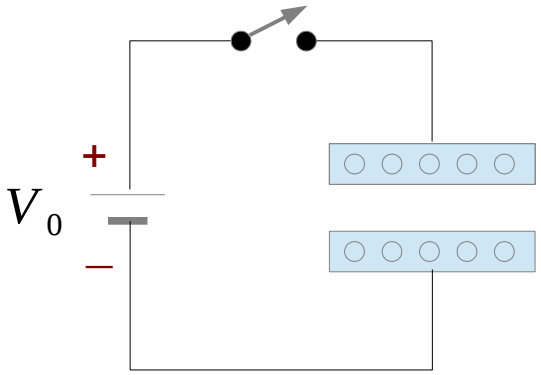
Delay Background

Please send corrections (or suggestions) to youngwlim@hotmail.com.

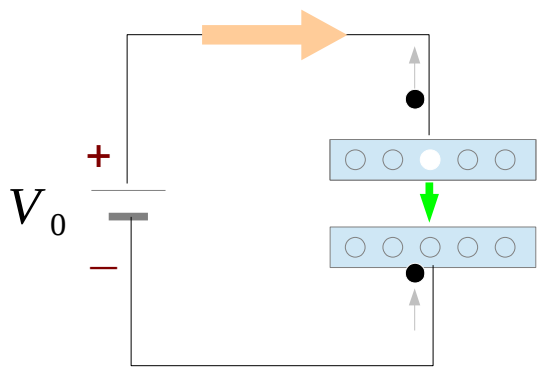
This document was produced by using OpenOffice and Octave.

Charge

Before

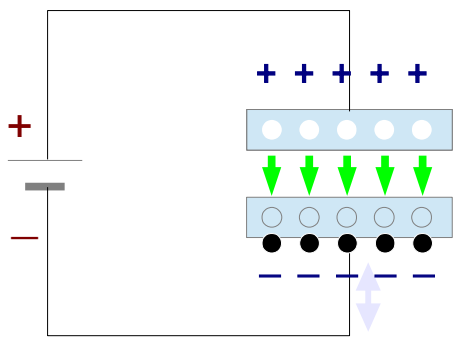


Initial



Final

No more electrons to leave



crowded electrons prevent other electrons from arriving

$$i_c = C \cdot \frac{d\,v_c}{d\,t}$$

Energy stored in Electric Field

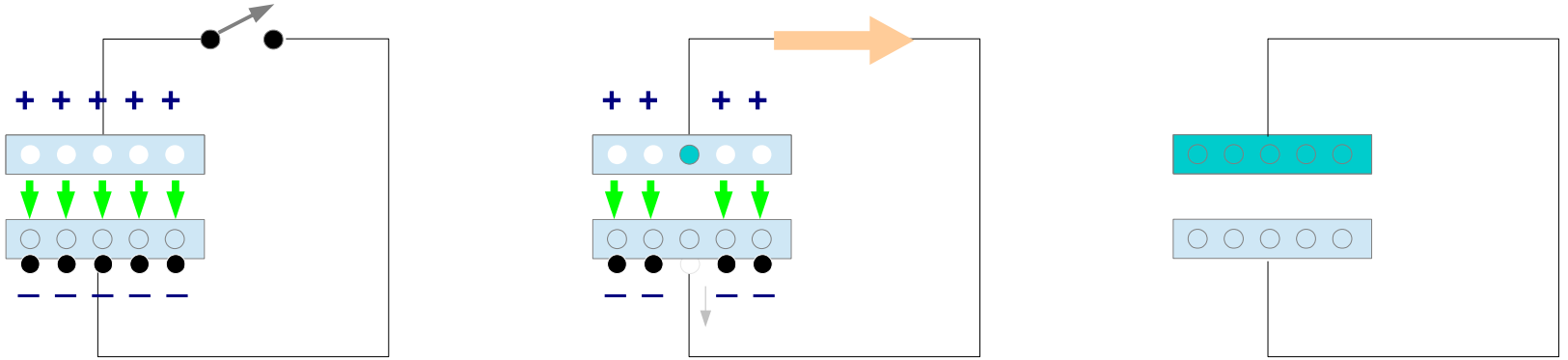
$$v_c(0^-) = v_c(0^+)$$ *unyielding voltage*

$$i_c(0^-) \neq i_c(0^+)$$ *current jump*

$$v_c(\infty) = V_0$$

$$i_c(\infty) = 0$$

Discharge



No more electrons moving

$$i_c = C \cdot \frac{d\,v_c}{d\,t}$$

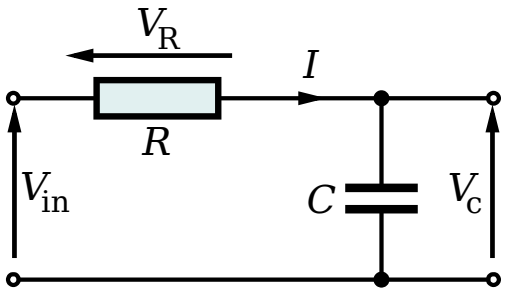
$v_c(0^-) = v_c(0^+)$ *unyielding voltage*

$i_c(0^-) \neq i_c(0^+)$ *current jump*

$v_c(\infty) = 0$

$i_c(\infty) = 0$

Charge



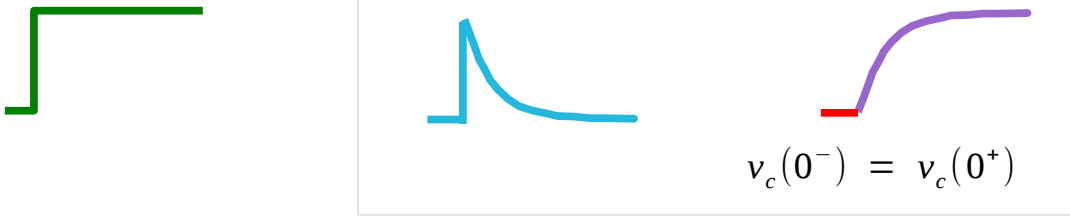
$$i_c = C \cdot \frac{d\,v_c}{d\,t}$$

unyielding voltage

current jump

the capacitor voltage slowly follows the shape of the applied step input voltage

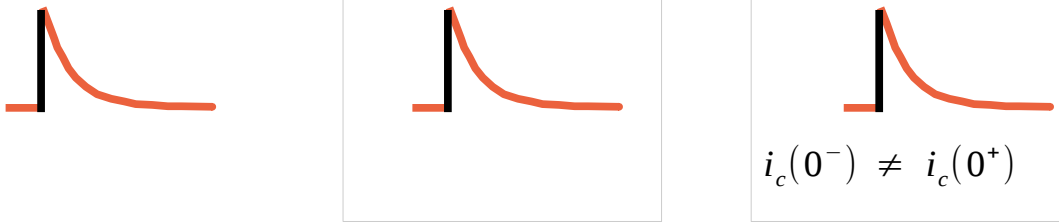
v = **v_R** + **v_C** ***continuous***



$$v_c(0^-) = v_c(0^+)$$

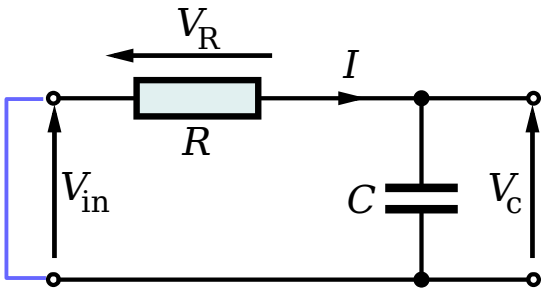
the capacitor current changes abruptly by the applied step input voltage and then slowly becomes zero

