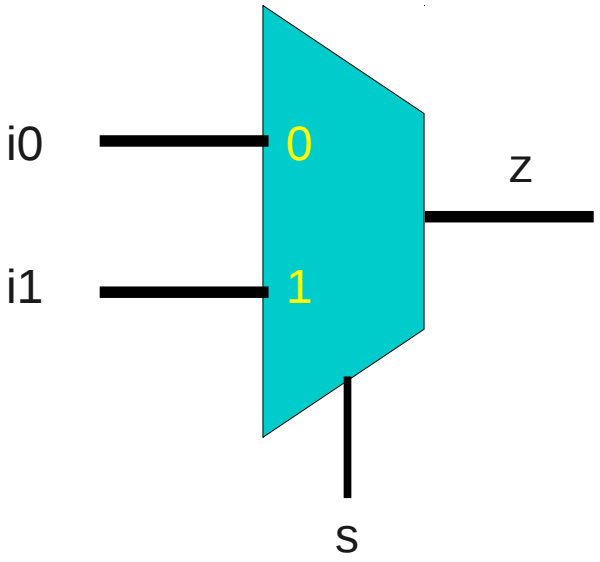
Design Levels of Abstraction : Overview

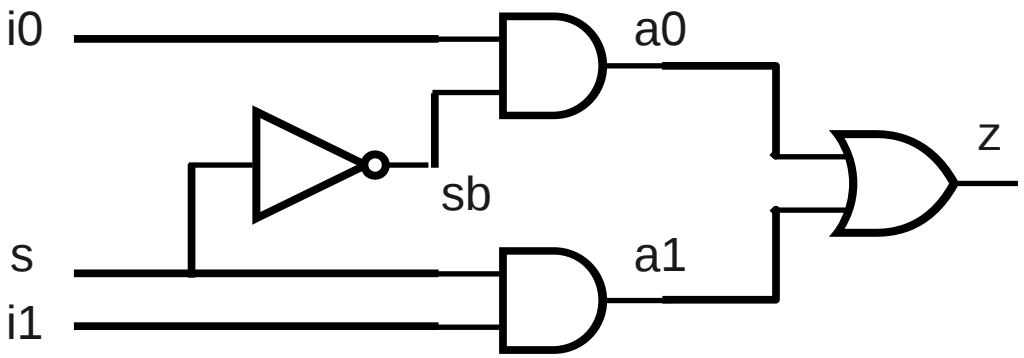
- Gate-level Modeling
- Dataflow Modeling
- Behavioral Modeling
- Structural Modeling

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

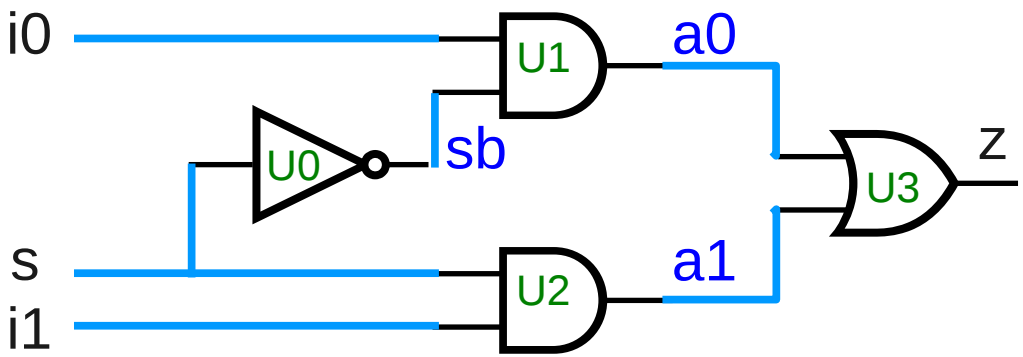
This document was produced by using OpenOffice and Octave.

