Path Delay

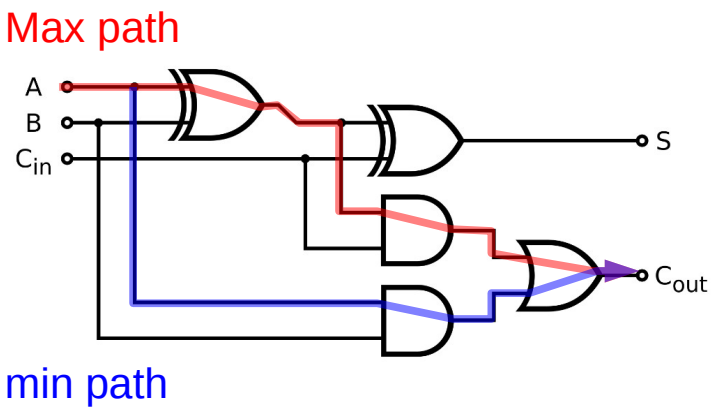
Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Path Delay

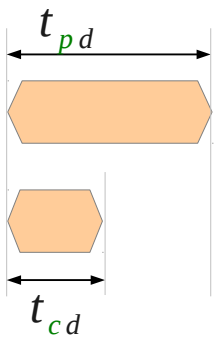
Max-Path
Min-Path
Critical Path
Timing Check
False Path
Multi-Cycle Path

Max Path / Min Path



Max delay

min delay

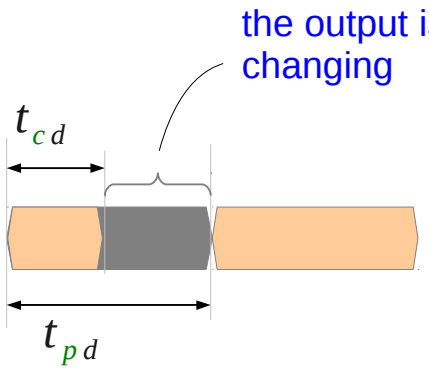


contamination delay

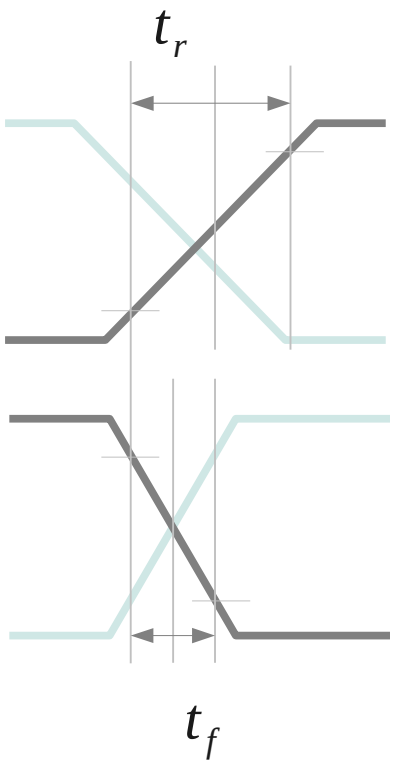
propagation delay

$$t_{cd} \leq t_{delay} \leq t_{pd}$$

min delay Max delay

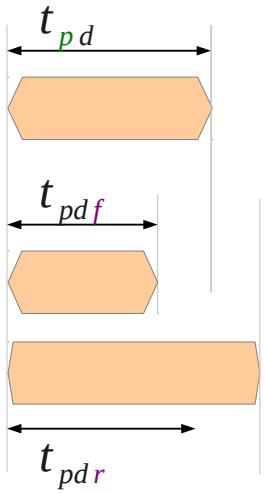


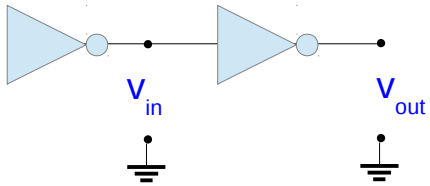
Rise / Fall Times



Max delay

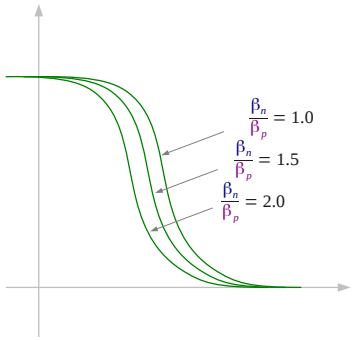
min delay

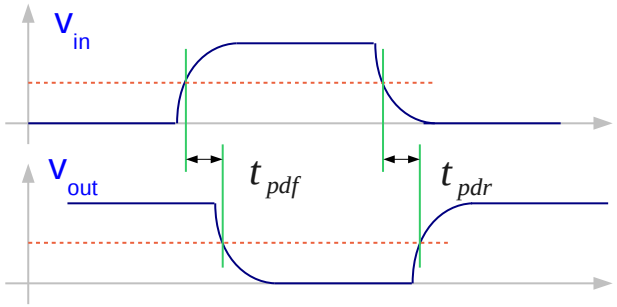




$$\frac{\beta_n}{\beta_p} > 1 \qquad \frac{R_n}{R_p} < 1$$

$$\frac{t_f}{t_r} = \frac{2.2\,\tau_n}{2.2\,\tau_p} \qquad \frac{\tau_n}{\tau_p} = \frac{R_n C_{out}}{R_p C_{out}} = \frac{R_n}{R_p} < 1$$