i = **i_R** = **i_C** ***jump***



$$i_c(0^-) \neq i_c(0^+)$$

Discharge

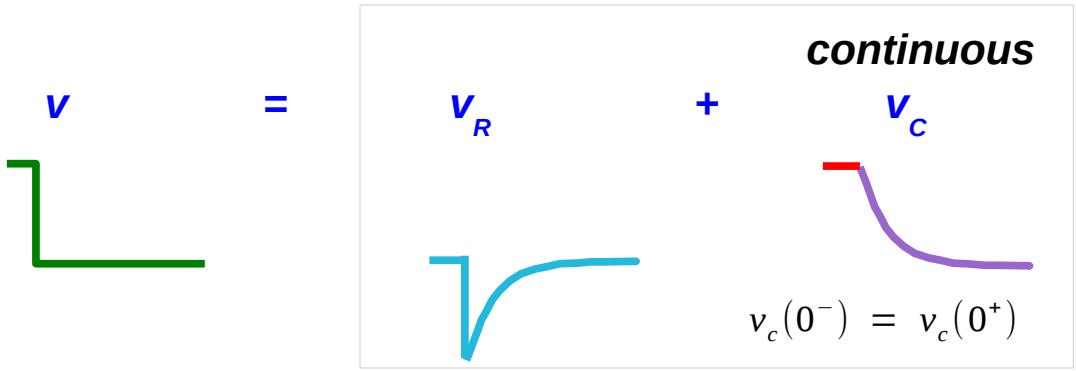


$$i_c = C \cdot \frac{d\,v_c}{d\,t}$$

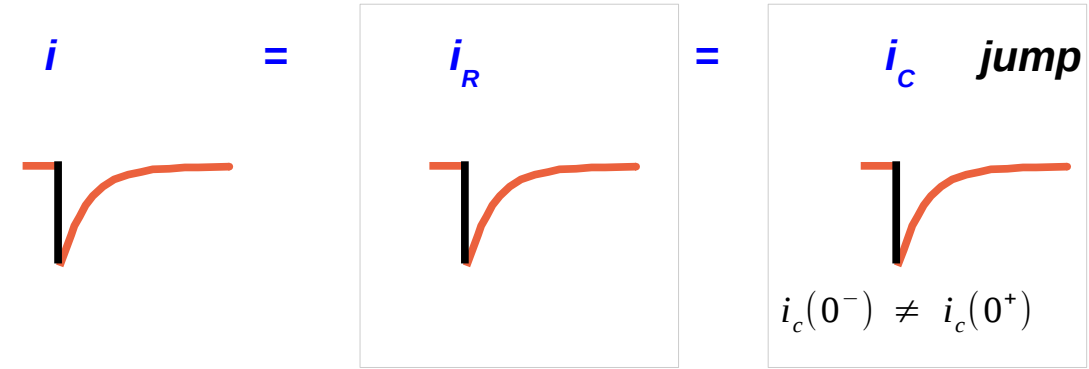
unyielding voltage

current jump

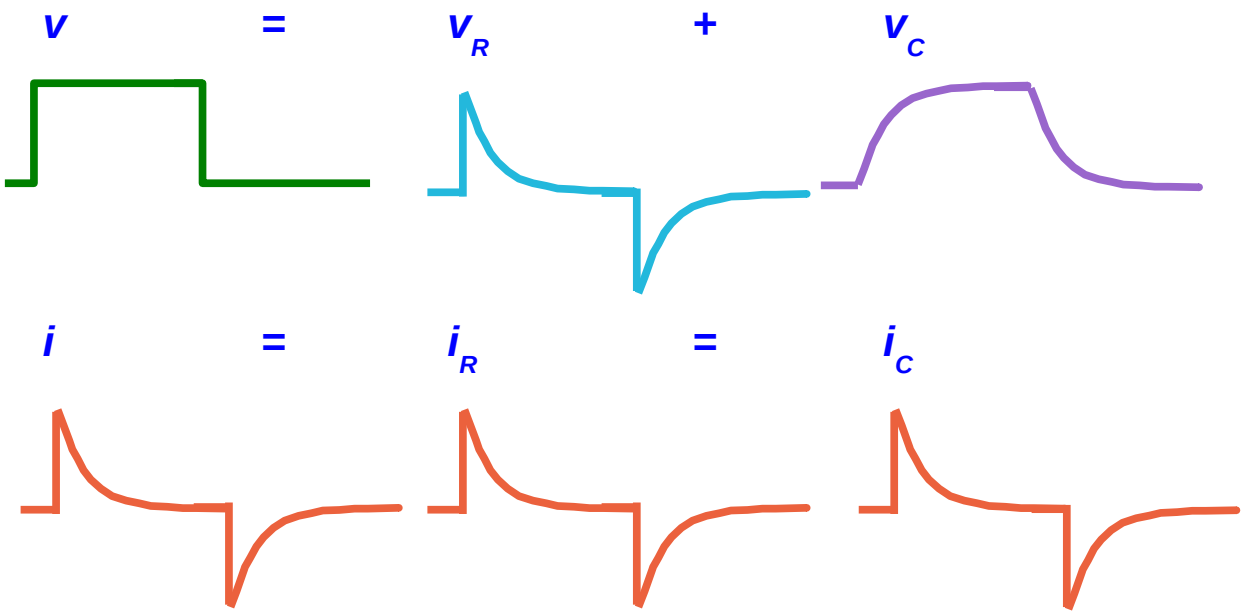
the capacitor voltage slowly follows the the shape of the applied step input voltage



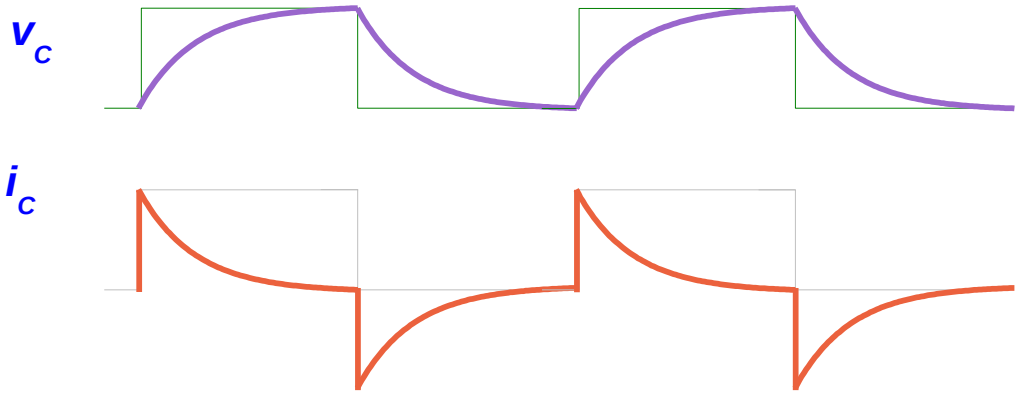
the capacitor current changes abruptly by the applied step input voltage and then slowly becomes zero



Pulse

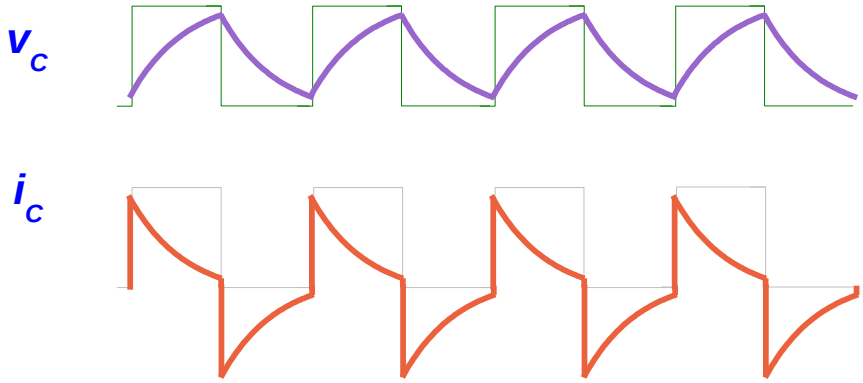


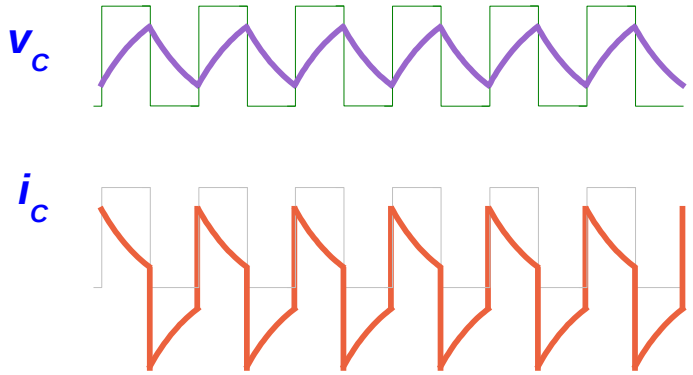
Pulses with different periods



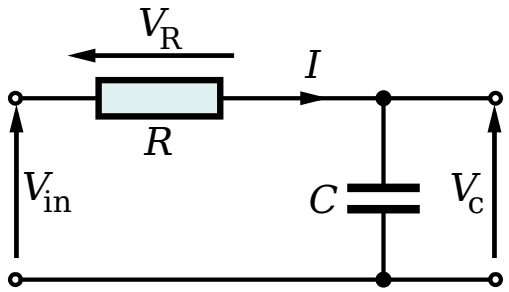
$$i_C = C \frac{d\,v_C}{d\,t}$$

ω ↑ i_c ↑ X_c ↓





Differential Equation (1)



$$RC \cdot \frac{dv_c}{dt} + v_c = V_s \qquad t \geq 0^+ \qquad v_c(0^+) = v_c(0^-) = V_0$$

$$\frac{dv_c}{dt} + \frac{1}{RC} v_c = \frac{1}{RC} V_s$$

$$\frac{dv_c}{dt} + \frac{1}{RC} v_c = 0 \qquad \frac{dv_c}{dt} = -\frac{1}{RC} v_c \qquad \frac{dv_c}{v_c} = -\frac{1}{RC} dt$$

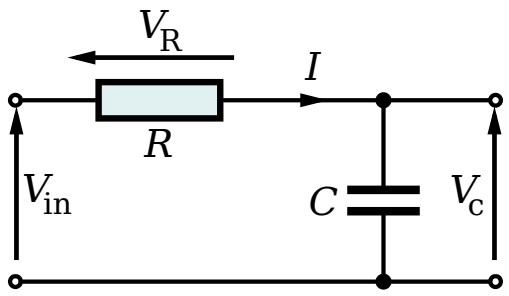
$$\ln v_c = -\int \frac{1}{RC} dt + C \qquad v_c = A e^{-\int \frac{1}{RC} dt} \qquad v_c = A e^{-\frac{t}{RC}}$$

$$\frac{dv_c}{dt} e^{\frac{t}{RC}} + \frac{1}{RC} v_c e^{\frac{t}{RC}} = \frac{V_s}{RC} e^{\frac{t}{RC}} \qquad \frac{d}{dt}\left(v_c e^{\frac{t}{RC}} \right) = \frac{V_s}{RC} e^{\frac{t}{RC}}$$

$$\left(v_c e^{\frac{t}{RC}} \right) = c + \int \frac{V_s}{RC} e^{\frac{t}{RC}} dt = c + V_s e^{\frac{t}{RC}} \qquad v_c(t) = c e^{-\frac{t}{RC}} + V_s$$

$$v_c(0) = c + V_s = V_0 \qquad v_c(t) = (V_0 - V_s) e^{-\frac{t}{RC}} + V_s$$

Differential Equation (2)



$$RC \cdot \frac{dv_c}{dt} + v_c = V_s \qquad t \geq 0^+ \qquad v_c(0^+) = v_c(0^-) = V_0$$

$$\frac{dv_c}{dt} + \frac{1}{RC} v_c = \frac{1}{RC} V_s$$

$$\frac{dv_c}{dt} + \frac{1}{RC} v_c = 0 \qquad \text{assume } v_c = Ae^{st} \qquad \frac{d}{dt}(Ae^{st}) + \frac{1}{RC} Ae^{st} = 0$$

$$\left(s + \frac{1}{RC}\right) Ae^{st} = 0 \qquad s = -\frac{1}{RC} \qquad v_c = Ae^{-\frac{t}{RC}}$$

homogeneous solution

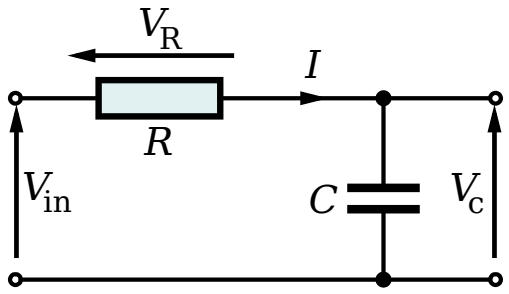
$$\frac{dv_c}{dt} + \frac{1}{RC} v_c = \frac{V_s}{RC} \qquad \text{assume } v_c = c \qquad \frac{d}{dt}(c) + \frac{1}{RC} c = \frac{V_s}{RC} \qquad v_c = V_s$$

particular solution

$$v_c(t) = V_s + Ae^{-\frac{t}{RC}} \qquad v_c(0) = A + V_s = V_0 \qquad v_c(t) = (V_0 - V_s)e^{-\frac{t}{RC}} + V_s$$

complete solution

Differential Equation (3)



$$R i + \frac{1}{C}\int_{-\infty}^{t} i\, d\tau = V_s \quad t \geq 0^+ \quad v_c(0^+) = v_c(0^-) = V_0$$

$$i = C\frac{d v_c}{d t} \qquad RC \cdot \frac{d v_c}{d t} + v_c = V_s$$

$$(RC D + 1) v_c = V_s \quad (RC\lambda + 1) = 0 \quad \lambda = -\frac{1}{RC} \quad v_{c0} = A e^{-\frac{t}{RC}}$$

$$v_{c0}(0) = A = V_0 \quad v_{c0} = V_0 e^{-\frac{t}{RC}}$$

zero-input response

$$RC \cdot \dot{h}(t) + h(t) = \delta(t)$$

assume $v_c = c$ $\quad \frac{d v_c}{d t} + \frac{1}{RC} v_c = \frac{V_s}{RC} \quad v_c = V_s$

particular solution

$$v_c(t) = V_s + A e^{-\frac{t}{RC}} \qquad v_c(t) = (V_0 - V_s) e^{-\frac{t}{RC}} + V_s$$