2-to-1 Multiplexer Example





Gate-level Modeling



Values are continuously driven by an output of a device

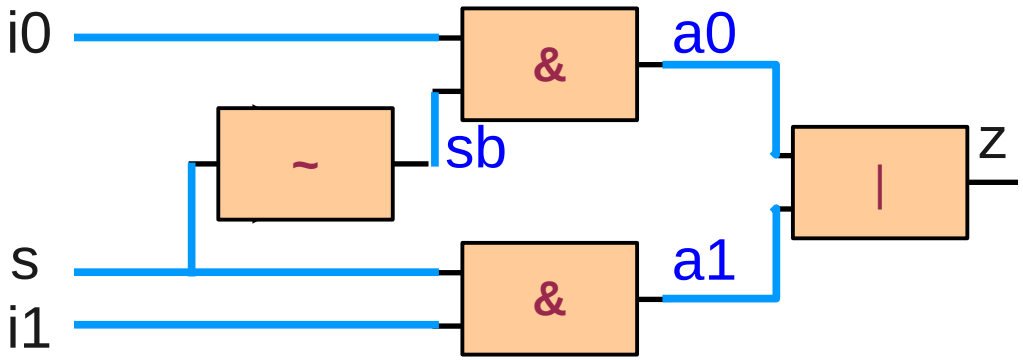
wire

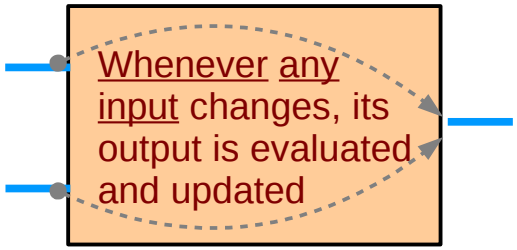
always active driving a 0, 1, x, z

```
not  U0 (sb, s);

and  U1 (a0, i0, sb),

     U2 (a1, i1, s);

or   U3 (z, a0, a1);
```

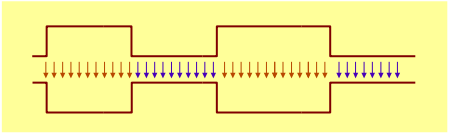
```
wire sb;

wire a0;

wire a1;

wire z;
```

Dataflow Modeling



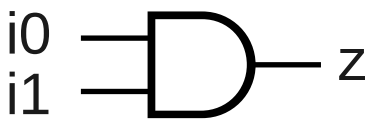


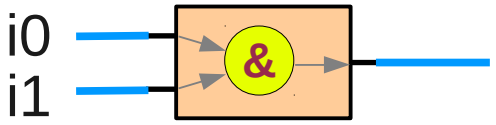
Continuous Assignment



```
not  U0 (sb, s);          ➡  assign sb = ~s;          wire sb;

and  U1 (a0, i0, sb),     ➡  assign a0 =  i0 & sb;    wire a0;

     U2 (a1, i1, s);      ➡  assign a1 =  i1 & s;     wire a1;

or   U3 (z, a0, a1);      ➡  assign z  =  a0 | a1;    wire z;
```