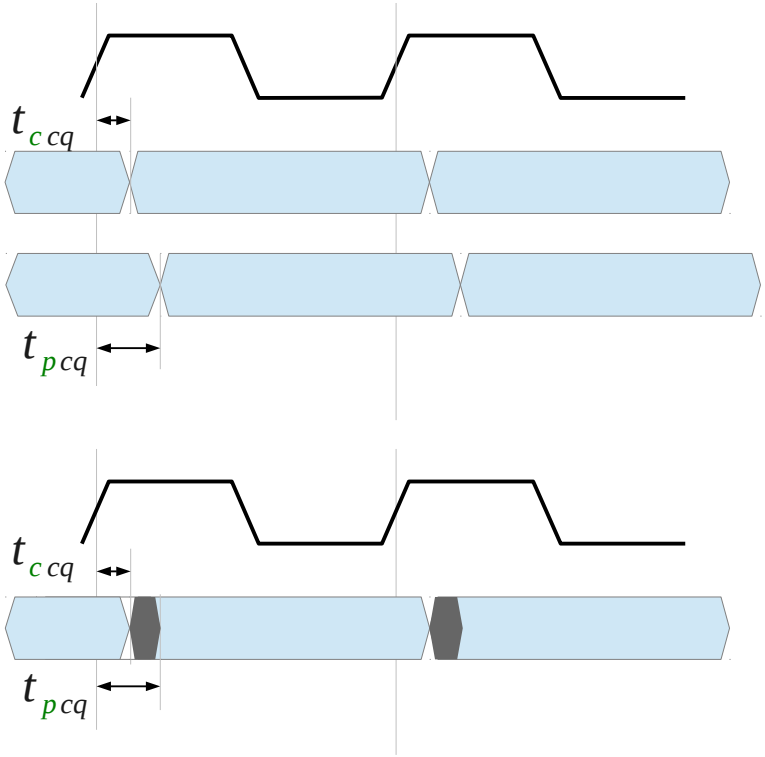


PVT Variation

- Process
- Voltage
- Temperature

High temperature	Max delay
Low temperature	min delay

FF Output Delay



contamination delay

propagation delay

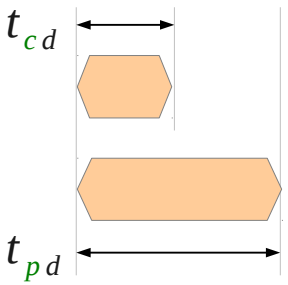
flipflop clock-to-q

$$t_{c\,cq} \leq t_{delay} \leq t_{p\,cq}$$

min delay

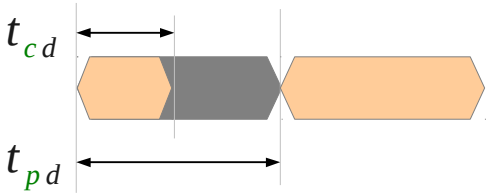
Max delay

Path Delay



contamination delay

propagation delay



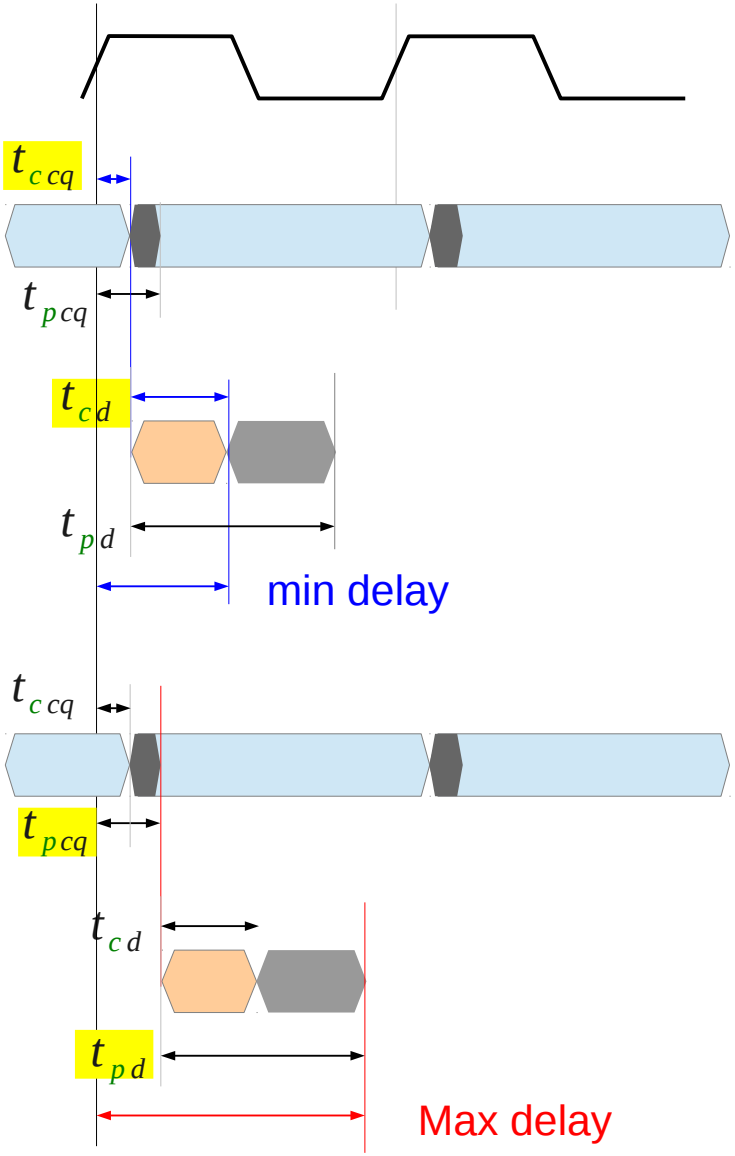
combinational logic delay

$$t_{cd} \leq t_{delay} \leq t_{pd}$$

min delay

Max delay

Reg-to-Reg Delay (1)



