System on Chip

Introduction

Contents

Chapter 1

System on a chip

The AMD Geode is an x86 compatible system on a chip.

A **system on a chip** or **system on chip** (**SoC** or **SOC**) is an integrated circuit (**IC**) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single chip substrate. SoCs are very common in the mobile electronics market because of their low power consumption.[1] A typical application is in the area of embedded systems.

The contrast with a microcontroller is one of degree. Microcontrollers typically have under 100 KB of RAM (often just a few kilobytes) and often really *are* single-chip-systems, whereas the term SoC is typically used for more powerful processors, capable of running software such as the desktop versions of Windows and Linux, which need external memory chips (flash, RAM) to be useful, and which are used with various external peripherals. In short, for larger systems, the term *system on a chip* is hyperbole, indicating technical direction more than reality: a high degree of chip integration, leading toward reduced manufacturing costs, and the production of smaller systems. Many systems are too complex to fit on just one chip built with a processor optimized for just one of the system's tasks.

When it is not feasible to construct a SoC for a particular application, an alternative is a system in package (SiP) comprising a number of chips in a single package. In large volumes, SoC is believed to be more cost-effective than SiP since it increases the yield of the fabrication and because its packaging is simpler.[2]

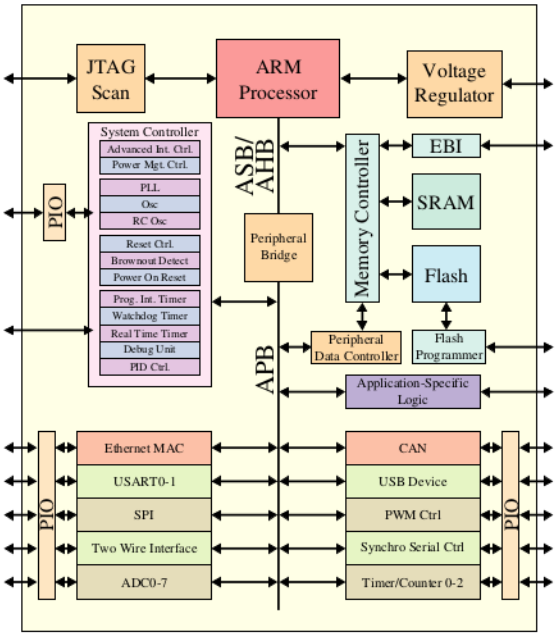
Another option, as seen for example in higher end cell phones is package on package stacking during board assembly. The SoC chip includes processors and numerous digital peripherals, and comes in a ball grid package with lower and upper connections. The lower balls connect to the board and various peripherals, with the upper balls in a ring holding the memory buses used to access NAND flash and DDR2 RAM. Memory packages could come from multiple vendors.

AMD Am286ZX/LX, SoC based on 80286

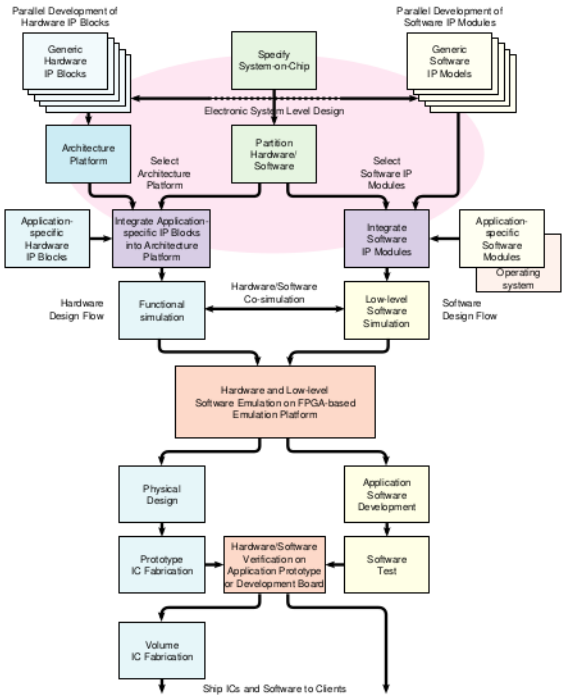
1.1 Structure

A typical SoC consists of:

- a microcontroller, microprocessor or digital signal processor (DSP) core – multiprocessor SoCs (MPSoC) having more than one processor core



Microcontroller-based system on a chip



System-on-a-chip design flow

- memory blocks including a selection of ROM, RAM, EEPROM and flash memory
- timing sources including oscillators and phase-locked loops
- peripherals including counter-timers, real-time timers and power-on reset generators
- external interfaces, including industry standards such as USB, FireWire, Ethernet, USART, SPI
- analog interfaces including ADCs and DACs
- voltage regulators and power management circuits

A bus – either proprietary or industry-standard such as the AMBA bus from ARM Holdings – connects these blocks. DMA controllers route data directly between external interfaces and memory, bypassing the processor core and thereby increasing the data throughput of the SoC.

1.2 Design flow

A SoC consists of both the hardware, described above, and the software controlling the microcontroller, microprocessor or DSP cores, peripherals and interfaces. The design flow for a SoC aims to develop this hardware and software in parallel.

Most SoCs are developed from pre-qualified hardware blocks for the hardware elements described above, together with the software drivers that control their operation. Of particular importance are the protocol stacks that drive industry-standard interfaces like USB. The hardware blocks are put together using CAD tools; the software modules are integrated using a software-development environment.

Chips are verified for logical correctness before being sent to foundry. This process is called functional verification and it accounts for a significant portion of the time and energy expended in the chip design life cycle (although the often quoted figure of 70% is probably an exaggeration).[3] With the growing complexity of chips, hardware verification languages like SystemVerilog, SystemC, e, and OpenVera are being used. Bugs found in the verification stage are reported to the designer.

Traditionally, engineers have employed simulation acceleration, emulation and/or an FPGA prototype to verify and debug both hardware and software for SoC designs prior to tapeout. With high capacity and fast compilation time, acceleration and emulation are powerful technologies that provide wide visibility into systems. Both technologies, however, operate slowly, on the order of MHz, which may be significantly slower – up to 100 times slower – than the SoC's operating frequency. Acceleration and emulation boxes are also very large and expensive at over US$1,000,000.

FPGA prototypes, in contrast, use FPGAs directly to enable engineers to validate and test at, or close to, a system's full operating frequency with real-world stimuli. Tools such as Certus[4] are used to insert probes in the FPGA RTL that make signals available for observation. This is used to debug hardware, firmware and software interactions across multiple FPGAs with capabilities sim-

ilar to a logic analyzer.

Once the hardware of the SoC is debugged, the place-and-route phase of the design of an integrated circuit or application-specific integrated circuit (ASIC) occurs before it is fabricated.

1.3 Fabrication

SoCs can be fabricated by several technologies, including:

- Full custom
- Standard cell
- Field-programmable gate array (FPGA)

SoC designs usually consume less power and have a lower cost and higher reliability than the multi-chip systems that they replace. And with fewer packages in the system, assembly costs are reduced as well.

However, like most VLSI designs, the total cost is higher for one large chip than for the same functionality distributed over several smaller chips, because of lower yields and higher non-recurring engineering costs.

1.4 Benchmarks

SoC research and development often compares many options. Benchmarks, such as COSMIC,[5] are developed to help such evaluations.

1.5 See also

- List of system-on-a-chip suppliers
- PSoC
- Electronic design automation
- Post-silicon validation
- Single-board computer
- Network on a chip
- Radio-on-a-chip
- ARM architecture
- Socionext

1.6 References

[1] Pete Bennett, EE Times. "The why, where and what of low-power SoC design." December 2, 2004. Retrieved July 28, 2015.

[2] EE Times. "The Great Debate: SOC vs. SIP." March 21, 2005. Retrieved July 28, 2015.

[3] EE Times. "Is verification really 70 percent?." June 14, 2004. Retrieved July 28, 2015.

[4] Brian Bailey, EE Times. "Tektronix hopes to shake up ASIC prototyping." October 30, 2012. Retrieved July 28, 2015.

[5] "COSMIC Heterogeneous Multiprocessor Benchmark Suite"

1.7 Further reading

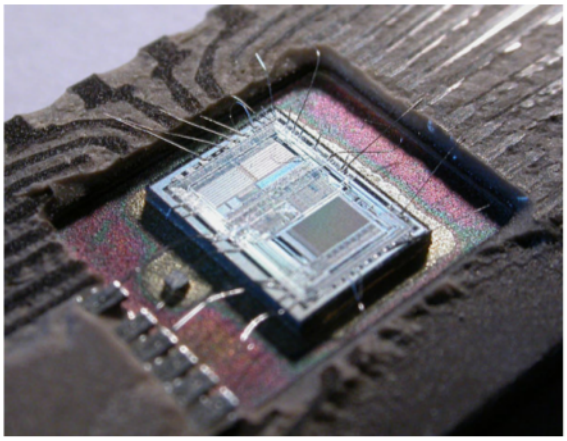
- Badawy, Wael; Jullien, Graham A., eds. (2003). *System-on-Chip for Real-Time Applications*. Kluwer international series in engineering and computer science, SECS 711. Boston: Kluwer Academic Publishers. ISBN 9781402072543. OCLC 50478525. 465 pages.
- Furber, Stephen B. (2000). *ARM system-on-chip architecture*. Boston: Addison-Wesley. ISBN 0-201-67519-6.

1.8 External links

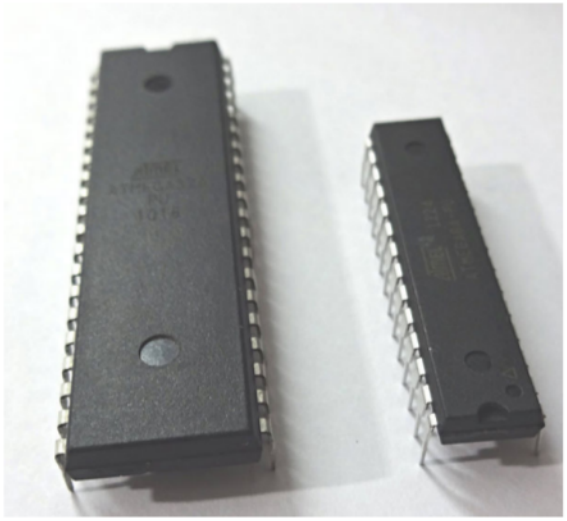
- SOCC Annual IEEE International SOC Conference
- Baya free SoC platform assembly and IP integration tool

Chapter 2

Microcontroller



The die from an Intel 8742, an 8-bit microcontroller that includes a CPU running at 12 MHz, 128 bytes of RAM, 2048 bytes of EPROM, and I/O in the same chip.



Two ATmega microcontrollers

A **microcontroller** is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Some microcontrollers may use four-bit words and operate at clock rate frequencies as low as 4 kHz, for low power consumption (single-digit milliwatts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

2.1 History

The first microprocessor was the 4-bit Intel 4004 released in 1971, with the Intel 8008 and other more capable microprocessors becoming available over the next several years. However, both processors required external chips to implement a working system, raising total system cost, and making it impossible to economically computerize appliances.

The Smithsonian Institution credits TI engineers Gary Boone and Michael Cochran with the successful creation of the first microcontroller in 1971. The result of their work was the TMS 1000, which became commercially available in 1974. It combined read-only memory, read/write memory, processor and clock on one chip and was targeted at embedded systems.[1]

Partly in response to the existence of the single-chip TMS 1000,[2] Intel developed a computer system on a chip optimized for control applications, the Intel 8048, with commercial parts first shipping in 1977.[2] It combined RAM and ROM on the same chip. This chip would find its way into over one billion PC keyboards, and other numerous applications. At that time Intel's President, Luke J. Valenter, stated that the microcontroller was one of the most successful in the company's history, and expanded the division's budget over 25%.

Most microcontrollers at this time had concurrent variants. One had an erasable EPROM program memory, with a transparent quartz window in the lid of the package to allow it to be erased by exposure to ultraviolet light, often used for prototyping. The other was either a mask programmed ROM from the manufacturer for large series, or a PROM variant which was only programmable once; sometimes this was signified with the designation OTP, standing for "one-time programmable". The PROM was of identical type of memory as the EPROM, but because there was no way to expose it to ultraviolet light, it could not be erased. The erasable versions required ceramic packages with quartz windows, making them significantly more expensive than the OTP versions, which could be made in lower-cost opaque plastic packages. For the erasable variants, quartz was required, instead of less expensive glass, for its transparency to ultraviolet—glass is largely opaque to UV—but the main cost differentiator was the ceramic package itself.

In 1993, the introduction of EEPROM memory allowed microcontrollers (beginning with the Microchip PIC16x84) to be electrically erased quickly without an expensive package as required for EPROM, allowing both rapid prototyping, and In System Programming. (EEPROM technology had been available prior to this time, but the earlier EEPROM was more expensive and less durable, making it unsuitable for low-cost mass-produced microcontrollers.) The same year, Atmel introduced the first microcontroller using Flash memory, a special type of EEPROM.[3] Other companies rapidly followed suit, with both memory types.

Cost has plummeted over time, with the cheapest 8-bit microcontrollers being available for under 0.25 USD in quantity (thousands) in 2009, and some 32-bit microcontrollers around US$1 for similar quantities.

Nowadays microcontrollers are cheap and readily available for hobbyists, with large online communities around certain processors.

In the future, MRAM could potentially be used in microcontrollers as it has infinite endurance and its incremental semiconductor wafer process cost is relatively low.

2.1.1 Volumes

In 2002, about 55% of all CPUs sold in the world were 8-bit microcontrollers and microprocessors.[4] Over two billion 8-bit microcontrollers were sold in 1997,[5] and according to Semico, over four billion 8-bit microcontrollers were sold in 2006.[6] More recently, Semico has claimed the MCU market grew 36.5% in 2010 and 12% in 2011.[7]

A typical home in a developed country is likely to have only four general-purpose microprocessors but around three dozen microcontrollers. A typical mid-range automobile has as many as 30 or more microcontrollers. They can also be found in many electrical devices such as washing machines, microwave ovens, and telephones.

> Historically, the 8-bit segment has dominated the MCU market [..] 16-bit microcontrollers became the largest volume MCU category in 2011, overtaking 8-bit devices for the first time that year [..] IC Insights believes the makeup of the MCU market will undergo substantial changes in the next five years with 32-bit devices steadily grabbing a greater share of sales and unit volumes. By 2017, 32-bit MCUs are expected to account for 55% of microcontroller sales [..] In terms of unit volumes, 32-bit MCUs are expected account for 38% of microcontroller shipments in 2017, while 16-bit devices will represent 34% of the total, and 4-/8-bit designs are forecast to be 28% of units sold that year.
>
> The 32-bit MCU market is expected to grow rapidly due to increasing demand for higher levels of precision in embedded-processing systems and the growth in connectivity using the Internet. [..] In the next few years, complex 32-bit MCUs are expected to account for over 25% of the processing power in vehicles.
>
> — IC Insights, MCU Market on Migration Path to 32-bit and ARM-based Devices[8]

In 2012, following a global crisis – a worst ever annual sales decline and recovery and average sales price year-over-year plunging 17% – the biggest reduction since the 1980s, the average price for a microcontroller was US$0.88 ($0.69 for 4-/8-bit, $0.59 for 16-bit, $1.76 for 32-bit).[8]

In 2012, worldwide sales of 8-bit microcontrollers were around $4 billion because they were so useful that many companies needed them to be able to progress into better technology. In 2012, 4-bit microcontrollers also see significant sales.[9]

In 2015, 8-bit microcontrollers can be bought for $0.311 (1,000 units),[10] 16-bit for $0.385 (1,000 units),[11] and

32-bit for $0.378 (1,000 units but at $0.35 for 5,000).[12]

A PIC 18F8720 ***microcontroller*** *in an 80-pin TQFP package.*

2.2 Embedded design

A microcontroller can be considered a self-contained system with a processor, memory and peripherals and can be used as an embedded system.[13] The majority of microcontrollers in use today are embedded in other machinery, such as automobiles, telephones, appliances, and peripherals for computer systems.

While some embedded systems are very sophisticated, many have minimal requirements for memory and program length, with no operating system, and low software complexity. Typical input and output devices include switches, relays, solenoids, LEDs, small or custom liquid-crystal displays, radio frequency devices, and sensors for data such as temperature, humidity, light level etc. Embedded systems usually have no keyboard, screen, disks, printers, or other recognizable I/O devices of a personal computer, and may lack human interaction devices of any kind.

2.2.1 Interrupts

Micro controllers must provide real time (predictable, though not necessarily fast) response to events in the embedded system they are controlling. When certain events occur, an interrupt system can signal the processor to suspend processing the current instruction sequence and to begin an interrupt service routine (ISR, or "interrupt handler"). The ISR will perform any processing required based on the source of the interrupt, before returning to the original instruction sequence. Possible interrupt sources are device dependent, and often include events such as an internal timer overflow, completing an analog to digital conversion, a logic level change on an input such as from a button being pressed, and data received on a communication link. Where power consumption is important as in battery operated devices, interrupts may also wake a microcontroller from a low power sleep state where the processor is halted until required to do something by a peripheral event.

2.2.2 Programs

Typically microcontroller programs must fit in the available on-chip program memory, since it would be costly to provide a system with external, expandable, memory. Compilers and assemblers are used to convert high-level language and assembler language codes into a compact machine code for storage in the microcontroller's memory. Depending on the device, the program memory may be permanent, read-only memory that can only be programmed at the factory, or program memory that may be field-alterable flash or erasable read-only memory.

Manufacturers have often produced special versions of their microcontrollers in order to help the hardware and software development of the target system. Originally these included EPROM versions that have a "window" on the top of the device through which program memory can be erased by ultraviolet light, ready for reprogramming after a programming ("burn") and test cycle. Since 1998, EPROM versions are rare and have been replaced by EEPROM and flash, which are easier to use (can be erased electronically) and cheaper to manufacture.

Other versions may be available where the ROM is accessed as an external device rather than as internal memory, however these are becoming increasingly rare due to the widespread availability of cheap microcontroller programmers.

The use of field-programmable devices on a microcontroller may allow field update of the firmware or permit late factory revisions to products that have been assembled but not yet shipped. Programmable memory also reduces the lead time required for deployment of a new product.

Where hundreds of thousands of identical devices are required, using parts programmed at the time of manufacture can be an economical option. These "mask programmed" parts have the program laid down in the same way as the logic of the chip, at the same time.

A customizable microcontroller incorporates a block of digital logic that can be personalized in order to provide additional processing capability, peripherals and interfaces that are adapted to the requirements of the application. For example, the AT91CAP from Atmel has a block of logic that can be customized during manufacture according to user requirements.

2.2.3 Other microcontroller features

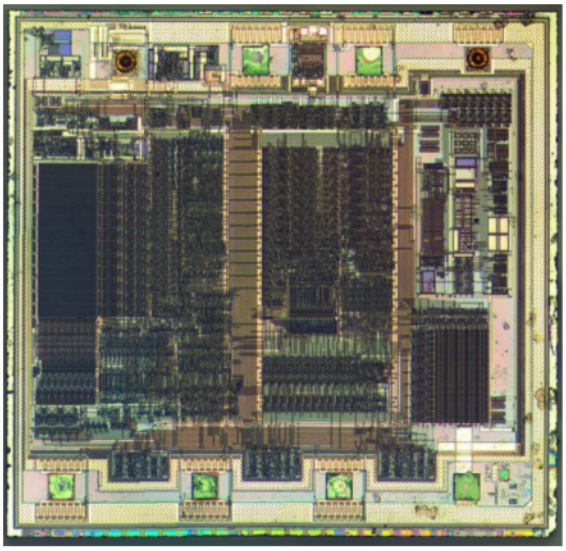
Microcontrollers usually contain from several to dozens of general purpose input/output pins (GPIO). GPIO pins are software configurable to either an input or an output state. When GPIO pins are configured to an input state, they are often used to read sensors or external signals. Configured to the output state, GPIO pins can drive external devices such as LEDs or motors, often indirectly, through external power electronics.

Many embedded systems need to read sensors that produce analog signals. This is the purpose of the analog-to-digital converter (ADC). Since processors are built to interpret and process digital data, i.e. 1s and 0s, they are not able to do anything with the analog signals that may be sent to it by a device. So the analog to digital converter is used to convert the incoming data into a form that the processor can recognize. A less common feature on some microcontrollers is a digital-to-analog converter (DAC) that allows the processor to output analog signals or voltage levels.

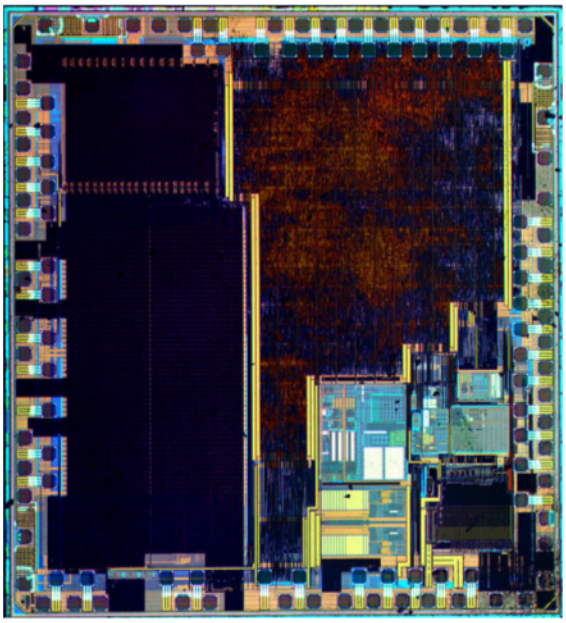
In addition to the converters, many embedded microprocessors include a variety of timers as well. One of the most common types of timers is the Programmable Interval Timer (PIT). A PIT may either count down from some value to zero, or up to the capacity of the count register, overflowing to zero. Once it reaches zero, it sends an interrupt to the processor indicating that it has finished counting. This is useful for devices such as thermostats, which periodically test the temperature around them to see if they need to turn the air conditioner on, the heater on, etc.

A dedicated Pulse Width Modulation (PWM) block makes it possible for the CPU to control power converters, resistive loads, motors, etc., without using lots of CPU resources in tight timer loops.

Universal Asynchronous Receiver/Transmitter (UART) block makes it possible to receive and transmit data over a serial line with very little load on the CPU. Dedicated on-chip hardware also often includes capabilities to communicate with other devices (chips) in digital formats such as Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), Universal Serial Bus (USB), and Ethernet.[14]



Die of a PIC12C508 8-bit, fully static, EEPROM/EPROM/ROM-based CMOS microcontroller manufactured by Microchip Technology using a 1200 nanometre process.



Die of a STM32F100C4T6B ARM Cortex-M3 microcontroller with 16 kilobytes flash memory, 24 MHz Central Processing Unit (CPU), motor control and Consumer Electronics Control (CEC) functions. Manufactured by STMicroelectronics.

2.3 Higher integration

Micro-controllers may not implement an external address or data bus as they integrate RAM and non-volatile memory on the same chip as the CPU. Using fewer pins, the chip can be placed in a much smaller, cheaper package.

Integrating the memory and other peripherals on a single chip and testing them as a unit increases the cost of that chip, but often results in decreased net cost of the embedded system as a whole. Even if the cost of a CPU that has integrated peripherals is slightly more than the cost of a CPU and external peripherals, having fewer chips typically allows a smaller and cheaper circuit board, and reduces the labor required to assemble and test the circuit board, in addition to tending to decrease the defect rate for the finished assembly.

A micro-controller is a single integrated circuit, commonly with the following features:

- central processing unit - ranging from small and simple 4-bit processors to complex 32-bit or 64-bit processors

- volatile memory (RAM) for data storage
- ROM, EPROM, EEPROM or Flash memory for program and operating parameter storage
- discrete input and output bits, allowing control or detection of the logic state of an individual package pin
- serial input/output such as serial ports (UARTs)
- other serial communications interfaces like I^2C, Serial Peripheral Interface and Controller Area Network for system interconnect
- peripherals such as timers, event counters, PWM generators, and watchdog
- clock generator - often an oscillator for a quartz timing crystal, resonator or RC circuit
- many include analog-to-digital converters, some include digital-to-analog converters
- in-circuit programming and in-circuit debugging support

This integration drastically reduces the number of chips and the amount of wiring and circuit board space that would be needed to produce equivalent systems using separate chips. Furthermore, on low pin count devices in particular, each pin may interface to several internal peripherals, with the pin function selected by software. This allows a part to be used in a wider variety of applications than if pins had dedicated functions.

Micro-controllers have proved to be highly popular in embedded systems since their introduction in the 1970s.

Some microcontrollers use a Harvard architecture: separate memory buses for instructions and data, allowing accesses to take place concurrently. Where a Harvard architecture is used, instruction words for the processor may be a different bit size than the length of internal memory and registers; for example: 12-bit instructions used with 8-bit data registers.

The decision of which peripheral to integrate is often difficult. The microcontroller vendors often trade operating frequencies and system design flexibility against time-to-market requirements from their customers and overall lower system cost. Manufacturers have to balance the need to minimize the chip size against additional functionality.

Microcontroller architectures vary widely. Some designs include general-purpose microprocessor cores, with one or more ROM, RAM, or I/O functions integrated onto the package. Other designs are purpose built for control applications. A micro-controller instruction set usually has many instructions intended for bit manipulation (bit-wise operations) to make control programs more compact.[15] For example, a general purpose processor might require several instructions to test a bit in a register and branch if the bit is set, where a micro-controller could have a single instruction to provide that commonly required function.

Microcontrollers typically do not have a math coprocessor, so floating point arithmetic is performed by software.

2.4 Programming environments

Microcontrollers were originally programmed only in assembly language, but various high-level programming languages are now also in common use to target microcontrollers. These languages are either designed specially for the purpose, or versions of general purpose languages such as the C programming language. Compilers for general purpose languages will typically have some restrictions as well as enhancements to better support the unique characteristics of microcontrollers. Some microcontrollers have environments to aid developing certain types of applications. Microcontroller vendors often make tools freely available to make it easier to adopt their hardware.

Many microcontrollers are so quirky that they effectively require their own non-standard dialects of C, such as SDCC for the 8051, which prevent using standard tools (such as code libraries or static analysis tools) even for code unrelated to hardware features. Interpreters are often used to hide such low level quirks.

Interpreter firmware is also available for some microcontrollers. For example, BASIC on the early microcontrollers Intel 8052;[16] BASIC and FORTH on the Zilog Z8[17] as well as some modern devices. Typically these interpreters support interactive programming.

Simulators are available for some microcontrollers. These allow a developer to analyze what the behavior of the microcontroller and their program should be if they were using the actual part. A simulator will show the internal processor state and also that of the outputs, as well as allowing input signals to be generated. While on the one hand most simulators will be limited from being unable to simulate much other hardware in a system, they can exercise conditions that may otherwise be hard to reproduce at will in the physical implementation, and can be the quickest way to debug and analyze problems.

Recent microcontrollers are often integrated with on-chip debug circuitry that when accessed by an in-circuit emulator via JTAG, allow debugging of the firmware with a debugger. A real-time ICE may allow viewing and/or manipulating of internal states while running. A tracing ICE can record executed program and MCU states before/after a trigger point.

2.5 Types of microcontrollers

See also: List of common microcontrollers

As of 2008, there are several dozen microcontroller architectures and vendors including:

- ARM core processors (many vendors)
  - ARM Cortex-M cores are specifically targeted towards microcontroller applications
- Atmel AVR (8-bit), AVR32 (32-bit), and AT91SAM (32-bit)
- Cypress Semiconductor's M8C Core used in their PSoC (Programmable System-on-Chip)
- Freescale ColdFire (32-bit) and S08 (8-bit)
- Freescale 68HC11 (8-bit), and others based on the Motorola 6800 family
- Intel 8051, also manufactured by NXP Semiconductors, Infineon and many others
- Infineon: 8-bit XC800, 16-bit XE166, 32-bit XMC4000 (ARM based Cortex M4F), 32-bit TriCore and, 32-bit Aurix Tricore Bit microcontrollers[18]
- MIPS
- Microchip Technology PIC, (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24), (32-bit PIC32)
- NXP Semiconductors LPC1000, LPC2000, LPC3000, LPC4000 (32-bit), LPC900, LPC700 (8-bit)
- Parallax Propeller
- PowerPC ISE
- Rabbit 2000 (8-bit)
- Renesas Electronics: RL78 16-bit MCU; RX 32-bit MCU; SuperH; V850 32-bit MCU; H8; R8C 16-bit MCU
- Silicon Laboratories Pipelined 8-bit 8051 Microcontrollers and mixed-signal ARM-based 32-bit microcontrollers
- STMicroelectronics STM8 (8-bit), ST10 (16-bit) and STM32 (32-bit)
- Texas Instruments TI MSP430 (16-bit), MSP432 (32-bit), C2000 (32-bit)
- Toshiba TLCS-870 (8-bit/16-bit)

Many others exist, some of which are used in very narrow range of applications or are more like applications processors than microcontrollers. The microcontroller market is extremely fragmented, with numerous vendors, technologies, and markets. Note that many vendors sell or have sold multiple architectures.

2.6 Interrupt latency

In contrast to general-purpose computers, microcontrollers used in embedded systems often seek to optimize interrupt latency over instruction throughput. Issues include both reducing the latency, and making it be more predictable (to support real-time control).

When an electronic device causes an interrupt, the intermediate results (registers) have to be saved before the software responsible for handling the interrupt can run. They must also be restored after that software is finished. If there are more registers, this saving and restoring process takes more time, increasing the latency. Ways to reduce such context/restore latency include having relatively few registers in their central processing units (undesirable because it slows down most non-interrupt processing substantially), or at least having the hardware not save them all (this fails if the software then needs to compensate by saving the rest “manually”). Another technique involves spending silicon gates on “shadow registers": One or more duplicate registers used only by the interrupt software, perhaps supporting a dedicated stack.

Other factors affecting interrupt latency include:

- Cycles needed to complete current CPU activities. To minimize those costs, microcontrollers tend to have short pipelines (often three instructions or less), small write buffers, and ensure that longer instructions are continuable or restartable. RISC design principles ensure that most instructions take the same number of cycles, helping avoid the need for most such continuation/restart logic.
- The length of any critical section that needs to be interrupted. Entry to a critical section restricts concurrent data structure access. When a data structure must be accessed by an interrupt handler, the critical section must block that interrupt. Accordingly, interrupt latency is increased by however long that interrupt is blocked. When there are hard external constraints on system latency, developers often need tools to measure interrupt latencies and track down which critical sections cause slowdowns.
  - One common technique just blocks all interrupts for the duration of the critical section. This is easy to implement, but sometimes critical sections get uncomfortably long.
  - A more complex technique just blocks the interrupts that may trigger access to that data

structure. This is often based on interrupt priorities, which tend to not correspond well to the relevant system data structures. Accordingly, this technique is used mostly in very constrained environments.

- Processors may have hardware support for some critical sections. Examples include supporting atomic access to bits or bytes within a word, or other atomic access primitives like the LDREX/STREX exclusive access primitives introduced in the ARMv6 architecture.

- Interrupt nesting. Some microcontrollers allow higher priority interrupts to interrupt lower priority ones. This allows software to manage latency by giving time-critical interrupts higher priority (and thus lower and more predictable latency) than less-critical ones.

- Trigger rate. When interrupts occur back-to-back, microcontrollers may avoid an extra context save/restore cycle by a form of tail call optimization.

Lower end microcontrollers tend to support fewer interrupt latency controls than higher end ones.

2.7 Microcontroller embedded memory technology

Since the emergence of microcontrollers, many different memory technologies have been used. Almost all microcontrollers have at least two different kinds of memory, a non-volatile memory for storing firmware and a read-write memory for temporary data.

2.7.1 Data

From the earliest microcontrollers to today, six-transistor SRAM is almost always used as the read/write working memory, with a few more transistors per bit used in the register file. FRAM or MRAM could potentially replace it as it is 4 to 10 times denser which would make it more cost effective.

In addition to the SRAM, some microcontrollers also have internal EEPROM for data storage; and even ones that do not have any (or not enough) are often connected to external serial EEPROM chip (such as the BASIC Stamp) or external serial flash memory chip.

A few recent microcontrollers beginning in 2003 have "self-programmable" flash memory.[3]

2.7.2 Firmware

The earliest microcontrollers used mask ROM to store firmware. Later microcontrollers (such as the early versions of the Freescale 68HC11 and early PIC microcontrollers) had quartz windows that allowed ultraviolet light in to erase the EPROM.

The Microchip PIC16C84, introduced in 1993,[19] was the first microcontroller to use EEPROM to store firmware. In the same year, Atmel introduced the first microcontroller using NOR Flash memory to store firmware.[3]

2.8 See also

- List of common microcontrollers
- List of open-source hardware projects
- Microbotics
- MCU with built in WiFi
- PIC microcontroller
- Programmable logic controller
- Single-board microcontroller

2.9 References

[1] Augarten, Stan (1983). *The Most Widely Used Computer on a Chip: The TMS 1000. State of the Art: A Photographic History of the Integrated Circuit* (New Haven and New York: Ticknor & Fields). ISBN 0-89919-195-9. Retrieved 2009-12-23.

[2] "Oral History Panel on the Development and Promotion of the Intel 8048 Microcontroller" (PDF). *Computer History Museum Oral History, 2008*. p. 4. Retrieved 2011-06-28.

[3] "Atmel's Self-Programming Flash Microcontrollers" (PDF). 2012-01-24. Retrieved 2008-10-25. by Odd Jostein Svendsli 2003

[4] Jim Turley. "The Two Percent Solution" 2002.

[5] Tom Cantrell "Microchip on the March". Circuit Cellar. 1998.

[6] http://www.semico.com

[7] Momentum Carries MCUs Into 2011 http://semico.com/content/momentum-carries-mcus-2011

[8] "MCU Market on Migration Path to 32-bit and ARM-based Devices". April 25, 2013. It typically takes a global economic recession to upset the diverse MCU marketplace, and that's exactly what occurred in 2009, when the microcontroller business suffered its worst-ever annual sales decline of 22% to $11.1 billion.

[9] Bill Giovino. "Zilog Buys Microcontroller Product Lines from Samsung". 2013.

[10] http://www.mouser.com/ProductDetail/Silicon-Labs/EFM8BB10F2G-A-QFN20/?qs=sGAEpiMZZMu9ReDVvI6ax9sqO0qrXlDW4ZuhKcnb2c%252bQvyUXU1UbuQ%3d%3d

[11] http://www.mouser.com/ProductDetail/Texas-Instruments/MSP430G2001IPW14R/?qs=sGAEpiMZZMvfhsTlJjecML5mLnp8Cec4esZ6%2f1aK7FQ%3d

[12] http://www.mouser.com/ProductDetail/Cypress-Semiconductor/CY8C4013SXI-400/?qs=sGAEpiMZZMuI9neUTtPr752e7iT1qQqS4inl2jxeWgxWqjKLOdzceQ%3d%3d

[13] Heath, Steve (2003). *Embedded systems design*. EDN series for design engineers (2 ed.). Newnes. pp. 11–12. ISBN 9780750655460.

[14] David Harris & Sarah Harris (2012). *Digital Design and Computer Architecture, Second Edition*, p. 515. Morgan Kaufmann. ISBN 0123944244.

[15] Easy Way to build a microcontroller project

[16] "8052-Basic Microcontrollers" by Jan Axelson 1994

[17] Edwards, Robert (1987). "Optimizing the Zilog Z8 Forth Microcontroller for Rapid Prototyping" (PDF). Martin Marietta: 3. Retrieved 9 December 2012.

[18] www.infineon.com/mcu

[19] Microchip unveils PIC16C84, a reprogrammable EEPROM-based 8-bit microcontroller 1993

2.10 External links

- Microcontroller at DMOZ
- Embedded Systems Design magazine

Chapter 3

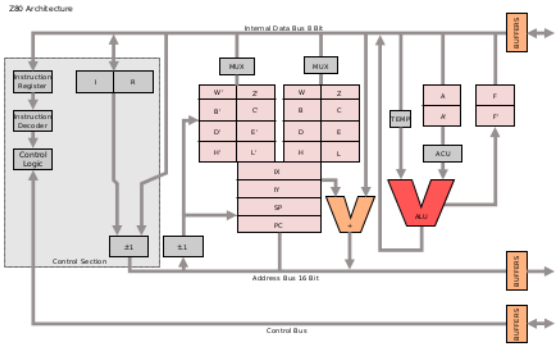
Microprocessor

See also: Processor (disambiguation), System on a chip, Microcontroller and Digital signal processor
This article is about Microprocessors. For Central Processing Units, see CPU.

A **microprocessor** is a computer processor that incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC),[1] or at most a few integrated circuits.[2] The microprocessor is a multipurpose, programmable device that accepts digital data as input, processes it according to instructions stored in its memory, and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.

The integration of a whole CPU onto a single chip or on a few chips greatly reduced the cost of processing power. Integrated circuit processors are produced in large numbers by highly automated processes resulting in a low per unit cost. Single-chip processors increase reliability as there are many fewer electrical connections to fail. As microprocessor designs get faster, the cost of manufacturing a chip (with smaller components built on a semiconductor chip the same size) generally stays the same.

Before microprocessors, small computers had been implemented using racks of circuit boards with many medium- and small-scale integrated circuits. Microprocessors integrated this into one or a few large-scale ICs. Continued increases in microprocessor capacity have since rendered other forms of computers almost completely obsolete (see history of computing hardware), with one or more microprocessors used in everything from the smallest embedded systems and handheld devices to the largest mainframes and supercomputers.



A block diagram of the internal architecture of the Z80 microprocessor, showing the arithmetic and logic section, register file, control logic section, and buffers to external address and data lines

3.1 Structure

The internal arrangement of a **microprocessor** varies depending on the age of the design and the intended purposes of the microprocessor. The complexity of an integrated circuit is bounded by physical limitations of the number of transistors that can be put onto one chip, the number of package terminations that can connect the processor to other parts of the system, the number of interconnections it is possible to make on the chip, and the heat that the chip can dissipate. Advancing technology makes more complex and powerful chips feasible to manufacture.

A minimal hypothetical microprocessor might only include an arithmetic logic unit (ALU) and a control logic section. The ALU performs operations such as addition, subtraction, and operations such as AND or OR. Each operation of the ALU sets one or more flags in a status register, which indicate the results of the last operation (zero value, negative number, overflow, or others). The control logic retrieves instruction codes from memory and initiates the sequence of operations required for the ALU to carry out the instruction. A single operation code might affect many individual data paths, registers, and other elements of the processor.

As integrated circuit technology advanced, it was feasible to manufacture more and more complex processors on a single chip. The size of data objects became larger; allowing more transistors on a chip allowed word sizes to increase from 4- and 8-bit words up to today's 64-bit words. Additional features were added to the processor architecture; more on-chip registers sped up programs, and complex instructions could be used to make more compact programs. Floating-point arithmetic, for example,

was often not available on 8-bit microprocessors, but had to be carried out in software. Integration of the floating point unit first as a separate integrated circuit and then as part of the same microprocessor chip, sped up floating point calculations.

Occasionally, physical limitations of integrated circuits made such practices as a bit slice approach necessary. Instead of processing all of a long word on one integrated circuit, multiple circuits in parallel processed subsets of each data word. While this required extra logic to handle, for example, carry and overflow within each slice, the result was a system that could handle, for example, 32-bit words using integrated circuits with a capacity for only four bits each.

With the ability to put large numbers of transistors on one chip, it becomes feasible to integrate memory on the same die as the processor. This CPU cache has the advantage of faster access than off-chip memory, and increases the processing speed of the system for many applications. Processor clock frequency has increased more rapidly than external memory speed, except in the recent past, so cache memory is necessary if the processor is not delayed by slower external memory.

3.1.1 Special-purpose designs

A microprocessor is a general purpose system. Several specialized processing devices have followed from the technology. Microcontrollers integrate a microprocessor with peripheral devices in embedded systems. A digital signal processor (DSP) is specialized for signal processing. Graphics processing units may have no limited or general programming facilities. For example, GPUs through the 1990s were mostly non-programmable and have only recently gained limited facilities like programmable vertex shaders.

32-bit processors have more digital logic than narrower processors, so 32-bit (and wider) processors produce more digital noise and have higher static consumption than narrower processors.[3] Reducing digital noise improves ADC conversion results.[4][5] So, 8-bit or 16-bit processors are better than 32-bit processors for system on a chip and microcontrollers that require extremely low-power electronics, or are part of a mixed-signal integrated circuit with noise-sensitive on-chip analog electronics such as high-resolution analog to digital converters, or both.

Nevertheless, trade-offs apply: running 32-bit arithmetic on an 8-bit chip could end up using more power, as the chip must execute software with multiple instructions. Modern microprocessors go into low power states when possible,[6] and a 8-bit chip running 32-bit software is active most of the time. This creates a delicate balance between software, hardware and use patterns, plus costs.

When manufactured on a similar process, 8-bit microprocessors use less power when operating and less power when sleeping than 32-bit microprocessors.[7]

However, some people say a 32-bit microprocessor may use less average power than an 8-bit microprocessor when the application requires certain operations such as floating-point math that take many more clock cycles on an 8-bit microprocessor than a 32-bit microprocessor so the 8-bit microprocessor spends more time in high-power operating mode.[7][8][9][10]

3.2 Embedded applications

Thousands of items that were traditionally not computer-related include microprocessors. These include large and small household appliances, cars (and their accessory equipment units), car keys, tools and test instruments, toys, light switches/dimmers and electrical circuit breakers, smoke alarms, battery packs, and hi-fi audio/visual components (from DVD players to phonograph turntables). Such products as cellular telephones, DVD video system and HDTV broadcast systems fundamentally require consumer devices with powerful, low-cost, microprocessors. Increasingly stringent pollution control standards effectively require automobile manufacturers to use microprocessor engine management systems, to allow optimal control of emissions over widely varying operating conditions of an automobile. Non-programmable controls would require complex, bulky, or costly implementation to achieve the results possible with a microprocessor.

A microprocessor control program (embedded software) can be easily tailored to different needs of a product line, allowing upgrades in performance with minimal redesign of the product. Different features can be implemented in different models of a product line at negligible production cost.