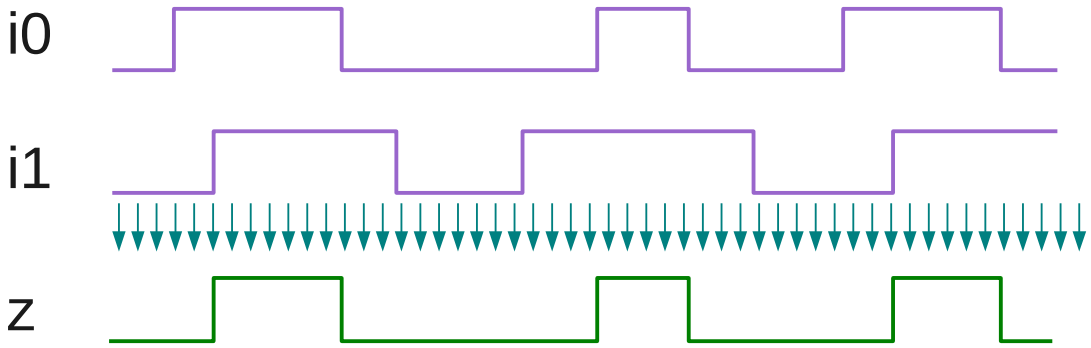
Continuous Assignment



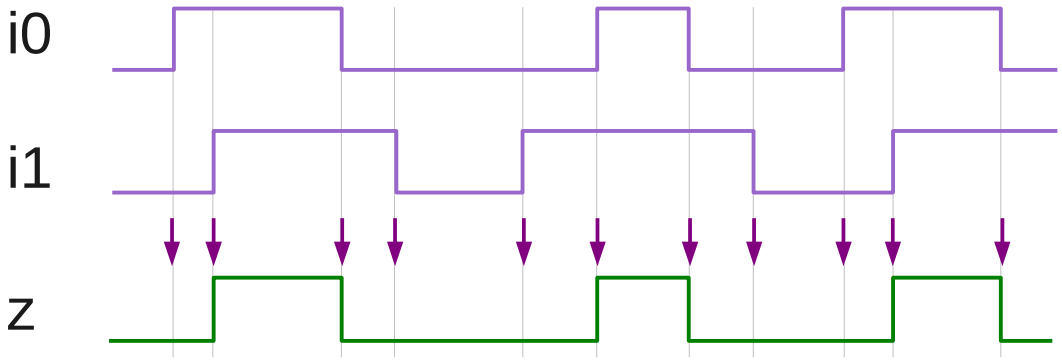



```
assign z =  i0 & i1;
```

Continuous Assignment



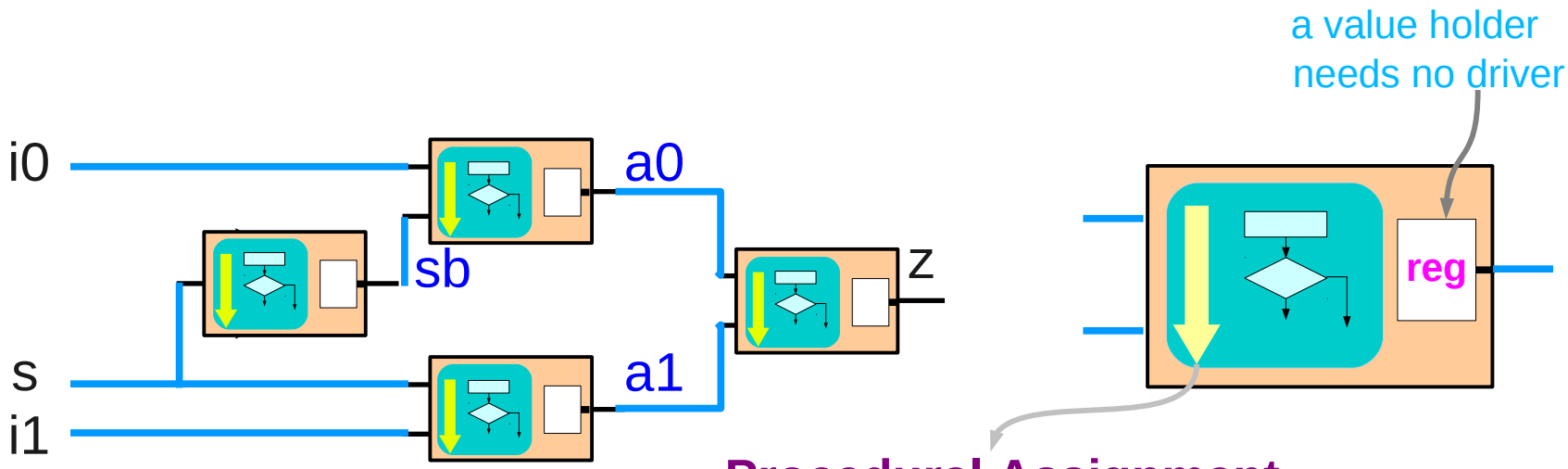
The output z always has the value of i0 **and** i1



Whenever inputs change, evaluate and propagate its output

The output z always has the value of i0 **and** i1

Behavioral Modeling – Combinational




```
assign sb = ~s;
```

```
always @(s)
      sb = ~s;
```

```
reg sb;
```

```
assign a0 =  i0 & sb;
```

```
always @(i0 or sb)
      a0 =  i0 & sb;
```

```
reg a0;
```

```
assign a1 =  i1 & s;
```

```
always @(i1 or s)
      a1 =  i1 & s;
```

```
reg a1;
```

```
assign z   =  a0 | a1;
```

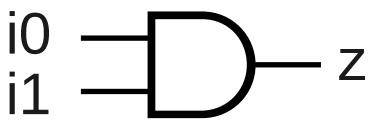
```
always @(a0 or a1)
      z =  a0 | a1;
```