$$t_{c\,cq} \le t_{FF} \le t_{p\,cq}$$

min delay Max delay

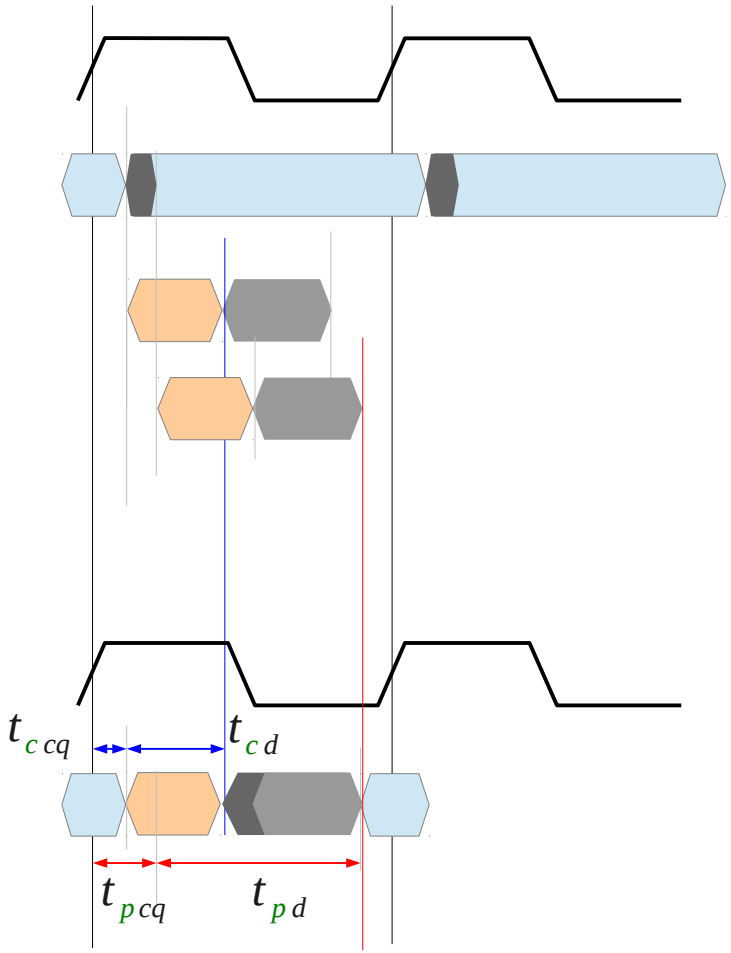
$$t_{c\,d} \le t_{comb} \le t_{p\,d}$$

min delay Max delay

$$t_{c\,cq} + t_{c\,d} \le t_{delay} \le t_{p\,cq} + t_{p\,d}$$

min delay Max delay

Reg-to-Reg Delay (2)



