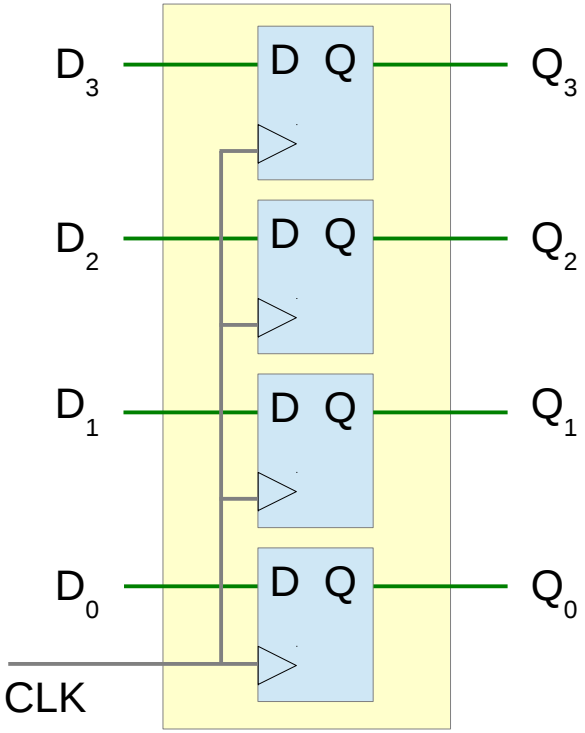
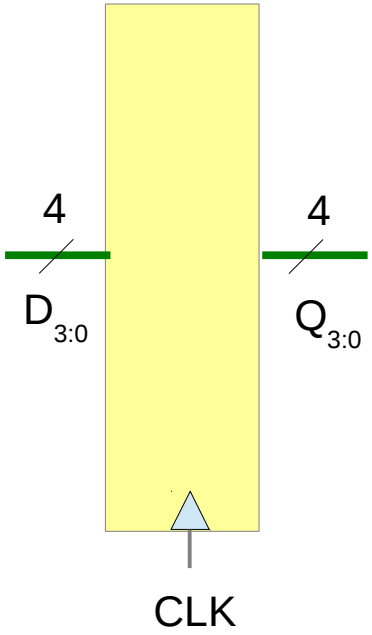
FlipFlop Timing (3D)

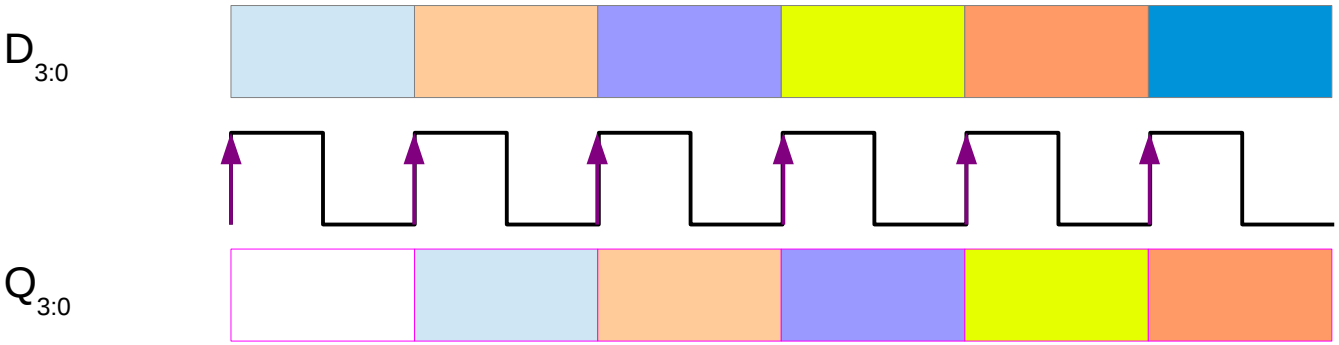
Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Register