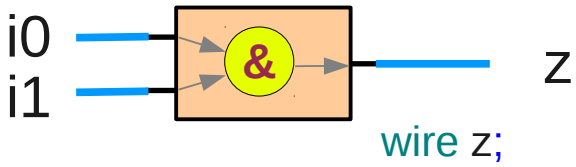
```
reg z;
```

Dataflow Modeling

Behavioral Modeling

Procedural Assignment





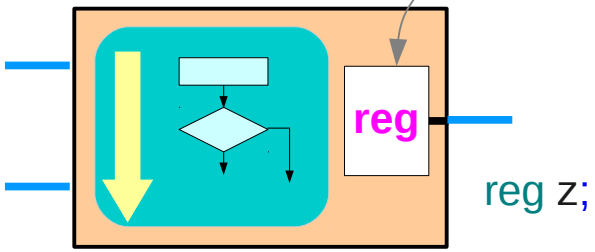

```
assign z =  i0 & i1;
```

Continuous Assignment

Whenever inputs change,
evaluate and propagate its output

a value holder
needs no driver

not a flipflop



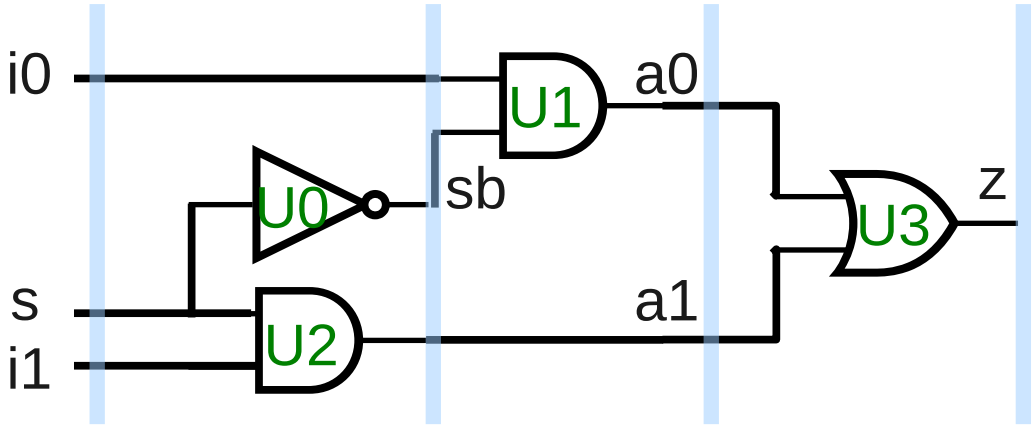

```
always @(i0 or sb)
    z =  i0 & i1;
```

Procedural Assignment

within always, initial

combined with if ( ) then – else –

Simulation




```
assign sb = ~s;
```

```
always @(s)
    sb = ~s;
```

```
whenever s changes
    sb = ~s;
```

```
assign a0 =  i0 & sb;
```

```
always @(i0 or sb)
    a0 =  i0 & sb;
```

```
whenever i0 or sb change
    a0 =  i0 & sb;
```

```
assign a1 =  i1 & s;
```

```
always @(i1 or s)
    a1 =  i1 & s;
```

```
whenever i1 or s change
    a1 =  i1 & s;
```

```
assign z  =  a0 | a1;
```

```
always @(a0 or a1)
    z =  a0 | a1;
```

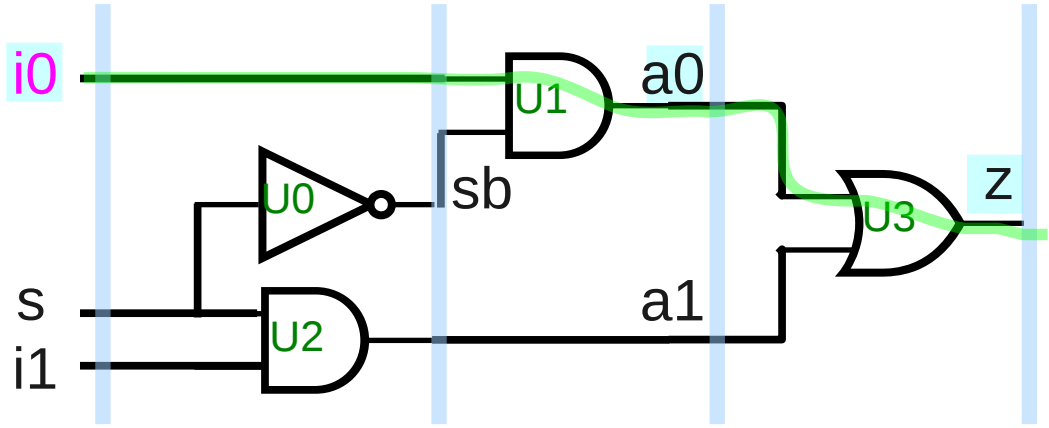
```
whenever a0 or a1 change
    z =  a0 | a1;
```