$$t_{c\,cq} \le t_{FF} \le t_{p\,cq}$$

min delay Max delay

$$t_{c\,d} \le t_{comb} \le t_{p\,d}$$

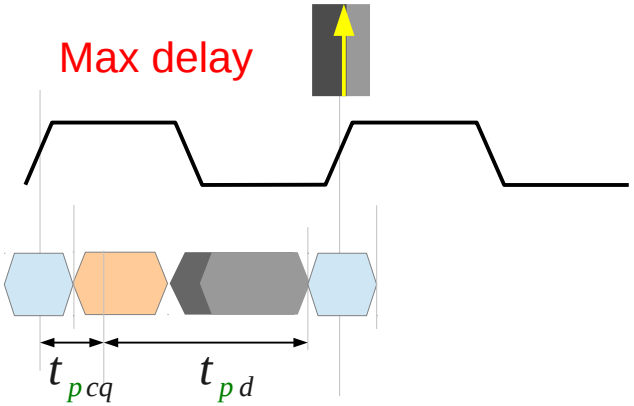
min delay Max delay

$$t_{c\,cq} + t_{c\,d} \le t_{delay} \le t_{p\,cq} + t_{p\,d}$$

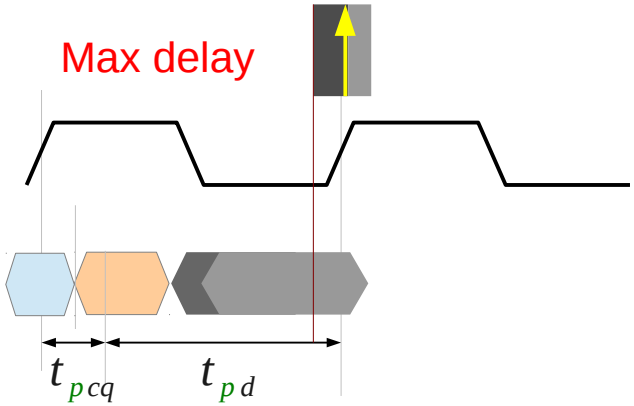
min delay Max delay

Setup Time / Hold Time

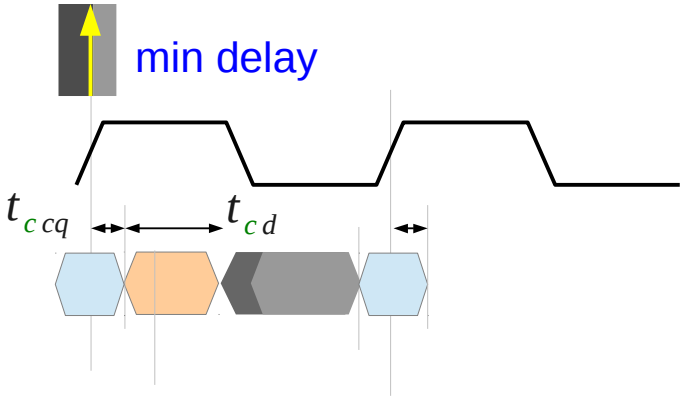
Setup Time OK



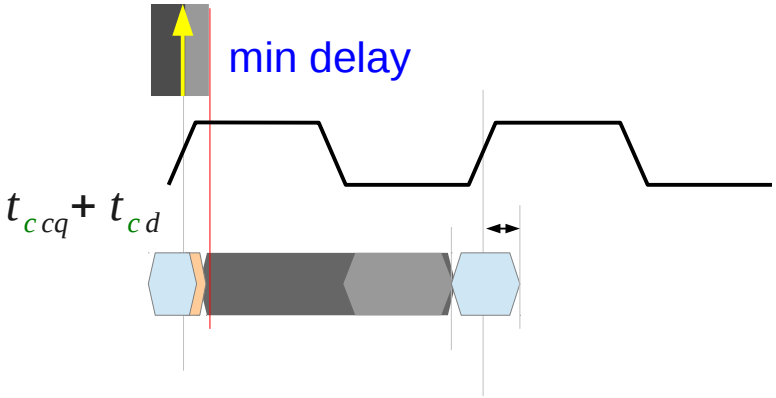
Setup Time Violation



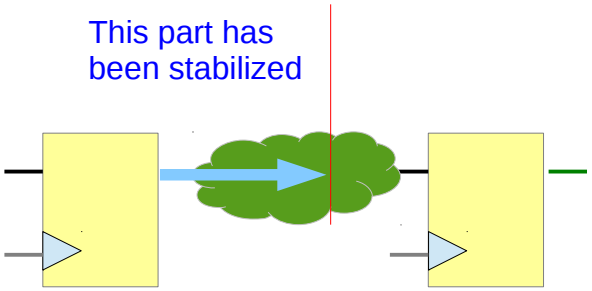
Hold Time OK



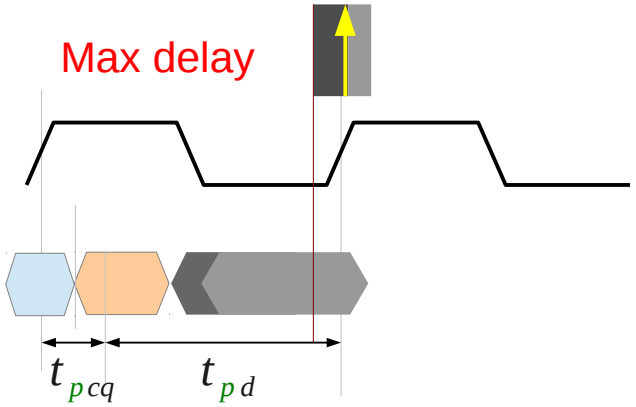
Hold Time Violation

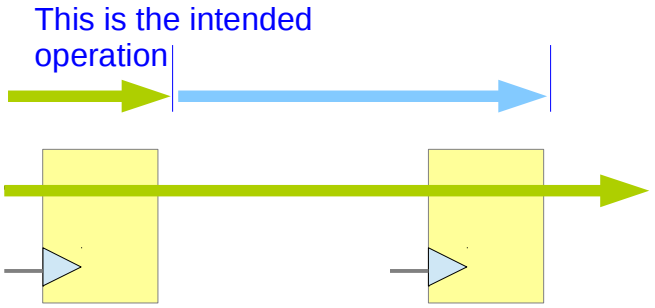


Setup Time / Hold Time



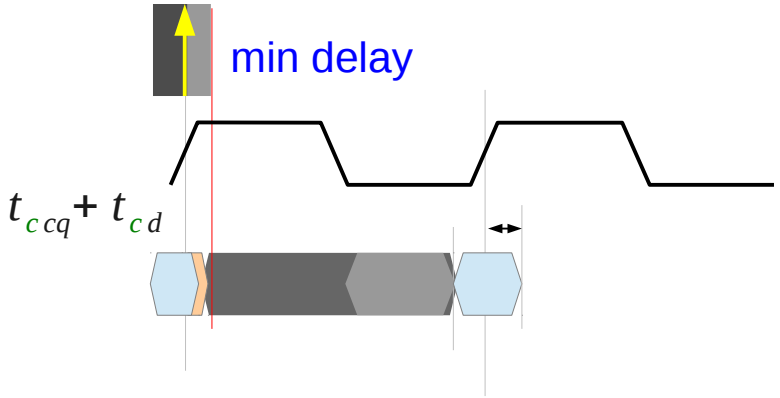