Microprocessor control of a system can provide control strategies that would be impractical to implement using electromechanical controls or purpose-built electronic controls. For example, an engine control system in an automobile can adjust ignition timing based on engine speed, load on the engine, ambient temperature, and any observed tendency for knocking—allowing an automobile to operate on a range of fuel grades.

3.3 History

The advent of low-cost computers on integrated circuits has transformed modern society. General-purpose microprocessors in personal computers are used for computation, text editing, multimedia display, and communication over the Internet. Many more microprocessors are part of embedded systems, providing digital control over myriad objects from appliances to automobiles to cellular

phones and industrial process control.

The first use of the term "microprocessor" is attributed to Viatron Computer Systems describing the custom integrated circuit used in their System 21 small computer system announced in 1968.

By the late-1960s, designers were striving to integrate the central processing unit (CPU) functions of a computer onto a handful of MOS LSI chips, called microprocessor unit (MPU) chip sets. Building on 8-bit arithmetic logic units (3800/3804) he designed earlier at Fairchild, in 1969 Lee Boysel created the Four-Phase Systems Inc. AL-1 an 8-bit CPU slice that was expandable to 32-bits. In 1970, Steve Geller and Ray Holt of Garrett AiResearch designed the MP944 chip set to implement the F-14A Central Air Data Computer on six metal-gate chips fabricated by AMI.

Intel introduced its first 4-bit microprocessor 4004 in 1971 and its 8-bit microprocessor 8008 in 1972. During the 1960s, computer processors were constructed out of small and medium-scale ICs—each containing from tens of transistors to a few hundred. These were placed and soldered onto printed circuit boards, and often multiple boards were interconnected in a chassis. The large number of discrete logic gates used more electrical power—and therefore produced more heat—than a more integrated design with fewer ICs. The distance that signals had to travel between ICs on the boards limited a computer's operating speed.

In the NASA Apollo space missions to the moon in the 1960s and 1970s, all onboard computations for primary guidance, navigation and control were provided by a small custom processor called "The Apollo Guidance Computer". It used wire wrap circuit boards whose only logic elements were three-input NOR gates.[11]

The first microprocessors emerged in the early 1970s and were used for electronic calculators, using binary-coded decimal (BCD) arithmetic on 4-bit words. Other embedded uses of 4-bit and 8-bit microprocessors, such as terminals, printers, various kinds of automation etc., followed soon after. Affordable 8-bit microprocessors with 16-bit addressing also led to the first general-purpose microcomputers from the mid-1970s on.

Since the early 1970s, the increase in capacity of microprocessors has followed Moore's law; this originally suggested that the number of components that can be fitted onto a chip doubles every year. With present technology, it is actually every two years,[12] and as such Moore later changed the period to two years.[13]

3.3.1 First projects

Three projects delivered a microprocessor at about the same time: Garrett AiResearch's Central Air Data Computer (CADC), Texas Instruments (TI) TMS 1000 (1971 September), and Intel's 4004 (1971 November).

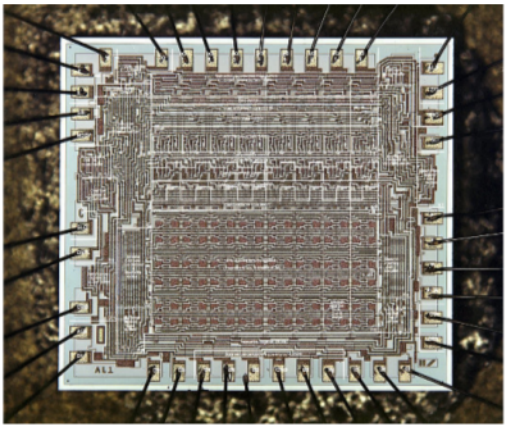
CADC

For more details on this topic, see Central Air Data Computer.

In 1968, Garrett AiResearch (which employed designers Ray Holt and Steve Geller) was invited to produce a digital computer to compete with electromechanical systems then under development for the main flight control computer in the US Navy's new F-14 Tomcat fighter. The design was complete by 1970, and used a MOS-based chipset as the core CPU. The design was significantly (approximately 20 times) smaller and much more reliable than the mechanical systems it competed against, and was used in all of the early Tomcat models. This system contained "a 20-bit, pipelined, parallel multi-microprocessor". The Navy refused to allow publication of the design until 1997. For this reason the CADC, and the MP944 chipset it used, are fairly unknown.[14] Ray Holt graduated from California Polytechnic University in 1968, and began his computer design career with the CADC. From its inception, it was shrouded in secrecy until 1998 when at Holt's request, the US Navy allowed the documents into the public domain. Since then people have debated whether this was the first microprocessor. Holt has stated that no one has compared this microprocessor with those that came later.[15] According to Parab et al. (2007), *"The scientific papers and literature published around 1971 reveal that the MP944 digital processor used for the F-14 Tomcat aircraft of the US Navy qualifies as the first microprocessor. Although interesting, it was not a single-chip processor, as was not the Intel 4004 – they both were more like a set of parallel building blocks you could use to make a general-purpose form. It contains a CPU, RAM, ROM, and two other support chips like the Intel 4004. It was made from the same P-channel technology, operated at military specifications and had larger chips -- an excellent computer engineering design by any standards. Its design indicates a major advance over Intel, and two year earlier. It actually worked and was flying in the F-14 when the Intel 4004 was announced. It indicates that today's industry theme of converging DSP-microcontroller architectures was started in 1971."*[16] This convergence of DSP and microcontroller architectures is known as a digital signal controller.[17]

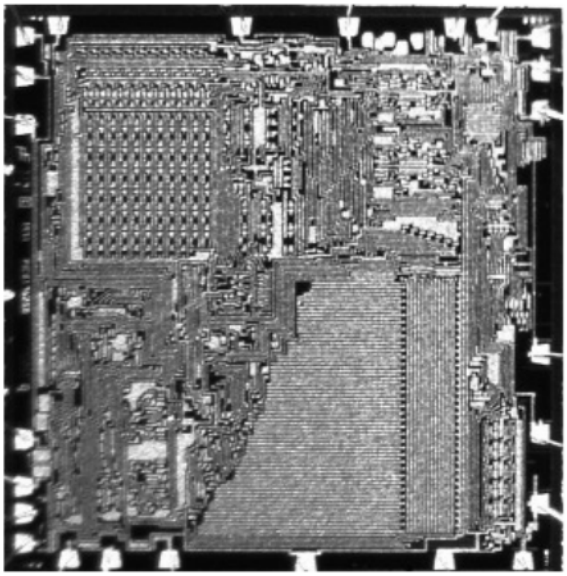
Four-Phase Systems AL1

The Four-Phase Systems AL1 was an 8-bit bit slice chip containing eight registers and an ALU.[18] It was designed by Lee Boysel in 1969.[19][20][21] At the time, it formed part of a nine-chip, 24-bit CPU with three AL1s, but it was later called a microprocessor when, in response to 1990s litigation by Texas Instruments, a demonstration system was constructed where a single AL1 formed part of a courtroom demonstration computer system, together with RAM, ROM, and an input-output device.[22]



AL1 by Four-Phase Systems Inc: one from the earliest inventions in the field of microprocessor technology

Pico/General Instrument



The PICO1/GI250 chip introduced in 1971. This was designed by Pico Electronics (Glenrothes, Scotland) and manufactured by General Instrument of Hicksville NY.

In 1971, Pico Electronics[23] and General Instrument (GI) introduced their first collaboration in ICs, a complete single chip calculator IC for the Monroe/Litton Royal Digital III calculator. This chip could also arguably lay claim to be one of the first microprocessors or microcontrollers having ROM, RAM and a RISC instruction set on-chip. The layout for the four layers of the PMOS process was hand drawn at x500 scale on mylar film, a significant task at the time given the complexity of the chip.

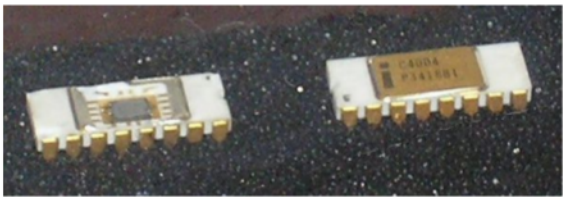
Pico was a spinout by five GI design engineers whose vision was to create single chip calculator ICs. They had significant previous design experience on multiple calculator chipsets with both GI and Marconi-Elliott.[24] The key team members had originally been tasked by Elliott Automation to create an 8-bit computer in MOS and had helped establish a MOS Research Laboratory in Glenrothes, Scotland in 1967.

Calculators were becoming the largest single market for semiconductors so Pico and GI went on to have significant success in this burgeoning market. GI continued to innovate in microprocessors and microcontrollers with products including the CP1600, IOB1680 and PIC1650.[25] In 1987 the GI Microelectronics business was spun out into the Microchip PIC microcontroller business.

Intel 4004

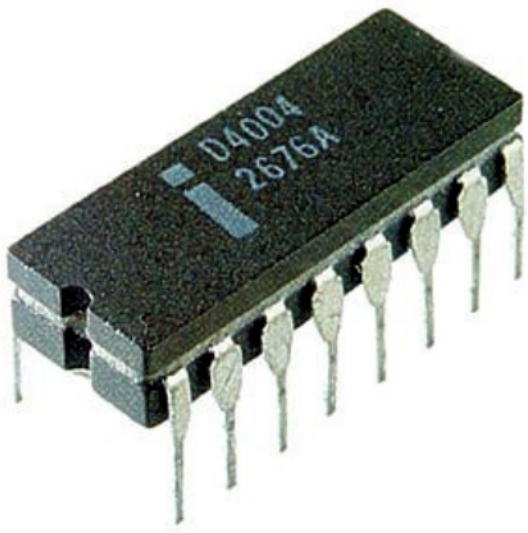
Main article: Intel 4004

The Intel 4004 is generally regarded as the first commercially available microprocessor,[26][27] and cost US$60 ($350.58 in 2016).[28] The first known advertisement for the 4004 is dated November 15, 1971 and appeared in Electronic News.[29] The project that produced the 4004 originated in 1969, when Busicom, a Japanese calculator manufacturer, asked Intel to build a chipset for high-performance desktop calculators. Busicom's original design called for a programmable chip set consisting of seven different chips. Three of the chips were to make a special-purpose CPU with its program stored in ROM and its data stored in shift register read-write memory. Ted Hoff, the Intel engineer assigned to evaluate the project, believed the Busicom design could be simplified by using dynamic RAM storage for data, rather than shift register memory, and a more traditional general-purpose CPU architecture. Hoff came up with a four-chip architectural proposal: a ROM chip for storing the programs, a dynamic RAM chip for storing data, a simple I/O device and a 4-bit central processing unit (CPU). Although not a chip designer, he felt the CPU could be integrated into a single chip, but as he lacked the technical know-how the idea remained just a wish for the time being.



The 4004 with cover removed (left) and as actually used (right)

While the architecture and specifications of the MCS-4 came from the interaction of Hoff with Stanley Mazor, a software engineer reporting to him, and with Busicom engineer Masatoshi Shima, during 1969, Mazor and Hoff moved on to other projects. In April 1970, Intel hired Italian-born engineer Federico Faggin as project leader, a move that ultimately made the single-chip CPU final design a reality (Shima meanwhile designed the Busicom calculator firmware and assisted Faggin during the first six months of the implementation). Faggin, who originally



Intel 4004, the first commercial microprocessor

developed the silicon gate technology (SGT) in 1968 at Fairchild Semiconductor[30] and designed the world's first commercial integrated circuit using SGT, the Fairchild 3708, had the correct background to lead the project into what would become the first commercial general purpose microprocessor. Since SGT was his very own invention, Faggin also used it to create his new methodology for random logic design that made it possible to implement a single-chip CPU with the proper speed, power dissipation and cost. The manager of Intel's MOS Design Department was Leslie L. Vadász at the time of the MCS-4 development but Vadász's attention was completely focused on the mainstream business of semiconductor memories so he left the leadership and the management of the MCS-4 project to Faggin, who was ultimately responsible for leading the 4004 project to its realization. Production units of the 4004 were first delivered to Busicom in March 1971 and shipped to other customers in late 1971.

Gilbert Hyatt

Gilbert Hyatt was awarded a patent claiming an invention pre-dating both TI and Intel, describing a "microcontroller".[31] The patent was later invalidated, but not before substantial royalties were paid out.[32][33]

TMS 1000

The Smithsonian Institution says TI engineers Gary Boone and Michael Cochran succeeded in creating the first microcontroller (also called a microcomputer) and the first single-chip CPU in 1971. The result of their work was the TMS 1000, which went on the market in 1974.[34] TI stressed the 4-bit TMS 1000 for use in pre-programmed embedded applications, introducing a version called the TMS1802NC on September 17, 1971 that implemented a calculator on a chip.

TI filed for a patent on the microprocessor. Gary Boone was awarded U.S. Patent 3,757,306 for the single-chip microprocessor architecture on September 4, 1973. In 1971 and again in 1976, Intel and TI entered into broad patent cross-licensing agreements, with Intel paying royalties to TI for the microprocessor patent. A history of these events is contained in court documentation from a legal dispute between Cyrix and Intel, with TI as inventor and owner of the microprocessor patent.

A computer-on-a-chip combines the microprocessor core (CPU), memory, and I/O (input/output) lines onto one chip. The computer-on-a-chip patent, called the "microcomputer patent" at the time, U.S. Patent 4,074,351, was awarded to Gary Boone and Michael J. Cochran of TI. Aside from this patent, the standard meaning of microcomputer is a computer using one or more microprocessors as its CPU(s), while the concept defined in the patent is more akin to a microcontroller.

3.3.2 8-bit designs

The Intel 4004 was followed in 1972 by the Intel 8008, the world's first 8-bit microprocessor. The 8008 was not, however, an extension of the 4004 design, but instead the culmination of a separate design project at Intel, arising from a contract with Computer Terminals Corporation, of San Antonio TX, for a chip for a terminal they were designing,[35] the Datapoint 2200—fundamental aspects of the design came not from Intel but from CTC. In 1968, CTC's Vic Poor and Harry Pyle developed the original design for the instruction set and operation of the processor. In 1969, CTC contracted two companies, Intel and Texas Instruments, to make a single-chip implementation, known as the CTC 1201.[36] In late 1970 or early 1971, TI dropped out being unable to make a reliable part. In 1970, with Intel yet to deliver the part, CTC opted to use their own implementation in the Datapoint 2200, using traditional TTL logic instead (thus the first machine to run "8008 code" was not in fact a microprocessor at all and was delivered a year earlier). Intel's version of the 1201 microprocessor arrived in late 1971, but was too late, slow, and required a number of additional support chips. CTC had no interest in using it. CTC had originally contracted Intel for the chip, and would have owed them US$50,000 ($292,153 in 2016) for their design work.[36] To avoid paying for a chip they did not want (and could not use), CTC released Intel from their contract and allowed them free use of the design.[36] Intel marketed it as the 8008 in April, 1972, as the world's first 8-bit microprocessor. It was the basis for the famous "Mark-8" computer kit advertised in the magazine *Radio-Electronics* in 1974. This processor had an 8-bit data bus and a 14-bit address bus.[37]

The 8008 was the precursor to the successful Intel 8080 (1974), which offered improved performance over the

8008 and required fewer support chips. Federico Faggin conceived and designed it using high voltage N channel MOS. The Zilog Z80 (1976) was also a Faggin design, using low voltage N channel with depletion load and derivative Intel 8-bit processors: all designed with the methodology Faggin created for the 4004. Motorola released the competing 6800 in August 1974, and the similar MOS Technology 6502 in 1975 (both designed largely by the same people). The 6502 family rivaled the Z80 in popularity during the 1980s.

A low overall cost, small packaging, simple computer bus requirements, and sometimes the integration of extra circuitry (e.g. the Z80's built-in memory refresh circuitry) allowed the home computer "revolution" to accelerate sharply in the early 1980s. This delivered such inexpensive machines as the Sinclair ZX-81, which sold for US$99 ($257.68 in 2016). A variation of the 6502, the MOS Technology 6510 was used in the Commodore 64 and yet another variant, the 8502, powered the Commodore 128.

The Western Design Center, Inc (WDC) introduced the CMOS 65C02 in 1982 and licensed the design to several firms. It was used as the CPU in the Apple IIe and IIc personal computers as well as in medical implantable grade pacemakers and defibrillators, automotive, industrial and consumer devices. WDC pioneered the licensing of microprocessor designs, later followed by ARM (32-bit) and other microprocessor intellectual property (IP) providers in the 1990s.

Motorola introduced the MC6809 in 1978. It was an ambitious and well thought-through 8-bit design that was source compatible with the 6800, and implemented using purely hard-wired logic (subsequent 16-bit microprocessors typically used microcode to some extent, as CISC design requirements were becoming too complex for pure hard-wired logic).

Another early 8-bit microprocessor was the Signetics 2650, which enjoyed a brief surge of interest due to its innovative and powerful instruction set architecture.

A seminal microprocessor in the world of spaceflight was RCA's RCA 1802 (aka CDP1802, RCA COSMAC) (introduced in 1976), which was used on board the Galileo probe to Jupiter (launched 1989, arrived 1995). RCA COSMAC was the first to implement CMOS technology. The CDP1802 was used because it could be run at very low power, and because a variant was available fabricated using a special production process, silicon on sapphire (SOS), which provided much better protection against cosmic radiation and electrostatic discharge than that of any other processor of the era. Thus, the SOS version of the 1802 was said to be the first radiation-hardened microprocessor.

The RCA 1802 had what is called a static design, meaning that the clock frequency could be made arbitrarily low, even to 0 Hz, a total stop condition. This let the Galileo spacecraft use minimum electric power for long uneventful stretches of a voyage. Timers or sensors would awaken the processor in time for important tasks, such as navigation updates, attitude control, data acquisition, and radio communication. Current versions of the Western Design Center 65C02 and 65C816 have static cores, and thus retain data even when the clock is completely halted.

3.3.3 12-bit designs

The Intersil 6100 family consisted of a 12-bit microprocessor (the 6100) and a range of peripheral support and memory ICs. The microprocessor recognised the DEC PDP-8 minicomputer instruction set. As such it was sometimes referred to as the **CMOS-PDP8**. Since it was also produced by Harris Corporation, it was also known as the **Harris HM-6100**. By virtue of its CMOS technology and associated benefits, the 6100 was being incorporated into some military designs until the early 1980s.

3.3.4 16-bit designs

The first multi-chip 16-bit microprocessor was the National Semiconductor IMP-16, introduced in early 1973. An 8-bit version of the chipset was introduced in 1974 as the IMP-8.

Other early multi-chip 16-bit microprocessors include one that Digital Equipment Corporation (DEC) used in the LSI-11 OEM board set and the packaged PDP 11/03 minicomputer—and the Fairchild Semiconductor MicroFlame 9440, both introduced in 1975–76. In 1975, National introduced the first 16-bit single-chip microprocessor, the National Semiconductor PACE, which was later followed by an NMOS version, the INS8900.

Another early single-chip 16-bit microprocessor was TI's TMS 9900, which was also compatible with their TI-990 line of minicomputers. The 9900 was used in the TI 990/4 minicomputer, the TI-99/4A home computer, and the TM990 line of OEM microcomputer boards. The chip was packaged in a large ceramic 64-pin DIP package, while most 8-bit microprocessors such as the Intel 8080 used the more common, smaller, and less expensive plastic 40-pin DIP. A follow-on chip, the TMS 9980, was designed to compete with the Intel 8080, had the full TI 990 16-bit instruction set, used a plastic 40-pin package, moved data 8 bits at a time, but could only address 16 KB. A third chip, the TMS 9995, was a new design. The family later expanded to include the 99105 and 99110.

The Western Design Center (WDC) introduced the CMOS 65816 16-bit upgrade of the WDC CMOS 65C02 in 1984. The 65816 16-bit microprocessor was the core of the Apple IIgs and later the Super Nintendo Entertainment System, making it one of the most popular 16-bit designs of all time.

Intel "upsized" their 8080 design into the 16-bit Intel 8086, the first member of the x86 family, which powers

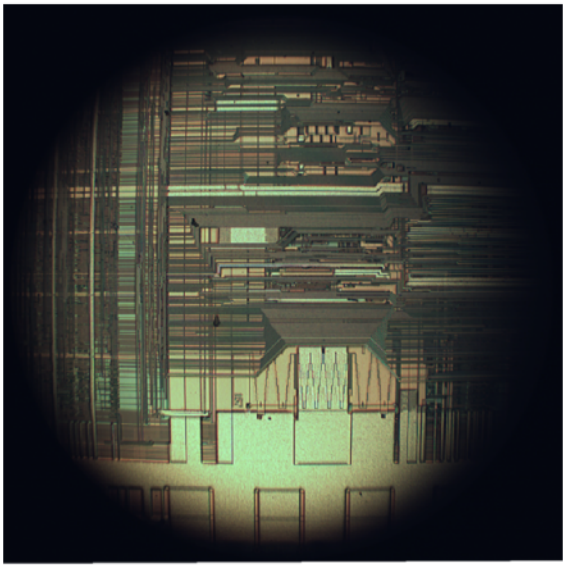
most modern PC type computers. Intel introduced the 8086 as a cost-effective way of porting software from the 8080 lines, and succeeded in winning much business on that premise. The 8088, a version of the 8086 that used an 8-bit external data bus, was the microprocessor in the first IBM PC. Intel then released the 80186 and 80188, the 80286 and, in 1985, the 32-bit 80386, cementing their PC market dominance with the processor family's backwards compatibility. The 80186 and 80188 were essentially versions of the 8086 and 8088, enhanced with some onboard peripherals and a few new instructions. Although Intel's 80186 and 80188 were not used in IBM PC type designs, second source versions from NEC, the V20 and V30 frequently were. The 8086 and successors had an innovative but limited method of memory segmentation, while the 80286 introduced a full-featured segmented memory management unit (MMU). The 80386 introduced a flat 32-bit memory model with paged memory management.

The 16-bit Intel x86 processors up to and including the 80386 do not include floating-point units (FPUs). Intel introduced the 8087, 80187, 80287 and 80387 math coprocessors to add hardware floating-point and transcendental function capabilities to the 8086 through 80386 CPUs. The 8087 works with the 8086/8088 and 80186/80188,[38] the 80187 works with the 80186 but not the 80188,[39] the 80287 works with the 80286 and the 80387 works with the 80386. The combination of an x86 CPU and an x87 coprocessor forms a single multichip microprocessor; the two chips are programmed as a unit using a single integrated instruction set.[40] The 8087 and 80187 coprocessors are connected in parallel with the data and address buses of their parent processor and directly execute instructions intended for them. The 80287 and 80387 coprocessors are interfaced to the CPU through I/O ports in the CPU's address space, this is transparent to the program, which does not need to know about or access these I/O ports directly; the program accesses the coprocessor and its registers through normal instruction opcodes.

3.3.5 32-bit designs

16-bit designs had only been on the market briefly when 32-bit implementations started to appear.

The most significant of the 32-bit designs is the Motorola MC68000, introduced in 1979. The 68k, as it was widely known, had 32-bit registers in its programming model but used 16-bit internal data paths, three 16-bit Arithmetic Logic Units, and a 16-bit external data bus (to reduce pin count), and externally supported only 24-bit addresses (internally it worked with full 32 bit addresses). In PC-based IBM-compatible mainframes the MC68000 internal microcode was modified to emulate the 32-bit System/370 IBM mainframe.[41] Motorola generally described it as a 16-bit processor. The combination of high performance, large (16 megabytes or 2^{24} bytes) memory space and fairly low cost made it the most popular CPU design of its class. The Apple Lisa and Macintosh designs made use of the 68000, as did a host of other designs in the mid-1980s, including the Atari ST and Commodore Amiga.



Upper interconnect layers on an Intel 80486DX2 die

The world's first single-chip fully 32-bit microprocessor, with 32-bit data paths, 32-bit buses, and 32-bit addresses, was the AT&T Bell Labs BELLMAC-32A, with first samples in 1980, and general production in 1982.[42][43] After the divestiture of AT&T in 1984, it was renamed the WE 32000 (WE for Western Electric), and had two follow-on generations, the WE 32100 and WE 32200. These microprocessors were used in the AT&T 3B5 and 3B15 minicomputers; in the 3B2, the world's first desktop super microcomputer; in the "Companion", the world's first 32-bit laptop computer; and in "Alexander", the world's first book-sized super microcomputer, featuring ROM-pack memory cartridges similar to today's gaming consoles. All these systems ran the UNIX System V operating system.

The first commercial, single chip, fully 32-bit microprocessor available on the market was the HP FOCUS.

Intel's first 32-bit microprocessor was the iAPX 432, which was introduced in 1981, but was not a commercial success. It had an advanced capability-based object-oriented architecture, but poor performance compared to contemporary architectures such as Intel's own 80286 (introduced 1982), which was almost four times as fast on typical benchmark tests. However, the results for the iAPX432 was partly due to a rushed and therefore suboptimal Ada compiler.

Motorola's success with the 68000 led to the MC68010, which added virtual memory support. The MC68020, introduced in 1984 added full 32-bit data and address buses. The 68020 became hugely popular in the Unix

supermicrocomputer market, and many small companies (e.g., Altos, Charles River Data Systems, Cromemco) produced desktop-size systems. The MC68030 was introduced next, improving upon the previous design by integrating the MMU into the chip. The continued success led to the MC68040, which included an FPU for better math performance. A 68050 failed to achieve its performance goals and was not released, and the follow-up MC68060 was released into a market saturated by much faster RISC designs. The 68k family faded from use in the early 1990s.

Other large companies designed the 68020 and follow-ons into embedded equipment. At one point, there were more 68020s in embedded equipment than there were Intel Pentiums in PCs.[44] The ColdFire processor cores are derivatives of the venerable 68020.

During this time (early to mid-1980s), National Semiconductor introduced a very similar 16-bit pinout, 32-bit internal microprocessor called the NS 16032 (later renamed 32016), the full 32-bit version named the NS 32032. Later, National Semiconductor produced the NS 32132, which allowed two CPUs to reside on the same memory bus with built in arbitration. The NS32016/32 outperformed the MC68000/10, but the NS32332—which arrived at approximately the same time as the MC68020—did not have enough performance. The third generation chip, the NS32532, was different. It had about double the performance of the MC68030, which was released around the same time. The appearance of RISC processors like the AM29000 and MC88000 (now both dead) influenced the architecture of the final core, the NS32764. Technically advanced—with a superscalar RISC core, 64-bit bus, and internally overclocked—it could still execute Series 32000 instructions through real-time translation.

When National Semiconductor decided to leave the Unix market, the chip was redesigned into the Swordfish Embedded processor with a set of on chip peripherals. The chip turned out to be too expensive for the laser printer market and was killed. The design team went to Intel and there designed the Pentium processor, which is very similar to the NS32764 core internally. The big success of the Series 32000 was in the laser printer market, where the NS32CG16 with microcoded BitBlt instructions had very good price/performance and was adopted by large companies like Canon. By the mid-1980s, Sequent introduced the first SMP server-class computer using the NS 32032. This was one of the design's few wins, and it disappeared in the late 1980s. The MIPS R2000 (1984) and R3000 (1989) were highly successful 32-bit RISC microprocessors. They were used in high-end workstations and servers by SGI, among others. Other designs included the Zilog Z80000, which arrived too late to market to stand a chance and disappeared quickly.

The ARM first appeared in 1985.[45] This is a RISC processor design, which has since come to dominate the 32-bit embedded systems processor space due in large part to its power efficiency, its licensing model, and its wide selection of system development tools. Semiconductor manufacturers generally license cores and integrate them into their own system on a chip products; only a few such vendors are licensed to modify the ARM cores. Most cell phones include an ARM processor, as do a wide variety of other products. There are microcontroller-oriented ARM cores without virtual memory support, as well as symmetric multiprocessor (SMP) applications processors with virtual memory.

From 1993 to 2003, the 32-bit x86 architectures became increasingly dominant in desktop, laptop, and server markets, and these microprocessors became faster and more capable. Intel had licensed early versions of the architecture to other companies, but declined to license the Pentium, so AMD and Cyrix built later versions of the architecture based on their own designs. During this span, these processors increased in complexity (transistor count) and capability (instructions/second) by at least three orders of magnitude. Intel's Pentium line is probably the most famous and recognizable 32-bit processor model, at least with the public at broad.

3.3.6 64-bit designs in personal computers

While 64-bit microprocessor designs have been in use in several markets since the early 1990s (including the Nintendo 64 gaming console in 1996), the early 2000s saw the introduction of 64-bit microprocessors targeted at the PC market.

With AMD's introduction of a 64-bit architecture backwards-compatible with x86, x86-64 (also called **AMD64**), in September 2003, followed by Intel's near fully compatible 64-bit extensions (first called IA-32e or EM64T, later renamed **Intel 64**), the 64-bit desktop era began. Both versions can run 32-bit legacy applications without any performance penalty as well as new 64-bit software. With operating systems Windows XP x64, Windows Vista x64, Windows 7 x64, Linux, BSD, and Mac OS X that run 64-bit native, the software is also geared to fully utilize the capabilities of such processors. The move to 64 bits is more than just an increase in register size from the IA-32 as it also doubles the number of general-purpose registers.

The move to 64 bits by PowerPC processors had been intended since the processors' design in the early 90s and was not a major cause of incompatibility. Existing integer registers are extended as are all related data pathways, but, as was the case with IA-32, both floating point and vector units had been operating at or above 64 bits for several years. Unlike what happened when IA-32 was extended to x86-64, no new general purpose registers were added in 64-bit PowerPC, so any performance gained when using the 64-bit mode for applications making no use of the larger address space is minimal.

In 2011, ARM introduced a new 64-bit ARM architecture.

3.3.7 RISC

Main article: Reduced instruction set computing

In the mid-1980s to early 1990s, a crop of new high-performance reduced instruction set computer (RISC) microprocessors appeared, influenced by discrete RISC-like CPU designs such as the IBM 801 and others. RISC microprocessors were initially used in special-purpose machines and Unix workstations, but then gained wide acceptance in other roles.

In 1986, HP released its first system with a PA-RISC CPU. The first commercial RISC microprocessor design was released in 1984 by MIPS Computer Systems, the 32-bit R2000 (the R1000 was not released). In 1987 in the non-Unix Acorn computers' 32-bit, then cache-less, ARM2-based Acorn Archimedes became the first commercial success using the ARM architecture, then known as Acorn RISC Machine (ARM); first silicon ARM1 in 1985. The R3000 made the design truly practical, and the R4000 introduced the world's first commercially available 64-bit RISC microprocessor. Competing projects would result in the IBM POWER and Sun SPARC architectures. Soon every major vendor was releasing a RISC design, including the AT&T CRISP, AMD 29000, Intel i860 and Intel i960, Motorola 88000, DEC Alpha.

In the late 1990s, only two 64-bit RISC architectures were still produced in volume for non-embedded applications: SPARC and Power ISA, but as ARM has become increasingly powerful, in the early 2010s, it became the third RISC architecture in the general computing segment.

3.3.8 Multi-core designs

Main article: Multi-core processor

A different approach to improving a computer's performance is to add extra processors, as in symmetric multiprocessing designs, which have been popular in servers and workstations since the early 1990s. Keeping up with Moore's Law is becoming increasingly challenging as chip-making technologies approach their physical limits. In response, microprocessor manufacturers look for other ways to improve performance so they can maintain the momentum of constant upgrades.

A multi-core processor is a single chip that contains more than one microprocessor core. Each core can simultaneously execute processor instructions in parallel. This effectively multiplies the processor's potential performance by the number of cores, if the software is designed to take advantage of more than one processor core. Some components, such as bus interface and cache, may be shared between cores. Because the cores are physically close to each other, they can communicate with each other much faster than separate (off-chip) processors in a multiprocessor system, which improves overall system performance.

In 2005, AMD released the first native dual-core processor, the Athlon X2. Intel's Pentium D had beaten the X2 to market by a few weeks, but it used two separate CPU dies and was less efficient than AMD's native design. As of 2012, dual- and quad-core processors are widely used in home PCs and laptops, while quad-, six-, eight-, ten-, twelve-, and sixteen-core processors are common in the professional and enterprise markets with workstations and servers.

Sun Microsystems has released the Niagara and Niagara 2 chips, both of which feature an eight-core design. The Niagara 2 supports more threads and operates at 1.6 GHz.

High-end Intel Xeon processors that are on the LGA 771, LGA 1366, and LGA 2011 sockets and high-end AMD Opteron processors that are on the C32 and G34 sockets are DP (dual processor) capable, as well as the older Intel Core 2 Extreme QX9775 also used in an older Mac Pro by Apple and the Intel Skulltrail motherboard. AMD's G34 motherboards can support up to four CPUs and Intel's LGA 1567 motherboards can support up to eight CPUs.

Modern desktop computers support systems with multiple CPUs, but few applications outside of the professional market can make good use of more than four cores. Both Intel and AMD currently offer fast quad, hex and octa-core desktop CPUs, making multi-CPU systems obsolete for many purposes. The desktop market has been in a transition towards quad-core CPUs since Intel's Core 2 Quad was released and are now common, although dual-core CPUs are still more prevalent. Older or mobile computers are less likely to have more than two cores than newer desktops. Not all software is optimised for multi-core CPUs, making fewer, more powerful cores preferable.

AMD offers CPUs with more cores for a given amount of money than similarly priced Intel CPUs—but the AMD cores are somewhat slower, so the two trade blows in different applications depending on how well-threaded the programs running are. For example, Intel's cheapest Sandy Bridge quad-core CPUs often cost almost twice as much as AMD's cheapest Athlon II, Phenom II, and FX quad-core CPUs but Intel has dual-core CPUs in the same price ranges as AMD's cheaper quad-core CPUs. In an application that uses one or two threads, the Intel dual-core CPUs outperform AMD's similarly priced quad-core CPUs—and if a program supports three or four threads the cheap AMD quad-core CPUs outperform the similarly priced Intel dual-core CPUs.

Historically, AMD and Intel have switched places as the company with the fastest CPU several times. Intel cur-

rently leads on the desktop side of the computer CPU market, with their Sandy Bridge and Ivy Bridge series. In servers, AMD's new Opterons seem to have superior performance for their price point. This means that AMD are currently more competitive in low- to mid-end servers and workstations that more effectively use fewer cores and threads.

3.4 Market statistics

In 1997, about 55% of all CPUs sold in the world are 8-bit microcontrollers, over two billion of which were sold.[46]

In 2002, less than 10% of all the CPUs sold in the world were 32-bit or more. Of all the 32-bit CPUs sold, about 2% are used in desktop or laptop personal computers. Most microprocessors are used in embedded control applications such as household appliances, automobiles, and computer peripherals. Taken as a whole, the average price for a microprocessor, microcontroller, or DSP is just over US$6 ($7.89 in 2016).[47]

In 2003, about US$44 ($56.6 in 2016) billion worth of microprocessors were manufactured and sold.[48] Although about half of that money was spent on CPUs used in desktop or laptop personal computers, those count for only about 2% of all CPUs sold.[47] The quality-adjusted price of laptop microprocessors improved −25% to −35% per year in 2004–10, and the rate of improvement slowed to −15% to −25% per year in 2010–13.[49]

About ten billion CPUs were manufactured in 2008. About 98% of new CPUs produced each year are embedded.[50]

3.5 See also

- Arithmetic logic unit
- Central processing unit
- Comparison of CPU architectures
- Computer architecture
- Computer engineering
- CPU design
- Floating point unit
- Instruction set
- List of instruction sets
- List of microprocessors
- Microarchitecture
- Microcode
- Microprocessor chronology

3.6 Notes

[1] Osborne, Adam (1980). *An Introduction to Microcomputers.* Volume 1: Basic Concepts (2nd ed.). Berkeley, California: Osborne-McGraw Hill. ISBN 0-931988-34-9.

[2] Krishna Kant *Microprocessors And Microcontrollers: Architecture Programming And System Design*, PHI Learning Pvt. Ltd., 2007 ISBN 81-203-3191-5, page 61, describing the iAPX 432.

[3] Kristian Saether, Ingar Fredriksen. "Introducing a New Breed of Microcontrollers for 8/16-bit Applications". p. 5.

[4] "AN1636: Understanding and minimising ADC conversion errors". 2003.

[5] Rahul Singh et. al. "Method and apparatus for reducing switching noise in a system-on-chip (SoC) integrated circuit including an analog-to-digital converter (ADC)". 2009.

[6] "Managing the Impact of Increasing Microprocessor Power Consumption" (PDF). *Rice University*. Retrieved October 1, 2015.

[7] CMicrotek. "8-bit vs 32-bit Micros". 2013.

[8] Richard York. "8-bit versus 32-bit MCUs - The impassioned debate goes on".

[9] "32-bit Microcontroller Technology: Reduced processing time".

[10] "Cortex-M3 Processor: Energy efficiency advantage".

[11] Back to the Moon: The Verification of a Small Microprocessor's Logic Design - NASA Office of Logic Design

[12] Moore, Gordon (19 April 1965). "Cramming more components onto integrated circuits" (PDF). *Electronics* **38** (8). Retrieved 2009-12-23.

[13] "Excerpts from A Conversation with Gordon Moore: Moore's Law" (PDF). Intel. 2005. Retrieved 2009-12-23.

[14] Holt, Ray M. "World's First Microprocessor Chip Set". Ray M. Holt website. Archived from the original on 2010-07-25. Retrieved 2010-07-25.

[15] Holt, Ray (27 September 2001). *Lecture: Microprocessor Design and Development for the US Navy F14 FighterJet* (Speech). Room 8220, Wean Hall, Carnegie Mellon University, Pittsburgh, PA, US. Retrieved 2010-07-25.

[16] Parab, Jivan S.; Shelake, Vinod G.; Kamat, Rajanish K.; Naik, Gourish M. (2007). *Exploring C for Microcontrollers: A Hands on Approach* (PDF). Springer. p. 4. ISBN 978-1-4020-6067-0. Retrieved 2010-07-25.

[17] Dyer, S. A.; Harms, B. K. (1993). "Digital Signal Processing". In Yovits, M. C. *Advances in Computers* **37**. Academic Press. pp. 104–107. doi:10.1016/S0065-2458(08)60403-9. ISBN 9780120121373.

[18] Basset, Ross (2003). "When is a Microprocessor not a Microprocessor? The Industrial Construction of Semiconductor Innovation". In Finn, Bernard. *Exposing Electronics*. Michigan State University Press. p. 121. ISBN 0-87013-658-5.

[19] "1971 - Microprocessor Integrates CPU Function onto a Single Chip". *The Silicon Engine*. Computer History Museum. Retrieved 2010-07-25.

[20] Shaller, Robert R. (15 April 2004). "Dissertation: Technological Innovation in the Semiconductor Industry: A Case Study of the International Technology Roadmap for Semiconductors" (PDF). George Mason University. Archived from the original (PDF) on 2006-12-19. Retrieved 2010-07-25.

[21] RW (3 March 1995). "Interview with Gordon E. Moore". *LAIR History of Science and Technology Collections*. Los Altos Hills, California: Stanford University.

[22] Bassett 2003. pp. 115, 122.

[23] McGonigal, James (20 September 2006). "Microprocessor History: Foundations in Glenrothes, Scotland". *McGonigal personal website accessdate=2009-12-23*.

[24] Tout, Nigel. "ANITA at its Zenith". *Bell Punch Company and the ANITA calculators*. Retrieved 2010-07-25.

[25] 16 Bit Microprocessor Handbook by Gerry Kane, Adam Osborne ISBN 0-07-931043-5 (0-07-931043-5)

[26] Mack, Pamela E. (30 November 2005). "The Microcomputer Revolution". Retrieved 2009-12-23.

[27] "History in the Computing Curriculum" (PDF). Retrieved 2009-12-23.

[28] Bright, Peter (November 15, 2011). "The 40th birthday of—maybe—the first microprocessor, the Intel 4004". arstechnica.com.

[29] Faggin, Federico; Hoff, Marcian E., Jr.; Mazor, Stanley; Shima, Masatoshi (December 1996). "The History of the 4004". *IEEE Micro* **16** (6): 10–20. doi:10.1109/40.546561.

[30] Faggin, F.; Klein, T.; Vadasz, L. (23 October 1968). *Insulated Gate Field Effect Transistor Integrated Circuits with Silicon Gates* (JPEG image). International Electronic Devices Meeting. IEEE Electron Devices Group. Retrieved 2009-12-23.

[31] Hyatt, Gilbert P., "Single chip integrated circuit computer architecture", Patent 4942516, issued July 17, 1990

[32] "The Gilbert Hyatt Patent". *intel4004.com*. Federico Faggin. Retrieved 2009-12-23.

[33] Crouch, Dennis (1 July 2007). "Written Description: CAFC Finds Prima Facie Rejection (Hyatt v. Dudas (Fed. Cir. 2007))". *Patently-O blog*. Retrieved 2009-12-23.

[34] Augarten, Stan (1983). *The Most Widely Used Computer on a Chip: The TMS 1000*. *State of the Art: A Photographic History of the Integrated Circuit* (New Haven and New York: Ticknor & Fields). ISBN 0-89919-195-9. Retrieved 2009-12-23.

[35] Ceruzzi, Paul E. (May 2003). *A History of Modern Computing* (2nd ed.). MIT Press. pp. 220–221. ISBN 0-262-53203-4.

[36] Wood, Lamont (August 2008). "Forgotten history: the true origins of the PC". Computerworld. Archived from the original on 2011-01-07. Retrieved 2011-01-07.

[37] Intel 8008 data sheet.

[38] Intel 8087 datasheet, pg. 1

[39] The 80187 only has a 16-bit data bus because it used the 80387SX core.

[40] "Essentially, the 80C187 can be treated as an additional resource or an extension to the CPU. The 80C186 CPU together with an 80C187 can be used as a single unified system." Intel 80C187 datasheet, p. 3, November 1992 (Order Number: 270640-004).

[41] "Priorartdatabase.com". Priorartdatabase.com. 1986-01-01. Retrieved 2014-06-09.

[42] "Shoji, M. Bibliography". Bell Laboratories. 7 October 1998. Retrieved 2009-12-23.

[43] "Timeline: 1982–1984". *Physical Sciences & Communications at Bell Labs*. Bell Labs, Alcatel-Lucent. 17 January 2001. Archived from the original on 2011-05-14. Retrieved 2009-12-23.