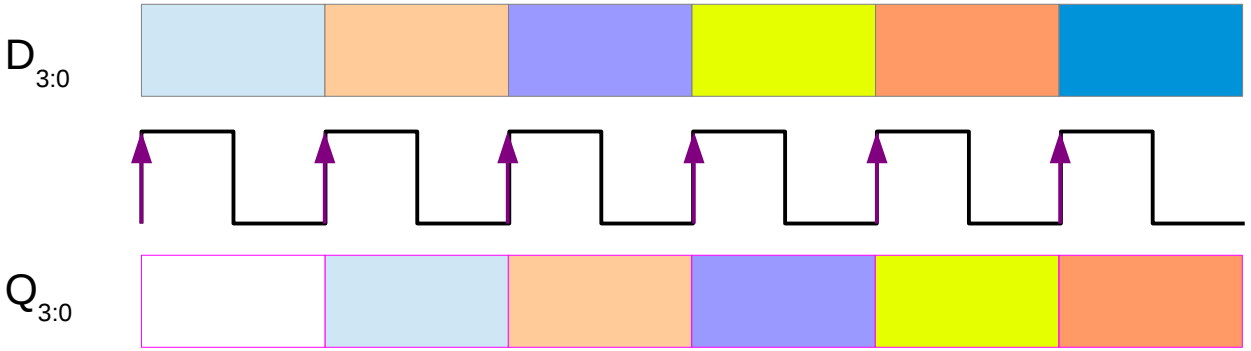


without delay

without delay

Types of Timing Diagrams

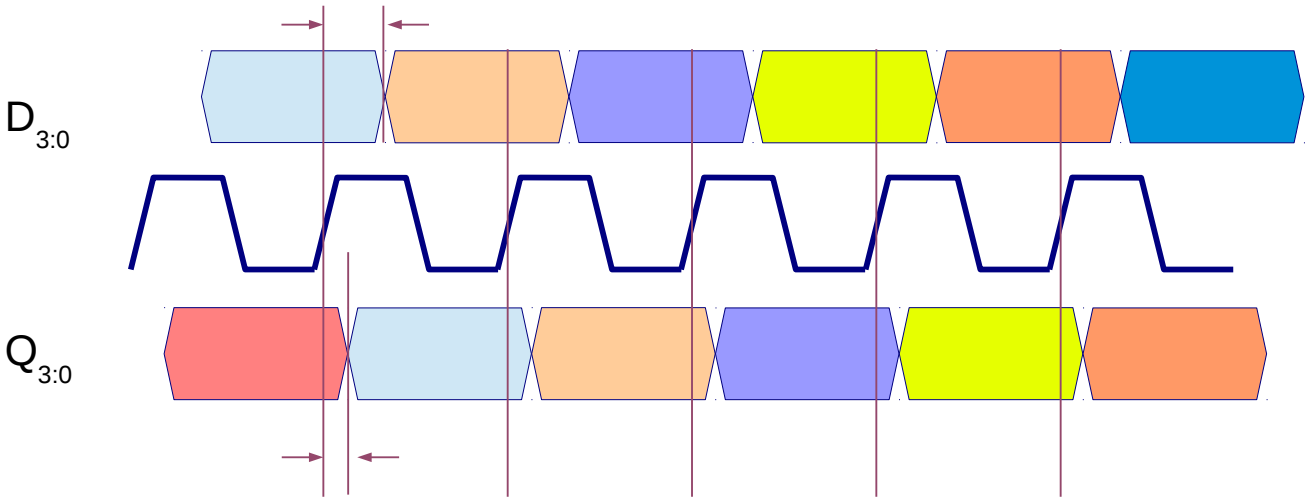
a timing diagram without delays



no delay (ideal case)

no delay (ideal case)

a timing diagram with delays



input delay

output delay Clk->Q

DFF Testbench

```
module dff(d, clk, rst, q, qb);
  input d, clk, rst;
  output q, qb;
  reg    q;

  always @(posedge clk)
  begin
    if (~rst) q = 0;
    else      q = d;
  end

  assign qb = ~q;
endmodule
```

Nonblocking Assignments **<=**

Blocking Assignments **=**

```
initial
begin
        clk =0;
        d   =0;
        rst =1;
        rst =0;
  #20   rst = 1;

  #10  d <=  1;
  #10  d <=  0;
  #10  d <=  1;
  #10  d <=  0;
  #10  d <=  1;
  #10  d <=  1;

  $finish;
end
```

```
#10  d =  1;
#10  d =  0;
#10  d =  1;
#10  d =  0;
#10  d =  1;
#10  d =  1;
```

```
`timescale 1ns/100ps

module dff_tb;

reg d, clk, rst;

dff U1 (d, clk, rst, q, qb);

always #10 clk = ~clk;

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

endmodule
```