Setup Time Violation



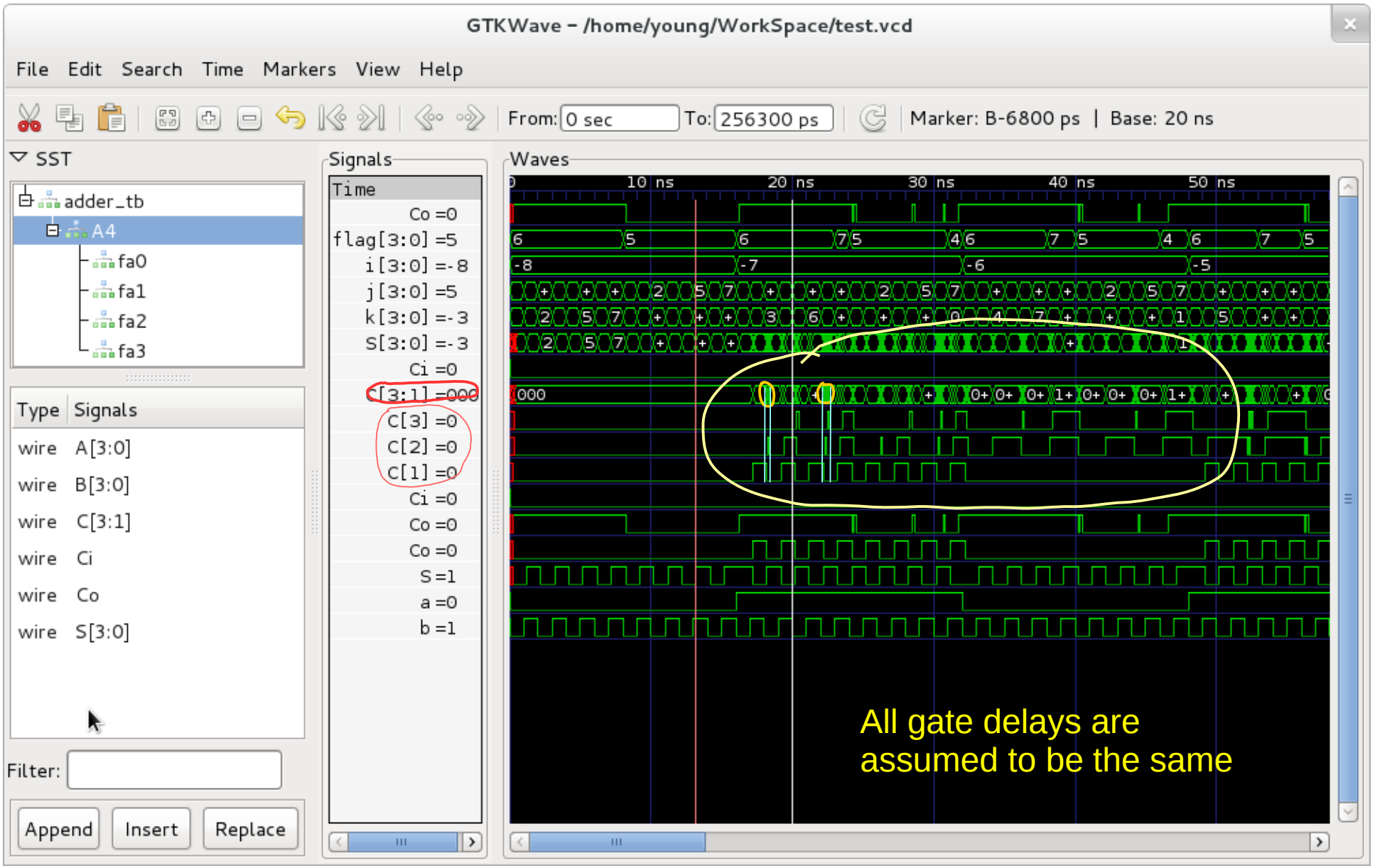


Since the delay is too
small signal passes
through the 2nd FF

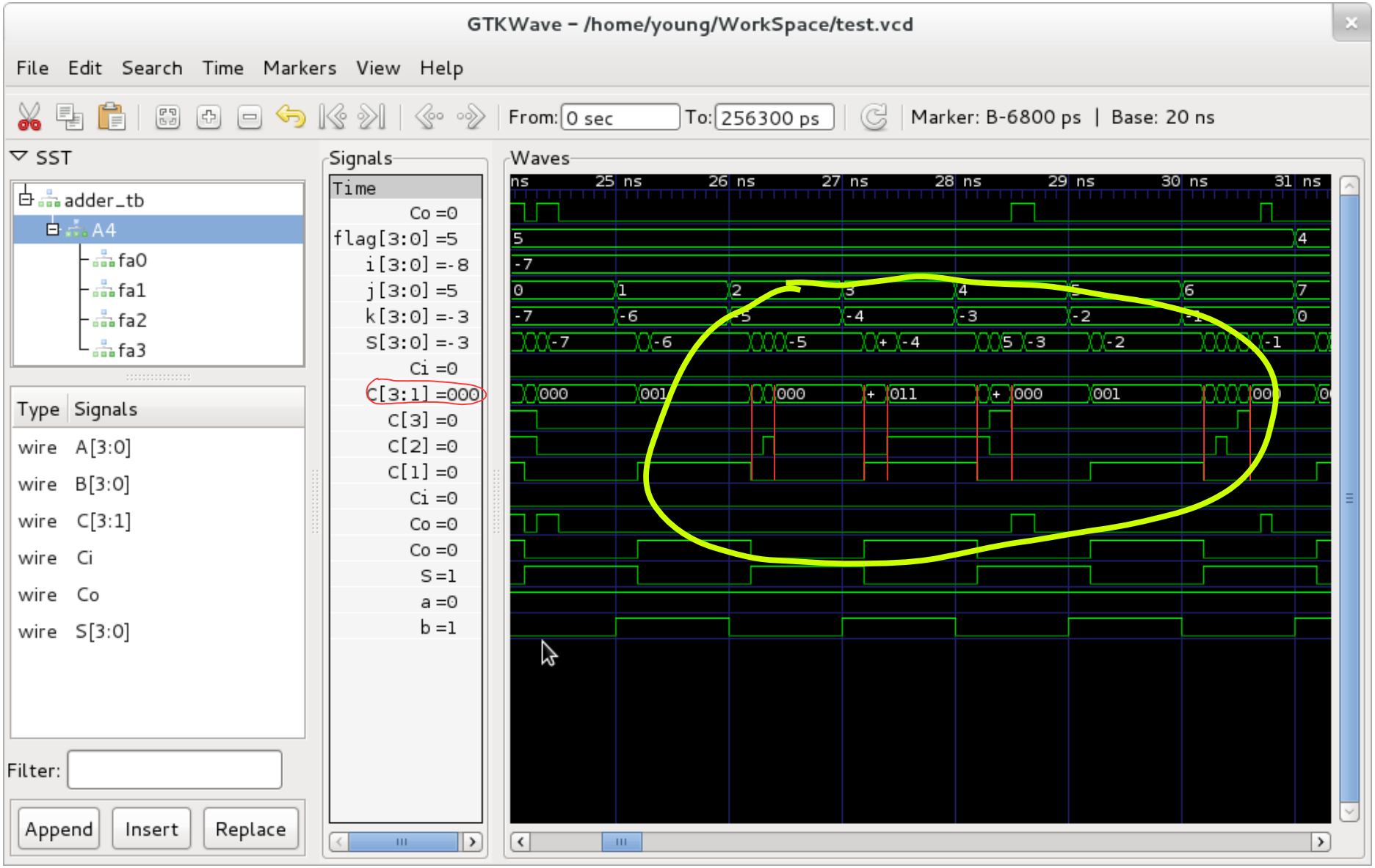
Hold Time Violation



Adder Simulation Waveform



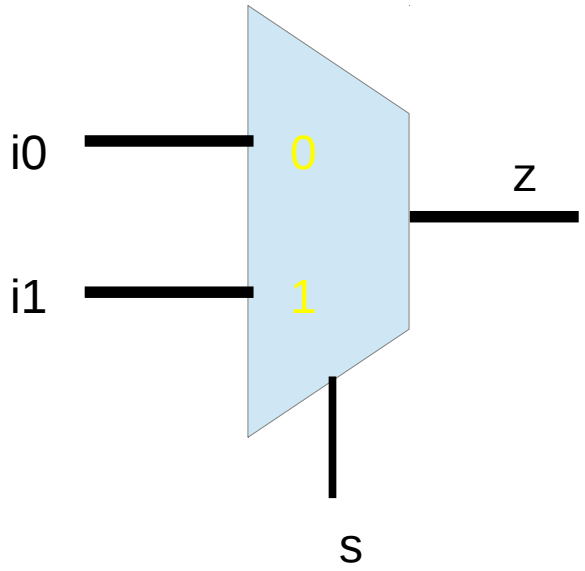
Glitches

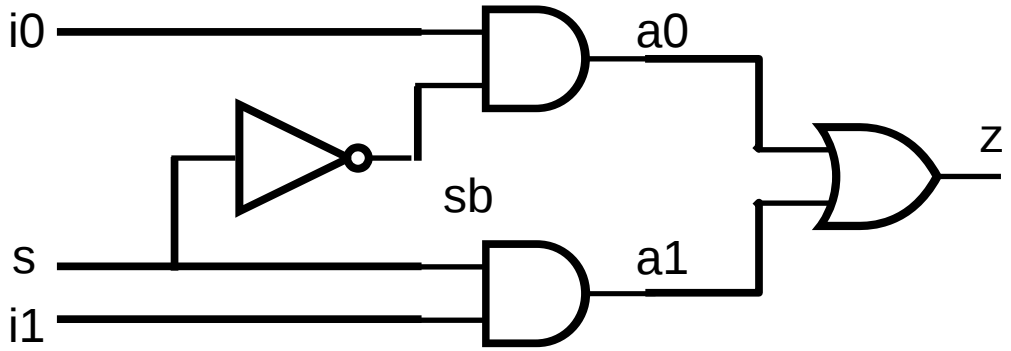

GTKWave – /home/young/WorkSpace/test.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 256300 ps
Marker: B-6800 ps | Base: 20 ns
SST
adder_tb
A4
fa0
fa1
fa2
fa3
Type Signals
wire A[3:0]
wire B[3:0]
wire C[3:1]
wire Ci
wire Co
wire S[3:0]
Filter:
Append
Insert
Replace
Signals
Time
Co =0
flag[3:0]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