[44] Turley, Jim (July 1998). "MCore: Does Motorola Need Another Processor Family?". *Embedded Systems Design*. TechInsights (United Business Media). Archived from the original on 1998-07-02. Retrieved 2009-12-23.

[45] Garnsey, Elizabeth; Lorenzoni, Gianni; Ferriani, Simone (March 2008). "Speciation through entrepreneurial spin-off: The Acorn-ARM story" (PDF). *Research Policy* **37** (2). doi:10.1016/j.respol.2007.11.006. Retrieved 2011-06-02. [...] the first silicon was run on April 26th 1985.

[46] Cantrell, Tom (1998). "Microchip on the March". Archived from the original on 2007-02-20.

[47] Turley, Jim (18 December 2002). "The Two Percent Solution". *Embedded Systems Design*. TechInsights (United Business Media). Retrieved 2009-12-23.

[48] WSTS Board Of Directors. "WSTS Semiconductor Market Forecast World Release Date: 1 June 2004 - 6:00 UTC". *Miyazaki, Japan, Spring Forecast Meeting 18–21 May 2004* (Press release). World Semiconductor Trade Statistics. Archived from the original on 2004-12-07.

[49] Sun, Liyang (2014-04-25). "What We Are Paying for: A Quality Adjusted Price Index for Laptop Microprocessors". Wellesley College. Retrieved 2014-11-07. … compared with −25% to −35% per year over 2004-2010, the annual decline plateaus around −15% to −25% over 2010-2013.

[50] Barr, Michael (1 August 2009). "Real men program in C". *Embedded Systems Design*. TechInsights (United Business Media). p. 2. Retrieved 2009-12-23.

3.7 References

- Ray, A. K.; Bhurchand, K.M. *Advanced Microprocessors and Peripherals*. India: Tata McGraw-Hill.

3.8 External links

- Patent problems
- Dirk Oppelt. "The CPU Collection". Retrieved 2009-12-23.
- Gennadiy Shvets. "CPU-World". Retrieved 2009-12-23.
- Jérôme Cremet. "The Gecko's CPU Library". Retrieved 2009-12-23.
- "How Microprocessors Work". Retrieved 2009-12-23.
- William Blair. "IC Die Photography". Retrieved 2009-12-23.
- John Bayko (December 2003). "Great Microprocessors of the Past and Present". Retrieved 2009-12-23.
- Wade Warner (22 December 2004). "Great moments in microprocessor history". IBM. Retrieved 2013-03-07.
- Ray M. Holt. "theDocuments". *World's First Microprocessor*. Retrieved 2009-12-23.

Chapter 4

Digital signal processor

See also: Digital signal processing

A **digital signal processor** (**DSP**) is a specialized

A digital signal processor chip found in a guitar effects unit.

microprocessor (or a SIP block), with its architecture optimized for the operational needs of digital signal processing.[1][2]

The goal of DSPs is usually to measure, filter and/or compress continuous real-world analog signals. Most general-purpose microprocessors can also execute digital signal processing algorithms successfully, but dedicated DSPs usually have better power efficiency thus they are more suitable in portable devices such as mobile phones because of power consumption constraints.[3] DSPs often use special memory architectures that are able to fetch multiple data and/or instructions at the same time.

4.1 Overview

A typical digital processing system

Digital signal processing algorithms typically require a large number of mathematical operations to be performed quickly and repeatedly on a series of data samples. Signals (perhaps from audio or video sensors) are constantly converted from analog to digital, manipulated digitally, and then converted back to analog form. Many DSP applications have constraints on latency; that is, for the system to work, the DSP operation must be completed within some fixed time, and deferred (or batch) processing is not viable.

Most general-purpose microprocessors and operating systems can execute DSP algorithms successfully, but are not suitable for use in portable devices such as mobile phones and PDAs because of power efficiency constraints.[3] A specialized digital signal processor, however, will tend to provide a lower-cost solution, with better performance, lower latency, and no requirements for specialized cooling or large batteries.

The architecture of a digital signal processor is optimized specifically for digital signal processing. Most also support some of the features as an applications processor or microcontroller, since signal processing is rarely the only task of a system. Some useful features for optimizing DSP algorithms are outlined below.

4.2 Architecture

4.2.1 Software architecture

By the standards of general-purpose processors, DSP instruction sets are often highly irregular. One implication for software architecture is that hand-optimized assembly-code routines are commonly packaged into li-

braries for re-use, instead of relying on advanced compiler technologies to handle essential algorithms.

Instruction sets

- multiply–accumulates (MACs, including fused multiply–add, FMA) operations
  - used extensively in all kinds of matrix operations
    - convolution for filtering
    - dot product
    - polynomial evaluation
  - Fundamental DSP algorithms depend heavily on multiply–accumulate performance
    - FIR filters
    - Fast Fourier transform (FFT)
- Instructions to increase parallelism:
  - SIMD
  - VLIW
  - superscalar architecture
- Specialized instructions for modulo addressing in ring buffers and bit-reversed addressing mode for FFT cross-referencing
- Digital signal processors sometimes use time-stationary encoding to simplify hardware and increase coding efficiency.
- Multiple arithmetic units may require memory architectures to support several accesses per instruction cycle
- Special loop controls, such as architectural support for executing a few instruction words in a very tight loop without overhead for instruction fetches or exit testing

Data instructions

- Saturation arithmetic, in which operations that produce overflows will accumulate at the maximum (or minimum) values that the register can hold rather than wrapping around (maximum+1 doesn't overflow to minimum as in many general-purpose CPUs, instead it stays at maximum). Sometimes various sticky bits operation modes are available.
- Fixed-point arithmetic is often used to speed up arithmetic processing
- Single-cycle operations to increase the benefits of pipelining

Program flow

- Floating-point unit integrated directly into the datapath
- Pipelined architecture
- Highly parallel multiplier–accumulators (MAC units)
- Hardware-controlled looping, to reduce or eliminate the overhead required for looping operations

4.2.2 Hardware architecture

Memory architecture

DSPs are usually optimized for streaming data and use special memory architectures that are able to fetch multiple data and/or instructions at the same time, such as the Harvard architecture or Modified von Neumann architecture, which use separate program and data memories (sometimes even concurrent access on multiple data buses).

DSPs can sometimes rely on supporting code to know about cache hierarchies and the associated delays. This is a tradeoff that allows for better performance. In addition, extensive use of DMA is employed.

Addressing and virtual memory DSPs frequently use multi-tasking operating systems, but have no support for virtual memory or memory protection. Operating systems that use virtual memory require more time for context switching among processes, which increases latency.

- Hardware modulo addressing
  - Allows circular buffers to be implemented without having to test for wrapping
- Bit-reversed addressing, a special addressing mode
  - useful for calculating FFTs
- Exclusion of a memory management unit
- Memory-address calculation unit

4.3 History

Prior to the advent of stand-alone DSP chips discussed below, most DSP applications were implemented using bit-slice processors. The AMD 2901 bit-slice chip with its family of components was a very popular choice.

There were reference designs from AMD, but very often the specifics of a particular design were application specific. These bit slice architectures would sometimes include a peripheral multiplier chip. Examples of these multipliers were a series from TRW including the TDC1008 and TDC1010, some of which included an accumulator, providing the requisite multiply–accumulate (MAC) function.

In 1976, Richard Wiggins proposed the Speak & Spell concept to Paul Breedlove, Larry Brantingham, and Gene Frantz at Texas Instrument's Dallas research facility. Two years later in 1978 they produced the first Speak & Spell, with the technological centerpiece being the TMS5100,[4] the industry's first digital signal processor. It also set other milestones, being the first chip to use Linear predictive coding to perform speech synthesis.[5]

In 1978, Intel released the 2920 as an "analog signal processor". It had an on-chip ADC/DAC with an internal signal processor, but it didn't have a hardware multiplier and was not successful in the market. In 1979, AMI released the S2811. It was designed as a microprocessor peripheral, and it had to be initialized by the host. The S2811 was likewise not successful in the market.

In 1980 the first stand-alone, complete DSPs – the NEC μPD7720 and AT&T DSP1 – were presented at the International Solid-State Circuits Conference '80. Both processors were inspired by the research in PSTN telecommunications.

The Altamira DX-1 was another early DSP, utilizing quad integer pipelines with delayed branches and branch prediction.

Another DSP produced by Texas Instruments (TI), the TMS32010 presented in 1983, proved to be an even bigger success. It was based on the Harvard architecture, and so had separate instruction and data memory. It already had a special instruction set, with instructions like load-and-accumulate or multiply-and-accumulate. It could work on 16-bit numbers and needed 390 ns for a multiply–add operation. TI is now the market leader in general-purpose DSPs.

About five years later, the second generation of DSPs began to spread. They had 3 memories for storing two operands simultaneously and included hardware to accelerate tight loops, they also had an addressing unit capable of loop-addressing. Some of them operated on 24-bit variables and a typical model only required about 21 ns for a MAC. Members of this generation were for example the AT&T DSP16A or the Motorola 56000.

The main improvement in the third generation was the appearance of application-specific units and instructions in the data path, or sometimes as coprocessors. These units allowed direct hardware acceleration of very specific but complex mathematical problems, like the Fourier-transform or matrix operations. Some chips, like the Motorola MC68356, even included more than one processor core to work in parallel. Other DSPs from 1995 are the TI TMS320C541 or the TMS 320C80.

The fourth generation is best characterized by the changes in the instruction set and the instruction encoding/decoding. SIMD extensions were added, VLIW and the superscalar architecture appeared. As always, the clock-speeds have increased, a 3 ns MAC now became possible.

4.4 Modern DSPs

Modern signal processors yield greater performance; this is due in part to both technological and architectural advancements like lower design rules, fast-access two-level cache, (E)DMA circuitry and a wider bus system. Not all DSPs provide the same speed and many kinds of signal processors exist, each one of them being better suited for a specific task, ranging in price from about US$1.50 to US$300

Texas Instruments produces the C6000 series DSPs, which have clock speeds of 1.2 GHz and implement separate instruction and data caches. They also have an 8 MiB 2nd level cache and 64 EDMA channels. The top models are capable of as many as 8000 MIPS (instructions per second), use VLIW (very long instruction word), perform eight operations per clock-cycle and are compatible with a broad range of external peripherals and various buses (PCI/serial/etc). TMS320C6474 chips each have three such DSPs, and the newest generation C6000 chips support floating point as well as fixed point processing.

Freescale produces a multi-core DSP family, the MSC81xx. The MSC81xx is based on StarCore Architecture processors and the latest MSC8144 DSP combines four programmable SC3400 StarCore DSP cores. Each SC3400 StarCore DSP core has a clock speed of 1 GHz.

XMOS produces a multi-core multi-threaded line of processor well suited to DSP operations, They come in various speeds ranging from 400 to 1600 MIPS. The processors have a multi-threaded architecture that allows up to 8 real-time threads per core, meaning that a 4 core device would support up to 32 real time threads. Threads communicate between each other with buffered channels that are capable of up to 80 Mbit/s. The devices are easily programmable in C and aim at bridging the gap between conventional micro-controllers and FPGAs

CEVA, Inc. produces and licenses three distinct families of DSPs. Perhaps the best known and most widely deployed is the CEVA-TeakLite DSP family, a classic memory-based architecture, with 16-bit or 32-bit word-widths and single or dual MACs. The CEVA-X DSP family offers a combination of VLIW and SIMD architectures, with different members of the family offering dual or quad 16-bit MACs. The CEVA-XC DSP family targets Software-defined Radio (SDR) modem designs

and leverages a unique combination of VLIW and Vector architectures with 32 16-bit MACs.

Analog Devices produce the SHARC-based DSP and range in performance from 66 MHz/198 MFLOPS (million floating-point operations per second) to 400 MHz/2400 MFLOPS. Some models support multiple multipliers and ALUs, SIMD instructions and audio processing-specific components and peripherals. The Blackfin family of embedded digital signal processors combine the features of a DSP with those of a general use processor. As a result, these processors can run simple operating systems like μCLinux, velOSity and Nucleus RTOS while operating on real-time data.

NXP Semiconductors produce DSPs based on TriMedia VLIW technology, optimized for audio and video processing. In some products the DSP core is hidden as a fixed-function block into a SoC, but NXP also provides a range of flexible single core media processors. The TriMedia media processors support both fixed-point arithmetic as well as floating-point arithmetic, and have specific instructions to deal with complex filters and entropy coding.

CSR produces the Quatro family of SoCs that contain one or more custom Imaging DSPs optimized for processing document image data for scanner and copier applications.

Most DSPs use fixed-point arithmetic, because in real world signal processing the additional range provided by floating point is not needed, and there is a large speed benefit and cost benefit due to reduced hardware complexity. Floating point DSPs may be invaluable in applications where a wide dynamic range is required. Product developers might also use floating point DSPs to reduce the cost and complexity of software development in exchange for more expensive hardware, since it is generally easier to implement algorithms in floating point.

Generally, DSPs are dedicated integrated circuits; however DSP functionality can also be produced by using field-programmable gate array chips (FPGAs).

Embedded general-purpose RISC processors are becoming increasingly DSP like in functionality. For example, the OMAP3 processors include a ARM Cortex-A8 and C6000 DSP.

In Communications a new breed of DSPs offering the fusion of both DSP functions and H/W acceleration function is making its way into the mainstream. Such Modem processors include ASOCS ModemX and CEVA's XC4000.

4.5 See also

- Digital signal controller
- Graphics processing unit
- MDSP - a multiprocessor DSP

4.6 References

[1] Dyer, S. A.; Harms, B. K. (1993). "Digital Signal Processing". In Yovits, M. C. *Advances in Computers* **37**. Academic Press. pp. 104–107. doi:10.1016/S0065-2458(08)60403-9. ISBN 9780120121373.

[2] Liptak, B. G. (2006). *Process Control and Optimization*. Instrument Engineers' Handbook **2** (4th ed.). CRC Press. pp. 11–12. ISBN 9780849310812.

[3] Ingrid Verbauwhede; Patrick Schaumont; Christian Piguet; Bart Kienhuis (2005-12-24). "Architectures and Design techniques for energy efficient embedded DSP and multimedia processing" (PDF). rijndael.ece.vt.edu. Retrieved 2014-06-11.

[4] "Speak & Spell, the First Use of a Digital Signal Processing IC for Speech Generation, 1978". *IEEE Milestones*. IEEE. Retrieved 2012-03-02.

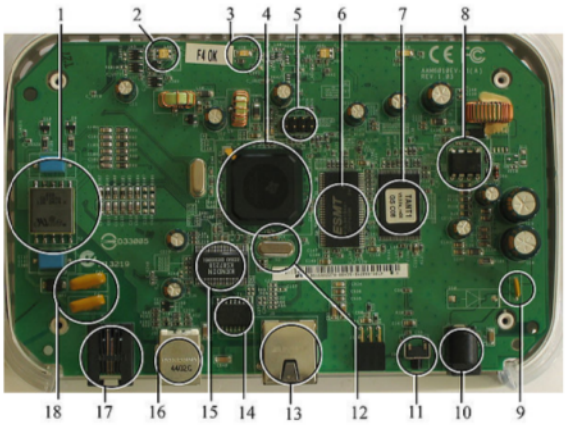
[5] Bogdanowicz, A. (2009-10-06). "IEEE Milestones Honor Three". *The Institute*. IEEE. Retrieved 2012-03-02.

4.7 External links

- DSP Online Book
- Pocket Guide to Processors for DSP - Berkeley Design Technology, INC

Chapter 5

Embedded system



Picture of the internals of an ADSL modem/router, a modern example of an embedded system. Labelled parts include a microprocessor (4), RAM (6), and flash memory (7).

An **embedded system** is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.[1][2] It is *embedded* as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today.[3] 98 percent of all microprocessors are manufactured as components of embedded systems.[4]

Examples of properties typical of embedded computers when compared with general-purpose counterparts are low power consumption, small size, rugged operating ranges, and low per-unit cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available.[5] For example, intelligent techniques can be designed to manage power consumption of embedded systems.[6]

Modern embedded systems are often based on microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces),[7] but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialised in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, and largely complex systems like hybrid vehicles, MRI, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

5.1 Varieties

Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical, commercial and military applications.

Telecommunications systems employ numerous embedded systems from telephone switches for the network to cell phones at the end-user. Computer networking uses dedicated routers and network bridges to route data.

Consumer electronics include personal digital assistants (PDAs), mp3 players, mobile phones, videogame consoles, digital cameras, DVD players, GPS receivers, and printers. Household appliances, such as microwave ovens, washing machines and dishwashers, include embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded

Embedded Computer Sub-Assembly for Accupoll Electronic Voting Machine[8]

devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors — brushless DC motors, induction motors and DC motors — use electric/electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.

Medical equipment uses embedded systems for vital signs monitoring, electronic stethoscopes for amplifying sounds, and various medical imaging (PET, SPECT, CT, MRI) for non-invasive internal inspections. Embedded systems within medical equipment are often powered by industrial computers.[9]

Embedded systems are used in transportation, fire safety, safety and security, medical applications and life critical systems, as these systems can be isolated from hacking and thus, be more reliable. For fire safety, the systems can be designed to have greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.

A new class of miniature wireless devices called motes are networked wireless sensors. Wireless sensor networking, WSN, makes use of miniaturization made possible by advanced IC design to couple full wireless subsystems to sophisticated sensors, enabling people and companies to measure a myriad of things in the physical world and act on this information through IT monitoring and control systems. These motes are completely self-contained, and will typically run off a battery source for years before the batteries need to be changed or charged.

Embedded Wi-Fi modules provide a simple means of wirelessly enabling any device which communicates via a serial port.

5.2 History

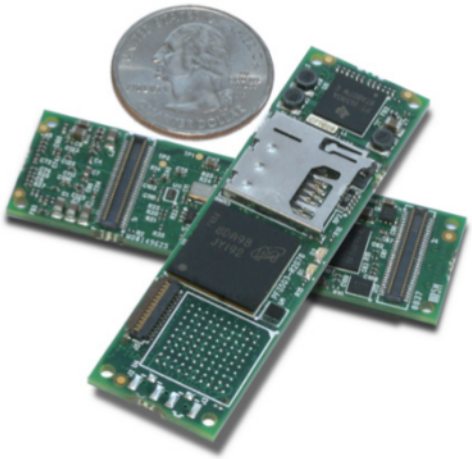
One of the very first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad nand gate ICs from $1000/each to $3/each, permitting their use in commercial products.

Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. An early microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expensive knob-based analog components such as potentiometers and variable capacitors with up/down buttons or knobs read out by a microprocessor even in consumer products. By the early 1980s, memory, input and output system components had been integrated into the same chip as the processor forming a microcontroller. Microcontrollers find applications where a general-purpose computer would be too costly.

A comparatively low-cost microcontroller may be programmed to fulfill the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself. Very few additional components may be needed and most of the design effort is in the software. Software prototype and test can be quicker compared with the design and construction of a new circuit not using an embedded processor.

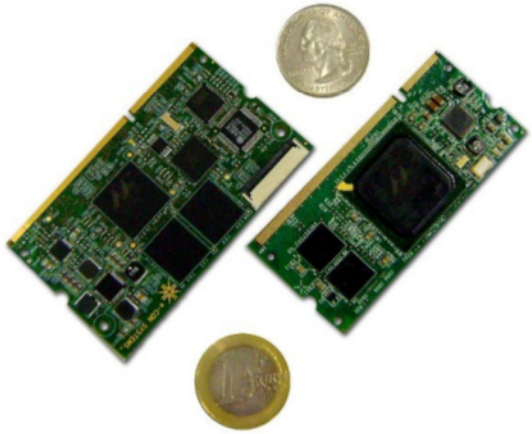
5.3 Characteristics



Gumstix Overo COM, a tiny, OMAP-based embedded computer-on-module with Wifi and Bluetooth.

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music.[10] Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.



e-con Systems eSOM270 & eSOM300 Computer on Modules

The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard or screen.

5.3.1 User interface

Embedded system text user interface using MicroVGA[nb 1]

Embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system.

More sophisticated devices which use a graphical screen with touch sensing or screen-edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what's desired. Handheld systems often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I^2C, etc.) or network (e.g. Ethernet) connection. This approach gives several advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies BSP and allows one to build a rich user interface on the PC. A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network router. The user interface is displayed in a web browser on a PC connected to the device, therefore needing no software to be installed.

5.3.2 Processors in embedded systems

Embedded processors can be broken into two broad categories. Ordinary microprocessors (µP) use separate integrated circuits for memory and peripherals. Microcontrollers (µC) have on-chip peripherals, thus reducing

power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

Numerous microcontrollers have been developed for embedded systems use. General-purpose microprocessors are also used in embedded systems, but generally require more support circuitry than microcontrollers.

Ready made computer boards

PC/104 and PC/104+ are examples of standards for *ready made* computer boards intended for small, low-volume embedded and ruggedized systems, mostly x86-based. These are often physically small compared to a standard PC, although still quite large compared to most simple (8/16-bit) embedded systems. They often use DOS, Linux, NetBSD, or an embedded real-time operating system such as MicroC/OS-II, QNX or VxWorks. Sometimes these boards use non-x86 processors.

In certain applications, where small size or power efficiency are not primary concerns, the components used may be compatible with those used in general purpose x86 personal computers. Boards such as the VIA EPIA range help to bridge the gap by being PC-compatible but highly integrated, physically smaller or have other attributes making them attractive to embedded engineers. The advantage of this approach is that low-cost commodity components may be used along with the same software development tools used for general software development. Systems built in this way are still regarded as embedded since they are integrated into larger devices and fulfill a single role. Examples of devices that may adopt this approach are ATMs and arcade machines, which contain code specific to the application.

However, most ready-made embedded systems boards are not PC-centered and do not use the ISA or PCI busses. When a System-on-a-chip processor is involved, there may be little benefit to having a standarized bus connecting discrete components, and the environment for both hardware and software tools may be very different.

One common design style uses a small system module, perhaps the size of a business card, holding high density BGA chips such as an ARM-based System-on-a-chip processor and peripherals, external flash memory for storage, and DRAM for runtime memory. The module vendor will usually provide boot software and make sure there is a selection of operating systems, usually including Linux and some real time choices. These modules can be manufactured in high volume, by organizations familiar with their specialized testing issues, and combined with much lower volume custom mainboards with application-specific external peripherals.

Implementation of embedded systems have advanced, embedded systems can easily be implemented with already made boards which are based on worldwide accepted platform. These platforms include but not limited to arduino, raspberry pi etc.

ASIC and FPGA solutions

A common array of n configuration for very-high-volume embedded systems is the system on a chip (SoC) which contains a complete system consisting of multiple processors, multipliers, caches and interfaces on a single chip. SoCs can be implemented as an application-specific integrated circuit (ASIC) or using a field-programmable gate array(FPGA).

5.3.3 Peripherals

A close-up of the SMSC LAN91C110 (SMSC 91x) chip, an embedded ethernet chip.

Embedded Systems talk with the outside world via peripherals, such as:

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485 etc.
- Synchronous Serial Communication Interface: I2C, SPI, SSC and ESSI (Enhanced Synchronous Serial Interface)
- Universal Serial Bus (USB)
- Multi Media Cards (SD Cards, Compact Flash etc.)
- Networks: Ethernet, LonWorks, etc.
- Fieldbuses: CAN-Bus, LIN-Bus, PROFIBUS, etc.

- Timers: PLL(s), Capture/Compare and Time Processing Units
- Discrete IO: aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging: JTAG, ISP, ICSP, BDM Port, BITP, and DB9 ports.

5.3.4 Tools

As with other software, embedded system designers use compilers, assemblers, and debuggers to develop embedded system software. However, they may also use some more specific tools:

- In circuit debuggers or emulators (see next section).
- Utilities to add a checksum or CRC to a program, so the embedded system can check if the program is valid.
- For systems using digital signal processing, developers may use a math workbench such as Scilab / Scicos, MATLAB / Simulink, EICASLAB, MathCad, Mathematica,or FlowStone DSP to simulate the mathematics. They might also use libraries for both the host and target which eliminates developing DSP routines as done in DSPnano RTOS.
- System Level Modeling and Simulation tools such as VisualSim helps designers to construct simulation models of a system with Hardware Components such as Processors, Memories, DMA, Interfaces, buses and Software behavior flow as a State diagram or flow diagram using configurable library blocks. Simulation is conducted to select right components by performing power vs performance trade-off, reliability analysis and bottleneck analysis. Typical reports that helps designer to make architecture decisions includes application latency, Device Throughput, Device Utilization, Power Consumption of full System as well as device level power consumption.
- A model based development tool like VisSim lets you create and simulate graphical data flow and UML State chart diagrams of components like digital filters, motor controllers, communication protocol decoding and multi-rate tasks. Interrupt handlers can also be created graphically. After simulation, you can automatically generate C-code to the VisSim RTOS which handles the main control task and preemption of background tasks, as well as automatic setup and programming of on-chip peripherals.
- Custom compilers and linkers may be used to optimize specialized hardware.
- An embedded system may have its own special language or design tool, or add enhancements to an existing language such as Forth or Basic.
- Another alternative is to add a real-time operating system or embedded operating system, which may have DSP capabilities like DSPnano RTOS.
- Modeling and code generating tools often based on state machines

Software tools can come from several sources:

- Software companies that specialize in the embedded market
- Ported from the GNU software development tools
- Sometimes, development tools for a personal computer can be used if the embedded processor is a close relative to a common PC processor

As the complexity of embedded systems grows, higher level tools and operating systems are migrating into machinery where it makes sense. For example, cellphones, personal digital assistants and other consumer computers often need significant software that is purchased or provided by a person other than the manufacturer of the electronics. In these systems, an open programming environment such as Linux, NetBSD, OSGi or Embedded Java is required so that the third-party software provider can sell to a large market.

5.3.5 Debugging

Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticated they can be roughly grouped into the following areas:

- Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
- External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multicore systems.
- An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
- An in-circuit emulator (ICE) replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.

- A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified, and allowing debugging on a normal PC. The downsides are expense and slow operation, in some cases up to 100X slower than the final system.
- For SoC designs, the typical approach is to verify and debug the design on an FPGA prototype board. Tools such as Certus[11] are used to insert probes in the FPGA RTL that make signals available for observation. This is used to debug hardware, firmware and software interactions across multiple FPGA with capabilities similar to a logic analyzer.

Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as HLL source-code, assembly code or mixture of both.

Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software- (and microprocessor-) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor). An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance.

Tracing

Real-time operating systems (RTOS) often supports tracing of operating system events. A graphical view is presented by a host PC tool, based on a recording of the system behavior. The trace recording can be performed in software, by the RTOS, or by special tracing hardware. RTOS tracing allows developers to understand timing and performance issues of the software system and gives a good understanding of the high-level system behaviors. Commercial tools like RTXC Quadros or IAR Systems exists.

5.3.6 Reliability

Embedded systems often reside in machines that are expected to run continuously for years without errors, and in some cases recover by themselves if an error occurs. Therefore, the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

- The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
- The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals.
- The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and also soft errors in the hardware:

- watchdog timer that resets the computer unless the software periodically notifies the watchdog
- subsystems with redundant spares that can be switched over to
- software "limp modes" that provide partial function
- Designing with a Trusted Computing Base (TCB) architecture[12] ensures a highly secure & reliable system environment
- A Hypervisor designed for embedded systems, is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.
- Immunity Aware Programming

5.3.7 High vs low volume

For high volume systems such as portable music players or mobile phones, minimizing cost is usually the primary design consideration. Engineers typically select hardware that is just "good enough" to implement the necessary functions.

For low-volume or prototype embedded systems, general purpose computers may be adapted by limiting the programs or by replacing the operating system with a real-time operating system.

5.4 Embedded software architectures

Main article: Embedded software

There are several different types of software architecture in common use.

5.4.1 Simple control loop

In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software.

5.4.2 Interrupt-controlled system

Some embedded systems are predominantly controlled by interrupts. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte.

These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

5.4.3 Cooperative multitasking

A nonpreemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to "run" in. When a task is idle, it calls an idle routine, usually called "pause", "wait", "yield", "nop" (stands for *no operation*), etc.

The advantages and disadvantages are similar to that of the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue.

5.4.4 Preemptive multitasking or multithreading

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non-blocking synchronization scheme.

Because of these complexities, it is common for organizations to use a real-time operating system (RTOS), allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a *generic* real time system, due to limitations regarding memory size, performance, or battery life. The choice that an RTOS is required brings in its own issues, however, as the selection must be done prior to starting to the application development process. This timing forces developers to choose the embedded operating system for their device based upon current requirements and so restricts future options to a large extent.[13] The restriction of future options becomes more of an issue as product life decreases. Additionally the level of complexity is continuously growing as devices are required to manage variables such as serial, USB, TCP/IP, Bluetooth, Wireless LAN, trunk radio, multiple channels, data and voice, enhanced graphics, multiple states, multiple threads, numerous wait states and so on. These trends are leading to the uptake of embedded middleware in addition to a real-time operating system.

5.4.5 Microkernels and exokernels

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast and fail when they are slow.

Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to and extensible by application programmers.

5.4.6 Monolithic kernels

In this case, a relatively large kernel with sophisticated capabilities is adapted to suit an embedded environment. This gives programmers an environment similar to a desktop operating system like Linux or Microsoft Windows, and is therefore very productive for development;

on the downside, it requires considerably more hardware resources, is often more expensive, and, because of the complexity of these kernels, can be less predictable and reliable.

Common examples of embedded monolithic kernels are embedded Linux and Windows CE.

Despite the increased cost in hardware, this type of embedded system is increasing in popularity, especially on the more powerful embedded devices such as wireless routers and GPS navigation systems. Here are some of the reasons:

- Ports to common embedded chip sets are available.
- They permit re-use of publicly available code for device drivers, web servers, firewalls, and other code.
- Development systems can start out with broad feature-sets, and then the distribution can be configured to exclude unneeded functionality, and save the expense of the memory that it would consume.
- Many engineers believe that running application code in user mode is more reliable and easier to debug, thus making the development process easier and the code more portable.
- Features requiring faster response than can be guaranteed can often be placed in hardware.

5.4.7 Additional software components

In addition to the core operating system, many embedded systems have additional upper-layer software components. These components consist of networking protocol stacks like CAN, TCP/IP, FTP, HTTP, and HTTPS, and also included storage capabilities like FAT and flash memory management systems. If the embedded device has audio and video capabilities, then the appropriate drivers and codecs will be present in the system. In the case of the monolithic kernels, many of these software layers are included. In the RTOS category, the availability of the additional software components depends upon the commercial offering.

5.5 See also

- Communications server
- Cyber-physical system
- DSP
- Electronic Control Unit
- Hypervisor
- Embedded operating systems
- Embedded software
- Firmware
- FPGA
- Information appliance
- Microprocessor
- Microcontroller
- Programming languages
- Real-time operating system
- Software engineering
- System on a chip
- System on module
- Ubiquitous computing

5.6 Notes

[1] For more details of MicroVGA see this PDF.

5.7 References

[1] Michael Barr. "Embedded Systems Glossary". *Neutrino Technical Library*. Retrieved 2007-04-21.

[2] Heath, Steve (2003). *Embedded systems design*. EDN series for design engineers (2 ed.). Newnes. p. 2. ISBN 978-0-7506-5546-0. An embedded system is a microprocessor based system that is built to control a function or a range of functions.

[3] Michael Barr; Anthony J. Massa (2006). "Introduction". *Programming embedded systems: with C and GNU development tools*. O'Reilly. pp. 1–2. ISBN 978-0-596-00983-0.

[4] Barr, Michael (1 August 2009). "Real men program in C". *Embedded Systems Design*. TechInsights (United Business Media). p. 2. Retrieved 2009-12-23.

[5] C.Alippi: *Intelligence for Embedded Systems*. Springer, 2014, 283pp, ISBN 978-3-319-05278-6.

[6] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems", IJCAET, 6(4), 440–459, 2014.

[7] Giovino, Bill. "Micro controller.com – Embedded Systems supersite".

[8] Electronic Voting Machine Information Sheet Accupoll AVS 1000

[9] Embedded Systems Dell OEM Solutions | Dell. Content.dell.com (2011-01-04). Retrieved on 2013-02-06.

[10] Embedded.com – Under the Hood: Robot Guitar embeds autotuning By David Carey, TechOnline EE Times (04/22/08, 11:10:00 AM EDT)Embedded Systems Design – Embedded.com

[11] "Tektronix Shakes Up Prototyping, Embedded Instrumentation Boosts Boards to Emulator Status". Electronic Engineering Journal. 2012-10-30. Retrieved 2012-10-30.

[12] Heiser, Gernot (December 2007). "Your System is secure? Prove it!" (PDF). *;login:* **2** (6): 35–8.

[13] "Working across Multiple Embedded Platforms" (PDF). clarinox. Retrieved 2010-08-17.

5.8 Further reading

- John Catsoulis (May 2005). *Designing Embedded Hardware, 2nd Edition*. O'Reilly. ISBN 0-596-00755-8.
- James M. Conrad; Alexander G. Dean (September 2011). *Embedded Systems, An Introduction Using the Renesas RX62N Microcontroller*. Micrium. ISBN 978-1935-7729-96.

5.9 External links

- Embedded Systems course with mbed YouTube, ongoing from 2015
- Trends in Cyber Security and Embedded Systems Dan Geer, November 2013
- Modern Embedded Systems Programming Video Course YouTube, ongoing from 2013
- Embedded Systems Week (ESWEEK) yearly event with conferences, workshops and tutorials covering all aspects of embedded systems and software
- Workshop on Embedded and Cyber-Physical Systems Education, workshop covering educational aspects of embedded systems

Chapter 6

MPSoC

The **multiprocessor System-on-Chip** (**MPSoC**) is a system-on-a-chip (SoC) which uses multiple processors (see multi-core), usually targeted for embedded applications. It is used by platforms that contain multiple, usually heterogeneous, processing elements with specific functionalities reflecting the need of the expected application domain, a memory hierarchy (often using scratchpad RAM and DMA) and I/O components. All these components are linked to each other by an on-chip interconnect. These architectures meet the performance needs of multimedia applications, telecommunication architectures, network security and other application domains while limiting the power consumption through the use of specialised processing elements and architecture.

6.1 Benchmarks

MPSoC research and development often compares many options. Benchmarks, such as COSMIC,[1] are developed to help such evaluations.

6.2 Examples

- CELL processor
- Adapteva epiphany architecture

6.3 See also

- Multi-core (computing)
- System-on-a-chip
- Many-core processing unit
- Multiprocessing
- Symmetric multiprocessing (SMP)
- multitasking
- Parallel computing

6.4 External links

- MPSoC - Annual Conference on MPSoC
- Annual Symposium

6.5 References

[1] "COSMIC Heterogeneous Multiprocessor Benchmark Suite"

Chapter 7

System in package

A **system in package** (**SiP**) or **system-in-a-package** is a number of integrated circuits enclosed in a single module (package). The SiP performs all or most of the functions of an electronic system, and is typically used inside a mobile phone, digital music player, etc. [1] Dies containing integrated circuits may be stacked vertically on a substrate. They are internally connected by fine wires that are bonded to the package. Alternatively, with a flip chip technology, solder bumps are used to join stacked chips together.

SiP dies can be stacked vertically or tiled horizontally, unlike slightly less dense multi-chip modules, which place dies horizontally on a carrier. SiP connects the dies with standard off-chip wire bonds or solder bumps, unlike slightly denser three-dimensional integrated circuits which connect stacked silicon dies with conductors running through the die.

Many different **3-D packaging** techniques have been developed for stacking many more-or-less standard chip dies into a compact area.[2]

An example SiP can contain several chips—such as a specialized processor, DRAM, flash memory—combined with passive components—resistors and capacitors—all mounted on the same substrate. This means that a complete functional unit can be built in a multi-chip package, so that few external components need to be added to make it work. This is particularly valuable in space constrained environments like MP3 players and mobile phones as it reduces the complexity of the printed circuit board and overall design. Despite its benefits, this technique decreases the yield of fabrication since any defective chip in the package will result in a non-functional packaged integrated circuit, even if all other modules in that same package are functional.

7.1 Suppliers

- Atmel
- NANIUM, S.A.
- Advanced Semiconductor Engineering, Inc.
- CeraMicro
- ChipSiP Technology
- STATS ChipPAC Ltd
- Toshiba
- Amkor
- Renesas
- SanDisk
- Samsung

7.2 See also

- System on a chip
- Package on package

7.3 References

[1] Pushkar Apte, W. R. Bottoms, William Chen and George Scalise, IEEE Spectrum. "Advanced Chip Packaging Satisfies Smartphone Needs." February 8, 2011. Retrieved July 31, 2015.

[2] R. Wayne Johnson, Mark Strickland and David Gerke, NASA Electronic Parts and Packaging Program. "3-D Packaging: A Technology Review." June 23, 2005. Retrieved July 31, 2015.

Chapter 8

Universal Synchronous/Asynchronous Receiver/Transmitter

A **Universal Synchronous/Asynchronous Receiver/Transmitter** (USART) is a type of a serial interface device that can be programmed to communicate asynchronously or synchronously. See Universal asynchronous receiver/transmitter (UART) for a discussion of the asynchronous capabilities of these devices.

8.1 Purpose and History

The USART's synchronous capabilities were primarily intended to support synchronous protocols like IBM's Synchronous transmit-receive (STR), Binary Synchronous Communications (BSC), Synchronous Data Link Control (SDLC), and the ISO-standard High-Level Data Link Control (HDLC) synchronous link-layer protocols, which were used with synchronous voice-frequency modems. These protocols were designed to make the best use of bandwidth when modems were analog devices. In those times, the fastest asynchronous voice-band modem could achieve at most speeds of 300 bps using frequency-shift keying, while synchronous modems could run at speeds up to 9600 bps using phase-shift keying. Synchronous transmission used only slightly over 80% of the bandwidth of the now more-familiar asynchronous transmission, since start and stop bits were unnecessary. Those modems are obsolete, having been replaced by modems with which convert asynchronous data to synchronous forms, but similar synchronous telecommunications protocols survive in numerous block-oriented technologies such as the widely-used IEEE 802.2 (Ethernet) link-level protocol. USARTs are still sometimes integrated with MCUs. USARTs are still used in routers that connect to external CSU/DSU devices, and they often use either Cisco's proprietary HDLC implementation or the IETF standard Point-to-Point Protocol in HDLC-like framing as defined in RFC 1662.

8.2 Operation

The operation of a USART is intimately related to the various protocols; refer to those pages for details. This section only provides a few general notes.

- USARTs in synchronous mode transmits data in frames. In synchronous operation, characters must be provided on time until a frame is complete; if the controlling processor does not do so, this is an *"underrun error*," and transmission of the frame is aborted.
- USARTs operating as synchronous devices used either character-oriented or bit-oriented mode. In character (STR and BSC) modes, the device relied on particular characters to define frame boundaries; in bit (HDLC and SDLC) modes earlier devices relied on physical-layer signals, while later devices took over the physical-layer recognition of bit patterns.
- A synchronous line is never silent; when the modem is transmitting, data is flowing. When the physical layer indicates that the modem is active, a USART will send a steady stream of padding, either characters or bits as appropriate to the device and protocol.

8.3 Devices

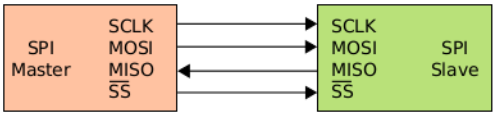
8.4 References

[1] "Intel 8251A Programmable Communications Interface," (PDF). *www.datasheetarchive.com*. Retrieved 2015-12-16.

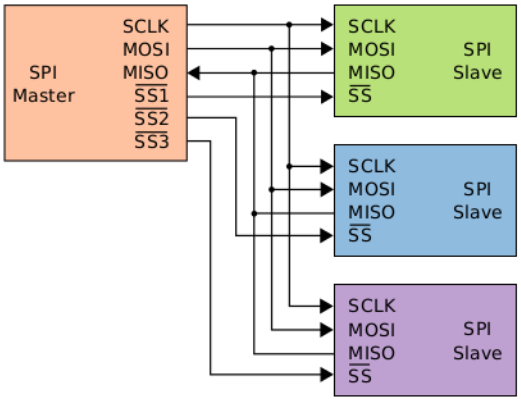
[2] "Enhanced Serial Communications Controllers,". *www.zilog.com*. Retrieved 2015-12-16.

Chapter 9

Serial Peripheral Interface Bus



SPI bus: single master and single slave



SPI bus: single master and multiple slaves

The **Serial Peripheral Interface** (**SPI**) bus is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems. The interface was developed by Motorola and has become a *de facto* standard. Typical applications include Secure Digital cards and liquid crystal displays.

SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing. Multiple slave devices are supported through selection with individual slave select (SS) lines.

Sometimes SPI is called a *four-wire* serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface,[1] but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol, but employs differential signaling and provides only a single simplex communication channel.

9.1 Interface

The SPI bus specifies four logic signals:

- SCLK : Serial Clock (output from master).
- MOSI : Master Output, Slave Input (output from master).
- MISO : Master Input, Slave Output (output from slave).
- SS : Slave Select (active low, output from master).

Alternative naming conventions are also widely used, and SPI port pin names for particular IC products may differ from those depicted in these illustrations:

Serial Clock:

- SCLK : SCK, CLK.

Master Output --> Slave Input:

- MOSI : SIMO, SDI(for slave devices), DI, DIN, SI, MTST.

Master Input <-- Slave Output:

- MISO : SOMI, SDO (for slave devices), DO, DOUT, SO, MRSR.

Slave Select:

- SS : nCS, CS, CSB, CSN, EN, nSS, STE, SYNC.

The MOSI/MISO convention requires that, on devices using the alternate names, SDI on the master be connected to SDO on the slave, and vice versa. Chip select polarity is rarely active high, although some notations (such as SS or CS instead of nSS or nCS) suggest otherwise. Slave select is used instead of an addressing concept.

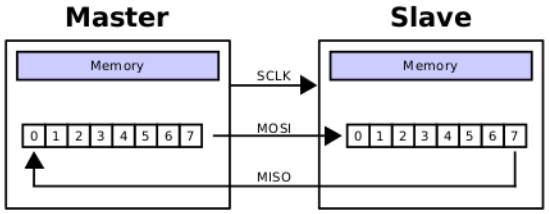
9.2 Operation

The SPI bus can operate with a single master device and with one or more slave devices.