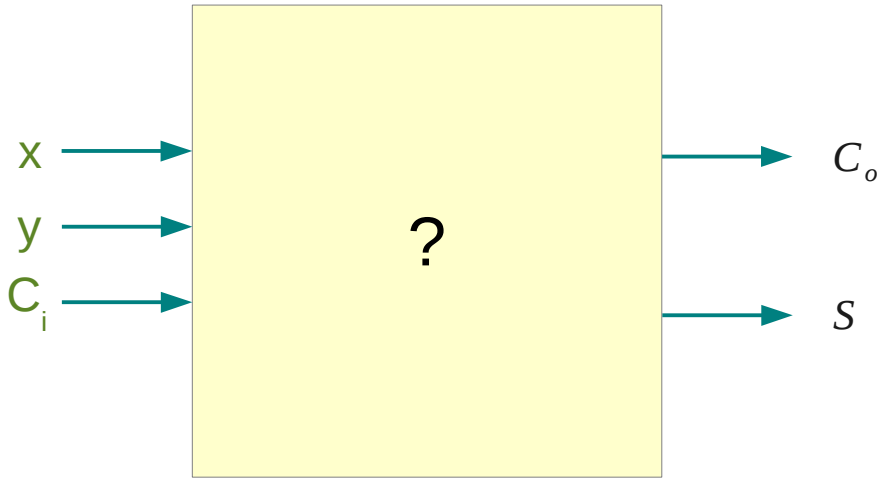
complete solution

Adder Example

Truth Table

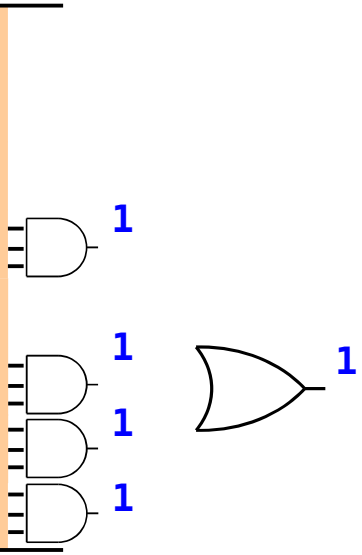
x	y	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

inputs output

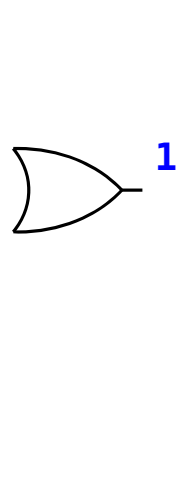


SOP

x	y	C_i	C_o
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



x	y	C_i	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



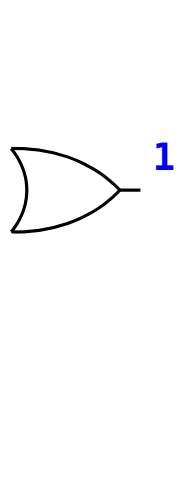
$$C_o = \bar{x}\,y\,C_i + x\,\bar{y}\,C_i + x\,y\,\bar{C}_i + x\,y\,C_i$$

$$S = \bar{x}\,\bar{y}\,C_i + \bar{x}\,y\,\bar{C}_i + x\,\bar{y}\,\bar{C}_i + x\,y\,C_i$$

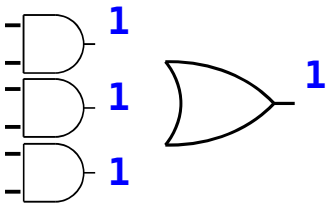
K-Map

x	y	C_i	C_o
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x	y	C_i	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



x \ y	0	1
0	0	0
1	0	1
1	1	1
C_i	1	0

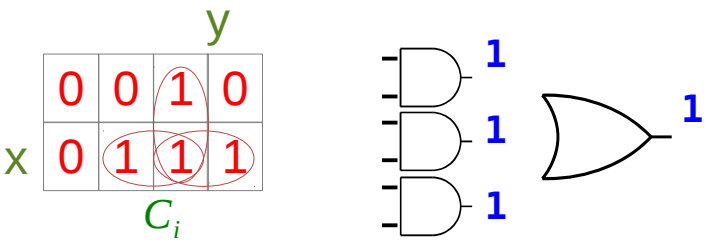


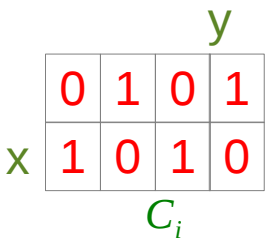
x \ y	0	1
0	0	1
1	1	0
C_i	1	0

$$C_o = y C_i + x C_i + x y$$

$$S = \bar{x}\,\bar{y}\,C_i + \bar{x}\,y\,\bar{C}_i + x\,\bar{y}\,\bar{C}_i + x\,y\,C_i$$

Boolean Algebra





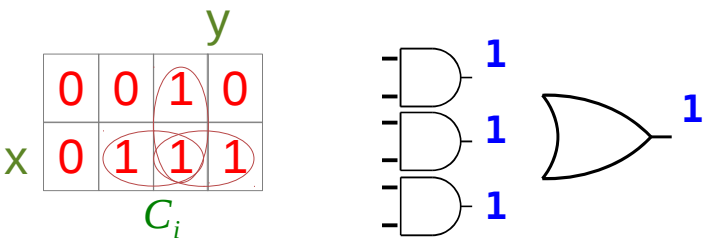
$$C_o = y C_i + x C_i + x y$$

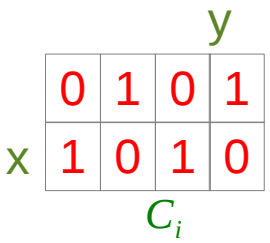
$$\begin{aligned} C_o &= (x + y) C_i + x y \\ &= (\bar{x} y + x \bar{y} + xy) C_i + x y \\ &= (\bar{x} y + x \bar{y}) C_i + xy (C_i + 1) \\ &= (x \oplus y) C_i + xy \end{aligned}$$

$$S = \bar{x}\,\bar{y}\,C_i + \bar{x}\,y\,\bar{C}_i + x\,\bar{y}\,\bar{C}_i + x\,y\,C_i$$

$$\begin{aligned} S &= (\bar{x}\,\bar{y} + xy) C_i + (\bar{x}\,y + x\,\bar{y}) \bar{C}_i \\ &= \overline{(x \oplus y)} C_i + (x \oplus y) \bar{C}_i \\ &= (x \oplus y) \oplus C_i \end{aligned}$$

Boolean Algebra



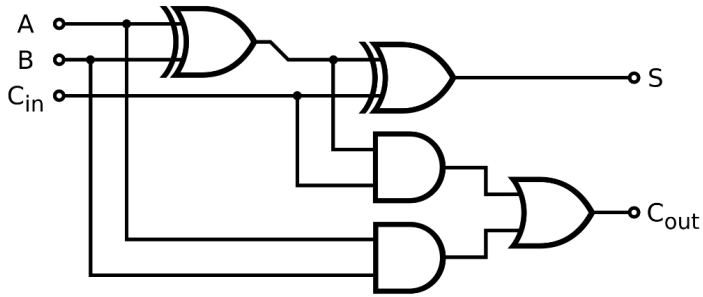


$$C_o = y C_i + x C_i + x y$$

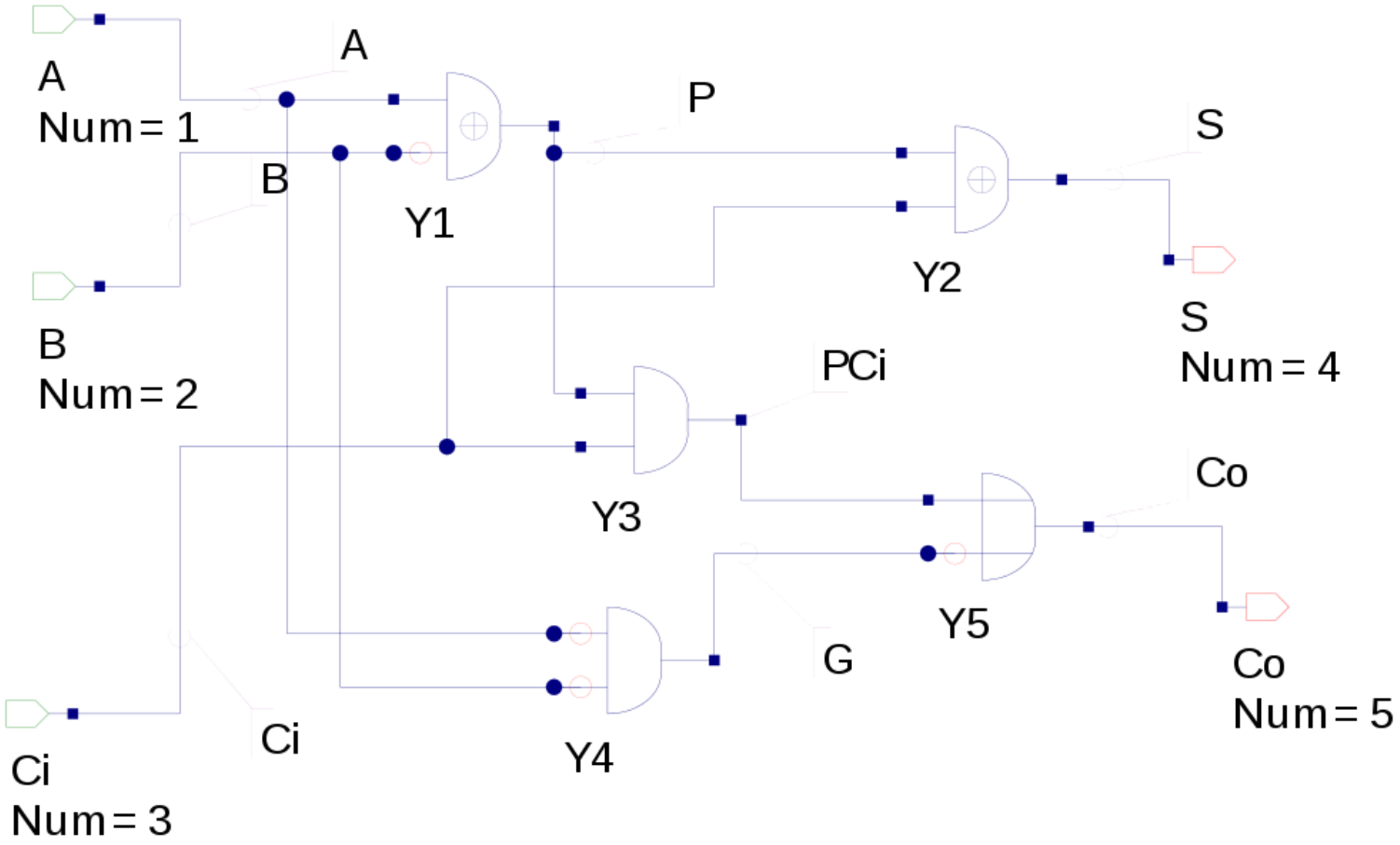
$$\begin{aligned} C_o &= (x + y) C_i + x y \\ &= (\bar{x} y + x \bar{y} + xy) C_i + x y \\ &= (\bar{x} y + x \bar{y}) C_i + xy (C_i + 1) \\ &= (x \oplus y) C_i + xy \end{aligned}$$

$$S = \bar{x}\,\bar{y}\,C_i + \bar{x}\,y\,\bar{C}_i + x\,\bar{y}\,\bar{C}_i + x\,y\,C_i$$