False Path

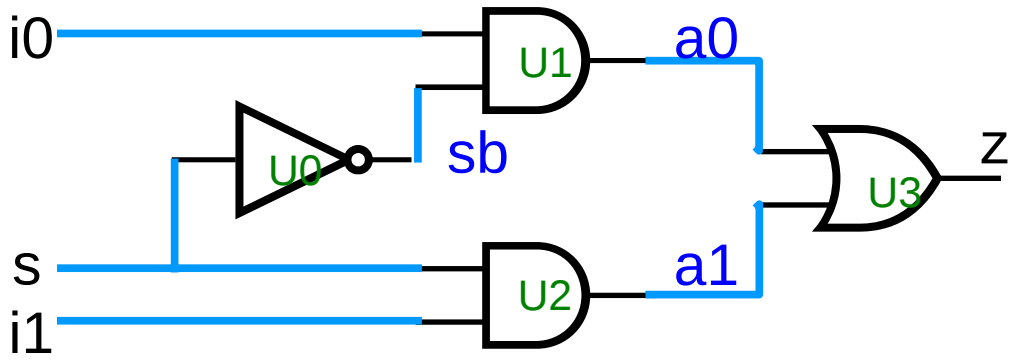
Multi-Cycle Path

Verilog Timing Model Examples





Gate-level Modeling



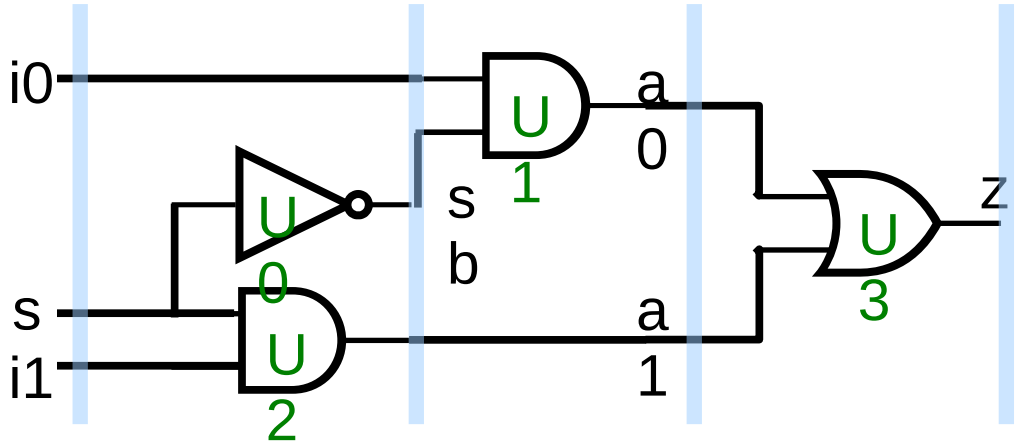
Values are continuously driven by an output of a device

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;
```

Simulation with Delta Delays




```
assign sb = ~s;

assign a0 =  i0 &
sb;

assign a1 =  i1 & s;

assign z   =  a0 |
a1;
```

Dataflow Modeling

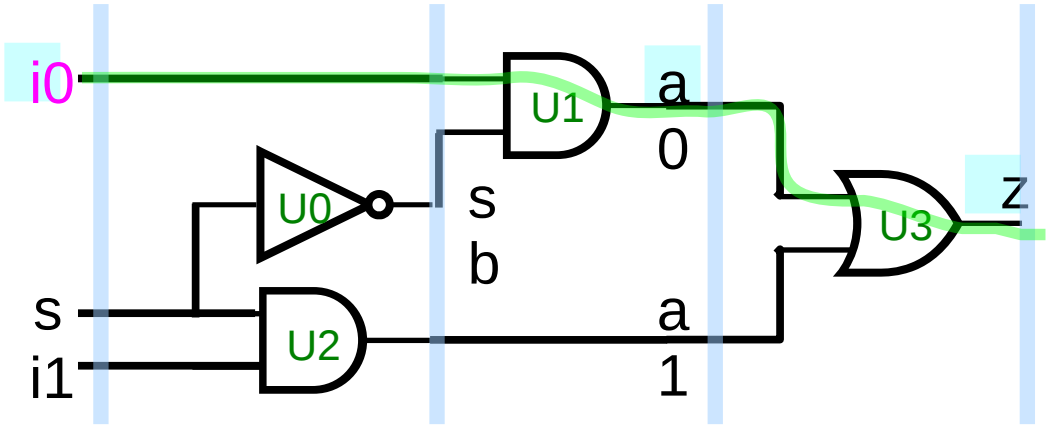
```
always @(s)
        sb = ~s;