If a single slave device is used, the SS pin *may* be fixed to logic low if the slave permits it. Some slaves require a falling edge of the chip select signal to initiate an action, an example is the Maxim MAX1242 ADC, which starts conversion on a high→low transition. With multiple slave devices, an independent SS signal is required from the master for each slave device.

Most slave devices have tri-state outputs so their MISO signal becomes high impedance (*logically disconnected*) when the device is not selected. Devices without tri-state outputs cannot share SPI bus segments with other devices; only one such slave could talk to the master.

9.2.1 Data transmission



A typical hardware setup using two shift registers to form an inter-chip circular buffer

To begin communication, the bus master configures the clock, using a frequency supported by the slave device, typically up to a few MHz. The master then selects the slave device with a logic level 0 on the select line. If a waiting period is required, such as for analog-to-digital conversion, the master must wait for at least that period of time before issuing clock cycles.

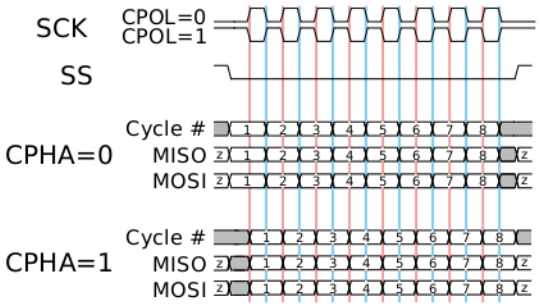
During each SPI clock cycle, a full duplex data transmission occurs. The master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it. This sequence is maintained even when only one-directional data transfer is intended.

Transmissions normally involve two shift registers of some given word size, such as eight bits, one in the master and one in the slave; they are connected in a virtual ring topology. Data is usually shifted out with the most-significant bit first, while shifting a new less-significant bit into the same register. At the same time, Data from the counterpart is shifted into the least-significant bit register. After the register bits have been shifted out and in, the master and slave have exchanged register values. If more data needs to be exchanged, the shift registers are reloaded and the process repeats. Transmission may continue for any number of clock cycles. When complete, the master stops toggling the clock signal, and typically deselects the slave.

Transmissions often consist of 8-bit words. However, other word sizes are also common, for example, 16-bit words for touchscreen controllers or audio codecs, such as the TSC2101 by Texas Instruments, or 12-bit words for many digital-to-analog or analog-to-digital converters.

Every slave on the bus that has not been activated using its chip select line must disregard the input clock and MOSI signals, and must not drive MISO.

9.2.2 Clock polarity and phase



A timing diagram showing clock polarity and phase. The red vertical line represents CPHA=0 and the blue vertical line represents CPHA=1

In addition to setting the clock frequency, the master must also configure the clock polarity and phase with respect to the data. Freescale's SPI Block Guide[2] names these two options as CPOL and CPHA respectively, and most vendors have adopted that convention.

The timing diagram is shown to the right. The timing is further described below and applies to both the master and the slave device.

- At CPOL=0 the base value of the clock is zero,i.e. the active state is 1 and idle state is 0.
  - For CPHA=0, data are captured on the clock's rising edge (low→high transition) and data is output on a falling edge (high→low clock transition).
  - For CPHA=1, data are captured on the clock's falling edge and data is output on a rising edge.
- At CPOL=1 the base value of the clock is one (inversion of CPOL=0), i.e. the active state is 0 and idle state is 1.
  - For CPHA=0, data are captured on clock's falling edge and data is output on a rising edge.
  - For CPHA=1, data are captured on clock's rising edge and data is output on a falling edge.

That is, CPHA=0 means sampling on the first clock edge, while CPHA=1 means sampling on the second clock

edge, regardless of whether that clock edge is rising or falling. Note that with CPHA=0, the data must be stable for a half cycle before the first clock cycle.

In other words, CPHA=0 means transmitting data on the active to idle state and CPHA=1 means that data is transmitted on the idle to active state edge. Note that if transmission happens on a particular edge, then capturing will happen on the opposite edge(i.e. if transmission happens on falling, then reception happens on rising and vice versa). The MOSI and MISO signals are usually stable (at their reception points) for the half cycle until the next clock transition. SPI master and slave devices may well sample data at different points in that half cycle.

This adds more flexibility to the communication channel between the master and slave.

9.2.3 Mode numbers

The combinations of polarity and phases are often referred to as modes which are commonly numbered according to the following convention, with CPOL as the high order bit and CPHA as the low order bit:

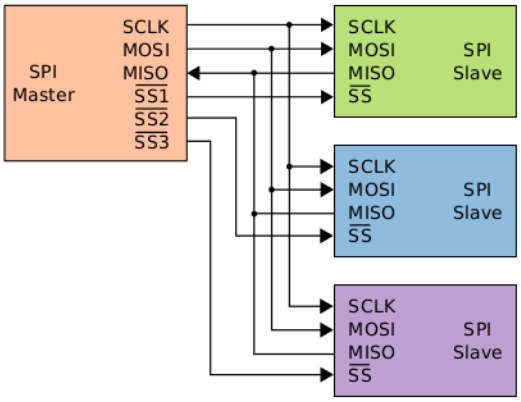
For “Microchip PIC” / “ARM-based” microcontrollers (note that NCPHA is the inversion of CPHA):

For PIC32MX : SPI mode configure CKP,CKE and SMP bits.Set SMP bit,and CKP,CKE two bits configured as above table.

For other microcontrollers:

Another commonly used notation represents the mode as a (CPOL, CPHA) tuple; e.g., the value '(0, 1)' would indicate CPOL=0 and CPHA=1.

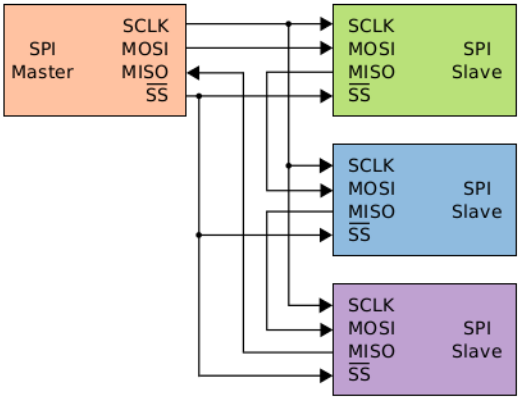
9.2.4 Independent slave configuration



Typical SPI bus: master and three independent slaves

In the independent slave configuration, there is an independent chip select line for each slave. A pull-up resistor between power source and chip select line is highly recommended for each independent device to reduce crosstalk between devices.[3] This is the way SPI is normally used. Since the MISO pins of the slaves are connected together, they are required to be tri-state pins (high, low or high-impedance).

9.2.5 Daisy chain configuration



Daisy-chained SPI bus: master and cooperative slaves

Some products that implement SPI may be connected in a daisy chain configuration, the first slave output being connected to the second slave input, etc. The SPI port of each slave is designed to send out during the second group of clock pulses an exact copy of the data it received during the first group of clock pulses. The whole chain acts as a communication shift register; daisy chaining is often done with shift registers to provide a bank of inputs or outputs through SPI. Such a feature only requires a single SS line from the master, rather than a separate SS line for each slave.[4]

Applications that require a daisy chain configuration include SGPIO and JTAG.

9.2.6 Valid communications

Some slave devices are designed to ignore any SPI communications in which the number of clock pulses is greater than specified. Others do not care, ignoring extra inputs and continuing to shift the same output bit. It is common for different devices to use SPI communications with different lengths, as, for example, when SPI is used to access the scan chain of a digital IC by issuing a command word of one size (perhaps 32 bits) and then getting a response of a different size (perhaps 153 bits, one for each pin in that scan chain).

9.2.7 Interrupts

SPI devices sometimes use another signal line to send an interrupt signal to a host CPU. Examples include pen-down interrupts from touchscreen sensors, thermal limit alerts from temperature sensors, alarms issued by real time clock chips, SDIO,[5] and headset jack insertions from the sound codec in a cell phone. Interrupts are not covered by the SPI standard; their usage is neither forbidden nor specified by the standard.

9.2.8 Example of bit-banging the master protocol

Below is an example of bit-banging the SPI protocol as an SPI master with CPOL=0, CPHA=0, and eight bits per transfer. The example is written in the C programming language. Because this is CPOL=0 the clock must be pulled low before the chip select is activated. The chip select line must be activated, which normally means being toggled low, for the peripheral before the start of the transfer, and then deactivated afterwards. Most peripherals allow or require several transfers while the select line is low; this routine might be called several times before deselecting the chip.

/* * Simultaneously transmit and receive a byte on the SPI. * * Polarity and phase are assumed to be both 0, i.e.: * - input data is captured on rising edge of SCLK. * - output data is propagated on falling edge of SCLK. * * Returns the received byte. */ uint8_t SPI_transfer_byte(uint8_t byte_out) { uint8_t byte_in = 0; uint8_t bit; for (bit = 0x80; bit; bit >>= 1) { /* Shift-out a bit to the MOSI line */ write_MOSI((byte_out & bit) ? HIGH : LOW); /* Delay for at least the peer's setup time */ delay(SPI_SCLK_LOW_TIME); /* Pull the clock line high */ write_SCLK(HIGH); /* Shift-in a bit from the MISO line */ if (read_MISO() == HIGH) byte_in |= bit; /* Delay for at least the peer's hold time */ delay(SPI_SCLK_HIGH_TIME); /* Pull the clock line low */ write_SCLK(LOW); } return byte_in; }

9.3 Pros and cons

9.3.1 Advantages

- Full duplex communication in the default version of this protocol.
- Push-pull drivers (as opposed to open drain) provide good signal integrity and high speed
- Higher throughput than I²C or SMBus
- Complete protocol flexibility for the bits transferred
  - Not limited to 8-bit words
  - Arbitrary choice of message size, content, and purpose
- Extremely simple hardware interfacing
  - Typically lower power requirements than I²C or SMBus due to less circuitry (including pull up resistors)
  - No arbitration or associated failure modes
  - Slaves use the master's clock, and do not need precision oscillators
  - Slaves do not need a unique address — unlike I²C or GPIB or SCSI
  - Transceivers are not needed
- Uses only four pins on IC packages, and wires in board layouts or connectors, much fewer than parallel interfaces
- At most one unique bus signal per device (chip select); all others are shared
- Signals are unidirectional allowing for easy Galvanic isolation
- Not limited to any maximum clock speed, enabling potentially high speed

9.3.2 Disadvantages

- Requires more pins on IC packages than I²C, even in the *three-wire* variant
- No in-band addressing; out-of-band chip select signals are required on shared buses
- No hardware flow control by the slave (but the master can delay the next clock edge to slow the transfer rate)
- No hardware slave acknowledgment (the master could be transmitting to nowhere and not know it)
- Typically supports only one master device (depends on device's hardware implementation)
- No error-checking protocol is defined
- Without a formal standard, validating conformance is not possible
- Only handles short distances compared to RS-232, RS-485, or CAN-bus
- Many existing variations, making it difficult to find development tools like host adapters that support those variations
- SPI does not support hot swapping (dynamically adding nodes).

- Interrupts must either be implemented with out-of-band signals or be faked by using periodic polling similarly to USB 1.1 and 2.0
- Some variants like Multi I/O SPI and three-wire serial buses defined below are half-duplex.

9.4 Applications

The board real estate savings compared to a parallel I/O bus are significant, and have earned SPI a solid role in embedded systems. That is true for most system-on-a-chip processors, both with higher end 32-bit processors such as those using ARM, MIPS, or PowerPC and with other microcontrollers such as the AVR, PIC, and MSP430. These chips usually include SPI controllers capable of running in either master or slave mode. In-system programmable AVR controllers (including blank ones) can be programmed using an SPI interface.[6]

Chip or FPGA based designs sometimes use SPI to communicate between internal components; on-chip real estate can be as costly as its on-board cousin.

The full-duplex capability makes SPI very simple and efficient for single master/single slave applications. Some devices use the full-duplex mode to implement an efficient, swift data stream for applications such as digital audio, digital signal processing, or telecommunications channels, but most off-the-shelf chips stick to half-duplex request/response protocols.

SPI is used to talk to a variety of peripherals, such as

- Sensors: temperature, pressure, ADC, touchscreens, video game controllers
- Control devices: audio codecs, digital potentiometers, DAC
- Camera lenses: Canon EF lens mount
- Communications: Ethernet, USB, USART, CAN, IEEE 802.15.4, IEEE 802.11, handheld video games
- Memory: flash and EEPROM
- Real-time clocks
- LCD, sometimes even for managing image data
- Any MMC or SD card (including SDIO variant[5])

For high performance systems, FPGAs sometimes use SPI to interface as a slave to a host, as a master to sensors, or for flash memory used to bootstrap if they are SRAM-based.

Although there are some similarities between the SPI bus and the JTAG (IEEE 1149.1-2013) protocol, They are not interchangeable. The SPI bus is intended for high speed, on board initialization of device peripherals, while the JTAG protocol is intended to provide reliable test access to the I/O pins from an off board controller with less precise signal delay and skew parameters. While not strictly a level sensitive interface, the JTAG protocol supports the recovery of both setup and hold violations between JTAG devices by reducing the clock rate or changing the clock's duty cycles. Consequently, the JTAG interface is not intended to support extremely high data rates.[7]

SGPIO is essentially another (incompatible) application stack for SPI designed for particular backplane management activities. SGPIO uses 3-bit messages.

9.5 Standards

The SPI bus is a *de facto* standard. However, the lack of a formal standard is reflected in a wide variety of protocol options. Different word sizes are common. Every device defines its own protocol, including whether it supports commands at all. Some devices are transmit-only; others are receive-only. Chip selects are sometimes active-high rather than active-low. Some protocols send the least significant bit first.

Some devices even have minor variances from the CPOL/CPHA modes described above. Sending data from slave to master may use the opposite clock edge as master to slave. Devices often require extra clock idle time before the first clock or after the last one, or between a command and its response. Some devices have two clocks, one to read data, and another to transmit it into the device. Many of the read clocks run from the chip select line.

Some devices require an additional flow control signal from slave to master, indicating when data are ready. This leads to a 5-wire protocol instead of the usual 4. Such a *ready* or *enable* signal is often active-low, and needs to be enabled at key points such as after commands or between words. Without such a signal, data transfer rates may need to be slowed down significantly, or protocols may need to have dummy bytes inserted, to accommodate the worst case for the slave response time. Examples include initiating an ADC conversion, addressing the right page of flash memory, and processing enough of a command that device firmware can load the first word of the response. (Many SPI masters do not support that signal directly, and instead rely on fixed delays.)

Many SPI chips only support messages that are multiples of 8 bits. Such chips can not interoperate with the JTAG or SGPIO protocols, or any other protocol that requires messages that are not multiples of 8 bits.

There are also hardware-level differences. Some chips combine MOSI and MISO into a single data line (SI/SO); this is sometimes called 'three-wire' signaling (in contrast

to normal 'four-wire' SPI). Another variation of SPI removes the chip select line, managing protocol state machine entry/exit using other methods. Anyone needing an external connector for SPI defines their own: UEXT, JTAG connector, Secure Digital card socket, etc. Signal levels depend entirely on the chips involved.

SafeSPI is an industry standard for SPI in automotive applications. Its main focus is the transmission of sensor data between different devices.

9.6 Development tools

When developing or troubleshooting systems using SPI, visibility at the level of hardware signals can be important.

9.6.1 Host adapters

There are a number of USB hardware solutions to provide computers, running Linux, Mac, or Windows, SPI master and/or slave capabilities. Many of them also provide scripting and/or programming capabilities (Visual Basic, C/C++, VHDL etc.).

An SPI host adapter lets the user play the role of a master on an SPI bus directly from PC. They are used for embedded systems, chips (FPGA/ASIC/SoC) and peripheral testing, programming and debugging.

The key parameters of SPI adapters are: the maximum supported frequency for the serial interface, command-to-command latency and the maximum length for SPI commands. It is possible to find SPI adapters on the market today that support up to 100 MHz serial interfaces, with virtually unlimited access length.

SPI protocol being a de facto standard, some SPI host adapters also have the ability of supporting other protocols beyond the traditional 4-wires SPI (for example, support of quad-SPI protocol or other custom serial protocol that derive from SPI[8]).

Examples of SPI adapters *(manufacturers in alphabetical order):*

9.6.2 Protocol analyzers

SPI protocol analyzers are tools which sample an SPI bus and decode the electrical signals to provide a higher-level view of the data being transmitted on a specific bus.

Examples of SPI protocol analyzers *(manufacturers in alphabetical order):*

9.6.3 Oscilloscopes

Every major oscilloscope vendor offers oscilloscope-based triggering and protocol decoding for SPI. Most support 2-, 3-, and 4-wire SPI. The triggering and decoding capability is typically offered as an optional extra. SPI signals can be accessed via analog oscilloscope channels or with digital MSO channels.[9]

9.6.4 Logic analyzers

When developing and/or troubleshooting the SPI bus, examination of hardware signals can be very important. Logic analyzers are tools which collect, analyze, decode, and store signals so people can view the high-speed waveforms at their leisure. Logic analyzers display time-stamps of each signal level change, which can help find protocol problems. Most logic analyzers have the capability to decode bus signals into high-level protocol data and show ASCII data.

9.7 Related terms

9.7.1 Intelligent SPI controllers

A **queued serial peripheral interface** (**QSPI**) is a type of SPI controller that uses a data queue to transfer data across the SPI bus.[10] It has a wrap-around mode allowing continuous transfers to and from the queue with only intermittent attention from the CPU. Consequently, the peripherals appear to the CPU as memory-mapped parallel devices. This feature is useful in applications such as control of an A/D converter. Other programmable features in QSPI are chip selects and transfer length/delay.

SPI controllers from different vendors support different feature sets; such DMA queues are not uncommon, although they may be associated with separate DMA engines rather than the SPI controller itself, such as used by **multichannel buffered serial port** (**MCBSP**).[11] Most SPI master controllers integrate support for up to four chip selects,[12] although some require chip selects to be managed separately through GPIO lines.

9.7.2 Microwire

Microwire,[13] often spelled **μWire**, is essentially a predecessor of SPI and a trademark of National Semiconductor. It's a strict subset of SPI: half-duplex, and using SPI mode 0. Microwire chips tend to need slower clock rates than newer SPI versions; perhaps 2 MHz vs. 20 MHz. Some Microwire chips also support a three-wire mode, which fits neatly with the restriction to half-duplex.

9.7.3 Microwire/Plus

Microwire/Plus[14] is an enhancement of Microwire and features full-duplex communication and support for SPI

modes 0 and 1. There was no specified improvement in serial clock speed.

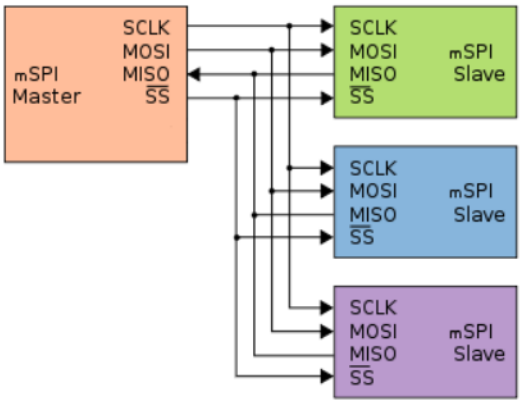
9.7.4 Three-wire serial buses

As mentioned, one variant of SPI uses single bidirectional data line (slave out/slave in, called SISO) instead of two unidirectional ones (MOSI and MISO). This variant is restricted to a half duplex mode. It tends to be used for lower performance parts, such as small EEPROMs used only during system startup and certain sensors, and Microwire. Few SPI master controllers support this mode; although it can often be easily bit-banged in software.

9.7.5 Multi I/O SPI

As opposed to three-wire serial buses, multi I/O SPI uses multiple parallel data lines (e.g., IO0 to IO3) to increase throughput. Dual I/O SPI using two data lines has comparable throughput to fast single I/O (MISO/MOSI). Quad I/O SPI using four data lines has approximately double the throughput.[15] Multi I/O SPI devices tend to be half duplex similar to three-wire devices to avoid adding too many pins. These serial memory devices combine the advantage of more speed with reduced pin count as compared to parallel memory.

9.7.6 mSPI



Typical mSPI bus: master and three independent slaves

mSPI (mini-SPI) is a modification initially developed by Dimitech for their programmable modules. Unlike the standard SPI, four signal lines are always required no matter of the number of slave devices. Its overall simplicity allows the use of standard SPI controllers with a very thin software layer.

All slave devices share the same SS (Slave Select; active low) line, along with the other three SPI signals: SCLK, MOSI and MISO. Additionally all slave devices normally have their MISO line disconnected from the bus in a high impedance state. As in the standard SPI, begin of transmission is marked by the activation of the SS line low and the end is marked by its return to high. mSPI requires the bus master to issue a "slave address" (typically 8 bits) as mandatory first word in every transmission. Since all slave devices share the same SS line, the address word will be received by all of them at the same time. From that point further, only the device with the specified address will connect its MISO line to the bus and start communicating, while all other slave devices will ignore any data and wait for a new start of transmission and address. mSPI solves some of the basic disadvantages of the standard SPI at the expense of a slight decrease in the overall communication speed due to the initial addressing.

9.7.7 Intel Enhanced Serial Peripheral Interface Bus

Intel is currently developing a successor to its Low Pin Count (LPC) bus that it calls the Enhanced Serial Peripheral Interface Bus, or eSPI for short. Intel aims to allow the reduction in the number of pins required on motherboards compared to systems using LPC, have more available throughput than LPC, reduce the working voltage to 1.8 volts to facilitate smaller chip manufacturing processes, allow eSPI peripherals to share SPI flash devices with the host (the LPC bus did not allow firmware hubs to be used by LPC peripherals), tunnel previous out-of-band pins through the eSPI bus, and allow system designers to trade off cost and performance.[16]

The eSPI bus can either be shared with SPI devices to save pins or be separate from the SPI bus to allow more performance, especially when eSPI devices need to use SPI flash devices.[16]

This proposed standard defines an Alert# signal that is used by an eSPI slave to request service from the master. In a performance-oriented design or a design with only one eSPI slave, each eSPI slave will have its Alert# pin connected to an Alert# pin on the eSPI master that is dedicated to each slave, allowing the eSPI master to grant low-latency service because the eSPI master will know which eSPI slave needs service and will not need to poll all of the slaves to determine which device needs service. In a budget design with more than one eSPI slave, all of the Alert# pins of the slaves are connected to one Alert# pin on the eSPI master in a wired-OR connection, which will require the master to poll all the slaves to determine which ones need service when the Alert# signal is pulled low by one or more peripherals that need service. Only after all of the devices are serviced will the Alert# signal be pulled high due to none of the eSPI slaves needing service and therefore pulling the Alert# signal low.[16]

This proposed standard allows designers to use 1-bit, 2-bit, or 4-bit communications at speeds from 20 to 66 MHz to further allow designers to trade off performance and

cost.[16]

All communications that were out-of-band of the LPC bus like general-purpose input/output (GPIO) and System Management Bus (SMBus) are tunneled through the eSPI bus via virtual wire cycles and out-of-band message cycles respectively in order to remove those pins from motherboard designs using eSPI.[16]

This proposed standard will support standard memory cycles with lengths of 1 byte to 4 kibibytes of data, short memory cycles with lengths of 1, 2, or 4 bytes that have much less overhead compared to standard memory cycles, and I/O cycles with lengths of 1, 2, or 4 bytes of data which are low overhead as well. This significantly reduces overhead compared to the LPC bus, whose throughput is nearly totally dominated by overhead. The standard memory cycle allows a length of anywhere from 1 byte to 4 kibibytes in order to allow its overhead to be amortized over a large transaction. eSPI slaves are allowed to initiate bus master versions of all of the memory cycles. Bus master I/O cycles, which were introduced by the LPC bus specification, and ISA-style DMA including the 32-bit variant introduced by the LPC bus specification, are not present in eSPI. Therefore, bus master memory cycles are the only allowed DMA in this standard.[16]

eSPI slaves are allowed to use the eSPI master as a proxy to perform flash operations on a standard SPI flash memory slave on behalf of the requesting eSPI slave.[16]

64-bit memory addressing is also added, but is only permitted when there is no equivalent 32-bit address.[16]

9.8 See also

- List of network buses
- UEXT Connector.
- Microsecond Bus.

9.9 References

[1] "What is Serial Synchronous Interface (SSI)?". Retrieved 2015-01-28.

[2] SPI Block Guide V03.06, Freescale Semiconductor

[3] "Better SPI Bus Design in 3 Steps". *dorkbot pdx*. Retrieved 3 September 2015.

[4] Maxim-IC application note 3947: "Daisy-Chaining SPI Devices"

[5] Not to be confused with the SDIO line of the half duplex implementation of the SPI bus, sometimes also called "3-wire SPI-bus". Here e.g. MOSI (via a resistor) and MISO (no resistor) of a master is connected to the SDIO line of a slave.

[6] AVR910 - In-system programming

[7] IEEE 1149.1-2013

[8] SPI Adapter with support of custom serial protocols, Byte Paradigm.

[9] "N5391B".

[10] Queued Serial Module Reference Manual, Freescale Semiconductor

[11] Such as with the MultiChannel Serial Port Interface, or McSPI, used in Texas Instruments OMAP chips.

[12] Such as the SPI controller on Atmel AT91 chips like the at91sam9G20, which is much simpler than TI's McSPI.

[13] MICROWIRE Serial Interface National Semiconductor Application Note AN-452

[14] MICROWIRE/PLUS Serial Interface for COP800 Family National Semiconductor Application Note AN-579

[15] Serial Peripheral Interface (SPI) Flash Memory Backgrounder, Spansion

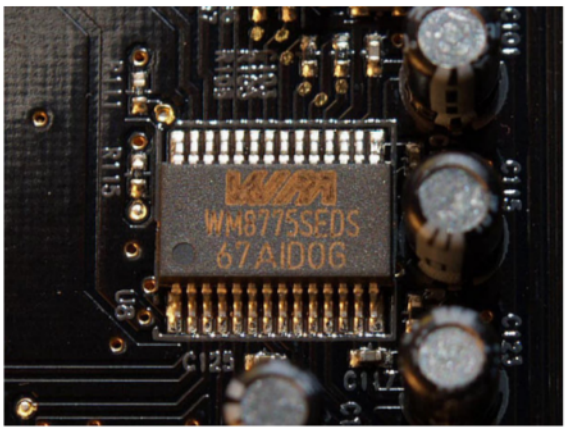
[16] https://downloadcenter.intel.com/Detail_Desc.aspx?lang=eng&changeLang=true&DwnldID=22112

9.10 External links

- Intel eSPI (Enhanced Serial Peripheral Interface)
- Introduction to SPI and I2C protocols
- Serial buses information page
- SPI Introduction
- SPI Tutorial

Chapter 10

Analog-to-digital converter



4-channel stereo multiplexed analog-to-digital converter WM8775SEDS made by Wolfson Microelectronics placed on an X-Fi Fatal1ty Pro sound card.

An **analog-to-digital converter** (**ADC**, **A/D**, or **A to D**) is a device that converts a continuous physical quantity (usually voltage) to a digital number that represents the quantity's amplitude.

The conversion involves quantization of the input, so it necessarily introduces a small amount of error. Furthermore, instead of continuously performing the conversion, an ADC does the conversion periodically, sampling the input. The result is a sequence of digital values that have been converted from a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal.

An ADC is defined by its bandwidth (the range of frequencies it can measure) and its signal to noise ratio (how accurately it can measure a signal relative to the noise it introduces). The actual bandwidth of an ADC is characterized primarily by its sampling rate, and to a lesser extent by how it handles errors such as aliasing. The dynamic range of an ADC is influenced by many factors, including the resolution (the number of output levels it can quantize a signal to), linearity and accuracy (how well the quantization levels match the true analog signal) and jitter (small timing errors that introduce additional noise). The dynamic range of an ADC is often summarized in terms of its effective number of bits (ENOB), the number of bits of each measure it returns that are on average not noise. An ideal ADC has an ENOB equal to its resolution. ADCs are chosen to match the bandwidth and required signal to noise ratio of the signal to be quantized. If an ADC operates at a sampling rate greater than twice the bandwidth of the signal, then perfect reconstruction is possible given an ideal ADC and neglecting quantization error. The presence of quantization error limits the dynamic range of even an ideal ADC, however, if the dynamic range of the ADC exceeds that of the input signal, its effects may be neglected resulting in an essentially perfect digital representation of the input signal.

An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number proportional to the magnitude of the voltage or current. However, some non-electronic or only partially electronic devices, such as rotary encoders, can also be considered ADCs. The digital output may use different coding schemes. Typically the digital output will be a two's complement binary number that is proportional to the input, but there are other possibilities. An encoder, for example, might output a Gray code.

The inverse operation is performed by a digital-to-analog converter (DAC).

10.1 Concepts

10.1.1 Resolution

The resolution of the converter indicates the number of discrete values it can produce over the range of analog values. The resolution determines the magnitude of the quantization error and therefore determines the maximum possible average signal to noise ratio for an ideal ADC without the use of oversampling. The values are usually stored electronically in binary form, so the resolution is usually expressed in bits. In consequence, the number of discrete values available, or "levels", is assumed to be a power of two. For example, an ADC with a resolution of 8 bits can encode an analog input to one in 256 different levels, since $2^8 = 256$. The values can represent the ranges from 0 to 255 (i.e. unsigned integer) or

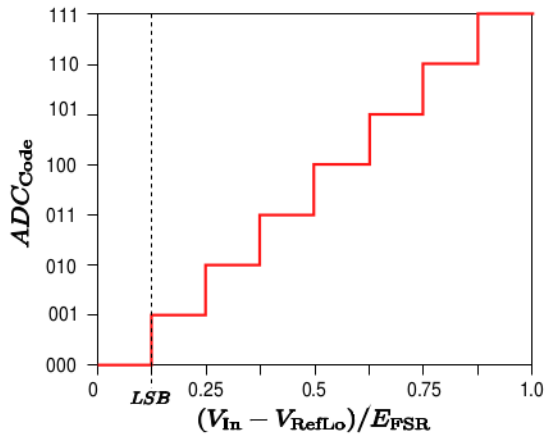


Fig. 1. An 8-level ADC coding scheme.

from −128 to 127 (i.e. signed integer), depending on the application.

Resolution can also be defined electrically, and expressed in volts. The minimum change in voltage required to guarantee a change in the output code level is called the least significant bit (LSB) voltage. The resolution Q of the ADC is equal to the LSB voltage. The voltage resolution of an ADC is equal to its overall voltage measurement range divided by the number of discrete values:

$$Q = \frac{E_{\text{FSR}}}{2^M},$$

where M is the ADC's resolution in bits and EFSR is the full scale voltage range (also called 'span'). EFSR is given by

$$E_{\text{FSR}} = V_{\text{RefHi}} - V_{\text{RefLow}},$$

where $V\text{R}_{\text{ef}}\text{H}_{\text{i}}$ and $V\text{R}_{\text{ef}}\text{L}_{\text{ow}}$ are the upper and lower extremes, respectively, of the voltages that can be coded.

Normally, the number of voltage intervals is given by

$$N = 2^M,$$

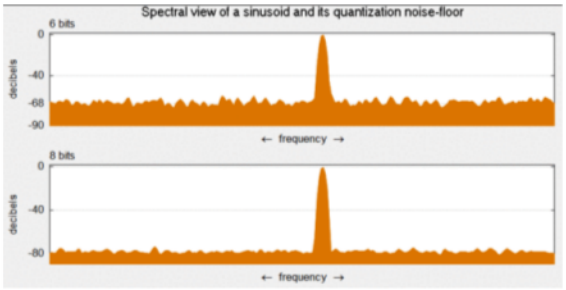
where M is the ADC's resolution in bits.[1]

That is, one voltage interval is assigned in between two consecutive code levels.

Example:

- Coding scheme as in figure 1 (assume input signal x(t) = Acos(t), A = 5V)
- Full scale measurement range = −5 to 5 volts
- ADC resolution is 8 bits: 2^8 = 256 quantization levels (codes)
- ADC voltage resolution, Q = (10 V − 0 V) / 256 = 10 V / 256 ≈ 0.039 V ≈ 39 mV.

In practice, the useful resolution of a converter is limited by the best signal-to-noise ratio (SNR) that can be achieved for a digitized signal. An ADC can resolve a signal to only a certain number of bits of resolution, called the effective number of bits (ENOB). One effective bit of resolution changes the signal-to-noise ratio of the digitized signal by 6 dB, if the resolution is limited by the ADC. If a preamplifier has been used prior to A/D conversion, the noise introduced by the amplifier can be an important contributing factor towards the overall SNR.



Comparison of quantizing a sinusoid to 64 levels (6 bits) and 256 levels (8 bits). The additive noise created by 6-bit quantization is 12 dB greater than the noise created by 8-bit quantization. When the spectral distribution is flat, as in this example, the 12 dB difference manifests as a measurable difference in the noise floors.

Quantization error

Main article: Quantization error

Quantization error is the noise introduced by quantization in an ideal ADC. It is a rounding error between the analog input voltage to the ADC and the output digitized value. The noise is non-linear and signal-dependent.

In an ideal analog-to-digital converter, where the quantization error is uniformly distributed between −1/2 LSB and +1/2 LSB, and the signal has a uniform distribution covering all quantization levels, the Signal-to-quantization-noise ratio (SQNR) can be calculated from

$$\text{SQNR} = 20\log_{10}(2^Q) \approx 6.02 \cdot Q \text{ dB}$$ [2]

Where Q is the number of quantization bits. For example, a 16-bit ADC has a maximum signal-to-noise ratio of 6.02 × 16 = 96.3 dB, and therefore the quantization error is 96.3 dB below the maximum level. Quantization error is distributed from DC to the Nyquist frequency, consequently if part of the ADC's bandwidth is not used (as in oversampling), some of the quantization error will fall out of band, effectively improving the SQNR. In an oversampled system, noise shaping can be used to further increase SQNR by forcing more quantization error out of the band.

Dither

Main article: dither

In ADCs, performance can usually be improved using dither. This is a very small amount of random noise (white noise), which is added to the input before conversion.

Its effect is to cause the state of the LSB to randomly oscillate between 0 and 1 in the presence of very low levels of input, rather than sticking at a fixed value. Rather than the signal simply getting cut off altogether at this low level (which is only being quantized to a resolution of 1 bit), it extends the effective range of signals that the ADC can convert, at the expense of a slight increase in noise – effectively the quantization error is diffused across a series of noise values which is far less objectionable than a hard cutoff. The result is an accurate representation of the signal over time. A suitable filter at the output of the system can thus recover this small signal variation.

An audio signal of very low level (with respect to the bit depth of the ADC) sampled without dither sounds extremely distorted and unpleasant. Without dither the low level may cause the least significant bit to "stick" at 0 or 1. With dithering, the true level of the audio may be calculated by averaging the actual quantized sample with a series of other samples [the dither] that are recorded over time.

A virtually identical process, also called dither or dithering, is often used when quantizing photographic images to a fewer number of bits per pixel—the image becomes noisier but to the eye looks far more realistic than the quantized image, which otherwise becomes banded. This analogous process may help to visualize the effect of dither on an analogue audio signal that is converted to digital.

Dithering is also used in integrating systems such as electricity meters. Since the values are added together, the dithering produces results that are more exact than the LSB of the analog-to-digital converter.

Note that dither can only increase the resolution of a sampler, it cannot improve the linearity, and thus accuracy does not necessarily improve.

10.1.2 Accuracy

An ADC has several sources of errors. Quantization error and (assuming the ADC is intended to be linear) non-linearity are intrinsic to any analog-to-digital conversion.

These errors are measured in a unit called the least significant bit (LSB). In the above example of an eight-bit ADC, an error of one LSB is 1/256 of the full signal range, or about 0.4%.

Non-linearity

All ADCs suffer from non-linearity errors caused by their physical imperfections, causing their output to deviate from a linear function (or some other function, in the case of a deliberately non-linear ADC) of their input. These errors can sometimes be mitigated by calibration, or prevented by testing.

Important parameters for linearity are integral non-linearity (INL) and differential non-linearity (DNL). These non-linearities reduce the dynamic range of the signals that can be digitized by the ADC, also reducing the effective resolution of the ADC.

10.1.3 Jitter

When digitizing a sine wave $x(t) = A\sin(2\pi f_0 t)$, the use of a non-ideal sampling clock will result in some uncertainty in when samples are recorded. Provided that the actual sampling time *uncertainty* due to the *clock jitter* is Δt , the error caused by this phenomenon can be estimated as $E_{ap} \leq |x'(t)\Delta t| \leq 2A\pi f_0 \Delta t$. This will result in additional recorded noise that will reduce the effective number of bits (ENOB) below that predicted by quantization error alone.

The error is zero for DC, small at low frequencies, but significant when high frequencies have high amplitudes. This effect can be ignored if it is drowned out by the *quantizing error*. Jitter requirements can be calculated using the following formula: $\Delta t < \frac{1}{2^q \pi f_0}$, where q is the number of ADC bits.

Clock jitter is caused by phase noise.[3][4] The resolution of ADCs with a digitization bandwidth between 1 MHz and 1 GHz is limited by jitter.[5]

When sampling audio signals at 44.1 kHz, the anti-aliasing filter should have eliminated all frequencies above 22 kHz. The input frequency (in this case, < 22 kHz), not the ADC clock frequency, is the determining factor with respect to jitter performance.[6]

10.1.4 Sampling rate

Main article: Sampling rate
See also: Sampling (signal processing)

The analog signal is continuous in time and it is necessary to convert this to a flow of digital values. It is therefore required to define the rate at which new digital values are sampled from the analog signal. The rate of new values is called the *sampling rate* or *sampling frequency* of the converter.

A continuously varying bandlimited signal can be sampled (that is, the signal values at intervals of time T, the sampling time, are measured and stored) and then the original signal can be *exactly* reproduced from the discrete-time values by an interpolation formula. The accuracy is limited by quantization error. However, this faithful reproduction is only possible if the sampling rate is higher than twice the highest frequency of the signal. This is essentially what is embodied in the Shannon-Nyquist sampling theorem.

Since a practical ADC cannot make an instantaneous conversion, the input value must necessarily be held constant during the time that the converter performs a conversion (called the *conversion time*). An input circuit called a sample and hold performs this task—in most cases by using a capacitor to store the analog voltage at the input, and using an electronic switch or gate to disconnect the capacitor from the input. Many ADC integrated circuits include the sample and hold subsystem internally.

Aliasing

Main article: Aliasing
See also: Undersampling

An ADC works by sampling the value of the input at discrete intervals in time. Provided that the input is sampled above the Nyquist rate, defined as twice the highest frequency of interest, then all frequencies in the signal can be reconstructed. If frequencies above half the Nyquist rate are sampled, they are incorrectly detected as lower frequencies, a process referred to as aliasing. Aliasing occurs because instantaneously sampling a function at two or fewer times per cycle results in missed cycles, and therefore the appearance of an incorrectly lower frequency. For example, a 2 kHz sine wave being sampled at 1.5 kHz would be reconstructed as a 500 Hz sine wave.

To avoid aliasing, the input to an ADC must be low-pass filtered to remove frequencies above half the sampling rate. This filter is called an *anti-aliasing filter*, and is essential for a practical ADC system that is applied to analog signals with higher frequency content. In applications where protection against aliasing is essential, oversampling may be used to greatly reduce or even eliminate it.

Although aliasing in most systems is unwanted, it should also be noted that it can be exploited to provide simultaneous down-mixing of a band-limited high frequency signal (see undersampling and frequency mixer). The alias is effectively the lower heterodyne of the signal frequency and sampling frequency.[7]

Oversampling

Main article: Oversampling

Signals are often sampled at the minimum rate required, for economy, with the result that the quantization noise introduced is white noise spread over the whole pass band of the converter. If a signal is sampled at a rate much higher than the Nyquist rate and then digitally filtered to limit it to the signal bandwidth there are the following advantages:

- digital filters can have better properties (sharper rolloff, phase) than analogue filters, so a sharper anti-aliasing filter can be realised and then the signal can be downsampled giving a better result
- a 20-bit ADC can be made to act as a 24-bit ADC with 256× oversampling
- the signal-to-noise ratio due to quantization noise will be higher than if the whole available band had been used. With this technique, it is possible to obtain an effective resolution larger than that provided by the converter alone
- The improvement in SNR is 3 dB (equivalent to 0.5 bits) per octave of oversampling which is not sufficient for many applications. Therefore, oversampling is usually coupled with noise shaping (see sigma-delta modulators). With noise shaping, the improvement is 6L+3 dB per octave where L is the order of loop filter used for noise shaping. e.g. – a 2nd order loop filter will provide an improvement of 15 dB/octave.

Oversampling is typically used in audio frequency ADCs where the required sampling rate (typically 44.1 or 48 kHz) is very low compared to the clock speed of typical transistor circuits (>1 MHz). In this case, by using the extra bandwidth to distribute quantization error onto out of band frequencies, the accuracy of the ADC can be greatly increased at no cost. Furthermore, as any aliased signals are also typically out of band, aliasing can often be completely eliminated using very low cost filters.

10.1.5 Relative speed and precision

The speed of an ADC varies by type. The Wilkinson ADC is limited by the clock rate which is processable by current digital circuits. Currently, frequencies up to 300 MHz are possible.[8] For a successive-approximation ADC, the conversion time scales with the logarithm of the resolution, e.g. the number of bits. Thus for high resolution, it is possible that the successive-approximation ADC is faster than the Wilkinson. However, the time consuming steps in the Wilkinson are digital, while those

in the successive-approximation are analog. Since analog is inherently slower than digital, as the resolution increases, the time required also increases. Thus there are competing processes at work. Flash ADCs are certainly the fastest type of the three. The conversion is basically performed in a single parallel step. For an 8-bit unit, conversion takes place in a few tens of nanoseconds.