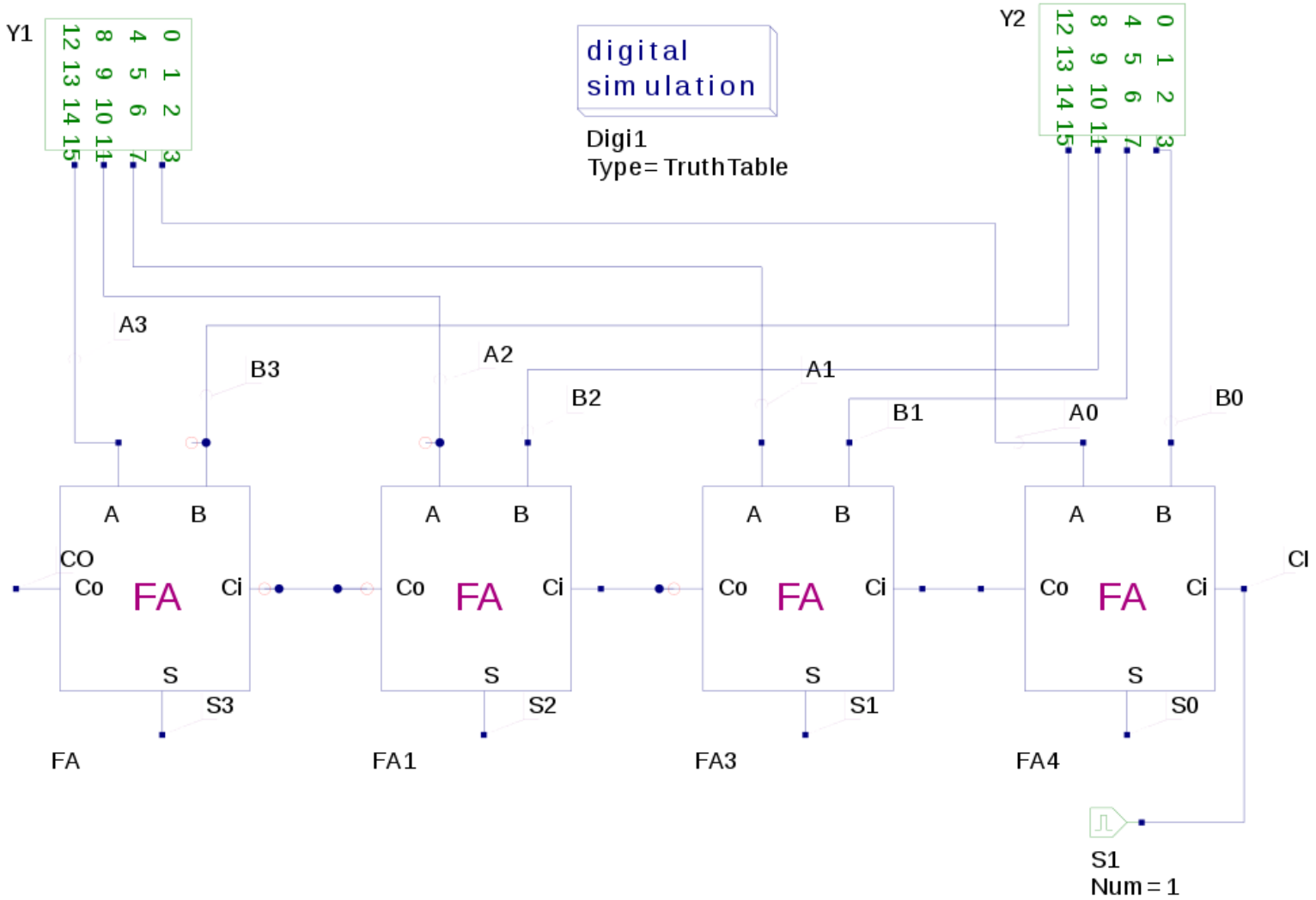
$$\begin{aligned} S &= (\bar{x}\,\bar{y} + xy) C_i + (\bar{x}\,y + x\,\bar{y}) \bar{C}_i \\ &= \overline{(x \oplus y)} C_i + (x \oplus y) \bar{C}_i \\ &= (x \oplus y) \oplus C_i \end{aligned}$$



Full Adder in Qucs



4-Bit Adder in Qucs



adder_tb.v

```
`timescale 1ns/100ps

module adder_tb;

reg signed [3:0] i, j, k, flag;

wire signed [3:0] S;


adder4 A4 (i, j, 1'b0, Co, S);

initial
begin
    $dumpfile("test.vcd");
    $dumpvars(0, adder_tb);
end



initial
begin
  i = -4'd8;
  j = -4'd8;
  repeat (16)
  begin
    repeat (16)
    begin

      k = i + j;

      flag = {i[3], j[3], k[3]};
      #1

      if ((flag == 3'b110) || (flag == 3'b001))
              $display("i=%d j=%d k=%d S=%d OV", i , j, k, S);
      else
              $display("i=%d j=%d k=%d S=%d", i , j, k, S);


      j = j + 4'b1;
    end
    i = i + 4'b1;
  end

end // initial begin

endmodule
```

adder4.v, fa_gate.v fa_flow.v

adder4.v

```
module adder4(A, B, Ci, Co, S);
  input [3:0] A, B;
  input Ci;

  output Co;
  output [3:0] S;

  wire[3:1] C;

  fa  fa0 (A[0], B[0], Ci,   C[1], S[0]);
  fa  fa1 (A[1], B[1], C[1], C[2], S[1]);
  fa  fa2 (A[2], B[2], C[2], C[3], S[2]);
  fa  fa3 (A[3], B[3], C[3], Co,   S[3]);


endmodule
```

fa_gate.v

```
module fa(a, b, c, Co, S);
  input a, b, c;
  output Co, S;

  wire P, S, G, PC;

  xor #0.1 U1 (P, a, b);
  xor #0.1 U2 (S, P, c);
  and #0.1 U3 (G, a, b);
  and #0.1 U4 (PC, P, c);
  or  #0.1 U5 (Co, G, PC);
endmodule
```

fa_flow.v

```
module fa(a, b, c, Co, S);
  input a, b, c;
  output Co, S;

  wire P, S, G, PC;

  assign #0.1 P = a ^ b;
  assign #0.1 S = P ^ c;
  assign #0.1 G = a & b;
  assign #0.1 PC = P & c;
  assign #0.1 Co = G | PC;
endmodule
```

iverilog adder_tb.v adder4.v fa_gate.v
vvp a.out

iverilog adder_tb.v adder4.v fa_flow.v
vvp a.out

Gate Delay

Gate Level Simulation

adder4.v, fa_behav.v, fa_arith.v

adder4.v

```
module adder4(A, B, Ci, Co, S);
  input [3:0] A, B;
  input Ci;

  output Co;
  output [3:0] S;

  wire[3:1] C;

  fa  fa0 (A[0], B[0], Ci,   C[1], S[0]);
  fa  fa1 (A[1], B[1], C[1], C[2], S[1]);
  fa  fa2 (A[2], B[2], C[2], C[3], S[2]);
  fa  fa3 (A[3], B[3], C[3], Co,   S[3]);


endmodule
```

iverilog adder_tb.v adder4.v fa_behav.v
vvp a.out

iverilog adder_tb.v adder4.v fa_arith.v
vvp a.out

fa_behav.v

```
module fa(a, b, c, Co, S);
  input a, b, c;
  output Co, S;

  reg P, S, G, PC, Co;

  always @(a or b)
    #0.1 P = a ^ b;

  always @(P or c)
    #0.1 S = P ^ c;

  always @(a or b)
    #0.1 G = a & b;

  always @(P or c)
    #0.1 PC = P & c;

  always @(G or PC)
    #0.1 Co = G | PC;


endmodule
```

fa_arith.v

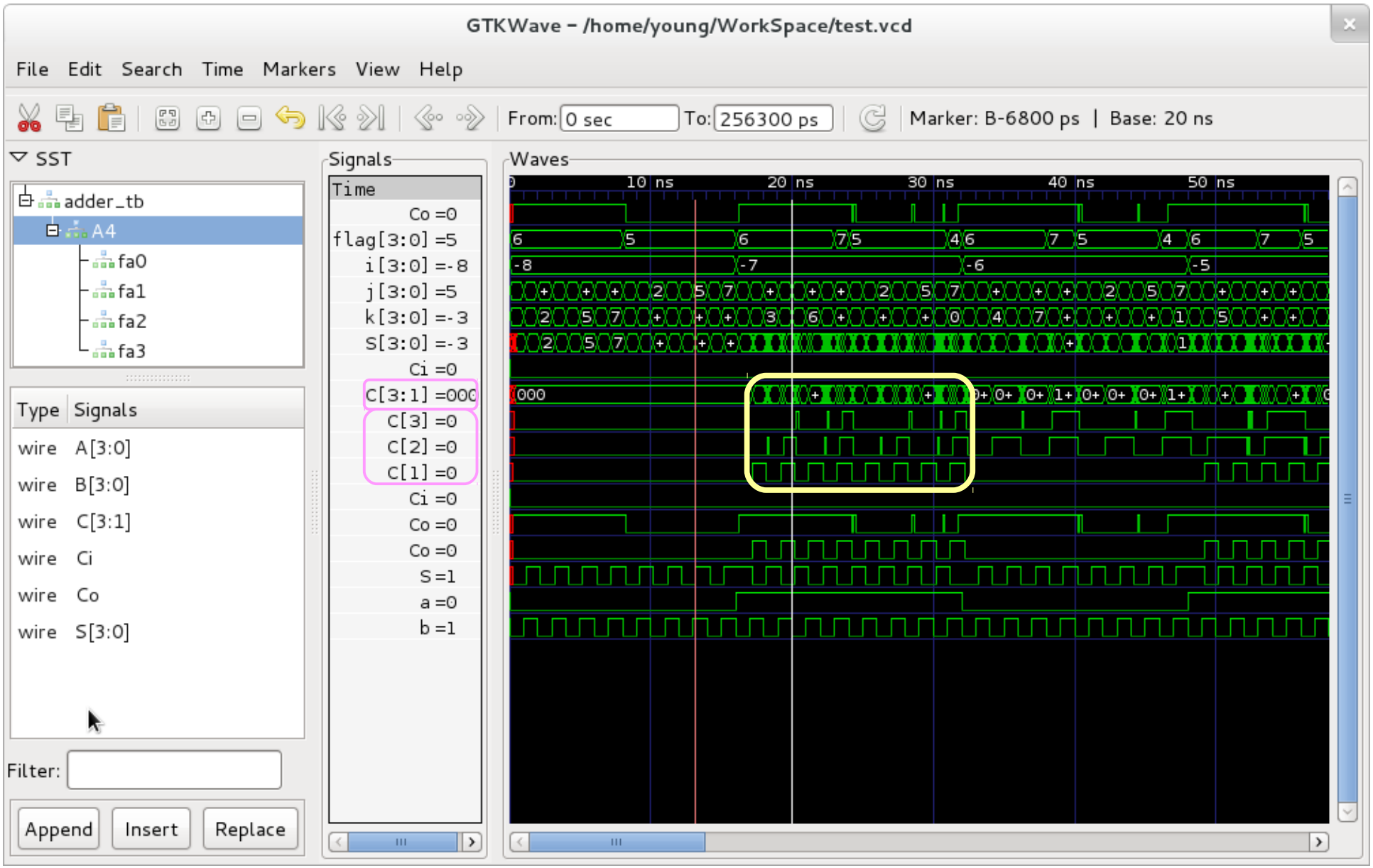
```
module fa(a, b, c, Co, S);
  input a, b, c;
  output Co, S;

  reg S, Co;

   always @(a or b or c)
     #0.1 {Co, S} = a + b + c;

endmodule
```