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

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

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

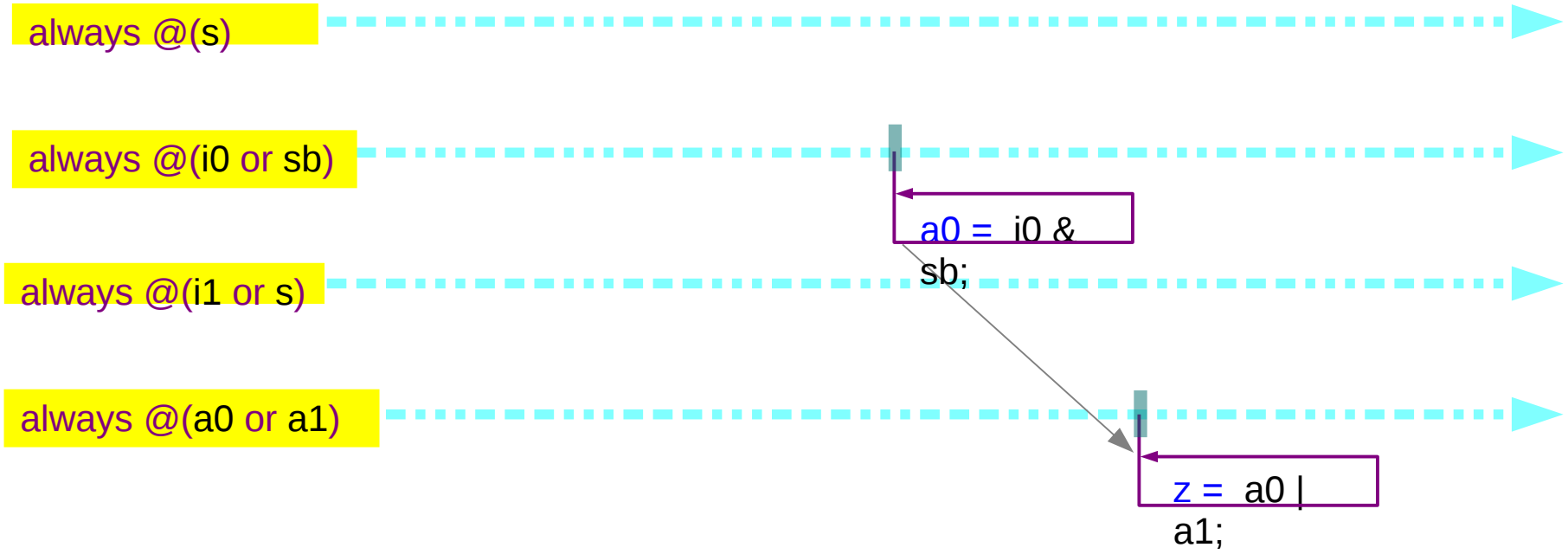
Behavioral Modeling

```
whenever s changes
        sb = ~s;

whenever i0 or sb change
        a0 =  i0 & sb;

whenever i1 or s change
        a1 =  i1 & s;

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

When i0 changes

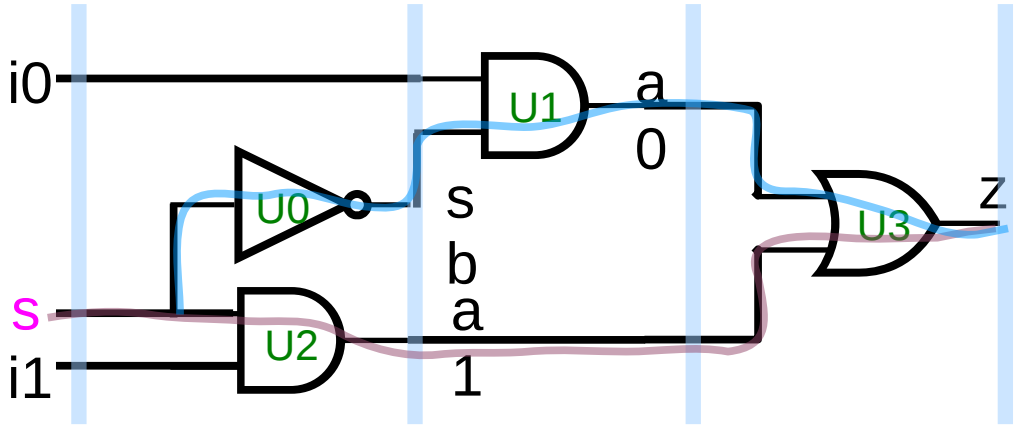
```
always @(s)
      sb = ~s;