There is, as expected, somewhat of a tradeoff between speed and precision. Flash ADCs have drifts and uncertainties associated with the comparator levels. This results in poor linearity. For successive-approximation ADCs, poor linearity is also present, but less so than for flash ADCs. Here, non-linearity arises from accumulating errors from the subtraction processes. Wilkinson ADCs have the highest linearity of the three. These have the best differential non-linearity. The other types require channel smoothing to achieve the level of the Wilkinson.[9][10]

10.1.6 The sliding scale principle

The sliding scale or randomizing method can be employed to greatly improve the linearity of any type of ADC, but especially flash and successive approximation types. For any ADC the mapping from input voltage to digital output value is not exactly a floor or ceiling function as it should be. Under normal conditions, a pulse of a particular amplitude is always converted to a digital value. The problem lies in that the ranges of analog values for the digitized values are not all of the same width, and the differential linearity decreases proportionally with the divergence from the average width. The sliding scale principle uses an averaging effect to overcome this phenomenon. A random, but known analog voltage is added to the sampled input voltage. It is then converted to digital form, and the equivalent digital amount is subtracted, thus restoring it to its original value. The advantage is that the conversion has taken place at a random point. The statistical distribution of the final levels is decided by a weighted average over a region of the range of the ADC. This in turn desensitizes it to the width of any specific level.[11][12]

10.2 ADC types

These are the most common ways of implementing an electronic ADC:

- A **direct-conversion ADC** or **flash ADC** has a bank of comparators sampling the input signal in parallel, each firing for their decoded voltage range. The comparator bank feeds a logic circuit that generates a code for each voltage range. Direct conversion is very fast, capable of gigahertz sampling rates, but usually has only 8 bits of resolution or fewer, since the number of comparators needed, $2^N - 1$, doubles with each additional bit, requiring a large, expensive circuit. ADCs of this type have a large die size, a high input capacitance, high power dissipation, and are prone to produce glitches at the output (by outputting an out-of-sequence code). Scaling to newer submicrometre technologies does not help as the device mismatch is the dominant design limitation. They are often used for video, wideband communications or other fast signals in optical storage.
- A **successive-approximation ADC** uses a comparator to successively narrow a range that contains the input voltage. At each successive step, the converter compares the input voltage to the output of an internal digital to analog converter which might represent the midpoint of a selected voltage range. At each step in this process, the approximation is stored in a successive approximation register (SAR). For example, consider an input voltage of 6.3 V and the initial range is 0 to 16 V. For the first step, the input 6.3 V is compared to 8 V (the midpoint of the 0–16 V range). The comparator reports that the input voltage is less than 8 V, so the SAR is updated to narrow the range to 0–8 V. For the second step, the input voltage is compared to 4 V (midpoint of 0–8). The comparator reports the input voltage is above 4 V, so the SAR is updated to reflect the input voltage is in the range 4–8 V. For the third step, the input voltage is compared with 6 V (halfway between 4 V and 8 V); the comparator reports the input voltage is greater than 6 volts, and search range becomes 6–8 V. The steps are continued until the desired resolution is reached.
- A **ramp-compare ADC** produces a saw-tooth signal that ramps up or down then quickly returns to zero. When the ramp starts, a timer starts counting. When the ramp voltage matches the input, a comparator fires, and the timer's value is recorded. Timed ramp converters require the least number of transistors. The ramp time is sensitive to temperature because the circuit generating the ramp is often a simple oscillator. There are two solutions: use a clocked counter driving a DAC and then use the comparator to preserve the counter's value, or calibrate the timed ramp. A special advantage of the ramp-compare system is that comparing a second signal just requires another comparator, and another register to store the voltage value. A very simple (non-linear) ramp-converter can be implemented with a microcontroller and one resistor and capacitor.[13] Vice versa, a filled capacitor can be taken from an integrator, time-to-amplitude converter, phase detector, sample and hold circuit, or peak and hold circuit and discharged. This has the advantage that a slow comparator cannot be disturbed by fast input changes.
- The **Wilkinson ADC** was designed by D. H.

Wilkinson in 1950. The Wilkinson ADC is based on the comparison of an input voltage with that produced by a charging capacitor. The capacitor is allowed to charge until its voltage is equal to the amplitude of the input pulse (a comparator determines when this condition has been reached). Then, the capacitor is allowed to discharge linearly, which produces a ramp voltage. At the point when the capacitor begins to discharge, a gate pulse is initiated. The gate pulse remains on until the capacitor is completely discharged. Thus the duration of the gate pulse is directly proportional to the amplitude of the input pulse. This gate pulse operates a linear gate which receives pulses from a high-frequency oscillator clock. While the gate is open, a discrete number of clock pulses pass through the linear gate and are counted by the address register. The time the linear gate is open is proportional to the amplitude of the input pulse, thus the number of clock pulses recorded in the address register is proportional also. Alternatively, the charging of the capacitor could be monitored, rather than the discharge.[14][15]

- An **integrating ADC** (also **dual-slope** or **multi-slope** ADC) applies the unknown input voltage to the input of an integrator and allows the voltage to ramp for a fixed time period (the run-up period). Then a known reference voltage of opposite polarity is applied to the integrator and is allowed to ramp until the integrator output returns to zero (the run-down period). The input voltage is computed as a function of the reference voltage, the constant run-up time period, and the measured run-down time period. The run-down time measurement is usually made in units of the converter's clock, so longer integration times allow for higher resolutions. Likewise, the speed of the converter can be improved by sacrificing resolution. Converters of this type (or variations on the concept) are used in most digital voltmeters for their linearity and flexibility.

- A **delta-encoded ADC** or **counter-ramp** has an up-down counter that feeds a digital to analog converter (DAC). The input signal and the DAC both go to a comparator. The comparator controls the counter. The circuit uses negative feedback from the comparator to adjust the counter until the DAC's output is close enough to the input signal. The number is read from the counter. Delta converters have very wide ranges and high resolution, but the conversion time is dependent on the input signal level, though it will always have a guaranteed worst-case. Delta converters are often very good choices to read real-world signals. Most signals from physical systems do not change abruptly. Some converters combine the delta and successive approximation approaches; this works especially well when high frequencies are known to be small in magnitude.

- A **pipeline ADC** (also called **subranging quantizer**) uses two or more steps of subranging. First, a coarse conversion is done. In a second step, the difference to the input signal is determined with a digital to analog converter (DAC). This difference is then converted finer, and the results are combined in a last step. This can be considered a refinement of the successive-approximation ADC wherein the feedback reference signal consists of the interim conversion of a whole range of bits (for example, four bits) rather than just the next-most-significant bit. By combining the merits of the successive approximation and flash ADCs this type is fast, has a high resolution, and only requires a small die size.

- A **sigma-delta ADC** (also known as a **delta-sigma ADC**) oversamples the desired signal by a large factor and filters the desired signal band. Generally, a smaller number of bits than required are converted using a Flash ADC after the filter. The resulting signal, along with the error generated by the discrete levels of the Flash, is fed back and subtracted from the input to the filter. This negative feedback has the effect of noise shaping the error due to the Flash so that it does not appear in the desired signal frequencies. A digital filter (decimation filter) follows the ADC which reduces the sampling rate, filters off unwanted noise signal and increases the resolution of the output (sigma-delta modulation, also called delta-sigma modulation).

- A **time-interleaved ADC** uses M parallel ADCs where each ADC samples data every M:th cycle of the effective sample clock. The result is that the sample rate is increased M times compared to what each individual ADC can manage. In practice, the individual differences between the M ADCs degrade the overall performance reducing the SFDR.[16] However, technologies exist to correct for these time-interleaving mismatch errors.

- An **ADC with intermediate FM stage** first uses a voltage-to-frequency converter to convert the desired signal into an oscillating signal with a frequency proportional to the voltage of the desired signal, and then uses a frequency counter to convert that frequency into a digital count proportional to the desired signal voltage. Longer integration times allow for higher resolutions. Likewise, the speed of the converter can be improved by sacrificing resolution. The two parts of the ADC may be widely separated, with the frequency signal passed through an opto-isolator or transmitted wirelessly. Some such ADCs use sine wave or square wave frequency modulation; others use pulse-frequency modulation. Such ADCs were once the most popular way to show a digital display of the status of a remote analog sensor.[17][18][19][20][21]

There can be other ADCs that use a combination of electronics and other technologies:

- A **time-stretch analog-to-digital converter (TS-ADC)** digitizes a very wide bandwidth analog signal, that cannot be digitized by a conventional electronic ADC, by time-stretching the signal prior to digitization. It commonly uses a photonic preprocessor frontend to time-stretch the signal, which effectively slows the signal down in time and compresses its bandwidth. As a result, an electronic backend ADC, that would have been too slow to capture the original signal, can now capture this slowed down signal. For continuous capture of the signal, the frontend also divides the signal into multiple segments in addition to time-stretching. Each segment is individually digitized by a separate electronic ADC. Finally, a digital signal processor rearranges the samples and removes any distortions added by the frontend to yield the binary data that is the digital representation of the original analog signal.

10.3 Commercial analog-to-digital converters

Commercial ADCs are usually implemented as integrated circuits.

Most converters sample with 6 to 24 bits of resolution, and produce fewer than 1 megasample per second. Thermal noise generated by passive components such as resistors masks the measurement when higher resolution is desired. For audio applications and in room temperatures, such noise is usually a little less than 1 μV (microvolt) of white noise. If the MSB corresponds to a standard 2 V of output signal, this translates to a noise-limited performance that is less than 20~21 bits, and obviates the need for any dithering. As of February 2002, Mega- and giga-sample per second converters are available. Megasample converters are required in digital video cameras, video capture cards, and TV tuner cards to convert full-speed analog video to digital video files.

Commercial converters usually have ±0.5 to ±1.5 LSB error in their output.

In many cases, the most expensive part of an integrated circuit is the pins, because they make the package larger, and each pin has to be connected to the integrated circuit's silicon. To save pins, it is common for slow ADCs to send their data one bit at a time over a serial interface to the computer, with the next bit coming out when a clock signal changes state, say from 0 to 5 V. This saves quite a few pins on the ADC package, and in many cases, does not make the overall design any more complex (even microprocessors which use memory-mapped I/O only need a few bits of a port to implement a serial bus to an ADC).

Commercial ADCs often have several inputs that feed the same converter, usually through an analog multiplexer. Different models of ADC may include sample and hold circuits, instrumentation amplifiers or differential inputs, where the quantity measured is the difference between two voltages.

10.4 Applications

10.4.1 Music recording

Analog-to-digital converters are integral to current music reproduction technology. People often produce music on computers using an analog recording and therefore need analog-to-digital converters to create the pulse-code modulation (PCM) data streams that go onto compact discs and digital music files.

The current crop of analog-to-digital converters utilized in music can sample at rates up to 192 kilohertz. Considerable literature exists on these matters, but commercial considerations often play a significant role. Most high-profile recording studios record in 24-bit/192-176.4 kHz pulse-code modulation (PCM) or in Direct Stream Digital (DSD) formats, and then downsample or decimate the signal for Red-Book CD production (44.1 kHz) or to 48 kHz for commonly used radio and television broadcast applications.

10.4.2 Digital signal processing

People must use ADCs to process, store, or transport virtually any analog signal in digital form. TV tuner cards, for example, use fast video analog-to-digital converters. Slow on-chip 8, 10, 12, or 16 bit analog-to-digital converters are common in microcontrollers. Digital storage oscilloscopes need very fast analog-to-digital converters, also crucial for software defined radio and their new applications.

10.4.3 Scientific instruments

Digital imaging systems commonly use analog-to-digital converters in digitizing pixels.

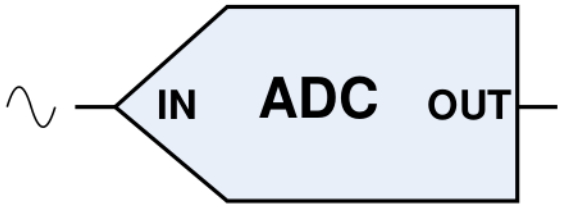
Some radar systems commonly use analog-to-digital converters to convert signal strength to digital values for subsequent signal processing. Many other in situ and remote sensing systems commonly use analogous technology.

The number of binary bits in the resulting digitized numeric values reflects the resolution, the number of unique discrete levels of quantization (signal processing). The correspondence between the analog signal and the digital signal depends on the quantization error. The quantization process must occur at an adequate speed, a constraint that may limit the resolution of the digital signal.

Many sensors produce an analog signal; temperature,

pressure, pH, light intensity etc. All these signals can be amplified and fed to an ADC to produce a digital number proportional to the input signal.

10.5 Electrical Symbol



10.6 Testing

Testing an Analog to Digital Converter requires an analog input source, hardware to send control signals and capture digital data output. Some ADCs also require an accurate source of reference signal.

The key parameters to test a SAR ADC are the following:

1. DC Offset Error
2. DC Gain Error
3. Signal to Noise Ratio (SNR)
4. Total Harmonic Distortion (THD)
5. Integral Non Linearity (INL)
6. Differential Non Linearity (DNL)
7. Spurious Free Dynamic Range
8. Power Dissipation

10.7 See also

- Audio codec
- Beta encoder
- Digitization
- Digital signal processing
- Integral linearity
- Modem

10.8 Notes

[1] "PRINCIPLES OF DATA ACQUISITION AND CONVERSION" (PDF). Burr-Brown. Retrieved 1994-05-01. Check date values in: |access-date= (help)

[2] Lathi, B.P. (1998). *Modern Digital and Analog Communication Systems (3rd edition)*. Oxford University Press.

[3] Maxim App 800: "Design a Low-Jitter Clock for High-Speed Data Converters". maxim-ic.com (July 17, 2002).

[4] "Jitter effects on Analog to Digital and Digital to Analog Converters" (PDF). Retrieved 19 August 2012.

[5] Löhning, Michael and Fettweis, Gerhard (2007). "The effects of aperture jitter and clock jitter in wideband ADCs". *Computer Standards & Interfaces archive* **29** (1): 11–18. doi:10.1016/j.csi.2005.12.005.

[6] Redmayne, Derek and Steer, Alison (8 December 2008) Understanding the effect of clock jitter on high-speed ADCs. eetimes.com

[7] "RF-Sampling and GSPS ADCs - Breakthrough ADCs Revolutionize Radio Architectures" (PDF). Texas Instruments. Retrieved 4 November 2013.

[8] 310 Msps ADC by Linear Technology, http://www.linear.com/product/LTC2158-14.

[9] Knoll (1989, pp. 664–665)

[10] Nicholson (1974, pp. 313–315)

[11] Knoll (1989, pp. 665–666)

[12] Nicholson (1974, pp. 315–316)

[13] Atmel Application Note AVR400: Low Cost A/D Converter. atmel.com

[14] Knoll (1989, pp. 663–664)

[15] Nicholson (1974, pp. 309–310)

[16] Vogel, Christian (2005). "The Impact of Combined Channel Mismatch Effects in Time-interleaved ADCs". *IEEE Transactions on Instrumentation and Measurement* **55** (1): 415–427. doi:10.1109/TIM.2004.834046.

[17] Analog Devices MT-028 Tutorial: "Voltage-to-Frequency Converters" by Walt Kester and James Bryant 2009, apparently adapted from Kester, Walter Allan (2005) *Data conversion handbook*, Newnes, p. 274, ISBN 0750678410.

[18] Microchip AN795 "Voltage to Frequency / Frequency to Voltage Converter" p. 4: "13-bit A/D converter"

[19] Carr, Joseph J. (1996) *Elements of electronic instrumentation and measurement*, Prentice Hall, p. 402, ISBN 0133416860.

[20] "Voltage-to-Frequency Analog-to-Digital Converters". globalspec.com

[21] Pease, Robert A. (1991) *Troubleshooting Analog Circuits*, Newnes, p. 130, ISBN 0750694998.

10.9 References

- Knoll, Glenn F. (1989). *Radiation Detection and Measurement* (2nd ed.). New York: John Wiley & Sons. ISBN 0471815047.
- Nicholson, P. W. (1974). *Nuclear Electronics*. New York: John Wiley & Sons. pp. 315–316. ISBN 0471636975.

10.10 Further reading

- Allen, Phillip E.; Holberg, Douglas R. *CMOS Analog Circuit Design*. ISBN 0-19-511644-5.
- Fraden, Jacob (2010). *Handbook of Modern Sensors: Physics, Designs, and Applications*. Springer. ISBN 978-1441964656.
- Kester, Walt, ed. (2005). *The Data Conversion Handbook*. Elsevier: Newnes. ISBN 0-7506-7841-0.
- Johns, David; Martin, Ken. *Analog Integrated Circuit Design*. ISBN 0-471-14448-7.
- Liu, Mingliang. *Demystifying Switched-Capacitor Circuits*. ISBN 0-7506-7907-7.
- Norsworthy, Steven R.; Schreier, Richard; Temes, Gabor C. (1997). *Delta-Sigma Data Converters*. IEEE Press. ISBN 0-7803-1045-4.
- Razavi, Behzad (1995). *Principles of Data Conversion System Design*. New York, NY: IEEE Press. ISBN 0-7803-1093-4.
- Ndjountche, Tertulien. *CMOS Analog Integrated Circuits: High-Speed and Power-Efficient Design*. Boca Raton, FL: CRC Press. ISBN 978-1-4398-5491-4.
- Staller, Len (February 24, 2005). "Understanding analog to digital converter specifications". *Embedded Systems Design*.
- Walden, R. H. (1999). "Analog-to-digital converter survey and analysis". *IEEE Journal on Selected Areas in Communications* **17** (4): 539–550. doi:10.1109/49.761034.

10.11 External links

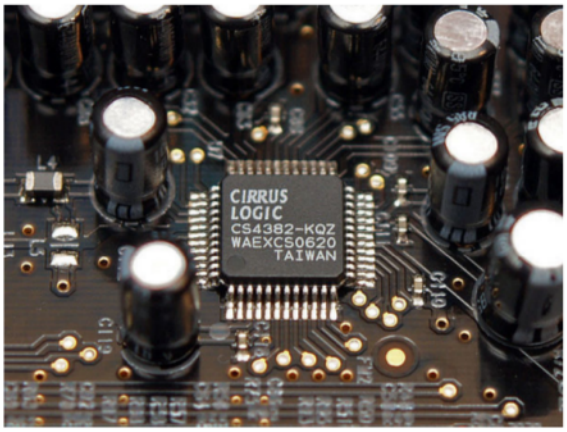
- Counting Type ADC A simple tutorial showing how to build your first ADC.
- An Introduction to Delta Sigma Converters A very nice overview of Delta-Sigma converter theory.
- Digital Dynamic Analysis of A/D Conversion Systems through Evaluation Software based on FFT/DFT Analysis RF Expo East, 1987
- Which ADC Architecture Is Right for Your Application? article by Walt Kester
- ADC and DAC Glossary Defines commonly used technical terms.
- Introduction to ADC in AVR – Analog to digital conversion with Atmel microcontrollers
- Signal processing and system aspects of time-interleaved ADCs.
- Explanation of analog-digital converters with interactive principles of operations.

Chapter 11

Digital-to-analog converter

For digital television converter boxes, see digital television adapter.

In electronics, a **digital-to-analog converter** (**DAC**, **D/A**, **D2A** or **D-to-A**) is a function that converts digital data (usually binary) into an analog signal (current, voltage, or electric charge). An analog-to-digital converter (ADC) performs the reverse function. Unlike analog signals, digital data can be transmitted, manipulated, and stored without degradation, albeit with more complex equipment. But a DAC is needed to convert the digital signal to analog to drive an earphone or loudspeaker amplifier in order to produce sound (analog air pressure waves).



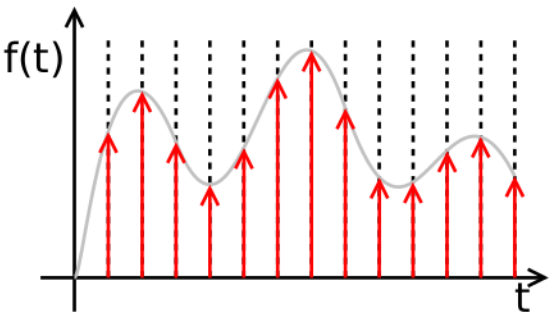
8-channel digital-to-analog converter Cirrus Logic CS4382 as used in a soundcard.

DACs and their inverse, ADCs, are part of an enabling technology that has contributed greatly to the digital revolution. To illustrate, consider a typical long-distance telephone call. The caller's voice is converted into an analog electrical signal by a microphone, then the analog signal is converted to a digital stream by an ADC. The digital stream is then divided into packets where it may be sent along with other digital data, not necessarily audio. The digital packets are then received at the destination, but each packet may take a completely different route and may not even arrive at the destination in the correct time order. The digital voice data is then extracted from the packets and assembled into a digital data stream. A DAC converts this into an analog electrical signal, which drives an audio amplifier, which in turn drives a loudspeaker, which finally produces sound.

There are several DAC architectures; the suitability of a DAC for a particular application is determined by six main parameters: physical size, power consumption, resolution, speed, accuracy, cost. Due to the complexity and the need for precisely matched components, all but the most specialist DACs are implemented as integrated circuits (ICs). Digital-to-analog conversion can degrade a signal, so a DAC should be specified that has insignificant errors in terms of the application.

DACs are commonly used in music players to convert digital data streams into analog audio signals. They are also used in televisions and mobile phones to convert digital video data into analog video signals which connect to the screen drivers to display monochrome or color images. These two applications use DACs at opposite ends of the speed/resolution trade-off. The audio DAC is a low speed high resolution type while the video DAC is a high speed low to medium resolution type. Discrete DACs would typically be extremely high speed low resolution power hungry types, as used in military radar systems. Very high speed test equipment, especially sampling oscilloscopes, may also use discrete DACs.

11.1 Overview



Ideally sampled signal.

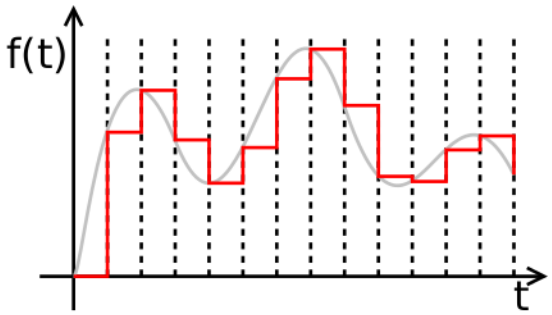
A DAC converts an abstract finite-precision number (usually a fixed-point binary number) into a physical quantity

(e.g., a voltage or a pressure). In particular, DACs are often used to convert finite-precision time series data to a continually varying physical signal.

An *ideal* DAC converts the abstract numbers into a conceptual sequence of impulses that are then processed by a reconstruction filter using some form of interpolation to fill in data between the impulses. A typical *practical* DAC converts the numbers into a piecewise constant function made up of a sequence of rectangular functions that is modeled with the zero-order hold. Other DAC methods (e.g., methods based on delta-sigma modulation) produce a pulse-density modulated signal that can then be filtered in a similar way to produce a smoothly varying signal.

As per the Nyquist–Shannon sampling theorem, a DAC can reconstruct the original signal from the sampled data provided that its bandwidth meets certain requirements (e.g., a baseband signal with bandwidth less than the Nyquist frequency). Digital sampling introduces quantization error that manifests as low-level noise added to the reconstructed signal.

11.2 Practical operation



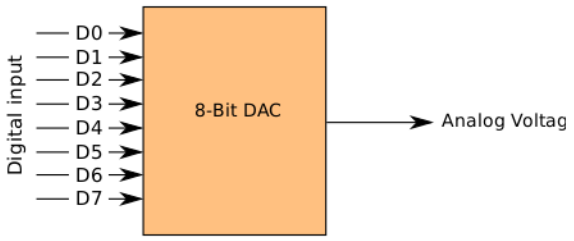
Piecewise constant output of an idealized DAC lacking a reconstruction filter. In a practical DAC, a filter or the finite bandwidth of the device smooths out the step response into a continuous curve.

Instead of impulses, they update the analog voltage at uniform sampling intervals, which are then often interpolated via a reconstruction filter to continuously varied levels.

These numbers are written to the DAC, typically with a clock signal that causes each number to be latched in sequence, at which time the DAC output voltage changes rapidly from the previous value to the value represented by the currently latched number. The effect of this is that the output voltage is *held* in time at the current value until the next input number is latched, resulting in a piecewise constant or staircase-shaped output. This is equivalent to a zero-order hold operation and has an effect on the frequency response of the reconstructed signal.

The fact that DACs output a sequence of piecewise constant values (known as zero-order hold in sample data textbooks) or rectangular pulses causes multiple harmonics above the Nyquist frequency. Usually, these are removed with a low pass filter acting as a reconstruction filter in applications that require it.

11.3 Applications



A simplified functional diagram of an 8-bit DAC

11.3.1 Audio

Most modern audio signals are stored in digital form (for example MP3s and CDs) and in order to be heard through speakers they must be converted into an analog signal. DACs are therefore found in CD players, digital music players, and PC sound cards.

Specialist standalone DACs can also be found in high-end hi-fi systems. These normally take the digital output of a compatible CD player or dedicated transport (which is basically a CD player with no internal DAC) and convert the signal into an analog line-level output that can then be fed into an amplifier to drive speakers.

Similar digital-to-analog converters can be found in digital speakers such as USB speakers, and in sound cards.

In VoIP (Voice over IP) applications, the source must first be digitized for transmission, so it undergoes conversion via an analog-to-digital converter, and is then reconstructed into analog using a DAC on the receiving party's end.

11.3.2 Video

Video sampling tends to work on a completely different scale altogether thanks to the highly nonlinear response both of cathode ray tubes (for which the vast majority of digital video foundation work was targeted) and the human eye, using a "gamma curve" to provide an appearance of evenly distributed brightness steps across the display's full dynamic range - hence the need to use RAMDACs in computer video applications with deep enough colour resolution to make engineering a hard-coded value into the DAC for each output level of each channel impractical (e.g. an Atari ST or Sega Genesis would require 24 such values; a 24-bit video card would

Top-loading CD player and external digital-to-analog converter.

need 768...). Given this inherent distortion, it is not unusual for a television or video projector to truthfully claim a linear contrast ratio (difference between darkest and brightest output levels) of 1000:1 or greater, equivalent to 10 bits of audio precision even though it may only accept signals with 8-bit precision and use an LCD panel that only represents 6 or 7 bits per channel.

Video signals from a digital source, such as a computer, must be converted to analog form if they are to be displayed on an analog monitor. As of 2007, analog inputs were more commonly used than digital, but this changed as flat panel displays with DVI and/or HDMI connections became more widespread. A video DAC is, however, incorporated in any digital video player with analog outputs. The DAC is usually integrated with some memory (RAM), which contains conversion tables for gamma correction, contrast and brightness, to make a device called a RAMDAC.

A device that is distantly related to the DAC is the digitally controlled potentiometer, used to control an analog signal digitally.

11.3.3 Mechanical

An unusual application of digital-to-analog conversion was the whiffletree electromechanical digital-to-analog converter linkage in the IBM Selectric typewriter.

11.4 DAC types

The most common types of electronic DACs are:

- The pulse-width modulator, the simplest DAC type. A stable current or voltage is switched into a low-pass analog filter with a duration determined by the digital input code. This technique is often used for electric motor speed control, but has many other applications as well.
- Oversampling DACs or interpolating DACs such as the delta-sigma DAC, use a pulse density conversion technique. The oversampling technique allows for the use of a lower resolution DAC internally. A simple 1-bit DAC is often chosen because the oversampled result is inherently linear. The DAC is driven with a pulse-density modulated signal, created with the use of a low-pass filter, step nonlinearity (the actual 1-bit DAC), and negative feedback loop, in a technique called delta-sigma modulation. This results in an effective high-pass filter acting on the quantization (signal processing) noise, thus steering this noise out of the low frequencies of interest into the megahertz frequencies of little interest, which is called noise shaping. The quantization noise at these high frequencies is removed or greatly attenuated by use of an analog low-pass filter at the output (sometimes a simple RC low-pass circuit is sufficient). Most very high resolution DACs (greater than 16 bits) are of this type due to its high linearity and low cost. Higher oversampling rates can relax the specifications of the output low-pass filter and enable further suppression of quantization noise. Speeds of greater than 100 thousand samples per second (for example, 192 kHz) and resolutions of 24 bits are attainable with delta-sigma DACs. A short comparison with pulse-width modulation shows that a 1-bit DAC with a simple first-order integrator would have to run at 3 THz (which is physically unrealizable) to achieve 24 meaningful bits of resolution, requiring a higher-order low-pass filter in the noise-shaping loop. A single integrator is a low-pass filter with a frequency response inversely proportional to frequency and using one such integrator in the noise-shaping loop is a first order delta-sigma modulator. Multiple higher order topologies (such as MASH) are used to achieve higher degrees of noise-shaping with a stable topology.
- The binary-weighted DAC, which contains individual electrical components for each bit of the DAC connected to a summing point. These precise voltages or currents sum to the correct output value. This is one of the fastest conversion methods but suffers from poor accuracy because of the high precision required for each individual voltage or current. Such high-precision components are expensive, so this type of converter is usually limited to 8-bit resolution or less.
    - Switched resistor DAC contains a parallel resistor network. Individual resistors are enabled or bypassed in the network based on the digital input.
    - Switched current source DAC, from which different current sources are selected based on the digital input.
    - Switched capacitor DAC contains a parallel capacitor network. Individual capacitors

are connected or disconnected with switches based on the input.

- The R-2R ladder DAC which is a binary-weighted DAC that uses a repeating cascaded structure of resistor values R and 2R. This improves the precision due to the relative ease of producing equal valued-matched resistors (or current sources).
- The Successive-Approximation or Cyclic DAC, which successively constructs the output during each cycle. Individual bits of the digital input are processed each cycle until the entire input is accounted for.
- The thermometer-coded DAC, which contains an equal resistor or current-source segment for each possible value of DAC output. An 8-bit thermometer DAC would have 255 segments, and a 16-bit thermometer DAC would have 65,535 segments. This is perhaps the fastest and highest precision DAC architecture but at the expense of high cost. Conversion speeds of >1 billion samples per second have been reached with this type of DAC.
- Hybrid DACs, which use a combination of the above techniques in a single converter. Most DAC integrated circuits are of this type due to the difficulty of getting low cost, high speed and high precision in one device.
    - The segmented DAC, which combines the thermometer-coded principle for the most significant bits and the binary-weighted principle for the least significant bits. In this way, a compromise is obtained between precision (by the use of the thermometer-coded principle) and number of resistors or current sources (by the use of the binary-weighted principle). The full binary-weighted design means 0% segmentation, the full thermometer-coded design means 100% segmentation.
- Most DACs, shown earlier in this list, rely on a constant reference voltage to create their output value. Alternatively, a *multiplying DAC*[1] takes a variable input voltage for their conversion. This puts additional design constraints on the bandwidth of the conversion circuit.

11.5 DAC performance

DACs are very important to system performance. The most important characteristics of these devices are:

Resolution The number of possible output levels the DAC is designed to reproduce. This is usually stated as the number of bits it uses, which is the base two logarithm of the number of levels. For instance a 1 bit DAC is designed to reproduce 2 (2^1) levels while an 8 bit DAC is designed for 256 (2^8) levels. Resolution is related to the effective number of bits which is a measurement of the actual resolution attained by the DAC. Resolution determines color depth in video applications and audio bit depth in audio applications.

Maximum sampling rate A measurement of the maximum speed at which the DACs circuitry can operate and still produce the correct output. As stated above, the Nyquist–Shannon sampling theorem defines a relationship between this and the bandwidth of the sampled signal.

Monotonicity The ability of a DAC's analog output to move only in the direction that the digital input moves (i.e., if the input increases, the output doesn't dip before asserting the correct output.) This characteristic is very important for DACs used as a low frequency signal source or as a digitally programmable trim element.

Total harmonic distortion and noise (THD+N) A measurement of the distortion and noise introduced to the signal by the DAC. It is expressed as a percentage of the total power of unwanted harmonic distortion and noise that accompany the desired signal. This is a very important DAC characteristic for dynamic and small signal DAC applications.

Dynamic range A measurement of the difference between the largest and smallest signals the DAC can reproduce expressed in decibels. This is usually related to resolution and noise floor.

Other measurements, such as phase distortion and jitter, can also be very important for some applications, some of which (e.g. wireless data transmission, composite video) may even *rely* on accurate production of phase-adjusted signals.

Linear PCM audio sampling usually works on the basis of each bit of resolution being equivalent to 6 decibels of amplitude (a 2x increase in volume or precision).

Non-linear PCM encodings (A-law / μ-law, ADPCM, NICAM) attempt to improve their effective dynamic ranges by a variety of methods - logarithmic step sizes between the output signal strengths represented by each data bit (trading greater quantisation distortion of loud signals for better performance of quiet signals)

11.6 DAC figures of merit

- Static performance:
    - Differential nonlinearity (DNL) shows how much two adjacent code analog values deviate from the ideal 1 LSB step.[2]

  - Integral nonlinearity (INL) shows how much the DAC transfer characteristic deviates from an ideal one. That is, the ideal characteristic is usually a straight line; INL shows how much the actual voltage at a given code value differs from that line, in LSBs (1 LSB steps).
  - Gain
  - Offset
  - Noise is ultimately limited by the thermal noise generated by passive components such as resistors. For audio applications and in room temperatures, such noise is usually a little less than 1 μV (microvolt) of white noise. This limits performance to less than 20~21 bits even in 24-bit DACs.
- Frequency domain performance
  - Spurious-free dynamic range (SFDR) indicates in dB the ratio between the powers of the converted main signal and the greatest undesired spur.
  - Signal-to-noise and distortion ratio (SNDR) indicates in dB the ratio between the powers of the converted main signal and the sum of the noise and the generated harmonic spurs
  - i-th harmonic distortion (HDi) indicates the power of the i-th harmonic of the converted main signal
  - Total harmonic distortion (THD) is the sum of the powers of all HDi
  - If the maximum DNL error is less than 1 LSB, then the D/A converter is guaranteed to be monotonic. However, many monotonic converters may have a maximum DNL greater than 1 LSB.
- Time domain performance:
  - Glitch impulse area (glitch energy)
  - Response uncertainty
  - Time nonlinearity (TNL)

11.7 See also

- Integral linearity
- I^2S
- Modem
- RAMDAC

11.8 References

[1] "Multiplying DACs, flexible building blocks" (PDF). Analog Devices inc. 2010. Retrieved 29 March 2012.

[2] ADC and DAC Glossary - Maxim

11.9 Further reading

- Kester, Walt, *The Data Conversion Handbook*, ISBN 0-7506-7841-0
- S. Norsworthy, Richard Schreier, Gabor C. Temes, *Delta-Sigma Data Converters*. ISBN 0-7803-1045-4.
- Mingliang Liu, *Demystifying Switched-Capacitor Circuits*. ISBN 0-7506-7907-7.
- Behzad Razavi, *Principles of Data Conversion System Design*. ISBN 0-7803-1093-4.
- Phillip E. Allen, Douglas R. Holberg, *CMOS Analog Circuit Design*. ISBN 0-19-511644-5.
- Robert F. Coughlin, Frederick F. Driscoll, *Operational Amplifiers and Linear Integrated Circuits*. ISBN 0-13-014991-8.
- A Anand Kumar, *Fundamentals of Digital Circuits*. ISBN 81-203-1745-9, ISBN 978-81-203-1745-1.
- Ndjountche Tertulien, "CMOS Analog Integrated Circuits: High-Speed and Power-Efficient Design". ISBN 978-1-4398-5491-4.

11.10 External links

- ADC and DAC Glossary

Chapter 12

Power management

For Management of energy in various contexts, see Energy management.

Power Management is a feature of some electrical appliances, especially copiers, computers, GPUs and computer peripherals such as monitors and printers, that turns off the power or switches the system to a low-power state when inactive. In computing this is known as PC power management and is built around a standard called ACPI. This supersedes APM. All recent (consumer) computers have ACPI support.

In the military, ""Power Management"" often refers to suites of equipment which permit soldiers and squads to share diverse energy sources, powering often incompatible equipment.[1]

12.1 Motivations

PC power management for computer systems is desired for many reasons, particularly:

- Reduce overall energy consumption
- Prolong battery life for portable and embedded systems
- Reduce cooling requirements
- Reduce noise
- Reduce operating costs for energy and cooling

Lower power consumption also means lower heat dissipation, which increases system stability, and less energy use, which saves money and reduces the impact on the environment.

12.2 Processor level techniques

The power management for microprocessors can be done over the whole processor,[2] or in specific components, such as cache memory[3] and main memory.[4]

With dynamic voltage scaling and dynamic frequency scaling, the CPU core voltage, clock rate, or both, can be altered to decrease power consumption at the price of potentially lower performance. This is sometimes done in real time to optimize the power-performance tradeoff.

Examples:

- AMD Cool'n'Quiet
- AMD PowerNow! [5]
- IBM EnergyScale [6]
- Intel SpeedStep
- Transmeta LongRun and LongRun2
- VIA LongHaul (PowerSaver)

Additionally, processors can selectively power off internal circuitry (power gating). For example:

- Newer Intel Core processors support ultra-fine power control over the functional units within the processors.
- AMD CoolCore technology get more efficient performance by dynamically activating or turning off parts of the processor.[7]

Intel VRT technology split the chip into a 3.3V I/O section and a 2.9V core section. The lower core voltage reduces power consumption.

12.2.1 Heterogenous computing

ARM's big.LITTLE architecture can migrate processes between faster “big” cores and more power efficient “LITTLE” cores.

12.3 Operating system level: Hibernation

Main article: Hibernation (computing)

When a computer system hibernates it saves the contents of the RAM to disk and powers down the machine. On startup it reloads the data. This allows the system to be completely powered off while in hibernate mode. This requires a file the size of the installed RAM to be placed on the hard disk, potentially using up space even when not in hibernate mode. Hibernate mode is enabled by default in some versions of Windows and can be disabled in order to recover this disk space.

12.4 Power Management in GPUs

Graphics processing unit (GPUs) are used together with a CPU to accelerate computing in variety of domains revolving around scientific, analytics, engineering, consumer and enterprise applications.[8] All of this do come with some drawbacks, the high computing capability of GPUs comes at the cost of high power dissipation. A lot of research has been done over the power dissipation issue of GPUs and a lot of different techniques have been proposed to address this issue. Dynamic voltage scaling/Dynamic frequency scaling(DVFS) and clock gating are two commonly used techniques for reducing dynamic power in GPUs.

12.4.1 DVFS Techniques

Experiments show that conventional processor DVFS policy can achieve power reduction of embedded GPUs with reasonable performance degradation.[9] New directions for designing effective DVFS schedulers for heterogeneous systems are also being explored.[10] A heterogeneous CPU-GPU architecture, GreenGPU[11] is presented which employs DVFS in a synchronized way, both for GPU and CPU. GreenGPU is implemented using the CUDA framework on a real physical testbed with Nvidia GeForce GPUs and AMD Phenom II CPUs. Experimentally it is shown that the GreenGPU achieves 21.04% average energy saving and outperforms several well-designed baselines. For the mainstream GPUs which are extensively used in all kinds of commercial and personal applications several DVFS techniques exist and are built into the GPUs alone, AMD PowerTune and AMD ZeroCore Power are the two dynamic frequency scaling technologies for AMD graphic cards. Practical tests showed that reclocking a Geforce GTX 480 can achieve a 28% lower power consumption while only decreasing performance by 1% for a given task.[12]

12.4.2 Power Gating Techniques

A lot of research has been done on the dynamic power reduction with the use of DVFS techniques. However, as technology continues to shrink, leakage power will become a dominant factor.[13] Power gating is a commonly used circuit technique to remove leakage by turning off the supply voltage of unused circuits. Power gating incurs energy overhead; therefore, unused circuits need to remain idle long enough to compensate this overheads. A novel micro-architectural technique[14] for runtime power-gating caches of GPUs saves leakage energy. Based on experiments on 16 different GPU workloads, the average energy savings achieved by the proposed technique is 54%. Shaders are the most power hungry component of a GPU, a predictive shader shut down power gating technique[15] achieves up to 46% leakage reduction on shader processors. The Predictive Shader Shutdown technique exploits workload variation across frames to eliminate leakage in shader clusters. Another technique called Deferred Geometry Pipeline seeks to minimize leakage in fixed-function geometry units by utilizing an imbalance between geometry and fragment computation across batches which removes up to 57% of the leakage in the fixed-function geometry units. A simple time-out power gating method can be applied to non-shader execution units which eliminates 83.3% of the leakage in non-shader execution units on average. All the three techniques stated above incur negligible performance degradation, less than 1%.[16]

12.5 See also

- CPU power dissipation
- Low-power electronics
- Dynamic voltage scaling
- Dynamic frequency scaling
- Advanced power management (APM)
- Advanced Configuration and Power Interface (ACPI)
  - Hibernate
  - Sleep
- BatteryMAX (idle detection)
- 80 Plus
- Energy Star
- Green computing
- pmset
- PowerTOP - diagnostic tool
- The Green Grid
- Sleep Proxy Service
- Standby power
- Thermal Design Power

- VESA Display Power Management Signaling (DPMS)
- Run-time estimation of system and sub-system level power consumption

12.6 References

[1] "Office of Naval Research - Power Management Kit". ONR. Retrieved 2015-01-15.

[2] "A Survey of Methods for Analyzing and Improving GPU Energy Efficiency", Mittal et al., ACM Computing Surveys, 2014.

[3] "A Survey of Architectural Techniques For Improving Cache Power Efficiency", S. Mittal, SUSCOM, 4(1), 33-43, 2014

[4] "A survey of architectural techniques for DRAM power management", S. Mittal, IJHPSA, 4(2), 110-119, 2012

[5] "AMD PowerNow! Technology with optimized power management". AMD. Retrieved 2009-04-23.

[6] "IBM EnergyScale for POWER6 Processor-Based Systems". IBM. Retrieved 2009-04-23.

[7] "AMD Cool'n'Quiet Technology Overview". AMD. Retrieved 2009-04-23.

[8] "What is GPU computing". Nvidia.

[9] "Dynamic voltage and frequency scaling framework for low-power embedded GPUs", Daecheol You et al., Electronics Letters (Volume:48, Issue: 21), 2012.