Testbench with Nonblocking Assignments

```
module dff(d, clk, rst, q, qb);
  input d, clk, rst;
  output q, qb;
  reg    q;

  always @(posedge clk)
  begin
    if (~rst) q = 0;
    else      q = d;
  end

  assign qb = ~q;
endmodule
```

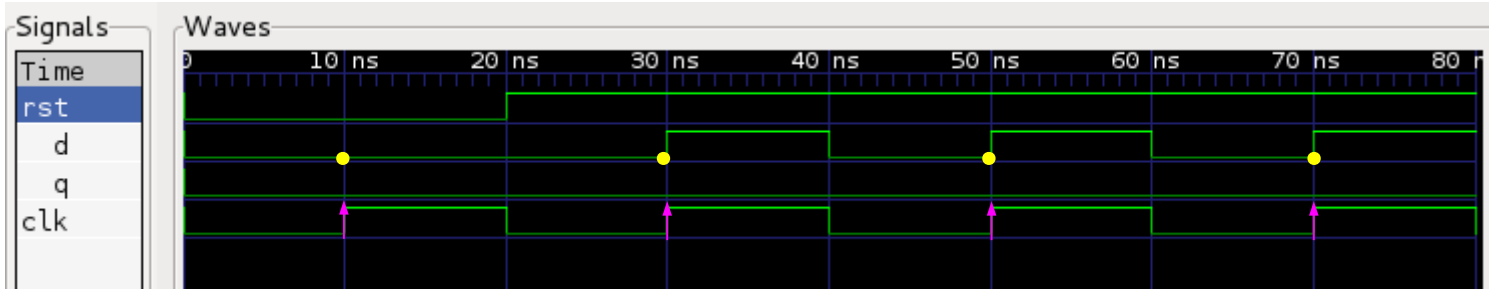
```
initial
begin
          clk =0;
          d   =0;

          rst =1;
          rst =0;
   #20    rst = 1;

   #10  d <=  1;
   #10  d <=  0;
   #10  d <=  1;
   #10  d <=  0;
   #10  d <=  1;
   #10  d <=  1;
  $finish;

end
```

```
`timescale 1ns/100ps

module dff_tb;

reg d, clk, rst;

dff U1 (d, clk, rst, q, qb);

always #10 clk = ~clk;

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

endmodule
```

DFF Testbench Waveforms

Nonblocking Assignments

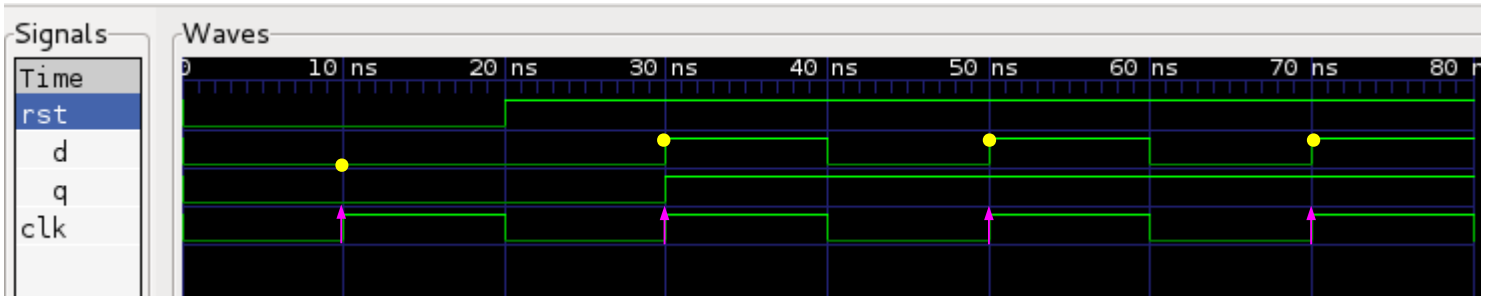
```
#10 d <= 1;
#10 d <= 0;
#10 d <= 1;
#10 d <= 0;
#10 d <= 1;
#10 d <= 1;
```