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

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

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

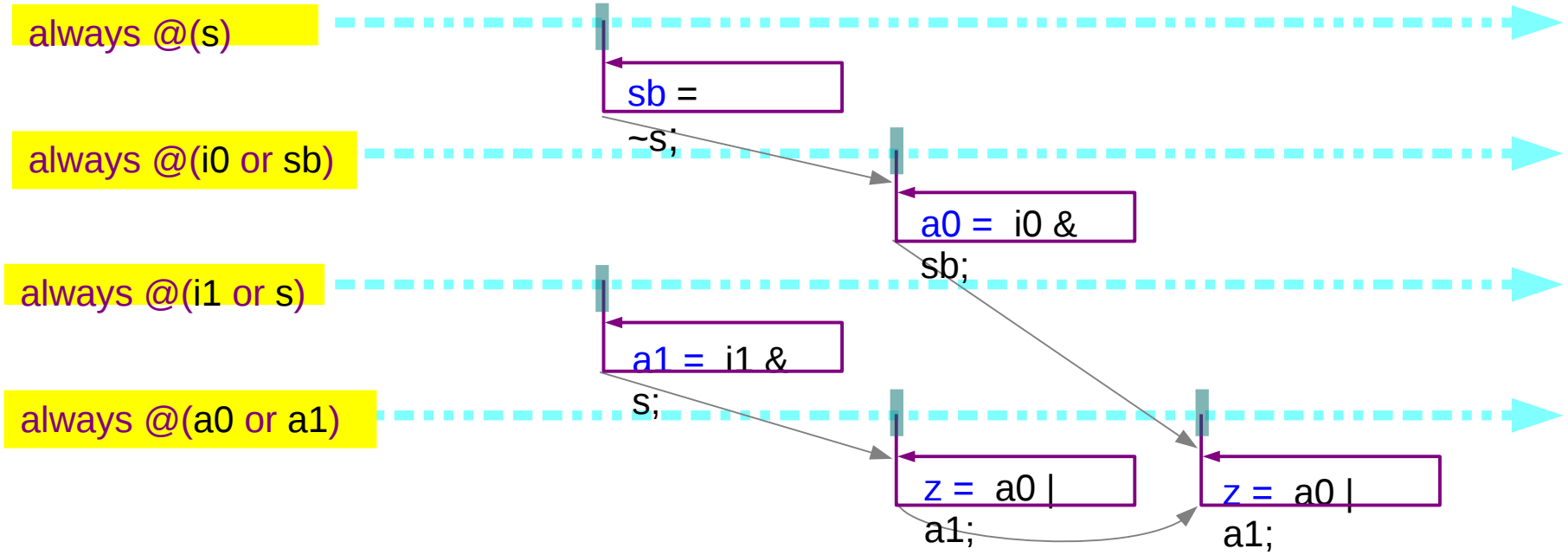




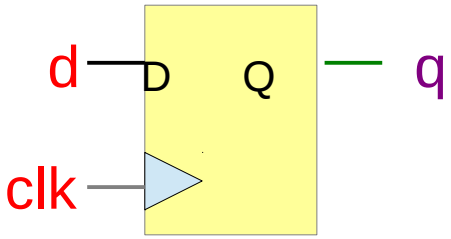
When s changes

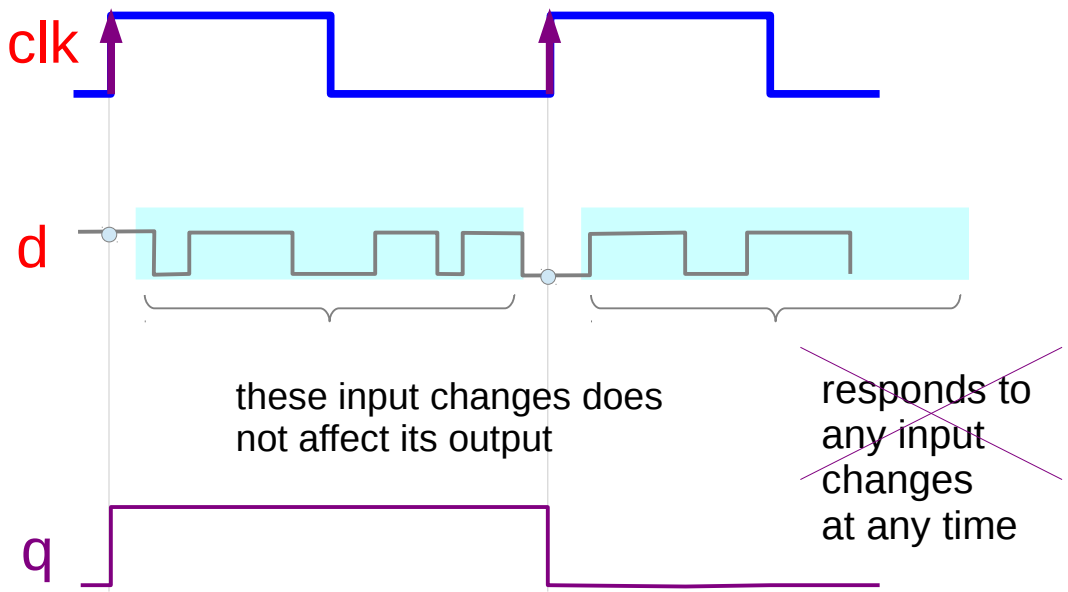
```
always @(s)
        sb = ~s;

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

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

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





Behavioral Modeling – Sequential

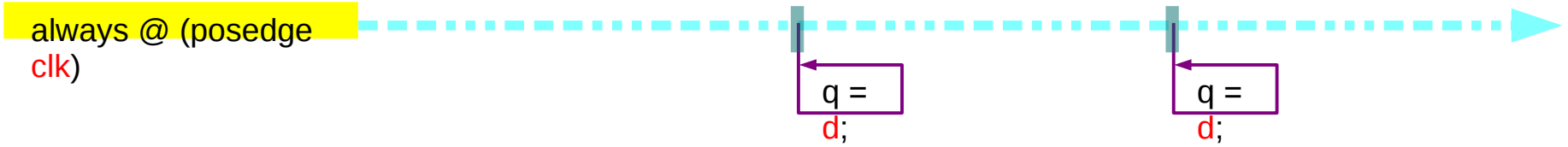




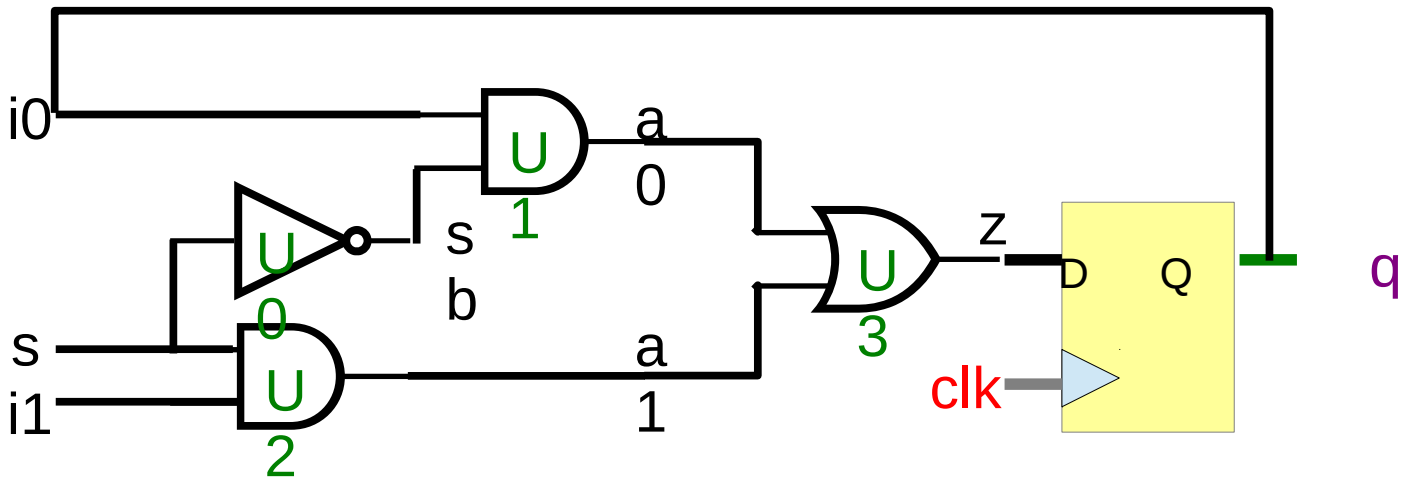
Only sensitive to a subset of of their inputs – sensitivity list

the two inputs (clk, d)

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



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

References

[1] http://en.wikipedia.org/
[2] http://www.allaboutcircuits.com/
[3] W. Wolf, “Modern VLSI Design : Systems on Silicon
[4] N. Weste, D. Harris, “CMOS VLSI Design: A Circuits and Systems Perspective”
[5] J. P. Uyemura, “Introduction to VLSI Circuits and Systems”
[6] https://en.wikiversity.org/wiki/The_necessities_in_SOC_Design
[7] https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design
[8] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design
[9] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Architecture
[10] https://en.wikiversity.org/wiki/The_necessities_in_Computer_Organization
[11] https://en.wikiversity.org/wiki/Verilog_programming_in_plain_view