[10] "Effects of Dynamic Voltage and Frequency Scaling on a K20 GPU", Rong Ge et al., 42nd International Conference on Parallel Processing Pages 826-833, 2013.

[11] "GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures", Kai Ma et al., 41st International Conference on Parallel Processing Pages 48-57, 2012.

[12] "Power and performance analysis of GPU-accelerated systems", Yuki Abe et al., USENIX conference on Power-Aware Computing and Systems Pages 10-10, 2012.

[13] "Design challenges of technology scaling", Borkar, S., Micro, IEEE (Volume:19 , Issue: 4), 1999.

[14] "Run-time power-gating in caches of GPUs for leakage energy savings", Yue Wang et al., Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012

[15] "A Predictive Shutdown Technique for GPU Shader Processors", Po-Han Wang et al., Computer Architecture Letters (Volume:8 , Issue: 1), 2009

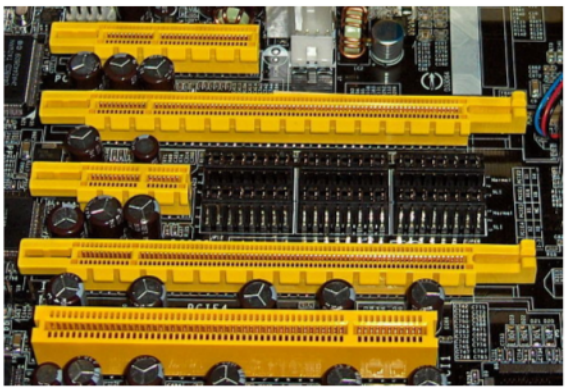
[16] "Power gating strategies on GPUs", Po-Han Wang et al., ACM Transactions on Architecture and Code Optimization (TACO) Volume 8 Issue 3, 2011

12.7 External links

- Energy Star - Independent List of Products
- Energy Star - Low Carbon IT Campaign
- Energy Consumption Calculator
- Research Bibliography on Power Management

Chapter 13

Bus (computing)



4 PCI Express bus card slots (from top to bottom: x4, x16, x1 and x16), compared to a 32-bit conventional PCI bus card slot (very bottom)

In computer architecture, a **bus** (related to the Latin "omnibus", meaning "for all") is a communication system that transfers data between components inside a computer, or between computers. This expression covers all related hardware components (wire, optical fiber, etc.) and software, including communication protocols.[1]

Early computer buses were parallel electrical wires with multiple connections, but the term is now used for any physical arrangement that provides the same logical functionality as a parallel electrical bus. Modern computer buses can use both parallel and bit serial connections, and can be wired in either a multidrop (electrical parallel) or daisy chain topology, or connected by switched hubs, as in the case of USB.

13.1 Background and nomenclature

Computer systems generally consist of three main parts: the central processing unit (CPU) that processes data, memory that holds the programs and data to be processed, and I/O (input/output) devices as peripherals that communicate with the outside world. An early computer might use a hand-wired CPU of vacuum tubes, a magnetic drum for main memory, and a punch tape and printer for reading and writing data. In a modern system we might find a multi-core CPU, DDR3 SDRAM for memory, a hard drive for secondary storage, a graphics card and LCD display as a display system, a mouse and keyboard for interaction, and a Wi-Fi connection for networking. In both examples, computer buses of one form or another move data between all of these devices.

In most traditional computer architectures, the CPU and main memory tend to be tightly coupled. A microprocessor conventionally is a single chip which has a number of electrical connections on its pins that can be used to select an "address" in the main memory and another set of pins to read and write the data stored at that location. In most cases, the CPU and memory share signalling characteristics and operate in synchrony. The bus connecting the CPU and memory is one of the defining characteristics of the system, and often referred to simply as the system bus.

It is possible to allow peripherals to communicate with memory in the same fashion, attaching adaptors in the form of expansion cards directly to the system bus. This is commonly accomplished through some sort of standardized electrical connector, several of these forming the expansion bus or local bus. However, as the performance differences between the CPU and peripherals varies widely, some solution is generally needed to ensure that peripherals do not slow overall system performance. Many CPUs feature a second set of pins similar to those for communicating with memory, but able to operate at very different speeds and using different protocols. Others use smart controllers to place the data directly in memory, a concept known as direct memory access. Most modern systems combine both solutions, where appropriate.

As the number of potential peripherals grew, using an expansion card for every peripheral became increasingly untenable. This has led to the introduction of bus systems designed specifically to support multiple peripherals. Common examples are the SATA ports in modern computers, which allow a number of hard drives to be connected without the need for a card. However, these high-performance systems are generally too expensive to implement in low-end devices, like a mouse. This has led to the parallel development of a number of low-performance bus systems for these solutions, the most

common example being Universal Serial Bus. All such examples may be referred to as peripheral buses, although this terminology is not universal.

In modern systems the performance difference between the CPU and main memory has grown so great that increasing amounts of high-speed memory is built directly into the CPU, known as a cache. In such systems, CPUs communicate using high-performance buses that operate at speeds much greater than memory, and communicate with memory using protocols similar to those used solely for peripherals in the past. These system buses are also used to communicate with most (or all) other peripherals, through adaptors, which in turn talk to other peripherals and controllers. Such systems are architecturally more similar to multicomputers, communicating over a bus rather than a network. In these cases, expansion buses are entirely separate and no longer share any architecture with their host CPU (and may in fact support many different CPUs, as is the case with PCI). What would have formerly been a system bus is now often known as a front-side bus.

Given these changes, the classical terms "system", "expansion" and "peripheral" no longer have the same connotations. Other common categorization systems are based on the buses primary role, connecting devices internally or externally, PCI vs. SCSI for instance. However, many common modern bus systems can be used for both; SATA and the associated eSATA are one example of a system that would formerly be described as internal, while in certain automotive applications use the primarily external IEEE 1394 in a fashion more similar to a system bus. Other examples, like InfiniBand and I^2C were designed from the start to be used both internally and externally.

13.1.1 Internal bus

The internal bus, also known as internal data bus, memory bus, system bus or Front-Side-Bus, connects all the internal components of a computer, such as CPU and memory, to the motherboard. Internal data buses are also referred to as a local bus, because they are intended to connect to local devices. This bus is typically rather quick and is independent of the rest of the computer operations.

13.1.2 External bus

The external bus, or expansion bus, is made up of the electronic pathways that connect the different external devices, such as printer etc., to the computer.

13.2 Implementation details

Buses can be parallel buses, which carry data words in parallel on multiple wires, or serial buses, which carry data in bit-serial form. The addition of extra power and control connections, differential drivers, and data connections in each direction usually means that most serial buses have more conductors than the minimum of one used in 1-Wire and UNI/O. As data rates increase, the problems of timing skew, power consumption, electromagnetic interference and crosstalk across parallel buses become more and more difficult to circumvent. One partial solution to this problem has been to double pump the bus. Often, a serial bus can be operated at higher overall data rates than a parallel bus, despite having fewer electrical connections, because a serial bus inherently has no timing skew or crosstalk. USB, FireWire, and Serial ATA are examples of this. Multidrop connections do not work well for fast serial buses, so most modern serial buses use daisy-chain or hub designs.

Network connections such as Ethernet are not generally regarded as buses, although the difference is largely conceptual rather than practical. An attribute generally used to characterize a bus is that power is provided by the bus for the connected hardware. This emphasizes the busbar origins of bus architecture as supplying switched or distributed power. This excludes, as buses, schemes such as serial RS-232, parallel Centronics, IEEE 1284 interfaces and Ethernet, since these devices also needed separate power supplies. Universal Serial Bus devices may use the bus supplied power, but often use a separate power source. This distinction is exemplified by a telephone system with a connected modem, where the RJ11 connection and associated modulated signalling scheme is not considered a bus, and is analogous to an Ethernet connection. A phone line connection scheme is not considered to be a bus with respect to signals, but the Central Office uses buses with cross-bar switches for connections between phones.

However, this distinction—that power is provided by the bus—is not the case in many avionic systems, where data connections such as ARINC 429, ARINC 629, MIL-STD-1553B (STANAG 3838), and EFABus (STANAG 3910) are commonly referred to as "data buses" or, sometimes, "databuses". Such avionic data buses are usually characterized by having several equipments or Line Replaceable Items/Units (LRI/LRUs) connected to a common, shared media. They may, as with ARINC 429, be simplex, i.e. have a single source LRI/LRU or, as with ARINC 629, MIL-STD-1553B, and STANAG 3910, be duplex, allow all the connected LRI/LRUs to act, at different times (half duplex), as transmitters and receivers of data.[2]

13.3 History

Over time, several groups of people worked on various computer bus standards, including the IEEE Bus Architecture Standards Committee (BASC), the IEEE "Superbus" study group, the open microprocessor initia-

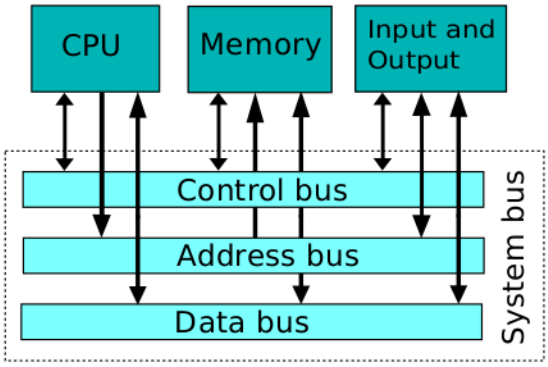
tive (OMI), the open microsystems initiative (OMI), the "Gang of Nine" that developed EISA, etc.

13.3.1 First generation

Early computer buses were bundles of wire that attached computer memory and peripherals. Anecdotally termed the "*digit trunk*",[3] they were named after electrical power buses, or busbars. Almost always, there was one bus for memory, and one or more separate buses for peripherals. These were accessed by separate instructions, with completely different timings and protocols.

One of the first complications was the use of interrupts. Early computer programs performed I/O by waiting in a loop for the peripheral to become ready. This was a waste of time for programs that had other tasks to do. Also, if the program attempted to perform those other tasks, it might take too long for the program to check again, resulting in loss of data. Engineers thus arranged for the peripherals to interrupt the CPU. The interrupts had to be prioritized, because the CPU can only execute code for one peripheral at a time, and some devices are more time-critical than others.

High-end systems introduced the idea of channel controllers, which were essentially small computers dedicated to handling the input and output of a given bus. IBM introduced these on the IBM 709 in 1958, and they became a common feature of their platforms. Other high-performance vendors like Control Data Corporation implemented similar designs. Generally, the channel controllers would do their best to run all of the bus operations internally, moving data when the CPU was known to be busy elsewhere if possible, and only using interrupts when necessary. This greatly reduced CPU load, and provided better overall system performance.



Single system bus

To provide modularity, memory and I/O buses can be combined into a unified system bus.[4] In this case, a single mechanical and electrical system can be used to connect together many of the system components, or in some cases, all of them.

Later computer programs began to share memory common to several CPUs. Access to this memory bus had to be prioritized, as well. The simple way to prioritize interrupts or bus access was with a daisy chain. In this case signals will naturally flow through the bus in physical or logical order, eliminating the need for complex scheduling.

13.3.2 Minis and micros

Digital Equipment Corporation (DEC) further reduced cost for mass-produced minicomputers, and mapped peripherals into the memory bus, so that the input and output devices appeared to be memory locations. This was implemented in the Unibus of the PDP-11 around 1969.[5]

Early microcomputer bus systems were essentially a passive backplane connected directly or through buffer amplifiers to the pins of the CPU. Memory and other devices would be added to the bus using the same address and data pins as the CPU itself used, connected in parallel. Communication was controlled by the CPU, which had read and written data from the devices as if they are blocks of memory, using the same instructions, all timed by a central clock controlling the speed of the CPU. Still, devices interrupted the CPU by signaling on separate CPU pins. For instance, a disk drive controller would signal the CPU that new data was ready to be read, at which point the CPU would move the data by reading the "memory location" that corresponded to the disk drive. Almost all early microcomputers were built in this fashion, starting with the S-100 bus in the Altair 8800 computer system.

In some instances, most notably in the IBM PC, although similar physical architecture can be employed, instructions to access peripherals (in and out) and memory (mov and others) have not been made uniform at all, and still generate distinct CPU signals, that could be used to implement a separate I/O bus.

These simple bus systems had a serious drawback when used for general-purpose computers. All the equipment on the bus had to talk at the same speed, as it shared a single clock.

Increasing the speed of the CPU becomes harder, because the speed of all the devices must increase as well. When it is not practical or economical to have all devices as fast as the CPU, the CPU must either enter a wait state, or work at a slower clock frequency temporarily,[6] to talk to other devices in the computer. While acceptable in embedded systems, this problem was not tolerated for long in general-purpose, user-expandable computers.

Such bus systems are also difficult to configure when constructed from common off-the-shelf equipment. Typically each added expansion card requires many jumpers in order to set memory addresses, I/O addresses, interrupt priorities, and interrupt numbers.

13.3.3 Second generation

"Second generation" bus systems like NuBus addressed some of these problems. They typically separated the computer into two "worlds", the CPU and memory on one side, and the various devices on the other. A *bus controller* accepted data from the CPU side to be moved to the peripherals side, thus shifting the communications protocol burden from the CPU itself. This allowed the CPU and memory side to evolve separately from the device bus, or just "bus". Devices on the bus could talk to each other with no CPU intervention. This led to much better "real world" performance, but also required the cards to be much more complex. These buses also often addressed speed issues by being "bigger" in terms of the size of the data path, moving from 8-bit parallel buses in the first generation, to 16 or 32-bit in the second, as well as adding software setup (now standardised as Plug-n-play) to supplant or replace the jumpers.

However these newer systems shared one quality with their earlier cousins, in that everyone on the bus had to talk at the same speed. While the CPU was now isolated and could increase speed, CPUs and memory continued to increase in speed much faster than the buses they talked to. The result was that the bus speeds were now very much slower than what a modern system needed, and the machines were left starved for data. A particularly common example of this problem was that video cards quickly outran even the newer bus systems like PCI, and computers began to include AGP just to drive the video card. By 2004 AGP was outgrown again by high-end video cards and other peripherals and has been replaced by the new PCI Express bus.

An increasing number of external devices started employing their own bus systems as well. When disk drives were first introduced, they would be added to the machine with a card plugged into the bus, which is why computers have so many slots on the bus. But through the 1980s and 1990s, new systems like SCSI and IDE were introduced to serve this need, leaving most slots in modern systems empty. Today there are likely to be about five different buses in the typical machine, supporting various devices.

13.3.4 Third generation

See also: Bus network

"Third generation" buses have been emerging into the market since about 2001, including HyperTransport and InfiniBand. They also tend to be very flexible in terms of their physical connections, allowing them to be used both as internal buses, as well as connecting different machines together. This can lead to complex problems when trying to service different requests, so much of the work on these systems concerns software design, as opposed to the hardware itself. In general, these third generation buses tend to look more like a network than the original concept of a bus, with a higher protocol overhead needed than early systems, while also allowing multiple devices to use the bus at once.

Buses such as Wishbone have been developed by the open source hardware movement in an attempt to further remove legal and patent constraints from computer design.

13.4 Examples of internal computer buses

13.4.1 Parallel

- ASUS Media Bus proprietary, used on some ASUS Socket 7 motherboards
- Computer Automated Measurement and Control (CAMAC) for instrumentation systems
- Extended ISA or EISA
- Industry Standard Architecture or ISA
- Low Pin Count or LPC
- MBus
- MicroChannel or MCA
- Multibus for industrial systems
- NuBus or IEEE 1196
- OPTi local bus used on early Intel 80486 motherboards.
- Conventional PCI
- Parallel ATA (also known as Advanced Technology Attachment, ATA, PATA, IDE, EIDE, ATAPI, etc.) disk/tape peripheral attachment bus
- S-100 bus or IEEE 696, used in the Altair and similar microcomputers
- SBus or IEEE 1496
- SS-50 Bus
- Runway bus, a proprietary front side CPU bus developed by Hewlett-Packard for use by its PA-RISC microprocessor family
- GSC/HSC, a proprietary peripheral bus developed by Hewlett-Packard for use by its PA-RISC microprocessor family
- Precision Bus, a proprietary bus developed by Hewlett-Packard for use by its HP3000 computer family
- STEbus

- STD Bus (for STD-80 [8-bit] and STD32 [16-/32-bit]), FAQ
- Unibus, a proprietary bus developed by Digital Equipment Corporation for their PDP-11 and early VAX computers.
- Q-Bus, a proprietary bus developed by Digital Equipment Corporation for their PDP and later VAX computers.
- VESA Local Bus or VLB or VL-bus
- VMEbus, the VERSAmodule Eurocard bus
- PC/104
- PC/104 Plus
- PC/104 Express
- PCI-104
- PCIe-104
- Zorro II and Zorro III, used in Amiga computer systems

13.4.2 Serial

- 1-Wire
- HyperTransport
- I^2C
- PCI Express or PCIe
- Serial ATA (SATA)
- Serial Peripheral Interface Bus or SPI bus
- UNI/O
- SMBus

13.5 Examples of external computer buses

13.5.1 Parallel

- HIPPI HIgh Performance Parallel Interface
- IEEE-488 (also known as GPIB, General-Purpose Interface Bus, and HPIB, Hewlett-Packard Instrumentation Bus)
- PC Card, previously known as *PCMCIA*, much used in laptop computers and other portables, but fading with the introduction of USB and built-in network and modem connections

13.5.2 Serial

- Controller area network ("CAN bus")
- eSATA
- ExpressCard
- Fieldbus
- IEEE 1394 interface (FireWire)
- Lightning
- RS-232
- RS-485
- Thunderbolt (interface)
- USB Universal Serial Bus, used for a variety of external devices

13.6 Examples of internal/external computer buses

- Futurebus
- InfiniBand
- PCI Express External Cabling
- QuickRing
- Scalable Coherent Interface (SCI)
- SCSI Small Computer System Interface, disk/tape peripheral attachment bus
- Serial Attached SCSI (SAS) and other serial SCSI buses
- Thunderbolt
- Yapbus, a proprietary bus developed for the Pixar Image Computer

13.7 See also

- Address bus
- Bus contention
- Control bus
- Data bus
- Front-side bus (FSB)
- External Bus Interface (EBI)
- Harvard architecture
- Network On Chip

- List of device bandwidths
- List of network buses
- Software bus

13.8 References

[1] "bus Definition from PC Magazine Encyclopedia". pcmag.com. 2014-05-29. Retrieved 2014-06-21.

[2] Avionic Systems Standardisation Committee, *Guide to Digital Interface Standards For Military Avionic Applications*, ASSC/110/6/2, Issue 2, September 2003

[3] See the early Australian CSIRAC computer

[4] Linda Null; Julia Lobur (2006). *The essentials of computer organization and architecture* (2nd ed.). Jones & Bartlett Learning. pp. 33,179–181. ISBN 978-0-7637-3769-6.

[5] C. Gordon Bell; R. Cady; H. McFarland; B. Delagi; J. O'Laughlin; R. Noonan; W. Wulf (1970). "A New Architecture for Mini-Computers—The DEC PDP-11" (PDF). *Spring Joint Computer Conference*: 657–675.

[6] Bray, Andrew C.; Dickens, Adrian C.; Holmes, Mark A. (1983). "28. The One Megahertz bus". *The Advanced User Guide for the BBC Microcomputer* (zipped PDF). Cambridge, UK: Cambridge Microcomputer Centre. pp. 442–443. ISBN 0-946827-00-1. Retrieved 2008-03-28.

13.9 External links

- Computer hardware buses at DMOZ
- Computer hardware buses and slots pinouts with brief descriptions

Chapter 14

Advanced Microcontroller Bus Architecture

The ARM **Advanced Microcontroller Bus Architecture (AMBA)** is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in system-on-a-chip (SoC) designs. It facilitates development of multi-processor designs with large numbers of controllers and peripherals. Since its inception, the scope of AMBA has, despite its name, gone far beyond micro controller devices. Today, AMBA is widely used on a range of ASIC and SoC parts including applications processors used in modern portable mobile devices like smartphones. AMBA is a registered trademark of ARM Ltd.[1]

AMBA was introduced by ARM in 1996. The first AMBA buses were Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). In its second version, AMBA 2, ARM added AMBA High-performance Bus (AHB) that is a single clock-edge protocol. In 2003, ARM introduced the third generation, AMBA 3, including AXI to reach even higher performance interconnect and the Advanced Trace Bus (ATB) as part of the CoreSight on-chip debug and trace solution. In 2010 the AMBA 4 specifications were introduced starting with AMBA 4 AXI4, then in 2011[2] extending system wide coherency with AMBA 4 ACE. In 2013[3] the AMBA 5 CHI (Coherent Hub Interface) specification was introduced, with a re-designed high-speed transport layer and features designed to reduce congestion.

These protocols are today the de facto standard for 32-bit embedded processors because they are well documented and can be used without royalties.

14.1 Design principles

An important aspect of a SoC is not only which components or blocks it houses, but also how they interconnect. AMBA is a solution for the blocks to interface with each other.

The objective of the AMBA specification is to:

- facilitate *right-first-time* development of embedded microcontroller products with one or more CPUs, GPUs or signal processors,
- be technology independent, to allow reuse of IP cores, peripheral and system macrocells across diverse IC processes,
- encourage modular system design to improve processor independence, and the development of reusable peripheral and system IP libraries
- minimize silicon infrastructure while supporting high performance and low power on-chip communication.

14.2 AMBA protocol specifications

The AMBA specification defines an on-chip communications standard for designing high-performance embedded microcontrollers. It is supported by ARM Limited with wide cross-industry participation.

The AMBA 5 specification defines following buses/interfaces:

- CHI Coherent Hub Interface (CHI) [3]

The AMBA 4 specification defines following buses/interfaces:

- AXI Coherency Extensions (ACE) - widely used on the latest ARM Cortex-A processors including Cortex-A7 and Cortex-A15
- AXI Coherency Extensions Lite (ACE-Lite)
- Advanced Extensible Interface 4 (AXI4)
- Advanced Extensible Interface 4 Lite (AXI4-Lite)
- Advanced Extensible Interface 4 Stream (AXI4-Stream v1.0)
- Advanced Trace Bus (ATB v1.1)

- Advanced Peripheral Bus (APB4 v2.0)

AMBA 3 specification defines four buses/interfaces:

- Advanced Extensible Interface (AXI3 or AXI v1.0) - widely used on ARM Cortex-A processors including Cortex-A9
- Advanced High-performance Bus Lite (AHB-Lite v1.0)
- Advanced Peripheral Bus (APB3 v1.0)
- Advanced Trace Bus (ATB v1.0)

AMBA 2 specification defines three buses/interfaces:

- Advanced High-performance Bus (AHB) - widely used on ARM7, ARM9 and ARM Cortex-M based designs
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB2 or APB)

AMBA specification (First version) defines two buses/interfaces:

- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)

The timing aspects and the voltage levels on the bus are not dictated by the specifications.

14.2.1 AXI Coherency Extensions (ACE and ACE-Lite)

ACE, defined as part of the AMBA 4 specification, extends AXI with additional signalling introducing system wide coherency.[4] This system coherency allows multiple processors to share memory and enables technology like ARM's big.LITTLE processing. The **ACE-Lite** protocol enables one-way aka IO coherency, for example a network interface that can read from the caches of a fully coherent ACE processor.

14.2.2 Advanced eXtensible Interface (AXI)

AXI, the third generation of AMBA interface defined in the AMBA 3 specification, is targeted at high performance, high clock frequency system designs and includes features that make it suitable for high speed submicrometer interconnect:

- separate address/control and data phases
- support for unaligned data transfers using byte strobes
- burst based transactions with only start address issued
- issuing of multiple outstanding addresses with out of order responses
- easy addition of register stages to provide timing closure.

14.2.3 Advanced High-performance Bus (AHB)

AHB is a bus protocol introduced in Advanced Microcontroller Bus Architecture version 2 published by ARM Ltd company.

In addition to previous release, it has the following features:

- large bus-widths (64/128 bit).

A simple transaction on the AHB consists of an address phase and a subsequent data phase (without wait states: only two bus-cycles). Access to the target device is controlled through a MUX (non-tristate), thereby admitting bus-access to one bus-master at a time.

AHB-Lite is a subset of AHB formally defined in the AMBA 3 standard. This subset simplifies the design for a bus with a single master.

14.2.4 Advanced Peripheral Bus (APB)

APB is designed for low bandwidth control accesses, for example register interfaces on system peripherals. This bus has an address and data phase similar to AHB, but a much reduced, low complexity signal list (for example no bursts).

14.3 AMBA products

A family of synthesizable intellectual property (IP) cores AMBA Products licensable from ARM Limited that implement a digital highway in a SoC for the efficient moving and storing of data using the AMBA protocol specifications. The AMBA family includes AMBA Network Interconnect (CoreLink NIC-400), Cache Coherent Interconnect (CoreLink CCI-500) SDRAM memory controllers (CoreLink DMC-400), DMA controllers (CoreLink DMA-230, DMA-330), level 2 cache controllers (L2C-310), etc.

A number of manufacturers utilize AMBA buses for non-ARM designs. As an example Infineon uses an AMBA bus for the ADM5120 SoC based on the MIPS architecture.

14.4 Competitors

- OpenCores Wishbone bus – Free and open bus architecture (formerly from Silicore)
- IBM CoreConnect bus technology, used in IBM's embedded Power Architecture products, but also in many other SoC-like systems with the Xilinx MicroBlaze or similar cores
- IDT IPBus
- Altera Avalon – proprietary bus system for Altera's Nios II SoCs [5]
- OCP Open Core Protocol
- Hyper Transport from AMD (though this is an off-chip interface, not on chip bus)
- Quick Path from Intel (though this is an off-chip interface, not on chip bus)

14.5 See also

- Functional specification

14.6 References

[1] AMBA Trademark License, http://arm.com/about/trademarks/arm-trademark-list/AMBA-trademark.php

[2] New AMBA 4 Specification Optimizes Coherency for Heterogeneous Multicore SoCs, http://www.arm.com/about/newsroom/new-amba-4-specification-optimizes-coherency-for-heterogeneous-multicore-socs.php

[3] ARM Announces AMBA 5 CHI Specification to Enable High Performance, Highly Scalable System on Chip Technology, http://www.arm.com/about/newsroom/arm-announces-amba-5-chi-specification-to-enable-high-performance-highly-scalable-system-on-chip.php

[4] Kriouile, A., & Serwe, W. (2013). Formal Analysis of the ACE Specification for Cache Coherent Systems-on-Chip. In Formal Methods for Industrial Critical Systems (pp. 108-122). Springer Berlin Heidelberg., http://link.springer.com/chapter/10.1007/978-3-642-41010-9_8#

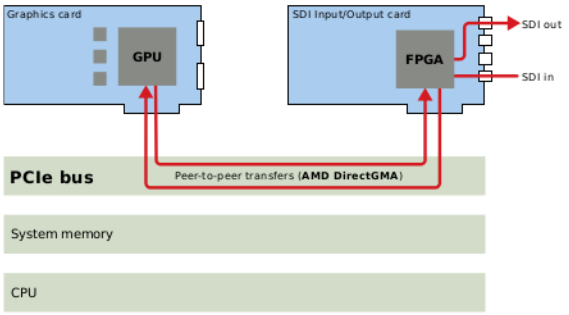
[5] Avalon

14.7 External links

- AMBA Specification home page - of ARM
- AMBA of ARM
- AMBA Documentation - from ARM
- AMBA 2 Specification including AHB - from ARM
- AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite - from ARM
- AMBA APB Specification including APB4, APB3, APB2 - from ARM

Chapter 15

Direct memory access



AMD DirectGMA is a form of DMA. It enables low-latency peer-to-peer data transfers between devices on the PCIe bus and AMD FirePro-branded products. SDI devices supporting DirectGMA can write directly into the graphics memory of the GPU and vice versa the GPU can directly access the memory of a peer device.

Direct memory access (**DMA**) is a feature of computer systems that allows certain hardware subsystems to access main system memory (RAM) independently of the central processing unit (CPU).

Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU first initiates the transfer, then it does other operations while the transfer is in progress, and it finally receives an interrupt from the DMA controller when the operation is done. This feature is useful at any time that the CPU cannot keep up with the rate of data transfer, or when the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer. Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards and sound cards. DMA is also used for intra-chip data transfer in multi-core processors. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without DMA channels. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel.

DMA can also be used for "memory to memory" copying or moving of data within memory. DMA can offload expensive memory operations, such as large copies or scatter-gather operations, from the CPU to a dedicated DMA engine. An implementation example is the I/O Acceleration Technology.

15.1 Principle

A DMA controller can generate memory addresses and initiate memory read or write cycles. It contains several processor registers that can be written and read by the CPU. These include a memory address register, a byte count register, and one or more control registers. The control registers specify the I/O port to use, the direction of the transfer (reading from the I/O device or writing to the I/O device), the transfer unit (byte at a time or word at a time), and the number of bytes to transfer in one burst.[1]

To carry out an input, output or memory-to-memory operation, the host processor initializes the DMA controller with a count of the number of words to transfer, and the memory address to use. The CPU then sends commands to a peripheral device to initiate transfer of data. The DMA controller then provides addresses and read/write control lines to the system memory. Each time a byte of data is ready to be transferred between the peripheral device and memory, the DMA controller increments its internal address register until the full block of data is transferred.

DMA transfers can either occur one byte at a time or all at once in burst mode. If they occur a byte at a time, this can allow the CPU to access memory on alternate bus cycles – this is called cycle stealing since the DMA controller and CPU contend for memory access. In *burst mode DMA*, the CPU can be put on hold while the DMA transfer occurs and a full block of possibly hundreds or thousands of bytes can be moved.[2] When memory cycles are much faster than processor cycles, an *interleaved* DMA cycle is possible, where the DMA controller uses memory while the CPU cannot.

In a bus mastering system, the CPU and peripherals can each be granted control of the memory bus. Where a peripheral can become bus master, it can directly write to

system memory without involvement of the CPU, providing memory address and control signals as required. Some measure must be provided to put the processor into a hold condition so that bus contention does not occur.

15.2 Modes of operation

15.2.1 Burst mode

An entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system bus by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU, but renders the CPU inactive for relatively long periods of time. The mode is also called "Block Transfer Mode". It is also used to stop unnecessary data.

15.2.2 Cycle stealing mode

The *cycle stealing mode* is used in systems in which the CPU should not be disabled for the length of time needed for burst transfer modes. In the cycle stealing mode, the DMA controller obtains access to the system bus the same way as in burst mode, using **BR (Bus Request) and BG (Bus Grant) signals**, which are the two signals controlling the interface between the CPU and the DMA controller. However, in cycle stealing mode, after one byte of data transfer, the control of the system bus is deasserted to the CPU via BG. It is then continually requested again via BR, transferring one byte of data per request, until the entire block of data has been transferred. By continually obtaining and releasing the control of the system bus, the DMA controller essentially interleaves instruction and data transfers. The CPU processes an instruction, then the DMA controller transfers one data value, and so on. On the one hand, the data block is not transferred as quickly in cycle stealing mode as in burst mode, but on the other hand the CPU is not idled for as long as in burst mode. Cycle stealing mode is useful for controllers that monitor data in real time.

15.2.3 Transparent mode

The *transparent mode* takes the most time to transfer a block of data, yet it is also the most efficient mode in terms of overall system performance. The DMA controller only transfers data when the CPU is performing operations that do not use the system buses. It is the primary advantage of the transparent mode that the CPU never stops executing its programs and the DMA transfer is free in terms of time. The disadvantage of the transparent mode is that the hardware needs to determine when the CPU is not using the system buses, which can be complex.

15.3 Cache coherency

Cache incoherence due to DMA

DMA can lead to cache coherency problems. Imagine a CPU equipped with a cache and an external memory that can be accessed directly by devices using DMA. When the CPU accesses location X in the memory, the current value will be stored in the cache. Subsequent operations on X will update the cached copy of X, but not the external memory version of X, assuming a write-back cache. If the cache is not flushed to the memory before the next time a device tries to access X, the device will receive a stale value of X.

Similarly, if the cached copy of X is not invalidated when a device writes a new value to the memory, then the CPU will operate on a stale value of X.

This issue can be addressed in one of two ways in system design: Cache-coherent systems implement a method in hardware whereby external writes are signaled to the cache controller which then performs a cache invalidation for DMA writes or cache flush for DMA reads. Non-coherent systems leave this to software, where the OS must then ensure that the cache lines are flushed before an outgoing DMA transfer is started and invalidated before a memory range affected by an incoming DMA transfer is accessed. The OS must make sure that the memory range is not accessed by any running threads in the meantime. The latter approach introduces some overhead to the DMA operation, as most hardware requires a loop to invalidate each cache line individually.

Hybrids also exist, where the secondary L2 cache is coherent while the L1 cache (typically on-CPU) is managed by software.

15.4 Examples

15.4.1 ISA

In the original IBM PC, there was only one Intel 8237 DMA controller capable of providing four DMA channels (numbered 0–3), as part of the so-called Industry Standard Architecture, or ISA. These DMA channels performed 8-bit transfers and could only address the first megabyte of RAM. With the IBM PC/AT, a second 8237 DMA controller was added (channels 5–7; channel 4 is dedicated as a cascade channel for the first 8237 controller), and the page register was rewired to address the full 16 MB memory address space of the 80286 CPU. This second controller performed 16-bit transfers.

Due to their lagging performance (2.5 Mbit/s[3]), these

devices have been largely obsolete since the advent of the 80386 processor in 1985 and its capacity for 32-bit transfers. They are still supported to the extent they are required to support built-in legacy PC hardware on modern machines. The only pieces of legacy hardware that use ISA DMA and are still fairly common are Super I/O devices on motherboards that often integrate a built-in floppy disk controller, an IrDA infrared controller when FIR (fast infrared) mode is selected, and a IEEE 1284 parallel port controller when ECP mode is selected.

Each DMA channel has a 16-bit address register and a 16-bit count register associated with it. To initiate a data transfer the device driver sets up the DMA channel's address and count registers together with the direction of the data transfer, read or write. It then instructs the DMA hardware to begin the transfer. When the transfer is complete, the device interrupts the CPU.

Scatter-gather or vectored I/O DMA allows the transfer of data to and from multiple memory areas in a single DMA transaction. It is equivalent to the chaining together of multiple simple DMA requests. The motivation is to off-load multiple input/output interrupt and data copy tasks from the CPU.

DRQ stands for *Data request*; DACK for *Data acknowledge*. These symbols, seen on hardware schematics of computer systems with DMA functionality, represent electronic signaling lines between the CPU and DMA controller. Each DMA channel has one Request and one Acknowledge line. A device that uses DMA must be configured to use both lines of the assigned DMA channel.

Standard ISA DMA assignments:

1. DRAM Refresh (obsolete),
2. User hardware, usually sound card 8-bit DMA
3. Floppy disk controller,
4. Hard disk (obsoleted by PIO modes, and replaced by UDMA modes), Parallel Port (ECP capable port), certain SoundBlaster Clones like the OPTi 928.
5. Cascade from XT DMA controller,
6. Hard Disk (PS/2 only), user hardware for all others, usually sound card 16-bit DMA
7. User hardware.
8. User hardware.

15.4.2 PCI

A PCI architecture has no central DMA controller, unlike ISA. Instead, any PCI component can request control of the bus ("become the bus master") and request to read from and write to system memory. More precisely, a PCI component requests bus ownership from the PCI bus controller (usually the southbridge in a modern PC design), which will arbitrate if several devices request bus ownership simultaneously, since there can only be one bus master at one time. When the component is granted ownership, it will issue normal read and write commands on the PCI bus, which will be claimed by the bus controller and will be forwarded to the memory controller using a scheme which is specific to every chipset.

As an example, on a modern AMD Socket AM2-based PC, the southbridge will forward the transactions to the northbridge (which is integrated on the CPU die) using HyperTransport, which will in turn convert them to DDR2 operations and send them out on the DDR2 memory bus. As can be seen, there are quite a number of steps involved in a PCI DMA transfer; however, that poses little problem, since the PCI device or PCI bus itself are an order of magnitude slower than the rest of the components (see list of device bandwidths).

A modern x86 CPU may use more than 4 GB of memory, utilizing PAE, a 36-bit addressing mode, or the native 64-bit mode of x86-64 CPUs. In such a case, a device using DMA with a 32-bit address bus is unable to address memory above the 4 GB line. The new Double Address Cycle (DAC) mechanism, if implemented on both the PCI bus and the device itself,[4] enables 64-bit DMA addressing. Otherwise, the operating system would need to work around the problem by either using costly double buffers (DOS/Windows nomenclature) also known as bounce buffers (FreeBSD/Linux), or it could use an IOMMU to provide address translation services if one is present.

15.4.3 I/OAT

Main article: I/O Acceleration Technology

As an example of DMA engine incorporated in a general-purpose CPU, newer Intel Xeon chipsets include a DMA engine technology called I/O Acceleration Technology (I/OAT), meant to improve network performance on high-throughput network interfaces, in particular gigabit Ethernet and faster.[5] However, various benchmarks with this approach by Intel's Linux kernel developer Andrew Grover indicate no more than 10% improvement in CPU utilization with receiving workloads, and no improvement when transmitting data.[6]

15.4.4 DDIO

Further performance-oriented enhancements to the DMA mechanism have been introduced in Intel Xeon E5 processors with their **Data Direct I/O** (**DDIO**) feature, allowing the DMA "windows" to reside within CPU caches instead of system RAM. As a result, CPU caches are used as the primary source and destination for I/O,

allowing network interface controllers (NICs) to talk directly to the caches of local CPUs and avoid costly fetching of the I/O data from system RAM. As a result, DDIO reduces the overall I/O processing latency, allows processing of the I/O to be performed entirely in-cache, prevents the available RAM bandwidth from becoming a performance bottleneck, and lowers the power consumption by allowing RAM to remain longer in low-powered state.[7][8][9][10]

15.4.5 AHB

Main article: Advanced Microcontroller Bus Architecture

In systems-on-a-chip and embedded systems, typical system bus infrastructure is a complex on-chip bus such as AMBA High-performance Bus. AMBA defines two kinds of AHB components: master and slave. A slave interface is similar to programmed I/O through which the software (running on embedded CPU, e.g. ARM) can write/read I/O registers or (less commonly) local memory blocks inside the device. A master interface can be used by the device to perform DMA transactions to/from system memory without heavily loading the CPU.

Therefore, high bandwidth devices such as network controllers that need to transfer huge amounts of data to/from system memory will have two interface adapters to the AHB: a master and a slave interface. This is because on-chip buses like AHB do not support tri-stating the bus or alternating the direction of any line on the bus. Like PCI, no central DMA controller is required since the DMA is bus-mastering, but an arbiter is required in case of multiple masters present on the system.

Internally, a multichannel DMA engine is usually present in the device to perform multiple concurrent scatter-gather operations as programmed by the software.

15.4.6 Cell

Main article: Cell (microprocessor)

As an example usage of DMA in a multiprocessor-system-on-chip, IBM/Sony/Toshiba's Cell processor incorporates a DMA engine for each of its 9 processing elements including one Power processor element (PPE) and eight synergistic processor elements (SPEs). Since the SPE's load/store instructions can read/write only its own local memory, an SPE entirely depends on DMAs to transfer data to and from the main memory and local memories of other SPEs. Thus the DMA acts as a primary means of data transfer among cores inside this CPU (in contrast to cache-coherent CMP architectures such as Intel's cancelled general-purpose GPU, Larrabee).

DMA in Cell is fully cache coherent (note however local stores of SPEs operated upon by DMA do not act as globally coherent cache in the standard sense). In both read ("get") and write ("put"), a DMA command can transfer either a single block area of size up to 16 KB, or a list of 2 to 2048 such blocks. The DMA command is issued by specifying a pair of a local address and a remote address: for example when a SPE program issues a put DMA command, it specifies an address of its own local memory as the source and a virtual memory address (pointing to either the main memory or the local memory of another SPE) as the target, together with a block size. According to a recent experiment, an effective peak performance of DMA in Cell (3 GHz, under uniform traffic) reaches 200 GB per second.[11]

15.5 See also

- AT Attachment
- Blitter
- Channel I/O
- DMA attack
- Polling (computer science)
- Remote direct memory access
- UDMA

15.6 Notes

[1] Osborne, Adam (1980). *An Introduction to Microcomputers: Volume 1: Basic Concepts* (2nd ed.). Osborne McGraw Hill. pp. 5–64 through 5–93. ISBN 0931988349.

[2] Horowitz, Paul; Hill, Winfield (1989). *The Art of Electronics* (Second ed.). Cambridge University Press. p. 702. ISBN 0521370957.

[3] Intel publication 03040, Aug 1989

[4] "Physical Address Extension — PAE Memory and Windows". Microsoft Windows Hardware Development Central. 2005. Retrieved 2008-04-07.

[5] Corbet, Jonathan (2005-12-06). "Memory copies in hardware". *LWN.net* (December 8, 2005). Retrieved 2006-11-12.

[6] Grover, Andrew (2006-06-01). "I/OAT on LinuxNet wiki". *Overview of I/OAT on Linux, with links to several benchmarks*. Retrieved 2006-12-12.

[7] "Intel Data Direct I/O (Intel DDIO): Frequently Asked Questions" (PDF). Intel. March 2012. Retrieved 2015-10-11.

[8] Rashid Khan (2015-09-29). "Pushing the Limits of Kernel Networking". *redhat.com*. Retrieved 2015-10-11.

[9] "Achieving Lowest Latencies at Highest Message Rates with Intel Xeon Processor E5-2600 and Solarflare SFN6122F 10 GbE Server Adapter" (PDF). *solarflare.com*. 2012-06-07. Retrieved 2015-10-11.

[10] Alexander Duyck (2015-08-19). "Pushing the Limits of Kernel Networking" (PDF). *linuxfoundation.org*. p. 5. Retrieved 2015-10-11.

[11] Kistler, Michael (May 2006). "Cell Multiprocessor Communication Network". *Extensive benchmarks of DMA performance in Cell Broadband Engine*.