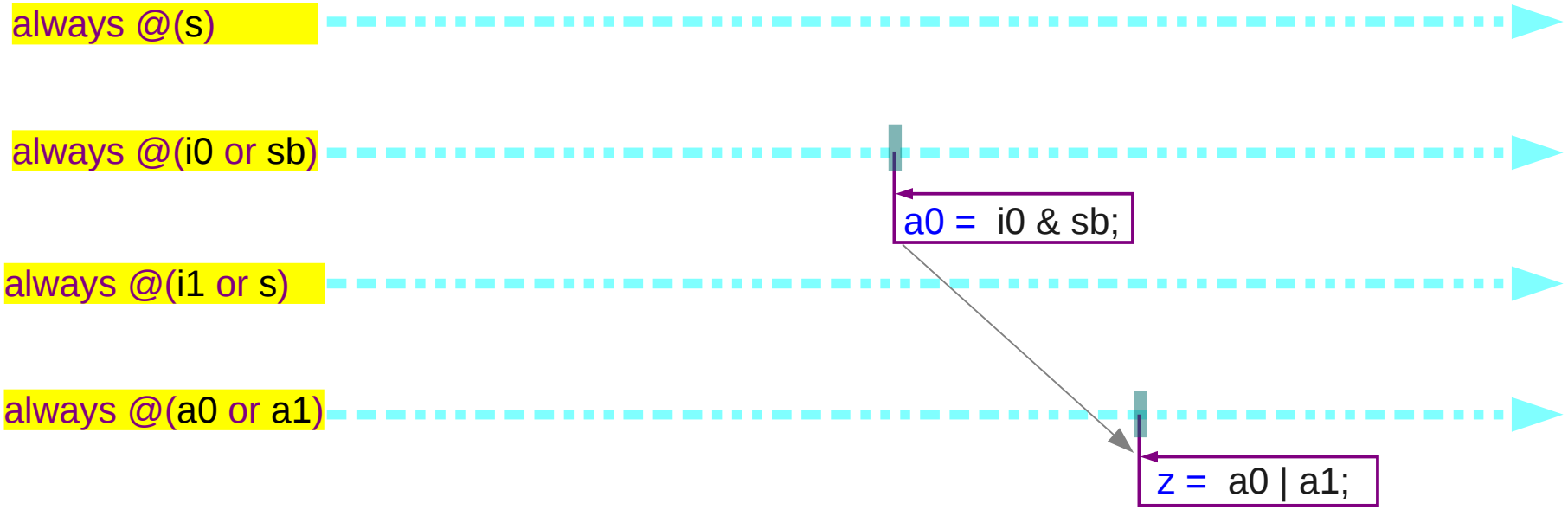
Dataflow Modeling

Behavioral Modeling

When i0 changes

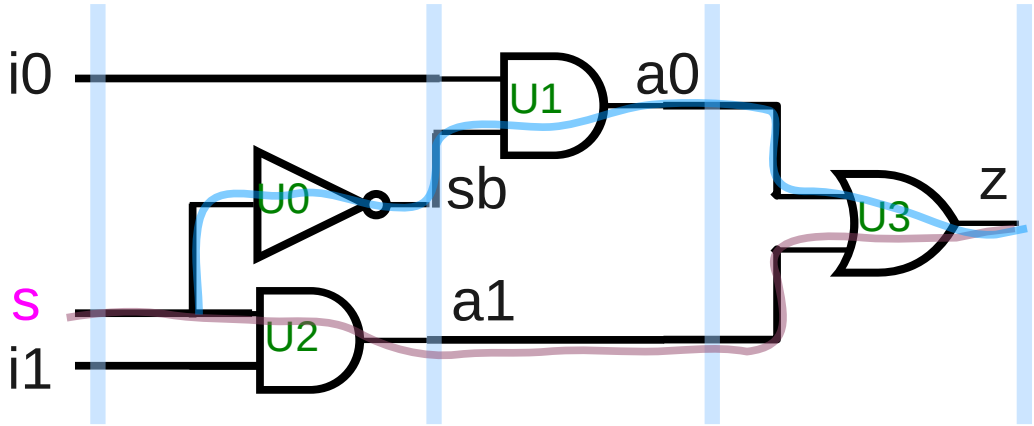
```
always @(s)
      sb = ~s;

always @(i0 or sb)
      a0 =  i0 & sb;

always @(i1 or s)
      a1 =  i1 & s;

always @(a0 or a1)
      z =  a0 | a1;
```