Behavioral Level Simulation

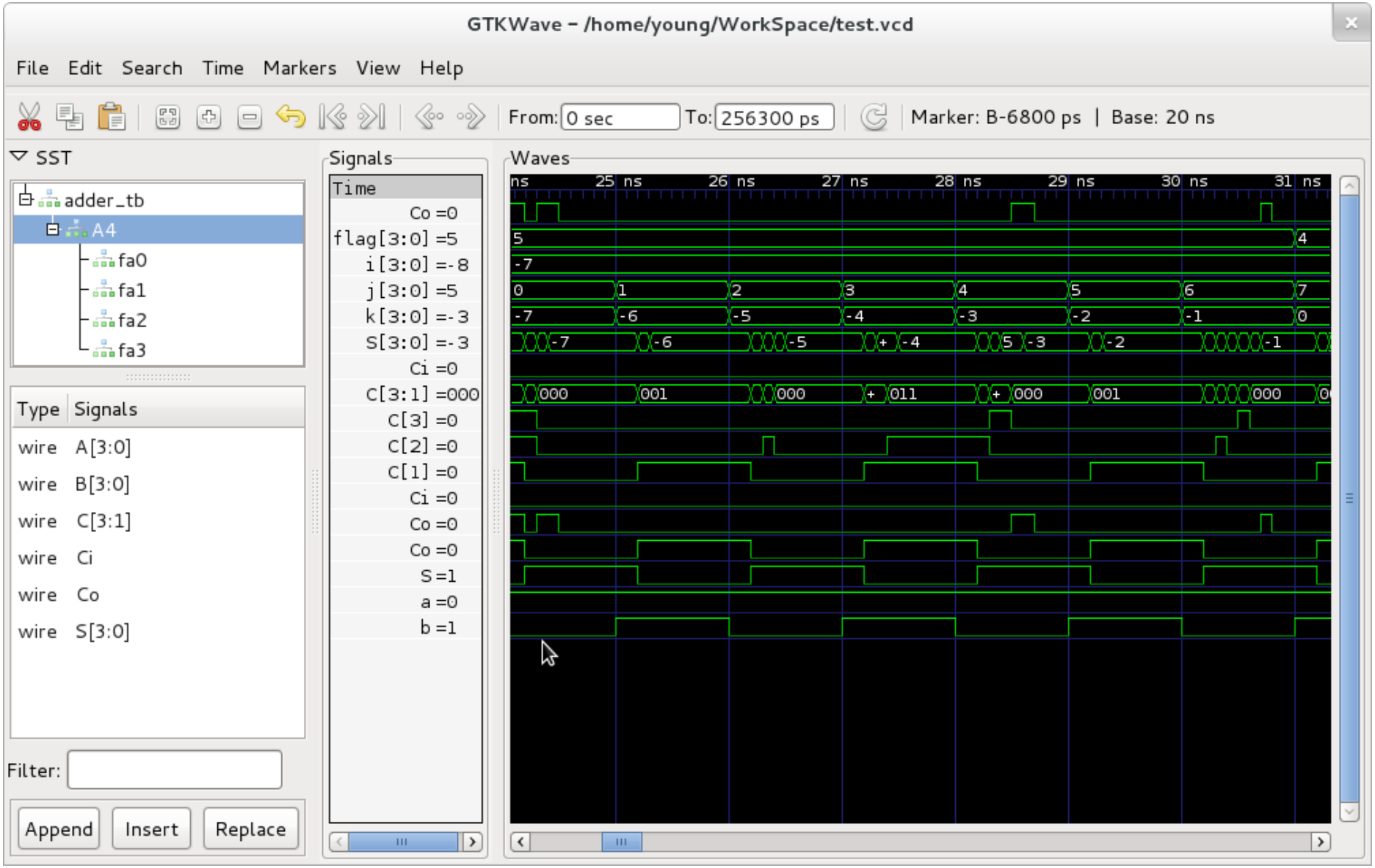
Output

```
[young@ubook WorkSpace]$ iverilog adder_tb.v adder4.v fa_gate.v
[young@ubook WorkSpace]$ vvp a.out
VCD info: dumpfile test.vcd opened for output.
i=-8 j=-8 k= 0 S= 0 OV
i=-8 j=-7 k= 1 S= 1 OV
i=-8 j=-6 k= 2 S= 2 OV
i=-8 j=-5 k= 3 S= 3 OV
i=-8 j=-4 k= 4 S= 4 OV
i=-8 j=-3 k= 5 S= 5 OV
i=-8 j=-2 k= 6 S= 6 OV
i=-8 j=-1 k= 7 S= 7 OV
i=-8 j= 0 k=-8 S=-8
i=-8 j= 1 k=-7 S=-7
i=-8 j= 2 k=-6 S=-6
i=-8 j= 3 k=-5 S=-5
i=-8 j= 4 k=-4 S=-4
i=-8 j= 5 k=-3 S=-3
i=-8 j= 6 k=-2 S=-2
i=-8 j= 7 k=-1 S=-1
i=-7 j=-8 k= 1 S= 1 OV
i=-7 j=-7 k= 2 S= 2 OV
i=-7 j=-6 k= 3 S= 3 OV
i=-7 j=-5 k= 4 S= 4 OV
i=-7 j=-4 k= 5 S= 5 OV
i=-7 j=-3 k= 6 S= 6 OV
i=-7 j=-2 k= 7 S= 7 OV
i=-7 j=-1 k=-8 S=-8
i=-7 j= 0 k=-7 S=-7
i=-7 j= 1 k=-6 S=-6
i=-7 j= 2 k=-5 S=-5
i=-7 j= 3 k=-4 S=-4
i=-7 j= 4 k=-3 S=-3
i=-7 j= 5 k=-2 S=-2
i=-7 j= 6 k=-1 S=-1
i=-7 j= 7 k= 0 S= 0
i=-6 j=-8 k= 2 S= 2 OV
i=-6 j=-7 k= 3 S= 3 OV
i=-6 j=-6 k= 4 S= 4 OV
i=-6 j=-5 k= 5 S= 5 OV
i=-6 j=-4 k= 6 S= 6 OV
i=-6 j=-3 k= 7 S= 7 OV
i=-6 j=-2 k=-8 S=-8
i=-6 j=-1 k=-7 S=-7
i=-6 j= 0 k=-6 S=-6
i=-6 j= 1 k=-5 S=-5
i=-6 j= 2 k=-4 S=-4
i=-6 j= 3 k=-3 S=-3
i=-6 j= 4 k=-2 S=-2
```

Waveform (1)


GTKWave – /home/young/WorkSpace/test.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 256300 ps
Marker: B-6800 ps | Base: 20 ns
SST
adder_tb
A4
fa0
fa1
fa2
fa3
Type Signals
wire A[3:0]
wire B[3:0]
wire C[3:1]
wire Ci
wire Co
wire S[3:0]
Filter:
Append
Insert
Replace
Signals
Time
Co =0
flag[3:0] =5
i[3:0] =-8
j[3:0] =5
k[3:0] =-3
S[3:0] =-3
Ci =0
C[3:1] =000
C[3] =0
C[2] =0
C[1] =0
Ci =0
Co =0
Co =0
S =1
a =0
b =1
Waves
10 ns
20 ns
30 ns
40 ns
50 ns

Waveform (2)


GTKWave – /home/young/WorkSpace/test.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 256300 ps
Marker: B-6800 ps | Base: 20 ns
SST
adder_tb
A4
fa0
fa1
fa2
fa3
Type Signals
wire A[3:0]
wire B[3:0]
wire C[3:1]
wire Ci
wire Co
wire S[3:0]
Filter:
Append
Insert
Replace
Signals
Time
Co =0
flag[3:0] =5
i[3:0] =-8
j[3:0] =5
k[3:0] =-3
S[3:0] =-3
Ci =0
C[3:1] =000
C[3] =0
C[2] =0
C[1] =0
Ci =0
Co =0
Co =0
S =1
a =0
b =1
Waves
ns 25 ns 26 ns 27 ns 28 ns 29 ns 30 ns 31 ns

$dumpvars()

$dumpfile("filename.vcd")
$dumpvar dumps all variables in the design.
$dumpvar(1, top) dumps all the variables in module top, but not modules instantiated in top.
$dumpvar(2, top) dumps all the variables in module top and 1 level below.
$dumpvar(n, top) dumps all the variables in module top and n-1 levels below.
$dumpvar(0, top) dumps all the variables in module top and all level below.
$dumpon initiates the dump.
$dumpoff stop dumping.

vvp

NAME

vvp - Icarus Verilog vvp runtime engine

SYNOPSIS

vvp [-sv] [-Mpath] [-mmodule] [-llogfile] inputfile [extended-args...]