15.7 References

- DMA Fundamentals on Various PC Platforms, from A. F. Harvey and Data Acquisition Division Staff NATIONAL INSTRUMENTS
- mmap() and DMA, from *Linux Device Drivers, 2nd Edition*, Alessandro Rubini & Jonathan Corbet
- Memory Mapping and DMA, from *Linux Device Drivers, 3rd Edition*, Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman
- DMA and Interrupt Handling
- DMA Modes & Bus Mastering

15.8 External links

- Mastering the DMA and IOMMU APIs, Embedded Linux Conference 2014, San Jose, by Laurent Pinchart

Chapter 16

Hardware verification language

A **hardware verification language**, or **HVL**, is a programming language used to verify the designs of electronic circuits written in a hardware description language. HVLs typically include features of a high-level programming language like C++ or Java as well as features for easy bit-level manipulation similar to those found in HDLs. Many HVLs will provide constrained random stimulus generation, and functional coverage constructs to assist with complex hardware verification.

SystemVerilog, OpenVera, e, and SystemC are the most commonly used HVLs. SystemVerilog attempts to combine HDL and HVL constructs into a single standard.

16.1 See also

- OpenVera
- e
- SystemC
- SystemVerilog
- Property Specification Language

16.2 References

16.3 External links

Chapter 17

Mixed-signal integrated circuit

A **mixed-signal integrated circuit** is any integrated circuit that has both analog circuits and digital circuits on a single semiconductor die. [1] [2] [3] [4] In real-life applications mixed-signal designs are everywhere, for example, a smart mobile phone. However, it is more accurate to call them mixed-signal systems. Mixed-signal ICs also process both analog and digital signals together. For example, an analog-to-digital converter is a mixed-signal circuit. Mixed-signal circuits or systems are typically cost-effective solutions for building any modern consumer electronics applications.

17.1 Introduction

A mixed-signal system-on-a-chip (AMS-SoC) can be a combination of analog circuits, digital circuits, intrinsic mixed-signal circuits (like ADC), and Embedded Software.

Integrated Circuits (ICs) are generally classified as digital (e.g. a microprocessors) or analog (e.g. an operational amplifier). Mixed-signal ICs are chips that contain both digital and analog circuits on the same chip. This category of chip has grown dramatically with the increased use of 3G cell phones and other portable technologies.

Mixed-signal ICs are often used to convert analog signals to digital signals so that digital devices can process them. For example, mixed-signal ICs are essential components for FM tuners in digital products such as media players, which have digital amplifiers. Any analog signal (such as an FM radio transmission, a light wave or a sound) can be digitized using a very basic analog-to-digital converter, and the smallest and most energy efficient of these would be in the form of mixed-signal ICs.

Mixed-signal ICs are more difficult to design and manufacture than analog-only or digital-only integrated circuits. For example, an efficient mixed-signal IC would have its digital and analog components share a common power supply. However, as one can imagine, analog and digital components have very different power needs and consumption characteristics that make this a non-trivial goal in chip design.

17.2 Examples

Typically, mixed-signal chips perform some whole function or sub-function in a larger assembly such as the radio subsystem of a cell phone, or the read data path and laser sled control logic of a DVD player. They often contain an entire system-on-a-chip.

Examples of mixed-signal integrated circuits include data converters using delta-sigma modulation, analog-to-digital converter/digital-to-analog converter using error detection and correction, and digital radio chips. Digitally controlled sound chips are also mixed-signal circuits. With the advent of cellular technology and network technology this category now includes cellular telephone, software radio, LAN and WAN router integrated circuits.

Because of the use of both digital signal processing and analog circuitry, mixed-signal ICs are usually designed for a very specific purpose and their design requires a high level of expertise and careful use of computer aided design (CAD) tools. Automated testing of the finished chips can also be challenging. Teradyne, Agilent, and Texas Instruments are the major suppliers of the test equipment for mixed-signal chips.

The particular challenges of mixed signal include:

- CMOS technology is usually optimal for digital performance and scaling while bipolar transistors are usually optimal for analog performance, yet until the last decade it has been difficult to either combine these cost-effectively or to design both analog and digital in a single technology without serious performance compromises. The advent of technologies like high performance CMOS, BiCMOS, CMOS SOI and SiGe have removed many of the compromises that previously had to be made.

- Testing functional operation of mixed-signal ICs remains complex, expensive and often a “one-off” implementation task.

- Systematic design methodologies comparable to digital design methods are far more primitive in the analog and mixed-signal arena. Analog circuit design can not generally be automated to nearly the

extent that digital circuit design can. Combining the two technologies multiplies this complication.

- Fast-changing digital signals send noise to sensitive analog inputs. One path for this noise is substrate coupling. A variety of techniques are used to attempt to block or cancel this noise coupling, such as fully differential amplifiers,[5] P+ guard-rings,[6] differential topology, on-chip decoupling, and triple-well isolation.[7]

17.3 Commercial examples

- ICsense
- AnSem
- Atari POKEY
- MOS Technology SID
- PSoC - Cypress PSoC Programmable System on Chip
- System to ASIC
- Texas Instruments' MSP430
- Triad Semiconductor
- Wolfson Microelectronics

Most modern radio and communications use mixed signal circuits.

17.4 See also

- Analog front-end

17.5 References

[1] Saraju Mohanty, Nanoelectronic Mixed-Signal System Design, McGraw-Hill, 2015, ISBN 978-0071825719 and 0071825711.

[2] "Mixed-Signal IC Design". quote: "mixed-signal (IC's with mixed analog and digital circuits on a single chip)"

[3] Mark Burns and Gordon W. Roberts, "An Introduction to Mixed-Signal IC Test and Measurement", 2001.

[4] "ESS Mixed Signal Circuits"

[5] "Fully differential current-input CMOS amplifier front-endsuppressing mixed signal substrate noise for opto-electronic applications" by Chang, J.J.; Myunghee Lee; Sungyong Jung; Brooke, M.A.; Jokerst, N.M.; Wills, D.S. 1999

[6] "Substrate noise issues in mixed-signal chip designs using Spice" by Singh, R.; Sali, S. 1997

[7] "Mixed-Signal IC Merges 14-Bit ADC With DSP In 0.18-µm CMOS"

17.6 Further reading

- Saraju Mohanty (2015). *Nanoelectronic Mixed-Signal System Design, ISBN 978-0071825719 and 0071825711, 1st Edition*. McGraw-Hill. External link in |title= (help)
- R. Jacob Baker (2009). *CMOS Mixed-Signal Circuit Design, Second Edition*. http://CMOSedu.com/

Chapter 18

Radio frequency

This article is about the generic oscillation. For the radiation, see Radio wave. For the electronics, see Radio frequency engineering.
"RF" redirects here. For other uses, see RF (disambiguation).

Radio frequency (**RF**) is any of the electromagnetic wave frequencies that lie in the range extending from around 3 kHz to 300 GHz, which include those frequencies used for communications or radar signals.[1] RF usually refers to electrical rather than mechanical oscillations. However, mechanical RF systems do exist (see mechanical filter and RF MEMS).

Although radio *frequency* is a rate of oscillation, the term "radio frequency" or its abbreviation "RF" are used as a synonym for radio – i.e., to describe the use of wireless communication, as opposed to communication via electric wires. Examples include:

- Radio-frequency identification
- ISO/IEC 14443−2 *Radio frequency power and signal interface*[2]

18.1 Special properties of RF current

Electric currents that oscillate at radio frequencies have special properties not shared by direct current or alternating current of lower frequencies.

- The energy in an RF current can radiate off a conductor into space as electromagnetic waves (radio waves); this is the basis of radio technology.
- RF current does not penetrate deeply into electrical conductors but tends to flow along their surfaces; this is known as the skin effect. For this reason, when the human body comes in contact with high power RF currents it can cause superficial but serious burns called *RF burns (Radiation burns)*.
- RF currents applied to the body often do not cause the painful sensation of electric shock as do lower frequency currents.[3][4] This is because the current changes direction too quickly to trigger depolarization of nerve membranes.
- RF current can easily ionize air, creating a conductive path through it. This property is exploited by "high frequency" units used in electric arc welding, which use currents at higher frequencies than power distribution uses.
- Another property is the ability to appear to flow through paths that contain insulating material, like the dielectric insulator of a capacitor.
- When conducted by an ordinary electric cable, RF current has a tendency to reflect from discontinuities in the cable such as connectors and travel back down the cable toward the source, causing a condition called standing waves. Therefore, RF current must be carried by specialized types of cable called transmission line.

18.2 Radio communication

To receive radio signals an antenna must be used. However, since the antenna will pick up thousands of radio signals at a time, a radio tuner is necessary to *tune into* a particular frequency (or frequency range).[5] This is typically done via a resonator – in its simplest form, a circuit with a capacitor and an inductor form a tuned circuit. The resonator amplifies oscillations within a particular frequency band, while reducing oscillations at other frequencies outside the band. Another method to isolate a particular radio frequency is by oversampling (which gets a wide range of frequencies) and picking out the frequencies of interest, as done in software defined radio.

The distance over which radio communications is useful depends significantly on things other than wavelength, such as transmitter power, receiver quality, type, size, and height of antenna, mode of transmission, noise, and interfering signals. Ground waves, tropospheric scatter and skywaves can all achieve greater ranges than line-of-sight

propagation. The study of radio propagation allows estimates of useful range to be made.

18.3 Frequency bands

Main article: Radio spectrum

18.4 In medicine

Radio frequency (RF) energy, in the form of radiating waves or electrical currents, has been used in medical treatments for over 75 years,[7] generally for minimally invasive surgeries, using radiofrequency ablation and cryoablation, including the treatment of sleep apnea.[8] Magnetic resonance imaging (MRI) uses radio frequency waves to generate images of the human body.

Radio frequencies at non-ablation energy levels are sometimes used as a form of cosmetic treatment that can tighten skin, reduce fat (lipolysis), or promote healing.[9]

RF diathermy is a medical treatment that uses RF induced heat as a form of physical or occupational therapy and in surgical procedures. It is commonly used for muscle relaxation. It is also a method of heating tissue electromagnetically for therapeutic purposes in medicine. Diathermy is used in physical therapy and occupational therapy to deliver moderate heat directly to pathologic lesions in the deeper tissues of the body. Surgically, the extreme heat that can be produced by diathermy may be used to destroy neoplasms, warts, and infected tissues, and to cauterize blood vessels to prevent excessive bleeding. The technique is particularly valuable in neurosurgery and surgery of the eye. Diathermy equipment typically operates in the short-wave radio frequency (range 1–100 MHz) or microwave energy (range 434–915 MHz).

Pulsed electromagnetic field therapy (PEMF) is a medical treatment that purportedly helps to heal bone tissue reported in a recent NASA study. This method usually employs electromagnetic radiation of different frequencies - ranging from static magnetic fields, through extremely low frequencies (ELF) to higher radio frequencies (RF) administered in pulses.

18.5 Effects on the human body

18.5.1 Extremely low frequency RF

High-power extremely low frequency RF with electric field levels in the low kV/m range are known to induce perceivable currents within the human body that create an annoying tingling sensation. These currents will typically flow to ground through a body contact surface such as the feet, or arc to ground where the body is well insulated.[10][11]

18.5.2 Microwaves

Main article: Microwave burn

Microwave exposure at low-power levels below the Specific absorption rate set by government regulatory bodies are considered harmless non-ionizing radiation and have no effect on the human body. However, levels above the Specific absorption rate set by the U.S. Federal Communications Commission are considered potentially harmful (see Mobile phone radiation and health).

Long-term human exposure to high-levels of microwaves is recognized to cause cataracts according to experimental animal studies and epidemiological studies. The mechanism is unclear but may include changes in heat sensitive enzymes that normally protect cell proteins in the lens. Another mechanism that has been advanced is direct damage to the lens from pressure waves induced in the aqueous humor.

High-power exposure to microwave RF is known to create a range of effects from lower to higher power levels, ranging from unpleasant burning sensation on the skin and microwave auditory effect, to extreme pain at the mid-range, to physical burning and blistering of skin and internals at high power levels (see microwave burn).

18.5.3 General RF exposure

The 1999 revision of Canadian Safety Code 6 recommended electric field limits of 100 kV/m for pulsed EMF to prevent air breakdown and spark discharges, mentioning rationale related to auditory effect and energy-induced unconsciousness in rats.[12] The pulsed EMF limit was removed in later revisions, however.[13]

For health effects see electromagnetic radiation and health.

For high-power RF exposure see radiation burn.

For low-power RF exposure see radiation-induced cancer.

18.6 As a weapon

See also: Directed energy weapons § Microwave weapons

A heat ray is an RF harassment device that makes use of microwave radio frequencies to create an unpleasant heating effect in the upper layer of the skin. A publicly known heat ray weapon called the Active Denial System was de-

veloped by the US military as an experimental weapon to deny the enemy access to an area. A death ray is a weapon that delivers heat ray electromagnetic energy at levels that injure human tissue. The inventor of the death ray, Harry Grindell Matthews, claims to have lost sight in his left eye while developing his death ray weapon based on a primitive microwave magnetron from the 1920s (note that a typical microwave oven induces a tissue damaging cooking effect inside the oven at about 2 kV/m.)

18.7 Measurement

Since radio frequency radiation has both an electric and a magnetic component, it is often convenient to express intensity of radiation field in terms of units specific to each component. The unit *volts per meter* (V/m) is used for the electric component, and the unit *amperes per meter* (A/m) is used for the magnetic component. One can speak of an electromagnetic field, and these units are used to provide information about the levels of electric and magnetic field strength at a measurement location.

Another commonly used unit for characterizing an RF electromagnetic field is *power density*. Power density is most accurately used when the point of measurement is far enough away from the RF emitter to be located in what is referred to as the far field zone of the radiation pattern. In closer proximity to the transmitter, i.e., in the "near field" zone, the physical relationships between the electric and magnetic components of the field can be complex, and it is best to use the field strength units discussed above. Power density is measured in terms of power per unit area, for example, milliwatts per square centimeter (mW/cm^2). When speaking of frequencies in the microwave range and higher, power density is usually used to express intensity since exposures that might occur would likely be in the far field zone.

18.8 See also

- Amplitude modulation
- Electromagnetic Interference
- Electromagnetic radiation
- Electromagnetic spectrum
- EMF measurement
- Frequency allocation
- Frequency bandwidth
- Frequency modulation
- Plastic welding
- Pulsed electromagnetic field therapy
- Spectrum management

18.9 References

[1] "Definition of RADIO FREQUENCY". *Merriam-Webster*. Encyclopædia Britannica. n.d. Retrieved 6 August 2015.

[2] "ISO/IEC 14443-2:2001 Identification cards — Contactless integrated circuit(s) cards — Proximity cards — Part 2: Radio frequency power and signal interface". Iso.org. 2010-08-19. Retrieved 2011-11-08.

[3] Curtis, Thomas Stanley (1916). *High Frequency Apparatus: Its Construction and Practical Application*. USA: Everyday Mechanics Company. p. 6.

[4] Mieny, C. J. (2003). *Principles of Surgical Patient Care* (2nd ed.). New Africa Books. p. 136. ISBN 9781869280055.

[5] Brain, Marshall (2000-12-07). "How Radio Works". HowStuffWorks.com. Retrieved 2009-09-11.

[6] Jeffrey S. Beasley; Gary M. Miller (2008). *Modern Electronic Communication* (9th ed.). pp. 4–5. ISBN 978-0132251136.

[7] Ruey J. Sung and Michael R. Lauer (2000). *Fundamental approaches to the management of cardiac arrhythmias*. Springer. p. 153. ISBN 978-0-7923-6559-4.

[8] Melvin A. Shiffman, Sid J. Mirrafati, Samuel M. Lam and Chelso G. Cueteaux (2007). *Simplified Facial Rejuvenation*. Springer. p. 157. ISBN 978-3-540-71096-7.

[9] Noninvasive Radio Frequency for Skin Tightening and Body Contouring, Frontline Medical Communications, 2013

[10] Limits of Human Exposure to Radiofrequency Electromagnetic Fields in the Frequency Range from 3 kHz to 300 GHz, Canada Safety Code 6, page 63

[11] Extremely Low Frequency Fields Environmental Health Criteria Monograph No.238, chapter 5, page 121, WHO

[12] Limits of Human Exposure to Radiofrequency Electromagnetic Fields in the Frequency Range from 3 kHz to 300 GHz, Canada Safety Code 6,page 62

[13] http://www.hc-sc.gc.ca/ewh-semt/pubs/radiation/radio_guide-lignes_direct/index-eng.php Safety Code 6: Health Canada's Radiofrequency Exposure Guidelines - Environmental and Workplace Health - Health Canada

18.10 External links

- Definition of frequency bands (VLF, ELF … etc.) IK1QFK Home Page (vlf.it)
- Radio, light, and sound waves, conversion between wavelength and frequency
- RF Terms Glossary

18.11 Text and image sources, contributors, and licenses

18.11.1 Text

- **System on a chip** *Source:* https://en.wikipedia.org/wiki/System_on_a_chip?oldid=697585543 *Contributors:* Butik, Fuzzie, Nixdorf, NuclearWinner, Julesd, Taxman, Robbot, Altenmann, Nurg, Ancheta Wis, Thv, DocWatson42, Uday, Khalid hassani, Pgan002, Calm, Pinnerup, Abdull, Imroy, Florian Blaschke, Dyl, CanisRufus, Kjkolb, Trevj, Jakew, Atlant, Twisp, Paul1337, Cburnett, Kenyon, Oleg Alexandrov, Brycen, Woohookitty, David Haslam, Hdante, AshishG, Alecv, Mekong Bluesman, Ashwin.rao, Lemoncurd, Qwertyus, Intgr, Chobot, RussBot, Toffile, SamJohnston, Welsh, Yahya Abdal-Aziz, Speedevil, Blitterbug, Deville, Nwk, Tevildo, Tsiaojian lee, ViperSnake151, SmackBot, Maelwys, Henriok, KelleyCook, Gilliam, Bluebot, TimBentley, Trebor, Thumperward, DHN-bot~enwiki, Frap, Maksim-bot, LouScheffer, Zvar, MichaelBillington, AThing, Parikshit Narkhede, Dicklyon, PeterJohnBishop, Hu12, Hetar, Pradeep.esg, Equendil, Keerthi msrit, Kozuch, Thijs!bot, Kubanczyk, Gerry Ashton, Dawnseeker2000, LachlanA, Stannered, Wiki0709, Raanoo, Mthermond86, Tedickey, Seashorewiki, Sajupa, Anantbhasu, Andre.holzner, RobinSummerhill, Whitethunder79, Fmltavares, Wbrito, Jnlin, Oshwah, Tim.daniels, Shovan3000, Duncan.Hull, AlleborgoBot, Struway, Zemoxian, Xelgen, Umar 3010, EnOreg, Casocco, Jan1nad, Nnemo, SuperHamster, Tbuechner, Faramarz.M, DanielAgorander, Salam32, AndrewScully, Pllevin, Addbot, Avi Ravner, Henridv, Hoenny, NicoleYu, Margin1522, Legobot, Luckas-bot, Yobot, TaBOT-zerem, KamikazeBot, Mohitjain1983, AnomieBOT, Momoricks, DemocraticLuntz, Hairhorn, Edward Nardella, GB fan, Xqbot, TheAMmollusc, Ceramic catfish, Brunonar, Bryawn, Savannah Kaylee, Tak'n, Jandalhandler, Dinamik-bot, PleaseStand, EmausBot, WikitanvirBot, BillyPreset, ZéroBot, Palosirkka, Jmsbush, ChuispastonBot, ClueBot NG, EnergyReporter, Powersjcb, Helpful Pixie Bot, 32alpha4tango, Privatechef, Wbm1058, BG19bot, Bshope, Thirdteam23423, Hans Haase, Dietcoke3.14, Stefek99, Rovingbite, DarafshBot, Arcandam, Kanai L Ghosh, Dexbot, KernelPone, UNOwenNYC, Comp.arch, JaconaFrere, Brettplease, BladeX1234, ClassicOnAStick, SageGreenRider, Suzana7 and Anonymous: 149
- **Microcontroller** *Source:* https://en.wikipedia.org/wiki/Microcontroller?oldid=706774699 *Contributors:* AxelBoldt, Derek Ross, Pierre-Abbat, Ben-Zin~enwiki, Maury Markowitz, Heron, Kwertii, Mahjongg, Mbessey, CesarB, Egil, Stan Shebs, Mac, Docu, Theresa knott, Glenn, Jonik, GRAHAMUK, JidGom, Omegatron, Wernher, Jni, Robbot, Izx, Chris 73, Hemanshu, Roscoe x, Giftlite, Andries, DavidCary, Cathy Linton, Solipsist, Matt Crypto, Bobblewik, Edderly, Imroy, CALR, Discospinster, Rich Farmbrough, Rhobite, Gejigeji~enwiki, Adam850, WikiPediaAid, Dyl, ESkog, Calamarain, CanisRufus, Femto, Bobo192, Netaimaid, Cje~enwiki, Shenme, R. S. Shaw, Matt Britt, Foobaz, Colin Douglas Howell, Cheung1303, Hooperbloob, Alansohn, Atlant, Corwin8, Nasukaren, Pion, Wtmitchell, Velella, Wtshymanski, Mlutfi, Cburnett, Suruena, RainbowOfLight, Voxadam, Bookandcoffee, Ceyockey, Mindmatrix, Hdante, Palica, Ketiltrout, Rjwilmsi, Seidenstud, Zbxgscqf, Jasoneth, FlaBot, Heycam, Margosbot~enwiki, RexNL, Ewlyahoocom, Lmatt, Jidan, Krishnavedala, Maluc, YurikBot, Wavelength, Spacepotato, RussBot, Stephenb, Gaius Cornelius, Rsrikanth05, TheMandarin, Ben b, Speedevil, Voidxor, Bota47, Scope creep, Microco~enwiki, LanguidMandala, Tevildo, Allens, Ianhopkins, GrinBot~enwiki, Benhoyt, Attilios, Yakudza, Joshbuddy, SmackBot, Sagie, Reedy, Maelwys, Henriok, Unyoyega, Davewild, Clpo13, InTheCastle~enwiki, Sloman, Gilliam, Lindosland, PATCheS, Chris the speller, DStoykov, TheParallax, CSWarren, ZyMOS, DHN-bot~enwiki, A. B., Hobiadami, Furby100, Jon-Harder, LouScheffer, Addshore, Allan McInnes, Easwarno1, Radagast83, Hgilbert, Yrogerg, Doodle77, Wizardman, Nishkid64, Kuru, Steve1608, The.happy.hippy, Petter73, Chrisch, CyrilB, Dicklyon, PeterJohnBishop, Jouthus, Lbokma, StephenBuxton, UncleDouggie, Drvanthorp, Menswear, Az1568, Tawkerbot2, Electron20, Mikiemike, Anon user, Ilikefood, Pfagerburg~enwiki, Nczempin, JPats, Timichal, HenkeB, Cbmeeks, AndrewHowse, Mblumber, Mato, Vwollan, Travelbird, Kozuch, Mtpaley, Gcalac, Thijs!bot, Tom d perkins, Electron9, Turkeyphant, Brewsum, Ebde, AntiVandalBot, Majorly, Guy Macon, QuiteUnusual, Quintote, Lovibond, Obsessiveatbest, Deadbeef, JAnDbot, Viskr, CosineKitty, RastaKins, Jahoe, LittleOldMe, Magioladitis, Bongwarrior, VoABot II, SoccerCore11, M 3bdelqader, Zedomax, ANONYMOUS COWARD0xC0DE, Kerdip, Kphowell, Calltech, Gwern, MartinBot, Moggie2002, GoldenMeadows, PJohnson, Axlq, Bogey97, Jiuguang Wang, Cpiral, Qratman, NewEnglandYankee, Stevemcl10, Cometstyles, Diwakarbista, VolkovBot, ICE77, Kevinkor2, AlnoktaBOT, Philip Trueman, Pfrulas, TXiKiBoT, Sarenne, Gcarter68, Timmh, Andy Dingley, Usman365, Falcon8765, Neoflame, SieBot, Scarian, BotMultichill, GlassCobra, Reinderien, Lisatwo, Lightmouse, Frappucino, A Sengupta, Asher196, ClueBot, GorillaWarfare, Snigbrook, The Thing That Should Not Be, Cambrasa, Rilak, Hysocc, Kiu77, Blanchardb, Na7id, Kanishafa, DragonBot, Foadsabori, Excirial, Anonymous101, Abrech, Alikian, Andreas Groß, Chaosdruid, C628, SoxBot III, Ginbot86, DumZiBoT, Joel Saks, XLinkBot, Jmont1, Parallelized, Skarebo, Suresh1686, Zodon, Nikhilb239, Otisjimmy1, MrOllie, Download, Glass Sword, Chzz, OlEnglish, Vanuan, Tartarus, Luckas-bot, Yobot, Ptbotgourou, Fraggle81, DJ LoPaTa, Jmicko, Thaiio, Clara Anthony, AnomieBOT, KDS4444, 1exec1, Jim1138, Kingpin13, Kushagraalankar, Materialscientist, Thisara.d.m, Rmaax, Juhuang, Xqbot, Capricorn42, Oborg, Victor Waiman, J04n, Geekstuff, Locobot, Secondwatch, Madison Alex, Aspwiki, FrescoBot, D'ohBot, Ionutzmovie, Fuadamjala, A8UDI, NKanngaz, Mikespedia, Gapaddict, TobeBot, Codinghead, Fox Wilson, Vrenator, Zvn, Ravikumar001, RjwilmsiBot, Dhiraj1984, PoPCulture69, Dead Horsey, Alex3yoyo, Winner 42, K6ka, Langermannia, John Cline, Fæ, Russell ti, A930913, Malikqasim, Willem7, Erianna, Sbmeirow, Coasterlover1994, Donner60, Whiteneon, Davewave88, Mcunerd, ClueBot NG, MSCI in space, Aleksandarbrain, Techeditor1, Kubing, Amith deshpande, Widr, Danim, Iswantoumy, Calabe1992, Microheat, BG19bot, Bmusician, Barewires, Wiki13, Akkazemi, Eagle c5, Snow Blizzard, Pratyya Ghosh, ChrisGualtieri, Dhx1, Usmanmustafa4u786, Gnetscher, Mogism, Rashid Mehmood Khokhar, Vahid alpha, Kyohyi, David CMT, Semonti, Comp.arch, Ugog Nizdast, Sam Sailor, Inaaaa, Withnoe, Div2005, Ali.Zeineddine93, Lagoset, TerryAlex, Engr Wasim Khan, Pramod Sahu957, Jfolkens, KasparBot, Srednuas Lenoroc, Jomonthomasnew, Crtrtrc, CLCStudent, Ramarokie60 and Anonymous: 519
- **Microprocessor** *Source:* https://en.wikipedia.org/wiki/Microprocessor?oldid=707374019 *Contributors:* WojPob, Mav, Zundark, Berek, Scipius, Aldie, Nate Silva, Roadrunner, Ray Van De Walker, SimonP, Hannes Hirzel, Maury Markowitz, Olivier, Ram-Man, Spiff~enwiki, Edward, Oystein, Michael Hardy, Mahjongg, Ixfd64, Komap, Dcljr, Seav, Minesweeper, Ahoerstemeier, Stan Shebs, Mac, Theresa knott, Jdforrester, Glenn, LouI, Netsnipe, HPA, Mxn, Crusadeonilliteracy, Rainer Wasserfuhr~enwiki, Vladmihaisima, Andrewman327, Greenrd, IceKarma, TEG24601, Tpbradbury, Grendelkhan, Wernher, Raul654, Frazzydee, AnthonyQBachler, Twang, Robbot, Murray Langton, RedWolf, Donreed, Alain~enwiki, Tim Ivorson, Pingveno, Yarvin, Blainster, Jondel, IOOI, Hadal, UtherSRG, Jason Michael Smithson, Tobias Bergemann, David Gerard, Ancheta Wis, Alf Boggis, Giftlite, Brouhaha, DavidCary, Mintleaf~enwiki, Nadavspi, Inkling, Ferkelparade, Ds13, Jonabbey, Archenzo, VampWillow, Mckaysalisbury, Bobblewik, Toytoy, MarkSweep, Jossi, MFNickster, Beerden, Jklamo, Kmweber, Oknazevad, Fermion, Moxfyre, Grunt, Mikef, Junexgamer, Gachet, Neckelmann, Rich Farmbrough, Milkmandan, Pmsyyz, Jpk, Pixel8, Mjpieters, Mani1, Goochelaar, Horsten, Bender235, Andrejj, Violetriga, CanisRufus, Phil websurfer@yahoo.com, Sietse Snel, Djd1219, Warpozio, Fir0002, Smalljim, Elipongo, Apyule, Matt Britt, Kjkolb, Kundor, Idleguy, Haham hanuka, Hooperbloob, Alansohn, Gary, Liao, Guy Harris, Snowolf, Angelic Wraith, Wtshymanski, RainbowOfLight, Fyngyrz, Rjhanson54, Axeman89, Bookandcoffee, DanielVonEhren, Conskeptical, Sylvain Mielot, Woohookitty, Poppafuze, Uncle G, Scootey, Crazysunshine, Wayward, Rmadams, Toussaint, Mandarax, Ilya, BD2412, Kbdank71, Island, Reisio, Search4Lancer, Zbxgscqf, Jake Wartenberg, Kinu, Arisa, Bubba73, Yamamoto Ichiro, FlaBot, Mirror Vax, SchuminWeb, RobertG, Windchaser, Arnero, Margosbot~enwiki,

- **Digital signal processor** *Source:* https://en.wikipedia.org/wiki/Digital_signal_processor?oldid=707221268 *Contributors:* Mav, SimonP, Maury Markowitz, Michael Hardy, Nixdorf, Nickrusnov, Oyd11, Minesweeper, Typhoon, Sphl~enwiki, Whkoh, Tristanb, Jrousseau, Dysprosia, Mrand, Wernher, Bevo, Topbanana, Murray Langton, RedWolf, Jondel, SpellBott, Giftlite, DavidCary, Mcapdevila, Chowbok, Pyle, Lockeownzj00, Sam Hocevar, Biot, Abdull, Flex, Alistair1978, Harriv, Violetriga, Matt Britt, Hooperbloob, Guy Harris, Cburnett, Suruena, GOKULAK, Nuno Tavares, Ruud Koot, Jeff3000, AlbertCahalan~enwiki, Meneth, MarkusHagenlocher, Toussaint, Palica, Zephyrxero, Kbdank71, Ketiltrout, Joe Decker, Alll~enwiki, Maxim Razin, Arnero, Webshared, Jidan, Chobot, Bgwhite, YurikBot, Borgx, Dmccarty, Toffile, Gaius Cornelius, TheMandarin, Darksidex, Kkmurray, Ninly, LeonardoRob0t, Curpsbot-unicodify, RichardYoung, Henriok, DomQ, Ohnoitsjamie, Oli Filth, Firetrap9254, Zvar, A5b, The undertow, Db1145, JimisDose, 16@r, Saerdnaer, Dicklyon, TeryKing, Kvng, Martin Kozák, Requestion, AstroPig7, Thijs!bot, WillMak050389, Electron9, Hcobb, Jauricchio, Jamesgor13579, Cevadsp, JAnDbot, Jahoe, Rivertorch, Soulbot, Yewyew66, GermanX, Jvienneau, Glrx, RockMFR, Jcurie, Harobikes34, STBotD, RobOnKnowledge, PNG crusade bot, Ultim, Starrymessenger, JhsBot, Broadbot, Andy Dingley, Haseo9999, SieBot, Sonicology, AlphaPyro, Jerryobject, Travelingseth, GAMER 20999, Dspanalyst, OSP Editor, Martarius, ClueBot, Rilak, Niceguyedc, DragonBot, Alexbot, LJanardhan, Jotterbot, Sebastien.m, GlasGhost, Pantech solutions, Johnuniq, Semitransgenic, Skarebo, Dsimic, Addbot, Mortense, Anschel, MrVanBot, Lightbot, Luckas-bot, Themfromspace, Lehuma, AnomieBOT, Adeliine, Thisara.d.m, Witguiota, Dspmandavid, RibotBOT, Kyng, Insomnia64, Prari, FrescoBot, BenzolBot, Jonathandeamer, Jschnur, RedBot, Serols, SpaceFlight89, Overjive, RjwilmsiBot, Bento00, 10 goal payne, DSP-user, EmausBot, Orphan Wiki, Sumudufdo, GoingBatty, Nrobbo, Mulletsrokkify, 28bot, ClueBot NG, Jaanus.kalde, Matarasephotos, BG19bot, Minsbot, Wikpoint, ChrisGualtieri, Tagremover, Poolborges, Frosty, Nutaq, Pkunk, Spencer.mccormick, Melonkelon, Yardimsever, ArmbrustBot, YiFeiBot, ScotXW, KasparBot and Anonymous: 186
- **Embedded system** *Source:* https://en.wikipedia.org/wiki/Embedded_system?oldid=707934571 *Contributors:* Damian Yerrick, Bryan Derksen, Jeronimo, Ap, William Avery, Rgvandewalker, Ray Van De Walker, SimonP, TomCerul, Edward, Patrick, RTC, Nixdorf, Kku, 7265, CesarB, Haakon, Mac, Nanshu, Elano, Mark Foskey, Glenn, Cyan, Ghewgill, GRAHAMUK, Gamma~enwiki, Wikiborg, Magnus.de, Tanolin, Maximus Rex, David Shay, Wernher, Joy, MD87, Robbot, Robin1225, Fredrik, Sanders muc, Netizen, Stewartadcock, Jondel, Hadal, Wikibot, Diberri, Tobias Bergemann, Giftlite, DavidCary, Akadruid, Kenny sh, Lupin, No Guru, Frencheigh, Maroux, Warlok42, VampWillow, Madoka, Wmspringer, Alexf, Tim Pritlove, Jimwilliams57, Ojw, Abdull, Zondor, Discospinster, Rich Farmbrough, Rhobite, Sbb, Mjpieters, Mani1, Harriv, Bender235, Limbo socrates, CanisRufus, El C, Walden, Sietse Snel, Femto, Smalljim, AllyUnion, Elipongo, Matt Britt, Dungodung, Kjkolb, Towel401, Helix84, (aeropagitica), Pearle, SeanGustafson, Mpeisenbr, Michael Drüing, Musiphil, Mrzaius, Guy Harris, Arthena, Conan, Atlant, Andrewpmk, Krischik, Corwin8, RoySmith, Wdfarmer, Wtshymanski, Cburnett, Suruena, Sciurinæ, Ceyockey, Forderud, Bruce89, Oleg Alexandrov, Woohookitty, TheNightFly, Millard73, MONGO, Ahazred8, Toussaint, Essjay, Gerbrant, GSlicer, Mandarax, Graham87, Kbdank71, Jclemens, Jorunn, Rjwilmsi, Benzamin, Vegaswikian, ElKevbo, Alll~enwiki, Brighterorange,

The wub, KageMonkey, Sumanch, SchuminWeb, Margosbot~enwiki, Mcleodm, Mathiastck, Alvin-cs, Chobot, ShadowHntr, Flcelloguy, Touseefliaqat, YurikBot, RobotE, Amnonc, Fabartus, Chris Capoccia, Polluxian, Shell Kinney, Schoen, Cpuwhiz11, TheMandarin, Wiki alf, InvaderJim42, Voidxor, Eclipsed, Fender123, Bota47, Oliverdl, Fadyfadl6, Zelikazi, Mike92591, Navstar, FF2010, Zzuuzz, Ninly, Closedmouth, Dspradau, Mike1024, Wbrameld, Dpotop, Curpsbot-unicodify, Attilios, SmackBot, Redslime, Mmernex, Bobet, McGeddon, Gary Kirk, InTheCastle~enwiki, Gilliam, Richfife, Jcarroll, Lakshmin, Rmosler2100, Chris the speller, Bluebot, Kurykh, Aidan Croft, GK tramrunner, EncMstr, DHN-bot~enwiki, Tommy1808, Simpsons contributor, Rrelf, Can't sleep, clown will eat me, JonHarder, Yorick8080, Zazpot, Zvar, Allan McInnes, SundarBot, BullRangifer, Luís Felipe Braga, Kvprav, Aaron Lawrence, Gennaro Prota, ArglebargleIV, Harryboyles, Soumyasch, Igor Markov, Linnell, JorisvS, NongBot~enwiki, Ckatz, Someguyonearth, SQGibbon, Dicklyon, Aeroniphus, Kvng, Hu12, Iridescent, Tawkerbot2, Mikiemike, CmdrObot, Amalas, Ilikefood, JPats, NickW557, HenkeB, Darren10000, Gregbard, Djg2006, Gogo Dodo, Morticae, DumbBOT, Poonacha, Kozuch, Lbertybell, Omicronpersei8, Pumpkin Pi, RickClements, Thijs!bot, Qwyrxian, Al Lemos, Natalie Erin, SubaruSVX, Ebde, AntiVandalBot, Gioto, Saimhe, Guy Macon, Didgeweb, Malcolm, Tyler Frederick, Myanw, Acrosser, MER-C, QuantumEngineer, IanOsgood, Albany NY, Jheiv, PhilKnight, Jahoe, Abu ali, Magioladitis, Vorenus, VoABot II, Donnay, Britton ohl, BigJouly, JamesBWatson, Shanerobinson, Tedickey, AMcKay, ForthOK, Americanhero, Japo, Cpl Syx, Highcount, Glen, Kerdip, Olivier Dupont, Oicumayberight, DGG, Grand Am, Vigyani, Microsp, R'n'B, CommonsDelinker, Commsserver, HEL, Worldedixor, J.delanoy, Raistlin11325, Lhbts~enwiki, Kottkrig, Katalaveno, Peterkimrowe, McSly, Chrabieh, NewEnglandYankee, Embedian, Gaussgauss, Samli1018, DavidCBryant, Thulasikce, Idioma-bot, Mistercupcake, ABF, ICE77, Jeff G., AlnoktaBOT, Philip Trueman, Entilsar~enwiki, DoorsAjar, Oshwah, Abbas ahmed, Gregfadein, Vipinhari, Hqb, Jozue, EM1SS&CSE, Maxim, SpecMode, Manik762007, Anderssjosvard, PlayStation 69, Billinghurst, Haseo9999, Falcon8765, Seraphiel, WereSpielChequers, Da Joe, Yintan, Askild, SmallRepair, Lightmouse, Vanished user kijsdion3i4jf, Emesee, Frappucino, Shooke, DoctorWhat2Know, Maralia, Mr. Stradivarius, Echo95, ClueBot, The Thing That Should Not Be, Petersburg, Rilak, Ignorance is strength, Razimantv, SuperHamster, Niceguyedc, OccamzRazor, Jameblo, Crazyman444, DhanushSKB, Mimosis, Muhandes, Estirabot, Sun Creator, Peter.C, 7, Versus22, SoxBot III, Michael.James.Daniel, HumphreyW, Vanished user uih38riiw4hjlsd, Expertjohn, DumZiBoT, Joel Saks, Smart boy abhishek, Caporaletti, XLinkBot, Cats AND hats, Stickee, Rror, Mk cr, Avoided, WikHead, Maxell 555, Nickbao, Dsimic, Addbot, Ghettoblaster, Crazy Ivan, Anorthup, MrOllie, LaaknorBot, Pmod, Favonian, AlexRWilson, Shwr, Zorrobot, Jarble, Sebenezer, Legobot, Luckas-bot, Yobot, Alexdemagalhaes, Embeddedsystemnews, Themfromspace, OrgasGirl, Bunnyhop11, MDuo13, Crispmuncher, Mirosamek, Stanza28, Linket, THEN WHO WAS PHONE?, AnomieBOT, Galoubet, Piano non troppo, AdjustShift, Law, Darolew, Materialscientist, Sunrise 87, Bseemba, Thisara.d.m, Rmaax, Renoj, GB fan, ArthurBot, RealityApologist, Xqbot, Capricorn42, Technopedian, Makeswell, SkiAustria, Majjigapumounika, Bryawn, MariaSantella, Shadowjams, Methcub, Savannah Kaylee, PM800, SD5, Steve Farnell, Bob Flintoff, Prari, FrescoBot, Voxii, Rackmount-guy, DrilBot, Pinethicket, I dream of horses, Martinstephene, A8UDI, Wikitanvir, SpaceFlight89, Zhuzheng, Jujutacular, Hybricon2009, Taylor2646, ItsZippy, Callanecc, Vrenator, Clarkcj12, Ravikumar001, Sideways713, Njbsankar, DARTH SIDIOUS 2, DigitalPowerExpert, NortuRE, RjwilmsiBot, DRAGON BOOSTER, चंद्रकांत धुतडमल, Socialjest404, Identime, EmausBot, John of Reading, Donnilad93, Dhishnawiki, Akjar13, Dewritech, Primefac, Bhas purk, Tommy2010, !ComputerAlert!, Wikipelli, K6ka, Systemsat69, Russell ti, Kiwi128, H3llBot, Wayne Slam, Sbmeirow, Donner60, Zlatan8621, JamesJeffersHunt, Pravin kumar0611, Noahmckinnon, Ficusreligiosa, JustinC474, Top Jim, Chicken.123nugget, Venkatkrish02, ClueBot NG, Jaanus.kalde, Matthiaspaul, Frietjes, Widr, Chjb, Helpful Pixie Bot, BG19bot, Borgopio, Nepster6705, 1335 HaXx0r, EmadIV, AdventurousSquirrel, Spence29, Comfr, Ssankar23, Mdann52, FoCuSandLeArN, Bhbuehler, Karl Magnus, Sfranson, Pippab3, Jamesx12345, The Anonymouse, Me, Myself, and I are Here, Passengerpigeon, Phamnhatkhanh, Epicgenius, Enock4seth, David CMT, Tentinator, Michel Dover, AnthonyJ Lock, Oophs, LethalDose51, X y Z s p, 296.x, Adhfuedgkseigjseritgs, Chloewhittingham, George8211, Jianhui67, Manul, Inaaaa, Warnerjhonson, ScotXW, JaconaFrere, Melcous, Monkbot, Stefenev, Aravindreddy203, Sawdust Restaurant, Sudheerelktronics, Himanshu Bhargava, Siddhu09, Kjerish, Sujithmat, Vedanga Kumar, Newwikieditor678, Kaldquez, Tuntunwin8683, Anselm94, KasparBot, FSFHM, Embeddedtechcon, Leonardoaraujo.santos, Jegan2510, MUTHUKUMAR SRINIVASAN, MRTHK and Anonymous: 737