samples the unchanged d input values
at the posedge of clk

Blocking Assignments

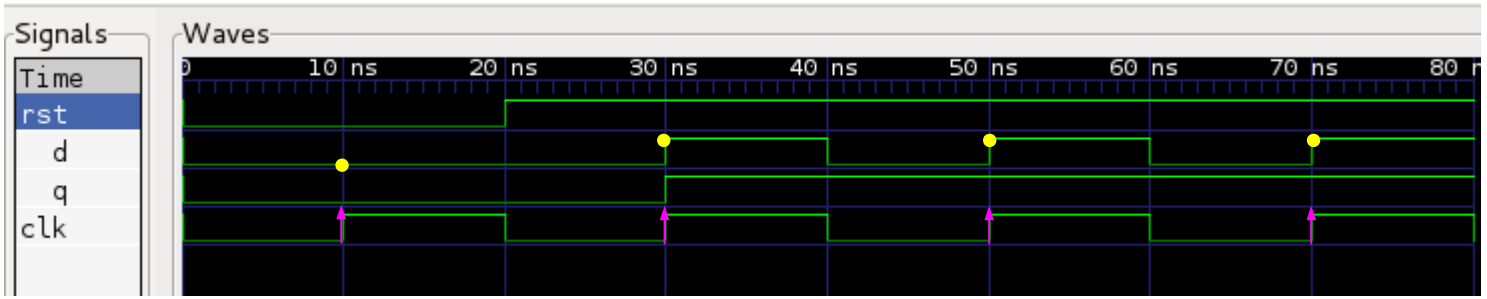
```
#10 d = 1;
#10 d = 0;
#10 d = 1;
#10 d = 0;
#10 d = 1;
#10 d = 1;
```



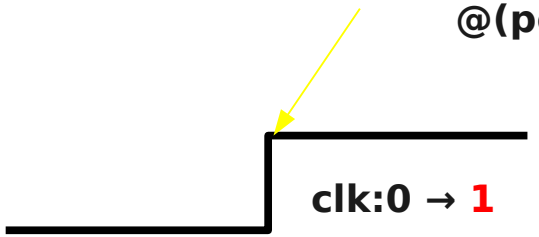
samples the changed d input values at
the posedge of clk

Blocking Assignments

```
#10  d = 1;
#10  d = 0;
#10  d = 1;
#10  d = 0;
#10  d = 1;
#10  d = 1;
```




```
always @(posedge clk)
begin
  if (~rst) q = 0;
  else      q = d;
end
```

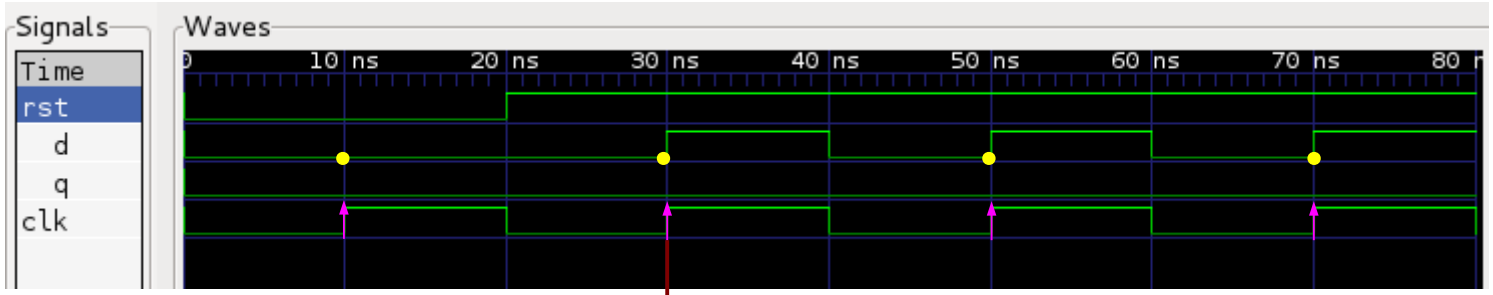


At the posedge of clk
clk has changed to 1
Also, d has changed to its new value

Nonblocking Assignments

Nonblocking Assignments

```
#10 d <= 1;
#10 d <= 0;
#10 d <= 1;
#10 d <= 0;
#10 d <= 1;
#10 d <= 1;
```