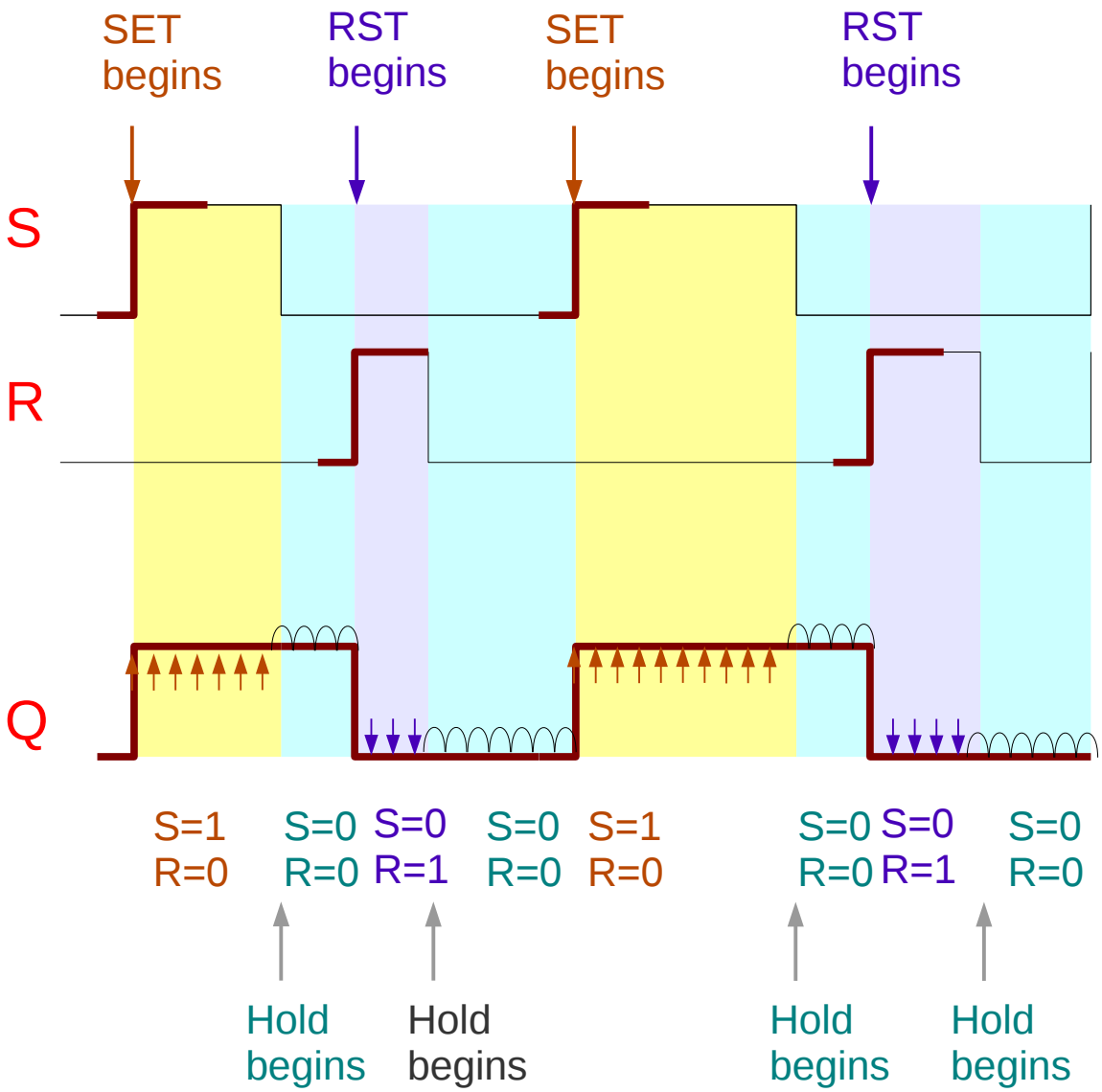
DESCRIPTION

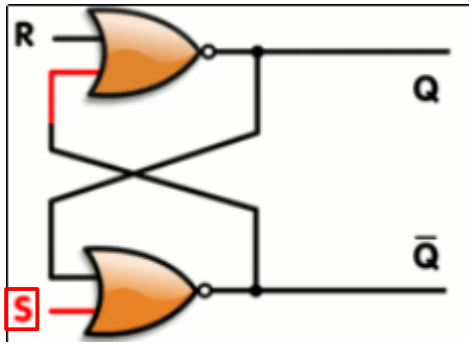
vvp is the run time engine that executes the default compiled form generated by Icarus Verilog. The output from the iverilog command is not by itself executable on any platform. Instead, the vvp program is invoked to execute the generated output file.

Sequential Circuits

Latches
Flip Flops

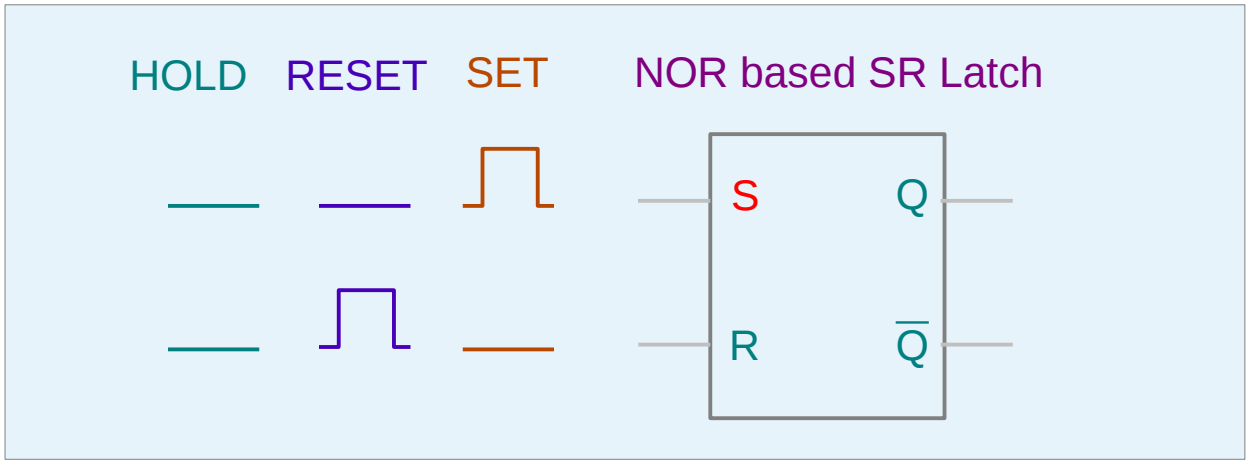
NOR-based SR Latch

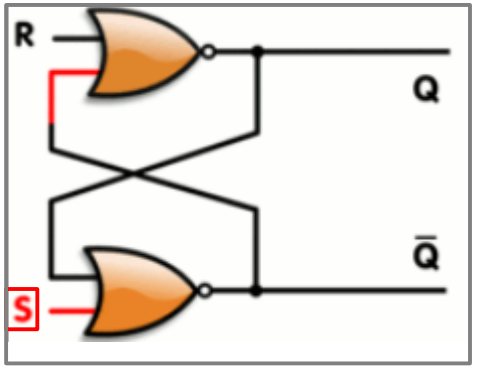


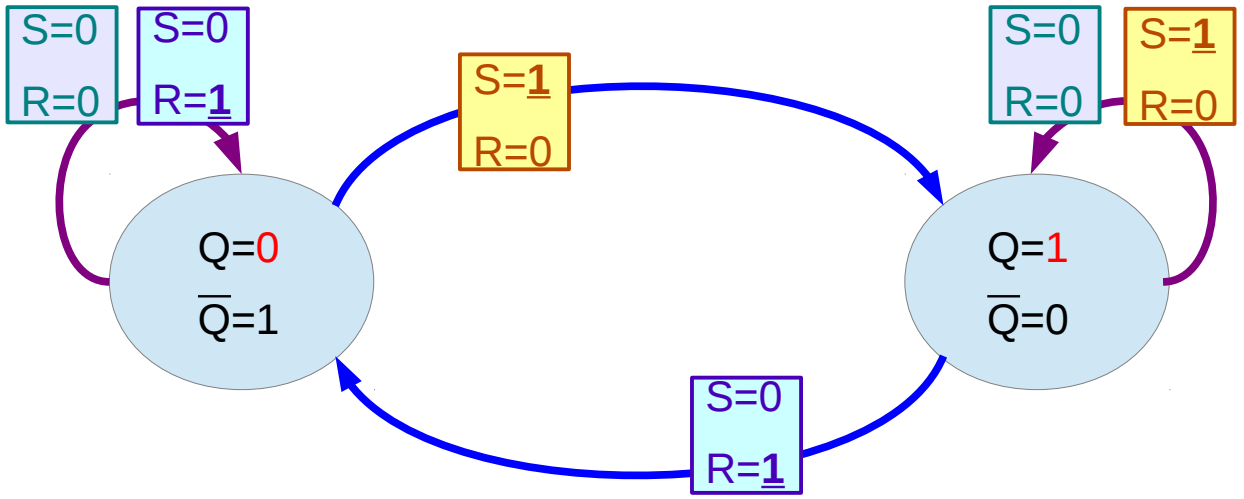


SET	S=1 R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=1	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