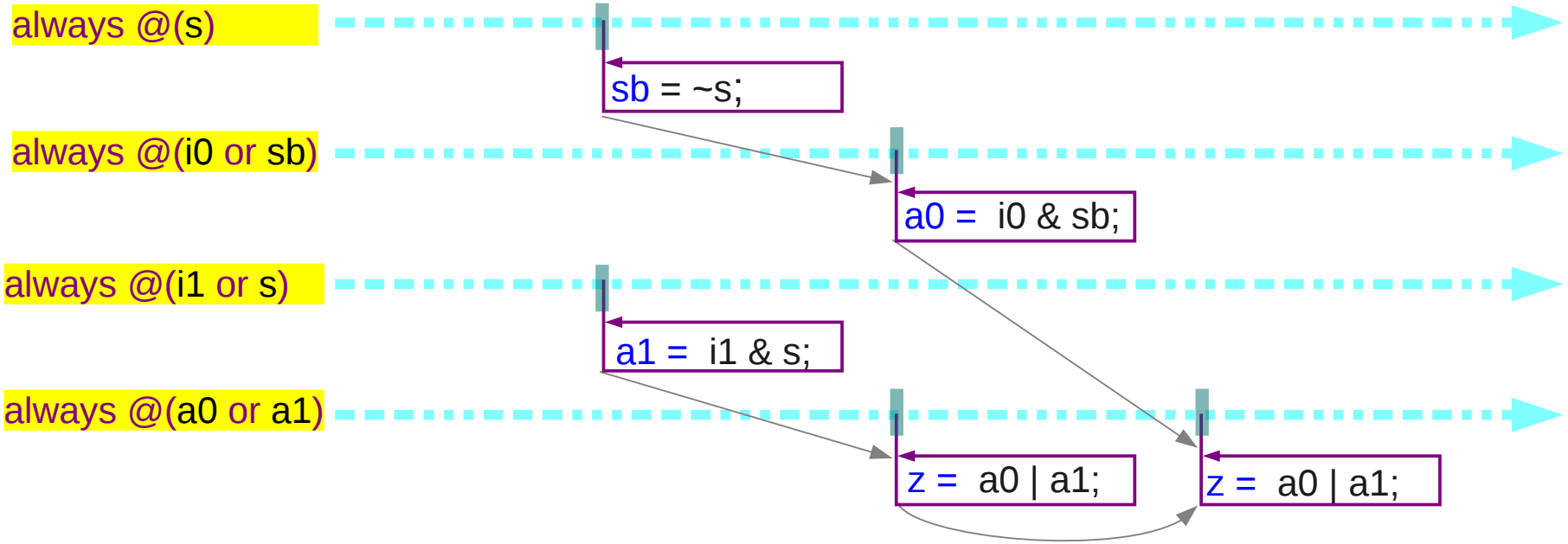




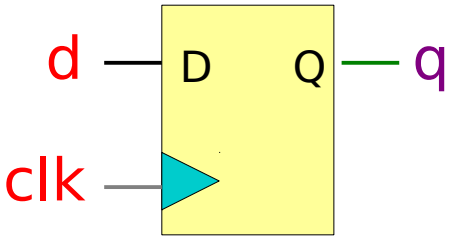
When s changes

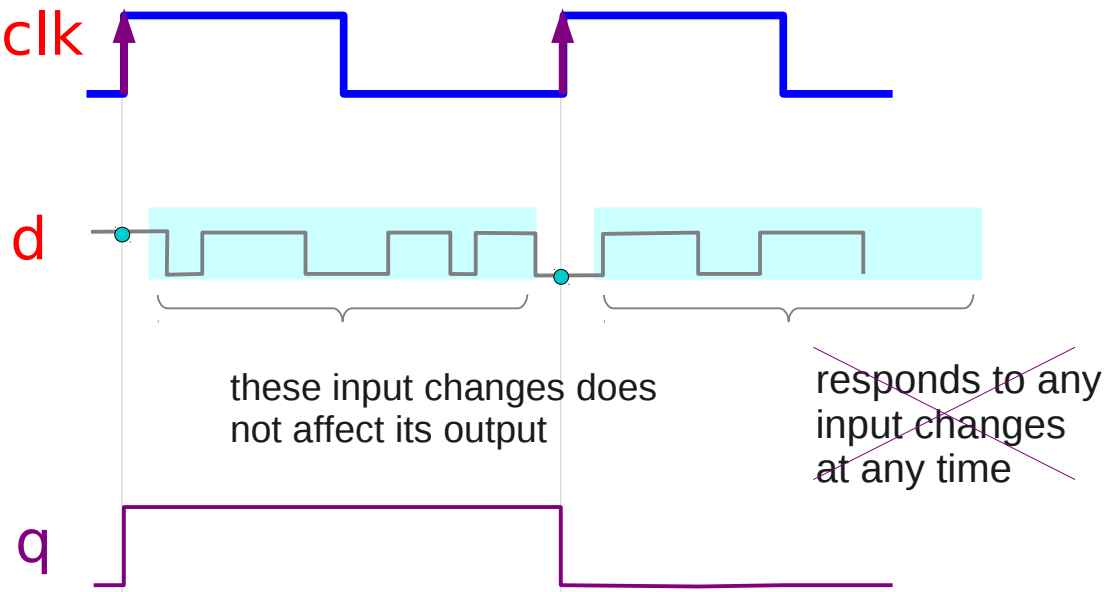
```
always @(s)
      sb = ~s;

always @(i0 or sb)
      a0 =  i0 & sb;

always @(i1 or s)
      a1 =  i1 & s;