- **MPSoC** *Source:* https://en.wikipedia.org/wiki/MPSoC?oldid=669925767 *Contributors:* Jdiemer, Malcolma, Pegship, Alaibot, Vjardin, Rilak, Addbot, I dream of horses, Gf uip, ZéroBot, Dameunadebravas, Jmsbush, Arcandam, John.fernadez and Anonymous: 14
- **System in package** *Source:* https://en.wikipedia.org/wiki/System_in_package?oldid=689827797 *Contributors:* Omegatron, Jpo, DavidCary, Abdull, Twisp, Alex.g, Intgr, Jared Preston, Bgwhite, TexasAndroid, Jpbowen, Mikeblas, Tevildo, SmackBot, Radagast83, PeterJohnBishop, Tawkerbot4, Underpants, Thijs!bot, Kubanczyk, Escarbot, Magioladitis, Cspan64, Warut, Whitethunder79, Rei-bot, The Seventh Taylor, Rilak, Yi Chen Chen, Addbot, Twirligig, Vickychen12345, Dileepand, Kevjonesin, EdoBot, TechGeek70, Privatechef, Dark Silver Crow, Nadavami, Comp.arch, Microfab guy, Aytk and Anonymous: 21
- **Universal Synchronous/Asynchronous Receiver/Transmitter** *Source:* https://en.wikipedia.org/wiki/Universal_Synchronous/Asynchronous_Receiver/Transmitter?oldid=703784344 *Contributors:* Boco XLVII, RussBot, Boggsa, Jesse Viviano, Matt B., Qu3a, PolyTekPatrick, Yobot, AnomieBOT, Raven Onthill and Anonymous: 2
- **Serial Peripheral Interface Bus** *Source:* https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus?oldid=704779254 *Contributors:* Damian Yerrick, Ray Van De Walker, Heron, Cyp, William M. Connolley, Glenn, HPA, Colin Marquardt, Darkhorse, Blades, Giftlite, Ds13, Gadfium, Kiteinthewind, Togo~enwiki, Elektron, Sam, Vijaykumar~enwiki, Snuffkin~enwiki, Imroy, Ralph Corderoy, Thomas Willerich, Plugwash, Mondalaci, Kwamikagami, Southen, Sietse Snel, Zoggie50, Polluks, Vapier, Kundor, Sebastian Goll, Hopp, Jlassoff, Tauwasser, Wtshymanski, Cburnett, Tocksin, David Haslam, Macaddct1984, Alecv, Rjwilmsi, Tizio, Allen Moore, FlaBot, Chobot, YurikBot, John2kx, Shaddack, Rsrikanth05, Marq Kole, Voidxor, Museo8bits, HereToHelp, SmackBot, Larry Doolittle, Ankitkankane, Pieleric, Toddintr, Oli Filth, Letdorf, Tsca.bot, Plasma16, Frap, Jinxed, Chlewbot, Vegard, Mosca, Cybercobra, Doodle77, ManiacK, Kvng, Vanisaac, Mamanakis, CmdrObot, Jesse Viviano, Rmallins, Garrickk, Libro0, Christian75, Mtpaley, Zalgo, Joel woodward, Djmdjm, Alex Forencich, Jtmoon, Widefox, Ebikeguy, Spencer, Alphachimpbot, Xoneca, Michael Stangeland, Japo, Kgfleischmann, Omnara, 99th Percentile, R'n'B, CommonsDelinker, Javawizard, Jreybert, Inwind, Scls19fr, Daltuna, Amikake3, Kunjan patel, Senarvi, Melsaran, Logan, Kbrose, SieBot, Ceson~enwiki, Cashannon, GeiwTeol, Crm123, EnOreg, Nskillen, Agunther, ClueBot, Nsk92, Iandiver, Sylvain Leroux, PixelBot, Amolhshah, Ginbot86, XLinkBot, Tonypdmtr, Dgtsyb, Broke Back Records, Addbot, Mortense, Tothwolf, MrOllie, Semiwiki, Crspybits, Tide rolls, OlEnglish, Luckas-bot, Yobot, Jinlei, Julia W, Jordsan, Alhaiksandher, AnomieBOT, Adeliine, Materialscientist, Simonjohndoherty, Xqbot, Yenz820, Nasa-verve, GrouchoBot, RibotBOT, DaleDe, Jcmcclurg, FrescoBot, LucienBOT, Username20090319, Ionutzmovie, Vishnu2011, Moggie100, Teuxe, Jusses2, MastiBot, RCelistrinoTeixeira, Ericbwiki, Hoptroff, Jnmbk, HDLfriki, Cwavnew, Cp82, TheMesquito, Dhiraj1984, EmausBot, Dewritech, GoingBatty, Matthieu CASTET, Kalin.KOZHUHAROV, Encijia, Sbmeirow, Tronixstuff, JohnBoxall, Edgar.bonet, Li Zaodie, Fftzyy, ClueBot NG, Matthiaspaul, O.Koslowski, Widr, Viswanathamk, Helpful Pixie Bot, Wbm1058, BG19bot, Charon77, Hz.tiang, IveGoneAway, Kendall-K1, Wikih101, Simon naylor, M.A.Elfar, Tonusamuel, Markuj7, ChrisGualtieri, Jzj1993, Makecat-bot, Frosty, Currystomper, Faizan, Mohan.PAKALAPATI, 1nafar, PirateKing42, Buntybhai, Wasquewhat, Knivd, Bbrice86, ESPI ROCKS, Neuliss, Bafeigum, Jegan2510, Flyxtop, RunnerRick08 and Anonymous: 336

- **Analog-to-digital converter** *Source:* https://en.wikipedia.org/wiki/Analog-to-digital_converter?oldid=702882001 *Contributors:* Damian Yerrick, Derek Ross, Tbackstr, Jkominek, Ellmist, Heron, Michael Hardy, Ixfd64, Ahoerstemeier, Stevenj, Snoyes, Andres, Tristanb, GRAHAMUK, Charles Matthews, Jukeboksi, Jitse Niesen, Colin Marquardt, EthanL, Omegatron, Wernher, Thue, Altenmann, Ojigiri~enwiki, Roscoe x, Hadal, Wikibot, Jleedev, Cutler, Giftlite, DavidCary, Wolfkeeper, BenFrantzDale, Everyking, Micru, Nayuki, Ablewisuk, Hellisp, Sonett72, BrianWilloughby, DJS~enwiki, Chepry, Talkstosocks, Dufekin, Guanabot, Pjacobi, Mecanismo, Kooo, Evice, STHayden, CanisRufus, Nwerneck, Meggar, Cmdrjameson, Johnteslade, Klo~enwiki, Hooperbloob, Nsaa, Alansohn, Sbeath, BernardH, Wtshymanski, Sciurinæ, Crystalllized, Gene Nygaard, Feezo, Davidkazuhiro, Csk, Jeff3000, Male1979, Tslocum, BD2412, Haikupoet, Pleiotrop3, Cat5nap, Alll~enwiki, Arnero, Margosbot~enwiki, Mcleodm, Lmatt, Goudzovski, Alvin-cs, Chobot, Bgwhite, Adoniscik, UkPaolo, YurikBot, RobotE, Shawn81, CambridgeBayWeather, Yyy, Shaddack, NawlinWiki, Jaxl, Erislover, Welsh, Neil.steiner, Nick C, Bota47, Brisvegas, Zipcube, David Underdown, LeonardoRob0t, Xorx, Garion96, GrinBot~enwiki, Yoshm, SmackBot, KnowledgeOfSelf, Eskimbot, Commander Keane bot, Gilliam, Chris the speller, Bluebot, Keegan, Oli Filth, EncMstr, Jerome Charles Potts, Adpete, Nbarth, Bob K, Southcaltree, RProgrammer, KaiserbBot, JonHarder, Rrburke, Stigwall, Romanski, Scientizzle, Saxbryn, Kvng, Hu12, Chetvorno, JohnTechnologist, CmdrObot, Pegasusbot, Requestion, Wsmarz, FakingNovember, Mblumber, Ring0, Thijs!bot, D4g0thur, Mbell, Kaleem str, Teh tennisman, Electron9, Nemilar, Legend Saber, AntiVandalBot, Whiteflye, JAnDbot, MER-C, Jheiv, GurchBot, SapnaArun, Jahoe, LittleOldMe, A4, WODUP, 28421u2232nfenfcenc, Americanhero, Tjwikipedia, Theslaw, Sicaspi, MartinBot, Scottr9, Nelbs, Glrx, J.delanoy, Ginsengbomb, Cullen kasunic, 4johnny, Krishfeelmylove, LordAnubisBOT, Iverson2, Coppertwig, Fmltavares, VoidLurker, VolkovBot, ICE77, Oshwah, Qllach, Peteraisher, Schickaneder, Crohnie, LeaveSleaves, VanishedUserABC, Purgatory Fubar, Spinningspark, Brianga, Biscuittin, SieBot, Bhimaji, Msadaghd, Josecampos, MinorContributor, Reinderien, Cindy141, Travelingseth, Steven Crossin, Sfan00 IMG, ClueBot, Binksternet, Jan1nad, Sandpiper800, Night Goblin, Timrprobocom, Nrpickle, Ivnryn, Jusdafax, ChardonnayNimeque, Wiki libs, Zootboy, DumZiBoT, Theo177, Analogkidr, XLinkBot, Vayalir, Stickee, Blowfishie, Ali Esfandiari, SilvonenBot, MystBot, Addbot, Mortense, Fgnievinski, GyroMagician, MrOllie, Redheylin, Dayewalker, Lightbot, Perry chesterfield, Pietrow, Legobot, Drpickem, Yobot, Themfromspace, KamikazeBot, AnomieBOT, MR7526, Materialscientist, DeadTotoro, Overmanic, LilHelpa, Xqbot, TheAMmollusc, TinucherianBot II, Shalabh24, Heddmj, J04n, GrouchoBot, SassoBot, Alvin Seville, Wikihitech, Shadowjams, Thehelpfulbot, FrescoBot, Febert, Smurfettekla, Berrinkursun, Gbalasandeep, Anitauky, Nacho Insular, ClickRick, Davidmeo, RobinK, Sjbfalchion, FoxBot, TecABC, Overjive, Arnab1984, Dhiraj1984, Alison22, EmausBot, Solarra, Satan131984, Dcirovic, Samurai meatwad, Zueignung, Puffin, Christian way, Pavan206, Wakebrdkid, Teapeat, Ttmartin2888, ClueBot NG, CocuBot, Techeditor1, AeroPsico, Helpful Pixie Bot, MusikAnimal, Bpromo7, Gayanmyte, Erynofwales, Windforce1989, Jimi75, Sparkie82, 08Peter15, Syl1729, ChrisGualtieri, AK456, Virrisho001, Radiodef, Binglau, EdSaWiki, Raka99, Michipedian, Gillsandeep2k, Rassom anatol, Hhm8, Delar303, Davo962, KasparBot, Gorplex and Anonymous: 402

- **Digital-to-analog converter** *Source:* https://en.wikipedia.org/wiki/Digital-to-analog_converter?oldid=701562596 *Contributors:* Damian Yerrick, The Anome, PierreAbbat, Waveguy, Mjb, Heron, Michael Hardy, TakuyaMurata, ZoeB, Glenn, GRAHAMUK, Bemoeial, Reddi, Zoicon5, Maximus Rex, Omegatron, Wernher, Thue, Topbanana, Robbot, Kizor, Ojigiri~enwiki, Lupo, Giftlite, Wolfkeeper, BenFrantzDale, Bradeos Graphon, Ssd, SWAdair, CryptoDerk, Stevenalex, Hellisp, Hugh Mason, BrianWilloughby, DJS~enwiki, Chmod007, Chepry, Imroy, TedPavlic, Paul August, Ht1848, CanisRufus, Meestaplu, Rbj, Johnteslade, Giraffedata, Photonique, Timecop, Hooperbloob, Alansohn, Jannev~enwiki, Cburnett, Algocu, Bookandcoffee, Kenyon, Bobrayner, Woohookitty, StradivariusTV, Jeff3000, Gradulov, CPES, Mandarax, BD2412, Joe Decker, FlaBot, Arnero, Mcleodm, Lsuff, Chobot, RobotE, Thane, Tharanath1981, Garion96, Yoshm, SmackBot, Fnfd, Tex23, Andy M. Wang, EncMstr, Southcaltree, RProgrammer, JonHarder, Adamantios, Sturm, UVnet, Kvng, Lee Carre, Nczempin, HenkeB, Dept of Alchemy, Thijs!bot, Ecclaim, AntiVandalBot, Salgueiro~enwiki, Father Goose, Rivertorch, Soulbot, Calltech, Sicaspi, MartinBot, Anaxial, Glrx, Abuthayar, R'n'B, Nono64, J.delanoy, Boodidha sampath, Ontarioboy, VolkovBot, ICE77, TXiKiBoT, Neildmartin, Spinningspark, Mahira75249, Crm123, Masgatotkaca, Travelingseth, Binksternet, VQuakr, Pointillist, ChardonnayNimeque, Arjayay, Andrebragareis, DanteLectro, DumZiBoT, Analogkidr, XLinkBot, StormtrooperTK421, Addbot, Mortense, Xx521xx, Cst17, MrOllie, Redheylin, Semiwiki, Tide rolls, Lightbot, Legobot, Luckas-bot, Yobot, AnomieBOT, Materialscientist, Danno uk, Xqbot, TheAMmollusc, RibotBOT, Louperibot, HRoestBot, 10metreh, RedBot, RobinK, Overjive, Brainmedley, Suffusion of Yellow, Ravenmewtwo, Incminister, Karkat-H-NJITWILL, John Siau, Lexusuns, Chad.Farmer, Rpal143, Wsko.ko, ClueBot NG, Blitzmut, Chester Markel, Helpful Pixie Bot, Bpromo7, Josvanehv, Binglau, Camyoung54, Glaisher, ScotXW, Mohammadali Aghakhani, KasparBot, Idahoprogrammer and Anonymous: 202

- **Power management** *Source:* https://en.wikipedia.org/wiki/Power_management?oldid=707526110 *Contributors:* Nixdorf, Cjmnyc, Itai, Vaceituno, Raul654, Mrdice, Niteowlneils, Jrdioko, Beland, Mako098765, EagleOne, Martpol, Rodtrent, Plugwash, Sietse Snel, BalooUrsidae, TheParanoidOne, Voxadam, Mindmatrix, Tokek, Coneslayer, Intgr, Bgwhite, Simesa, RussBot, Stephenb, Grafen, AGToth, Eptin, SmackBot, McGeddon, Gilliam, Oli Filth, Frap, Verycharpie, Mellery, Cydebot, Gogo Dodo, Kozuch, Thijs!bot, Widefox, Jared Hunt, MER-C, Beagel, Oehlingb, Wikimike2007, PolarYukon, Mgmcginn, Johnuniq, Zodon, Muffinon, Addbot, Ramu50, Yobot, A.amitkumar, Verismic, Probinso99, Christoph hausner, EmausBot, Primefac, Matthiaspaul, MarkPeters45, BG19bot, Gkakkar88, Civeel, BattyBot, Greenstruck, Phamnhatkhanh, SJ Defender, Abhinandan tiwari2, Hlo u fang axaxaxas mlo, ClearBlueSky85, Newwikieditor678, Ljsingh, Jegan2510 and Anonymous: 29

- **Bus (computing)** *Source:* https://en.wikipedia.org/wiki/Bus_(computing)?oldid=701784335 *Contributors:* Uriyan, The Anome, Drj, Css, Dachshund, Shd~enwiki, Aldie, Fubar Obfusco, Deb, SimonP, Maury Markowitz, Heron, Olivier, Michael Hardy, Mahjongg, Nixdorf, Egil, ArnoLagrange, Mac, Nanshu, Glenn, Nikai, Rl, GRAHAMUK, CAkira, Reddi, Magnus.de, Colin Marquardt, Tschild, Saltine, Wernher, Thue, Veghead, Jni, Chuunen Baka, Robbot, MrJones, Fredrik, Scott McNay, RedWolf, Naddy, Merovingian, Hadal, Marc Venot, DavidCary, Kenny sh, Guanaco, Ezhiki, Tom-, Ferdinand Pienaar, AlistairMcMillan, VampWillow, Uzume, Sam Hocevar, Klemen Kocjancic, Mike Rosoft, Olki, Imroy, Slady, JTN, Discospinster, Guanabot, Bert490, Xezbeth, Mjpieters, Horsten, WegianWarrior, Johannes Rohr, Limbo socrates, Violetriga, Tooto, CanisRufus, R. S. Shaw, Brim, SpeedyGonsales, Tritium6, James Foster, MatthewWilcox, Civvi~enwiki, Hopp, Riana, Fritzpoll, Mailer diablo, Helixblue, Wtshymanski, Proton, Nuno Tavares, Woohookitty, Timharwoodx, Rocastelo, Hbdragon88, JRHorse, Isnow, Cyberman, Wayward, Graham87, Arunib, Kbdank71, Pako, FlaBot, SchuminWeb, Moreati, Numa, RexNL, Nimur, Revolving Bugbear, Intgr, CiaPan, Chobot, DVdm, Bgwhite, YurikBot, Fabartus, CambridgeBayWeather, Rsrikanth05, Pseudomonas, Cpuwhiz11, Jeword, ENeville, Nowa, Pagrashtak, Dijxtra, Maverick Leonhart, Nick, Ospalh, Samir, Scope creep, Wknight94, Johndburger, Phgao, Sean Whitton, Kevin, Curpsbot-unicodify, Tom Morris, Attilios, SmackBot, KelleyCook, Commander Keane bot, Gilliam, Kurykh, Jerome Charles Potts, Craig t moore, Fjmustak, Can't sleep, clown will eat me, Egsan Bacon, Frap, HarisM, Autodmc, A5b, Luigi.a.cruz, SashatoBot, SpareHeadOne, DHR, Mathias-S, Beetstra, Kvng, DabMachine, Iridescent, Ihatethetv, SkyWalker, Raysonho, Nhumfrey, Jesse Viviano, ShoobyD, ST47, Epbr123, Kubanczyk, Sobreira, I do not exist, Doyley, Yettie0711, Andrew sh, Mentifisto, AntiVandalBot, Wayiran, JAnDbot, NapoliRoma, BenB4, Acroterion, Bongwarrior, VoABot II, Wikidudeman, Becksguy, Twsx, Schily, Avicennasis, BrianGV, GermanX, Amazonite, AVRS, MartinBot, Dima373, Rpclod, J.delanoy, Cpiral, Pyams, McSly, Peskydan, Cometstyles, CompNerd11, Idioma-bot, Priceman86, VolkovBot, AndyLandy, Philip Trueman, Technopat, Hqb, Anna Lincoln, BotKung,

Natg 19, Legoktm, Cowlinator, EmxBot, Regregex, SieBot, Ham Pastrami, Bentogoa, Flyer22 Reborn, Oda Mari, Egrian, Lightmouse, ClueBot, GorillaWarfare, Rilak, No such user, Jusdafax, Sun Creator, Lunchscale, BOTarate, Jonverve, Redhill54, XLinkBot, Dsimic, Deineka, Addbot, GargoyleBot, CanadianLinuxUser, Graham.Fountain, CarsracBot, BepBot, Lightbot, OlEnglish, Mike88chan, Legobot, Hydrofiber, Luckas-bot, Yobot, OrgasGirl, Crispmuncher, Tuxraider reloaded, Mmxx, Nallimbot, IW.HG, Keithbob, Toko50, Materialscientist, Citation bot, TinucherianBot II, Capricorn42, Nasa-verve, Wearingaredhat, RibotBOT, FrescoBot, קשיו, Krj373, W Nowicki, Amaka555, Yahia.barie, RedBot, MastiBot, Serols, Trappist the monk, Boriss111, Jeffrd10, Cp82, DARTH SIDIOUS 2, RjwilmsiBot, BlakeD360, Midhart90, Brightbulb, EmausBot, John of Reading, Stryn, ValC, Wikipelli, Thecheesykid, Prof Karl, Surya Prakash.S.A., Donner60, Atrivo, 28bot, ClueBot NG, Gilderien, Firowkp, Widr, Oddbodz, Helpful Pixie Bot, Wbm1058, Snaevar-bot, Kokkkikumar, Maxxdxx, ChrisGualtieri, EE JRW, Zeeyanwiki, Lugia2453, Frosty, Greenstruck, Lsmll, LarryWiki2009, Ugog Nizdast, Doenymo, Jianhui67, Reddraggone9, Dmtech, 42315413ferq, Derped Monkey Fish, MonkeysEatSwag, Some Gadget Geek, Dghgdsh, KasparBot, 3 of Diamonds, ToaneeM, Saad elayas khan, Wiser87, Qzd and Anonymous: 310

- **Advanced Microcontroller Bus Architecture** *Source:* https://en.wikipedia.org/wiki/Advanced_Microcontroller_Bus_Architecture?oldid=707542217 *Contributors:* Nixdorf, Jake Nelson, Giftlite, Chowbok, Imroy, Rich Farmbrough, Anthony Appleyard, Kocio, Kenyon, Conskeptical, Xiaowen, Alecv, Ketiltrout, Rjwilmsi, Feydey, Intgr, Philpem, SmackBot, Henriok, Frap, Poposhka, Balrog, 2help, Mrpeteheller, TXiKiBoT, Cootiequits, Lightmouse, Mild Bill Hiccup, SchreiberBike, Airplaneman, Addbot, Lightbot, Softy, Yobot, Patrickyip, Materialscientist, Jeanmarc.ayotte, Thehelpfulbot, W Nowicki, Lissajous, AndyHe829, EmausBot, John of Reading, Orphan Wiki, Gabriele.svelto, Rohit7401, Bnmguy, Jionpedia, Gord207, NeriDavide, EvergreenFir, Muhammadumairzafar, Abhishekreddy kola, ConcurrencyTheory, Jegan2510 and Anonymous: 63
- **Direct memory access** *Source:* https://en.wikipedia.org/wiki/Direct_memory_access?oldid=706265084 *Contributors:* Bryan Derksen, Tarquin, Shd~enwiki, Fuzzynerd, Michael Hardy, Llywrch, Notheruser, Nikai, LordK, Furrykef, Pilaf~enwiki, David.Monniaux, Davidmaxwaterman, Catskul, Robbot, Liotier, Ninjamask, DocWatson42, Erpel~enwiki, Kenny sh, VampWillow, Uzume, Pgan002, Sjjung, Abdull, NightMonkey, Richie, Discospinster, Dolda2000, WegianWarrior, Bender235, PutzfetzenORG, CanisRufus, Dudboi, Smalljim, R. S. Shaw, SpeedyGonsales, Keenan Pepper, Wtmitchell, Wtshymanski, Suruena, Blaxthos, Kenyon, Timharwoodx, Rocastelo, Ae-a, Mangojuice, Isnow, Kesla, MassGalactusUniversum, Vary, Z-4195, LjL, Firebug, Arnero, RAMChYLD, Pathoschild, Intgr, Chobot, Arodrig6, YurikBot, RussBot, Thoreaulylazy, Wimt, PhilipO, Pnorcks, BOT-Superzerocool, Zzuuzz, SmackBot, InverseHypercube, KnowledgeOfSelf, Chronodm, Aksi great, Brianski, Rmosler2100, Thumperward, Lubos, Kostmo, OrphanBot, JonHarder, A5b, Paulish, SashatoBot, Augast15, Fedallah, Kvng, Tawkerbot2, Jesse Viviano, Xaariz, Keli666, Kubanczyk, W Hukriede, Electron9, Mentifisto, AntiVandalBot, Gioto, Widefox, Seaphoto, Mk*, JAnDbot, MER-C, Kipholbeck, Ferritecore, Michi.bo, Alleborgo, JamesR, Xonicx, Петър Петров, VolkovBot, RainierHa, TXiKiBoT, Oshwah, Wicher Minnaard, Ferengi, Chris.franson, EmxBot, SieBot, Galileo seven, Lightmouse, Astrale01, Hmmmmike, Denisarona, ClueBot, 718 Bot, DragonBot, Excirial, Alexbot, Socrates2008, Goodone121, Dekisugi, Muro Bot, Jonverve, Damianesteves, DumZiBoT, Dsimic, Addbot, Rjpryan, Eivindbot, Roux, SpBot, Forbidmario, Numbo3-bot, Legobot, Luckas-bot, Yobot, OrgasGirl, MDuo13, Eric-Wester, Magog the Ogre, Law, Materialscientist, ArthurBot, Richarddonkin, BenzolBot, MastiBot, Thái Nhi, Surendhar Murugan, Jfmantis, EmausBot, Dewritech, Racerx11, The Blade of the Northern Lights, Ergaurav.ce, Music Sorter, Sbmeirow, Carmichael, Bomazi, Snubcube, ITMaverix, Neil P. Quinn, LZ6387, ClueBot NG, Matthiaspaul, Frietjes, Rezabot, Widr, Helpful Pixie Bot, Pjjanani, BattyBot, Jimw338, The Illusive Man, Mediran, Gajalkar.ashish, Numbermaniac, Frosty, Blakeawesome10, ScotXW, Calebcox314, TzachiNoy and Anonymous: 218
- **Hardware verification language** *Source:* https://en.wikipedia.org/wiki/Hardware_verification_language?oldid=594025569 *Contributors:* Jpbowen, JLaTondre, Jpape, Cydebot, PierreCA22, Cristian Amitroaie, Eslchip, Galileo seven, Coyote83, Addbot, Mortense, Julia W, AnomieBOT, Locobot, AndyHe829, Gf uip, SFK2 and Anonymous: 2
- **Mixed-signal integrated circuit** *Source:* https://en.wikipedia.org/wiki/Mixed-signal_integrated_circuit?oldid=686028201 *Contributors:* Heron, Nixdorf, Charles Matthews, DavidCary, A2Kafir, Atlant, Wtshymanski, RyanGerbil10, Marasmusine, Woohookitty, Isnow, Toffile, SmackBot, Rwender, Oli Filth, RedHillian, Noahspurrier, Caiaffa, BananaFiend, JForget, Amalas, Phatom87, DumbBOT, EdJohnston, Ste4k, Lasai~enwiki, J.delanoy, Jakejuliebaker, ICE77, Indubitably, Synthebot, Spinningspark, Malcolmxl5, EldenCrom, Cyfal, Excirial, Jeeter07, Addbot, Yangandjiao, MrOllie, Yobot, AnomieBOT, GrouchoBot, Deepon, Wbm1058, Henrique.hirata, Vikochka, Kahtar, Kiranbehara, Ordercrazy, Marieke marieke, Teupdeg, Rickynevada, Amraut77 and Anonymous: 33
- **Radio frequency** *Source:* https://en.wikipedia.org/wiki/Radio_frequency?oldid=699649871 *Contributors:* WojPob, Zundark, The Anome, Fredbauder, Aldie, SimonP, Waveguy, Rcingham, Heron, Mintguy, Stevertigo, Kku, Prefect, Bogdangiusca, Palmpilot900, Wfeidt, Conti, Mulad, RadarCzar, Emperorbma, Reddi, Radiojon, Tero~enwiki, SEWilco, Jerzy, Denelson83, Robbot, Moriori, RedWolf, Arkuat, Stewartadcock, Blainster, Hadal, Danceswithzerglings, DocWatson42, Average Earthman, Fleminra, Cantus, Jfdwolff, Bobblewik, Utcursch, Beland, Ary29, Ukexpat, Klemen Kocjancic, Deglr6328, Jakro64, Discospinster, Guanabot, LindsayH, Harriv, Bender235, El C, PhilHibbs, Bobo192, Sparkgap, MARQUIS111, Haham hanuka, ClementSeveillac, Atlant, Wtmitchell, Wtshymanski, Cburnett, Suruena, DV8 2XL, Gene Nygaard, WojciechSwiderski~enwiki, Kenyon, Alex.g, Camw, Pol098, Plaws, Zilog Jones, Ch'marr, Isnow, MarkPos, Zpb52, LimoWreck, Kotukunui, Koavf, Misternuvistor, Mike Peel, Vegaswikian, Fred Bradstadt, Ground Zero, RexNL, Smileyrepublic, Alvin-cs, Srleffler, King of Hearts, Wavelength, Fabartus, Anonymous editor, Yyy, Giro720, Teb728, Wiki alf, Retired username, Mortein, Anetode, Hyandat, Voidxor, Shadowblade, Kermi3, User27091, Jules.LT, Aparna82, GraemeL, Alureiter, SmackBot, Rutja76, Aim Here, C.Fred, Tbonnie, Yamaguchi, Gilliam, Bluebot, Avin, EncMstr, Bazonka, Can't sleep, clown will eat me, Harumphy, Frap, JustUser, Adamantios, Khoikhoi, NoIdeaNick, A.R., Anlace, Stattouk, Jcoy, Dicklyon, Stijak, NetBMC, Dsongman, Chetvorno, JohnTechnologist, Daggerstab, Chrumps, Requestion, Cydebot, Kanags, The Ultimate Koopa, Tsenapathy, L7HOMAS, Jstuby, Epbr123, Headbomb, Rosarinagazo, John254, Dawnseeker2000, AntiVandalBot, Orionus, Nitrous231, Myanw, MagiMaster, JAnDbot, Harryzilber, Ph.eyes, Ccrrccrr, Erpel13, .anacondabot, Bubba hotep, Benmcgraw, DonVincenzo, Rettetast, Lcabanel, PrestonH, Cpiral, JA.Davidson, Osndok, Coppertwig, NewEnglandYankee, DAID, Ibrahimyu, Петър Петров, Radioactivebloke, Steel1943, Deor, ICE77, Metroccfd, Mill haru, Anonymous Dissident, Anna Lincoln, Onevim, BwDraco, BotKung, Spinningspark, Hertz1888, Caltas, Bentogoa, Hovev~enwiki, Callidior, OP8, Lucyjuice, Regushee, Vcaeken, ClueBot, Noabar, Dean Wormer, CounterVandalismBot, Duane-light, Auntof6, PhySusie, M.O.X, Cexycy, Elizium23, Thefirm96, Jonverve, HumphreyW, InternetMeme, Ean5533, BarretB, Hotcrocodile, TopherGZ, Mitch Ames, Tvargy, Alexius08, Addbot, Yoenit, Zhipengye, Fgnievinski, Thaejas, Ronhjones, GyroMagician, Redheylin, Xicer9, Tide rolls, מבשת-זרם, Bssquirrel, Neilforcier, Dede2008, Cflm001, Galoubet, Bluerasberry, Materialscientist, Limideen, 45Factoid44, Madjar, DSisyphBot, Barkinfool, RadiX, Mathonius, Nedim Ardoğa, Fotaun, Darwinius, Prari, Jc3s5h, Vhann, Drew R. Smith, Hawkpride3000, -jem-, Esar100, Tbhotch, RHC3, RjwilmsiBot, Paulmasters, Orphan Wiki, Wikipelli, OnePt618, Milliemchi, Mohsen.1987, Coasterlover1994, Vietcuong1212, Doris Camire, Taylor10897, RockMagnetist, Weisspiloti, ClueBot NG, AeroPsico, Mclinch, AlagrecoNJITWILL, Solanki.3108, JordoCo, MerlIwBot, Helpful Pixie Bot, BG19bot, Dragon2531, Shaun, Jeanlyhautyetienne, MeanMotherJr, NPSao, Theo's Little Bot, EstonianMan, DetroitSeattle, Shivansh Chaudhary, Soham, Spyglasses, Hanthoec, Airwoz, Symphero, JaconaFrere, Monkbot, Thadlooms32, Ff9473, Hafsa1982, SrihariThalla, Mcstacheattack, KasparBot and Anonymous: 383

18.11.2 Images

- **File:153056995_5ef8b01016_o.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/64/Intel_8742_153056995.jpg *License:* CC BY-SA 2.0 *Contributors:* http://www.flickr.com/photos/biwook/153056995/ *Original artist:* Ioan Sameli
- **File:350px-mSPI_three_slaves_svg.png** *Source:* https://upload.wikimedia.org/wikipedia/en/f/fc/350px-mSPI_three_slaves_svg.png *License:* CC-BY-SA-3.0 *Contributors:*

  Own work

  Original artist:

  Knivd
- **File:80486DX2_200x.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/2b/80486DX2_200x.png *License:* CC BY-SA 2.5 *Contributors:* ? *Original artist:* ?
- **File:8_bit_DAC.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b1/8_bit_DAC.svg *License:* Attribution *Contributors:* Text Book: Operational Amplifiers *Original artist:* Robert F Coughlin
- **File:ADC_Symbol.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f0/ADC_Symbol.svg *License:* Public domain *Contributors:* Own work *Original artist:* Tjwikcom
- **File:ADC_voltage_resolution.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/5/52/ADC_voltage_resolution.svg *License:* CC-BY-SA-3.0 *Contributors:*

  Self created using Inkscape

  Original artist:

  SpinningSpark
- **File:ADSL_modem_router_internals_labeled.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/2c/ADSL_modem_router_internals_labeled.jpg *License:* Public domain *Contributors:* Photographed by User:Mike1024 *Original artist:* User Mike1024 on en.wikipedia
- **File:AL1_-_Four-Phase_Systems_Inc.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/9c/AL1_-_Four-Phase_Systems_Inc.jpg *License:* Public domain *Contributors:* http://www.tayloredge.com/museum/processor/1969-AL1.jpg *Original artist:* John Taylor
- **File:AMD_DirectGMA.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/df/AMD_DirectGMA.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* ScotXW
- **File:AMD_Geode_LX_800_CPU.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/33/AMD_Geode_LX_800_CPU.jpg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Kozuch
- **File:ARMSoCBlockDiagram.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/85/ARMSoCBlockDiagram.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work in Inkscape based on en:Image:ARMSoCBlockDiagram.gif *Original artist:* en:User:Cburnett
- **File:Accupoll-embedded-computer.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/af/Accupoll-embedded-computer.jpg *License:* CC BY-SA 3.0 *Contributors:* http://www.chassis-plans.com/custom/Accupoll-Front-Oblique-Larg.jpg *Original artist:* David Lippincott for Chassis Plans
- **File:Ambox_important.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b4/Ambox_important.svg *License:* Public domain *Contributors:* Own work, based off of Image:Ambox scales.svg *Original artist:* Dsmurat (talk · contribs)
- **File:C4004_(Intel).jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/09/C4004_%28Intel%29.jpg *License:* Public domain *Contributors:* en.wikipedia.org *Original artist:* Photo by John Pilge.
- **File:Cache_incoherence_write.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/49/Cache_incoherence_write.svg *License:* Public domain *Contributors:* Own work *Original artist:* Snubcube
- **File:Cd-player-top-loading-and-DAC.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/6b/Cd-player-top-loading-and-DAC.jpg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Adamantios
- **File:CirrusLogicCS4282-AB.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/ea/CirrusLogicCS4282-AB.jpg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Andrzej Barabasz (Chepry)
- **File:Commons-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:Computer-aj_aj_ashton_01.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/d7/Desktop_computer_clipart_-_Yellow_theme.svg *License:* CC0 *Contributors:* https://openclipart.org/detail/105871/computeraj-aj-ashton-01 *Original artist:* AJ from openclipart.org
- **File:Computer_system_bus.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/68/Computer_system_bus.svg *License:* CC BY-SA 3.0 *Contributors:* Own work, based on a diagram which seems to in turn be based on page 36 of *The Essentials of Computer Organization and Architecture* By Linda Null, Julia Lobur, http://books.google.com/books?id=f83XxoBC_8MC&pg=PA36 *Original artist:* W Nowicki
- **File:Crystal_energy.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/14/Crystal_energy.svg *License:* LGPL *Contributors:* Own work conversion of Image:Crystal_128_energy.png *Original artist:* Dhatfield
- **File:DSP_block_diagram.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bc/DSP_block_diagram.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:Dsp_chip.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a9/Dsp_chip.jpg *License:* CC BY 3.0 *Contributors:* Own work *Original artist:* Matarese photos
- **File:ESOM270_eSOM300_Computer_on_Modules.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/6f/ESOM270_eSOM300_Computer_on_Modules.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Lakshmin

- **File:Folder_Hexagonal_Icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/48/Folder_Hexagonal_Icon.svg *License:* Cc-by-sa-3.0 *Contributors:* ? *Original artist:* ?
- **File:Frequency_spectrum_of_a_sinusoid_and_its_quantization_noise_floor.gif** *Source:* https://upload.wikimedia.org/wikipedia/commons/7/7a/Frequency_spectrum_of_a_sinusoid_and_its_quantization_noise_floor.gif *License:* CC0 *Contributors:* Own work *Original artist:* Bob K
- **File:GI250_PICO1_die_photo.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f9/GI250_PICO1_die_photo.jpg *License:* CC BY 3.0 *Contributors:* Own work *Original artist:* Jamo spingal
- **File:HelloWorld.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/28/HelloWorld.svg *License:* Public domain *Contributors:* Own work *Original artist:* Wooptoo
- **File:Intel_4004.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/52/Intel_4004.jpg *License:* CC-BY-SA-3.0 *Contributors:* Transfered from it.wikipedia *Original artist:* Original uploader was LucaDetomi at it.wikipedia
- **File:KL_AMD_Am286LX_ZX.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fb/KL_AMD_Am286LX_ZX.jpg *License:* CC-BY-SA-3.0 *Contributors:* CPU collection Konstantin Lanzet *Original artist:* Konstantin Lanzet
- **File:Merge-arrow.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/aa/Merge-arrow.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:MicroVGA_TUI_demoapp.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a1/MicroVGA_TUI_demoapp.jpg *License:* CC-BY-SA-3.0 *Contributors:* Digital camera :-) *Original artist:* Martin Hinner
- **File:Microcontrollers_Atmega32_Atmega8.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/c8/Microcontrollers_Atmega32_Atmega8.jpg *License:* CC BY 3.0 *Contributors:* Transferred from en.wikipedia to Commons. *Original artist:* Vahid alpha at English Wikipedia
- **File:Microelectronics_stub.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/c3/Microelectronics_stub.svg *License:* LGPL *Contributors:* Integrated circuit icon.svg: *Original artist:* Integrated_circuit_icon.svg: Everaldo Coelho and YellowIcon
- **File:Nuvola_apps_ksim.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/8d/Nuvola_apps_ksim.png *License:* LGPL *Contributors:* http://icon-king.com *Original artist:* David Vignoni / ICON KING
- **File:Overo_with_coin.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/95/Overo_with_coin.jpg *License:* Public domain *Contributors:* Own work *Original artist:* JustinC474
- **File:PCIExpress.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fc/PCIExpress.jpg *License:* CC-BY-SA-3.0 *Contributors:* come from en.wikipedia *Original artist:* w:user:snickerdo
- **File:PIC12C508-HD.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/d0/PIC12C508-HD.jpg *License:* CC BY 3.0 *Contributors:* http://zeptobars.ru/en/read/open-microchip-asic-what-inside-II-msp430-pic-z80 *Original artist:* ZeptoBars
- **File:PIC18F8720.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/18/PIC18F8720.jpg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Portal-puzzle.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/f/fd/Portal-puzzle.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Question_book-new.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*

  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:* Tkgd2007
- **File:SMSC_LAN91C110_ethernet_chip.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/82/SMSC_LAN91C110_ethernet_chip.jpg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Nixdorf
- **File:SPI_8-bit_circular_transfer.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bb/SPI_8-bit_circular_transfer.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:SPI_single_slave.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/ed/SPI_single_slave.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:SPI_three_slaves.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fc/SPI_three_slaves.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:SPI_three_slaves_daisy_chained.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/97/SPI_three_slaves_daisy_chained.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:SPI_timing_diagram2.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/6b/SPI_timing_diagram2.svg *License:* CC-BY-SA-3.0 *Contributors:*
- SPI_timing_diagram.svg *Original artist:* SPI_timing_diagram.svg: en:User:Cburnett
- **File:STM32F100C4T6B-HD.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/67/STM32F100C4T6B-HD.jpg *License:* CC BY 3.0 *Contributors:* http://zeptobars.ru/en/read/how-to-open-microchip-asic-what-inside *Original artist:* ZeptoBars
- **File:Sampled.signal.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/88/Sampled.signal.svg *License:* Public domain *Contributors:* No machine-readable source provided. Own work assumed (based on copyright claims). *Original artist:* No machine-readable author provided. Rbj assumed (based on copyright claims).
- **File:SoCDesignFlow.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/58/SoCDesignFlow.svg *License:* CC-BY-SA-3.0 *Contributors:* en:Image:SoCDesignFlow.gif *Original artist:* Traced by User:Stannered

- **File:Symbol_list_class.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/d/db/Symbol_list_class.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Telecom-icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/4e/Telecom-icon.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Text_document_with_red_question_mark.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a4/Text_document_with_red_question_mark.svg *License:* Public domain *Contributors:* Created by bdesham with Inkscape; based upon Text-x-generic.svg from the Tango project. *Original artist:* Benjamin D. Esham (bdesham)
- **File:WM_WM8775SEDS-AB.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a9/WM_WM8775SEDS-AB.jpg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Andrzej Barabasz (Chepry)
- **File:Wikibooks-logo-en-noslogan.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-noslogan.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.
- **File:Wikibooks-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fa/Wikibooks-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.
- **File:Wikiversity-logo-Snorky.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/1b/Wikiversity-logo-en.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Snorky
- **File:Wikiversity-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/91/Wikiversity-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Snorky (optimized and cleaned up by verdy_p) *Original artist:* Snorky (optimized and cleaned up by verdy_p)
- **File:Wiktionary-logo-en.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f8/Wiktionary-logo-en.svg *License:* Public domain *Contributors:* Vector version of Image:Wiktionary-logo-en.png. *Original artist:* Vectorized by Fvasconcellos (talk · contribs), based on original logo tossed together by Brion Vibber
- **File:Z80_arch.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/db/Z80_arch.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Appaloosa
- **File:Zeroorderhold.signal.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/15/Zeroorderhold.signal.svg *License:* Public domain *Contributors:* en:Zeroorderhold.signal.svg *Original artist:* image source obtained from en:User:Petr.adamek (with permission) and previously saved as PD in PNG format. touched up a little and converted to SVG by en:User:Rbj

18.11.3 Content license

- Creative Commons Attribution-Share Alike 3.0