NOR-based SR Latch States

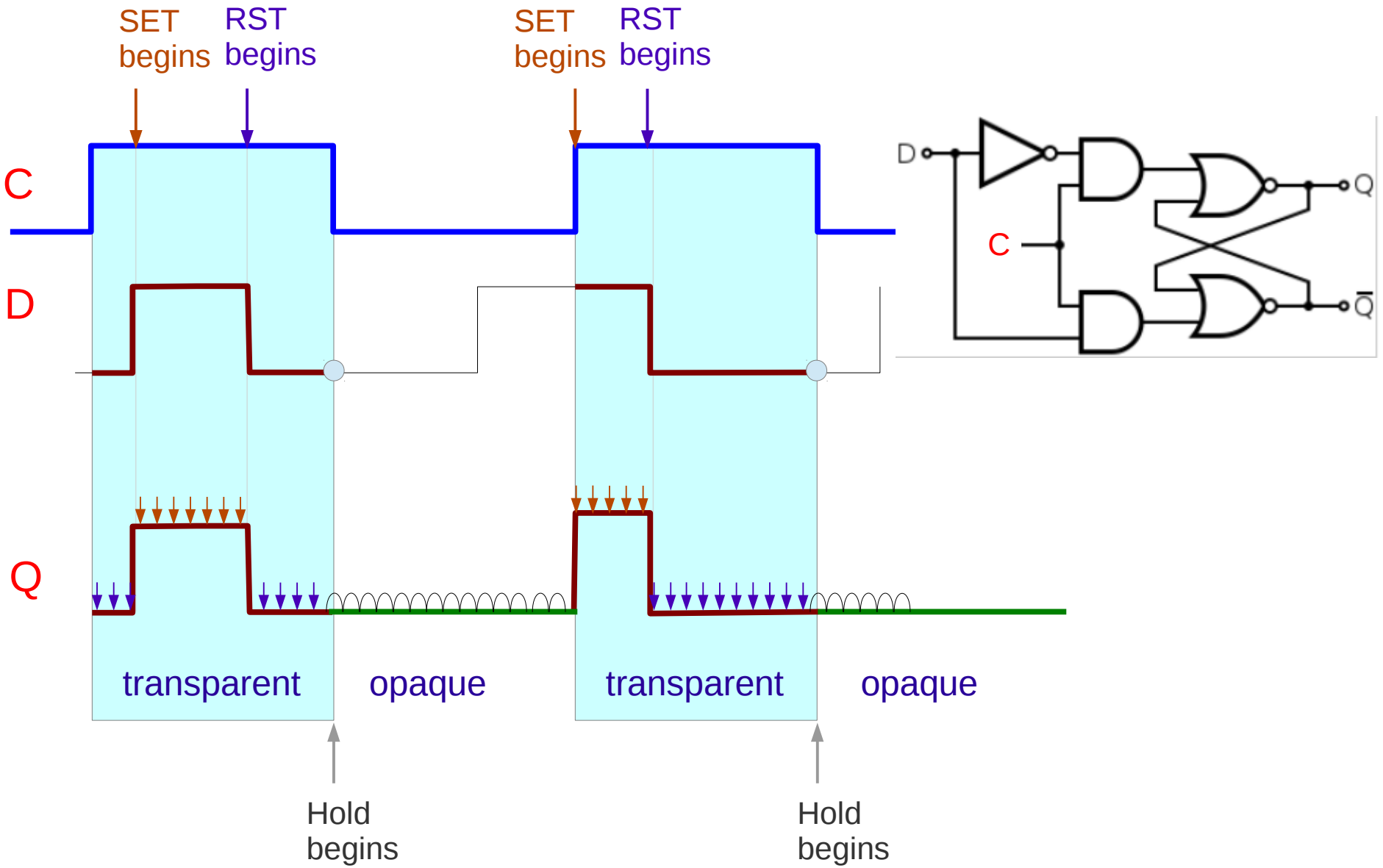




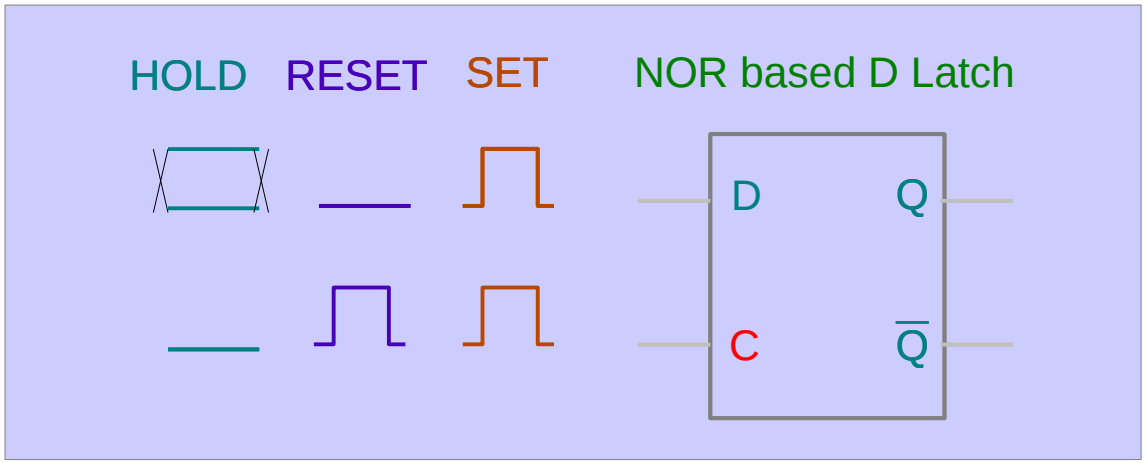


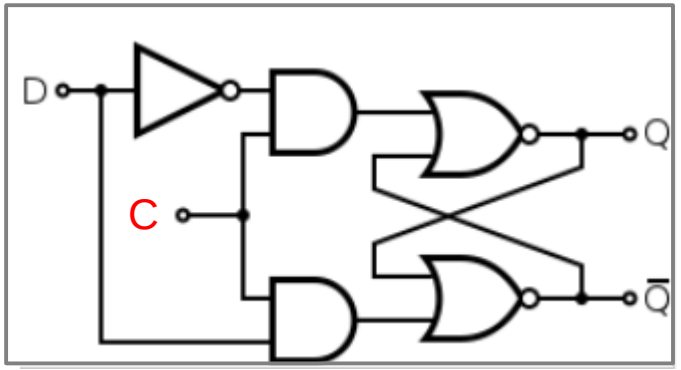
SET	S=1 R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=1	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