always @(a0 or a1)
      z =  a0 | a1;
```





Behavioral Modeling – Sequential

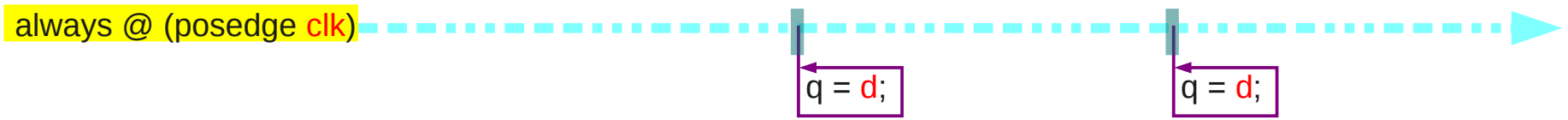




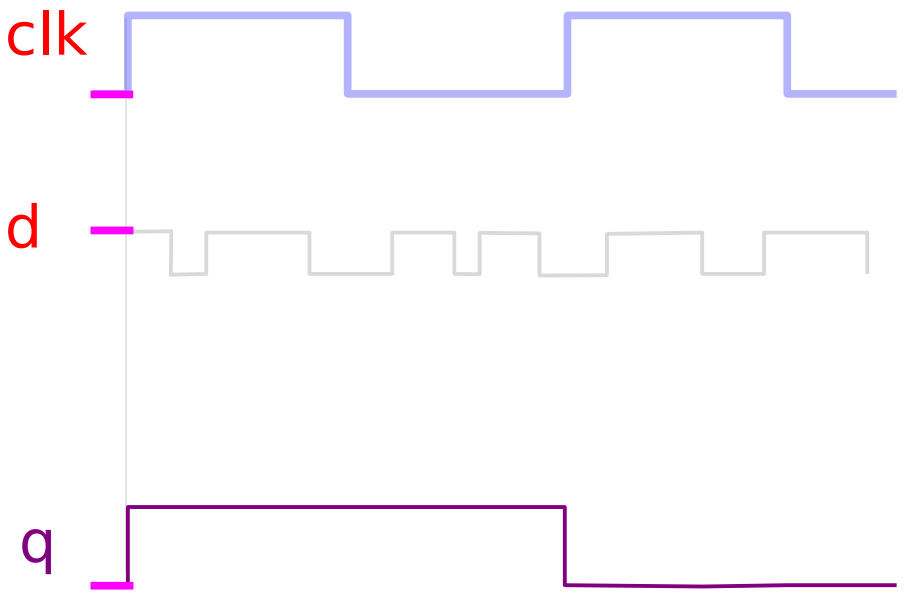
Only sensitive to a
subset of of their inputs
– sensitivity list

the two inputs (clk, d)