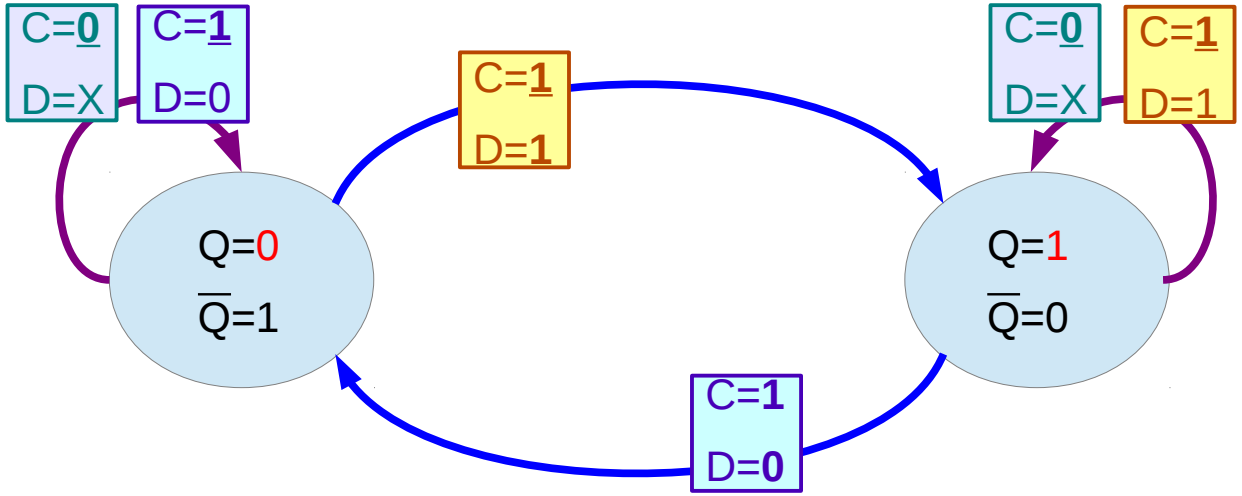
NOR-based D Latch



NOR-based D Latch

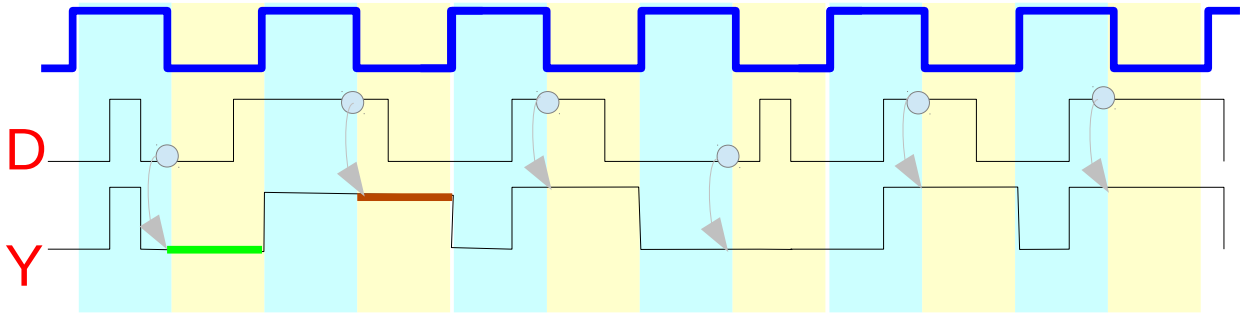


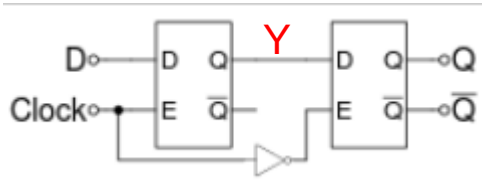




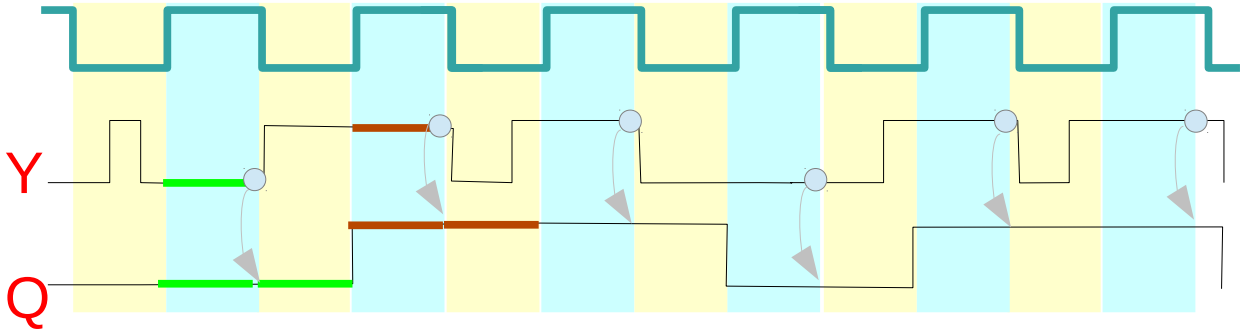
Master-Slave D FlipFlop

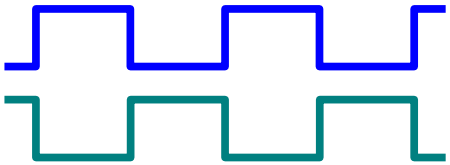
Master D Latch

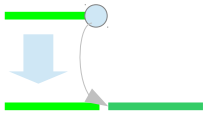




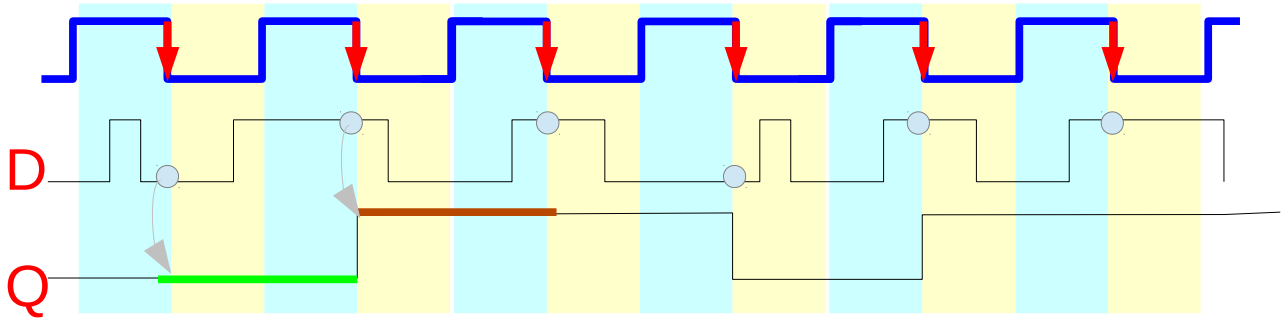
Slave D Latch







Master-Slave D F/F

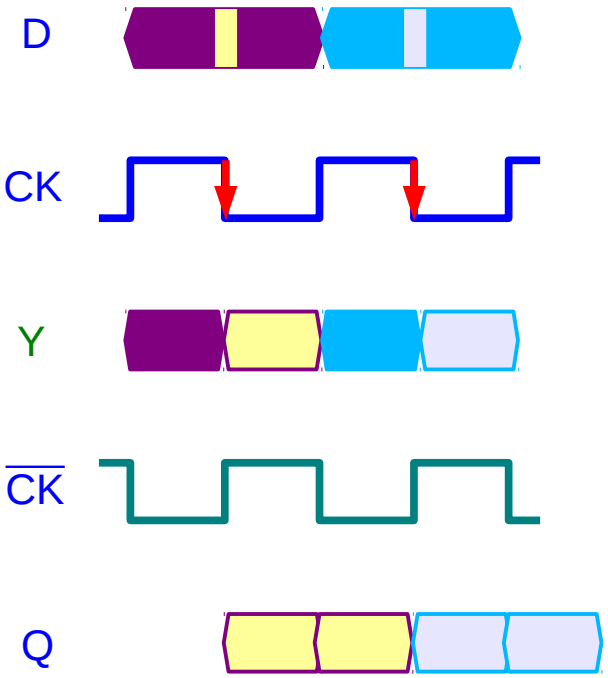


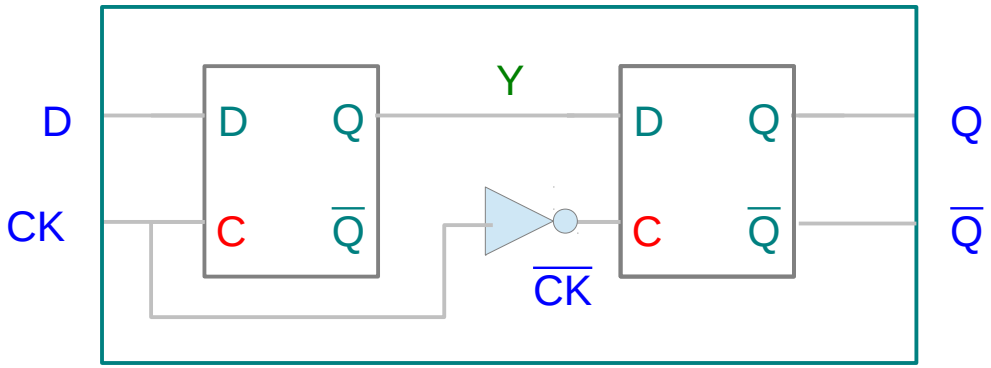
the hold output of the master is transparently reaches the output of the slave

this value is held for another half period

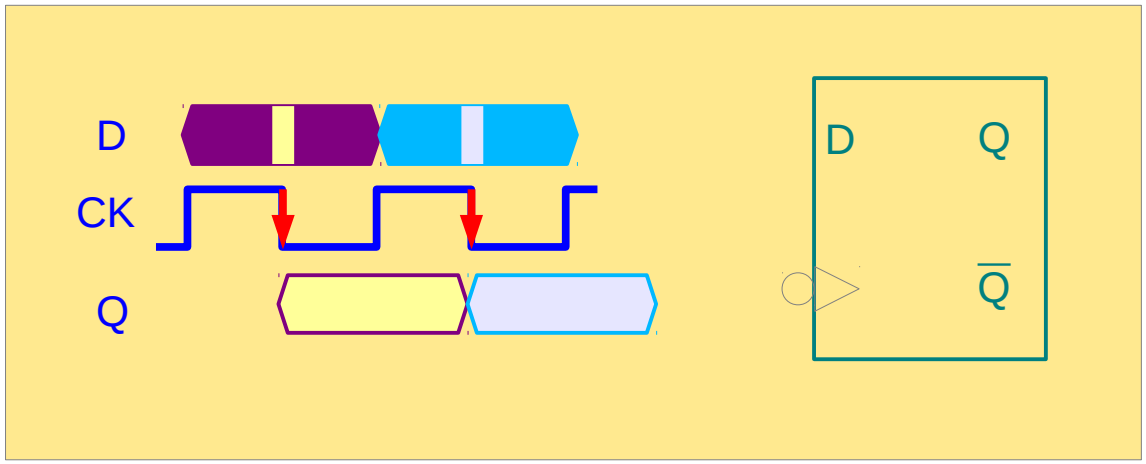
Master-Slave D FlipFlop – Falling Edge

Master D Latch



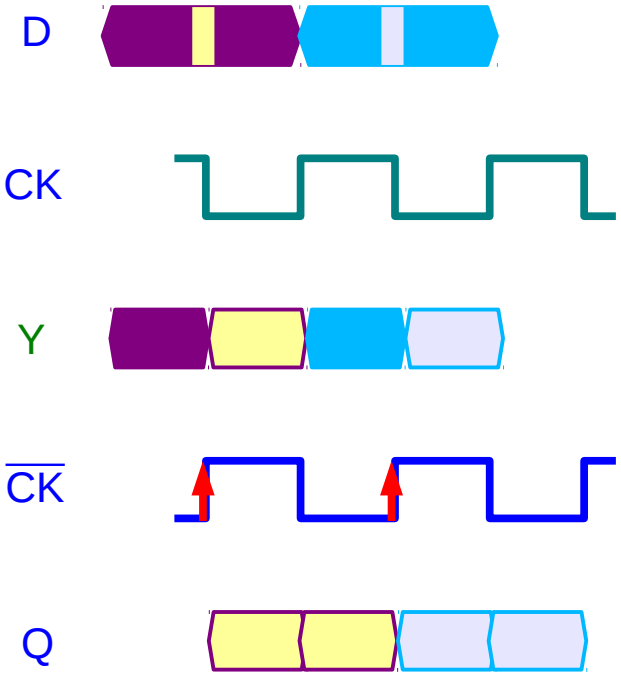


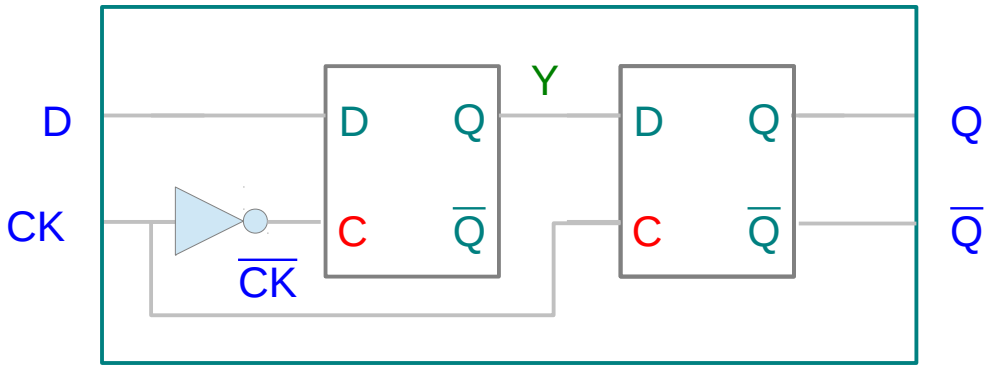
Slave D Latch



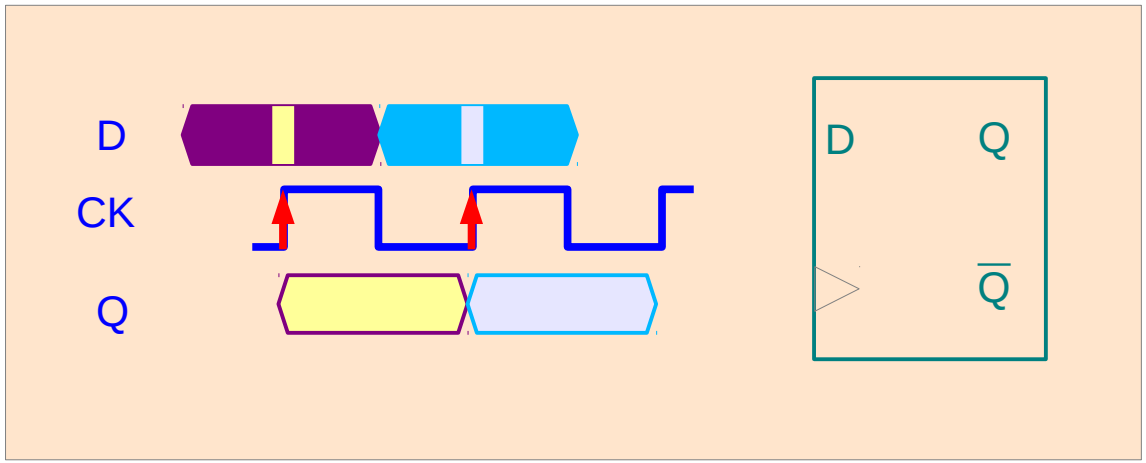
Master-Slave D FlipFlop – Rising Edge

Master D Latch





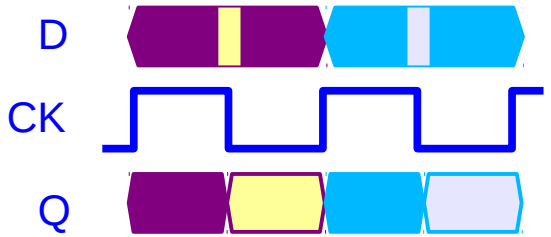
Slave D Latch

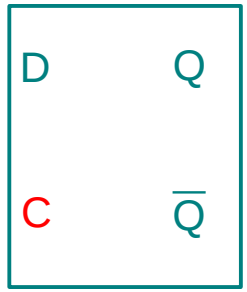


D Latch & D FlipFlop

Level Sensitive D Latch

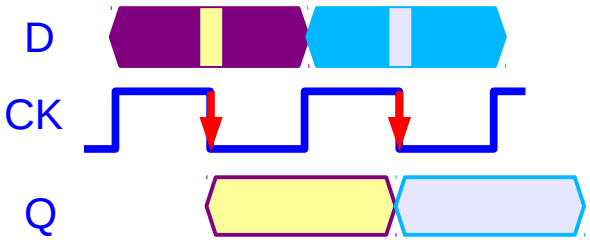
CK=1 transparent
CK=0 opaque

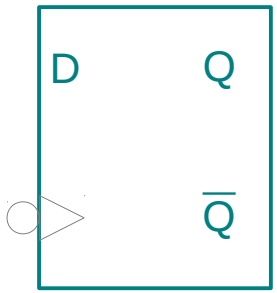




Edge Sensitive D FlipFlop

CK=1→0 transparent
else opaque





Resolution Time

References

[1] http://en.wikipedia.org/
[2] M. M. Mano, C. R. Kime, “Logic and Computer Design Fundamentals”, 4th ed.
[3] J. Stephenson, Understanding Metastability in FPGAs. Altera Corporation white paper. July 2009.