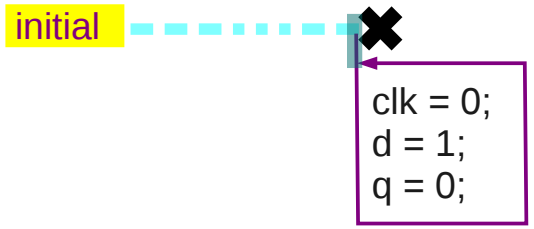
```
always @(posedge clk)
  q = d;
```



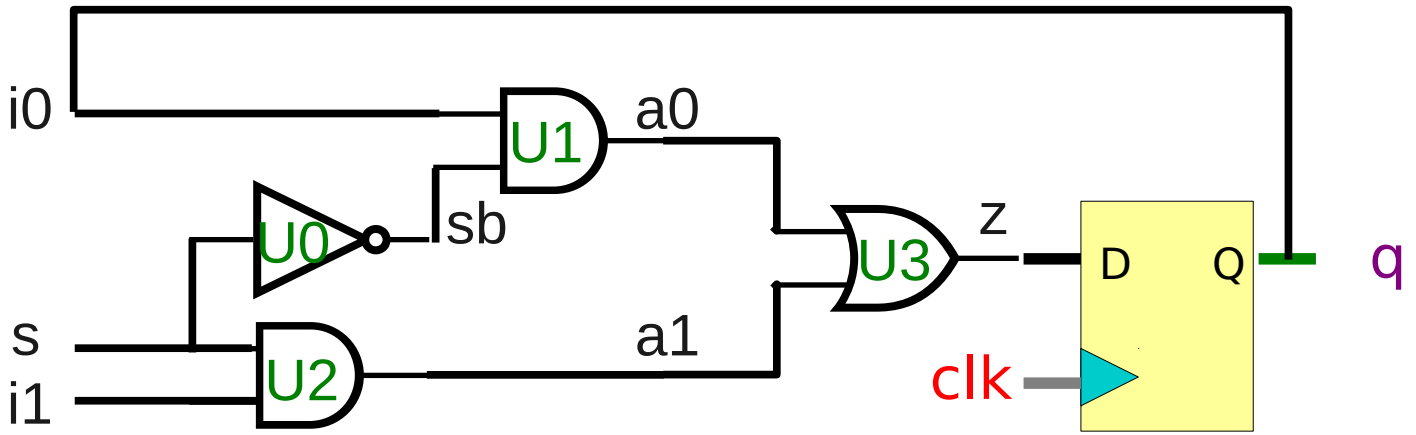
Behavioral Modeling – Initialization




```
initial
  begin
        clk = 0;
        d = 1;
        q = 0;
  end
```



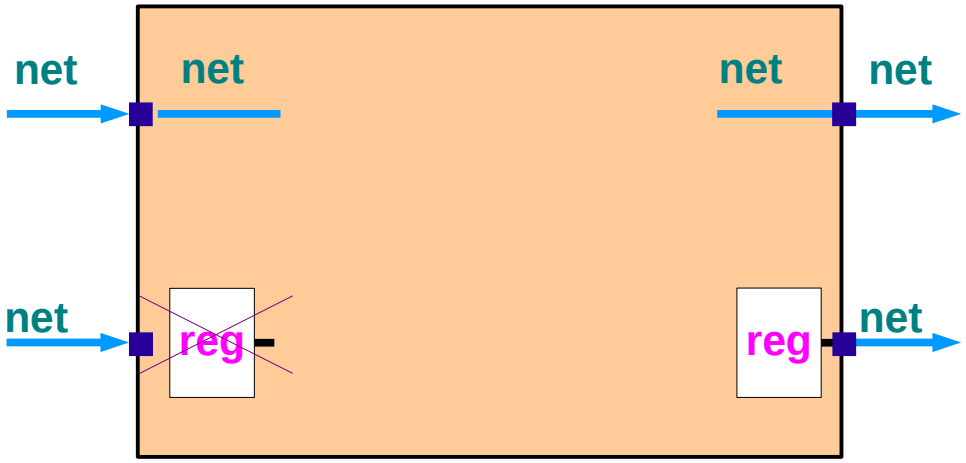
Parallel Processes




```
always @(s)
      sb = ~s;

always @(i0 or sb)
      a0 =  i0 & sb;

always @(i1 or s)
      a1 =  i1 & s;

always @(a0 or a1)
      z =  a0 | a1;

always @(posedge clk)
      q = d;
```

Five Parallel Processes

Structural Modeling



wire tri	wand wor triand trior	tri0 tri1	supply0 supply1	trireg

Sequential Assignment (2)

References

[1] http://en.wikipedia.org/
[2] T.R. Padmanabhan, B.T. Sundari, “Design Through Verilog HDL
[3] D.E. Thomas, P.R. Moorby, “The Verilog Hardware Description Language”, 3rd ed