In each of time slot
which is delayed by 10 time units,
there is only one assignment
that can be scheduled simultaneously,

samples the unchanged d input values
at the posedge of clk

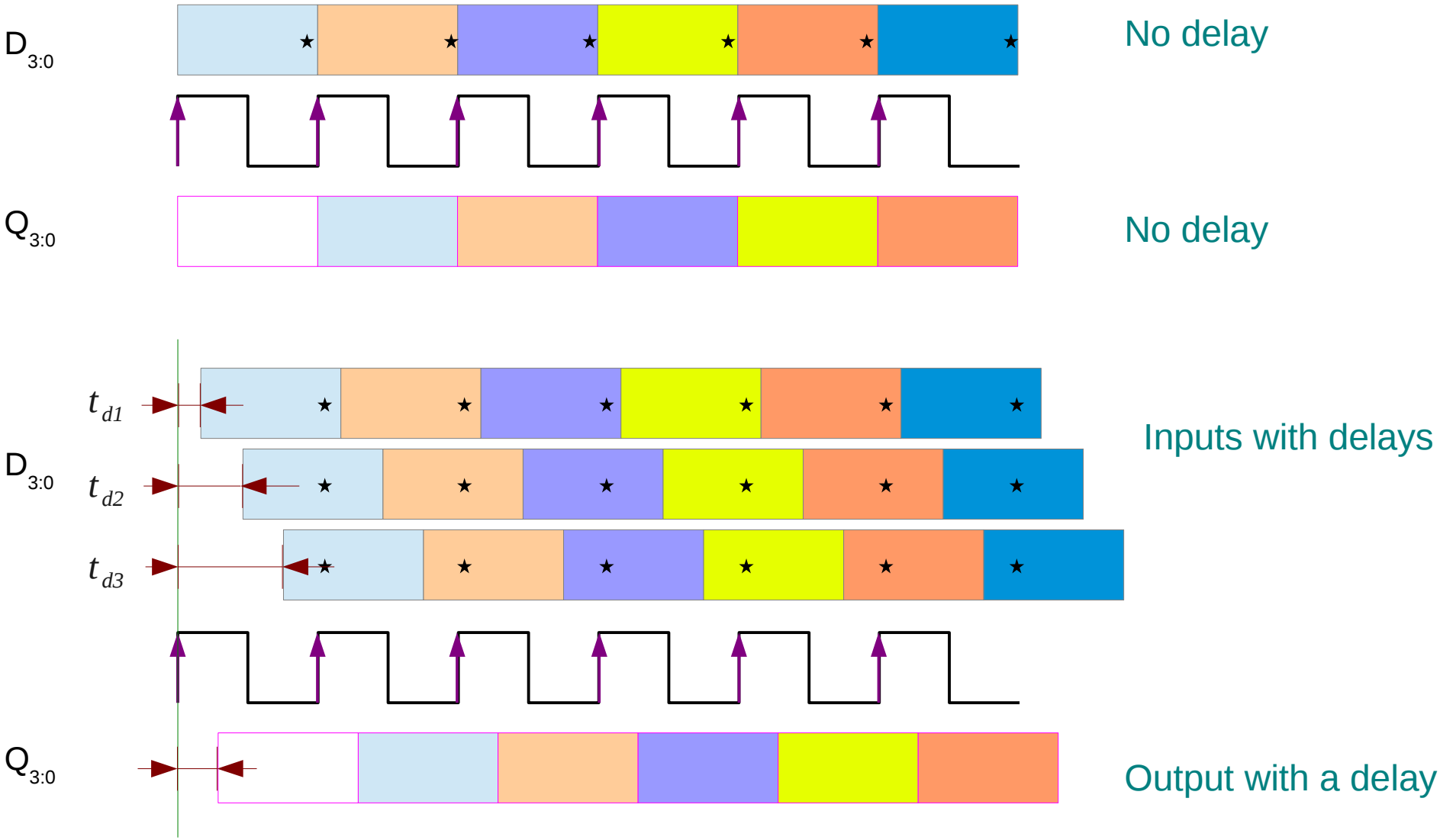
(1) Allowing scheduling of assignments without waiting for its completion
(2) Executed last in the time step in which it is scheduled (after all blocking assignments' execution)

Regular Events	Nonblocking Update Events	Monitor Events

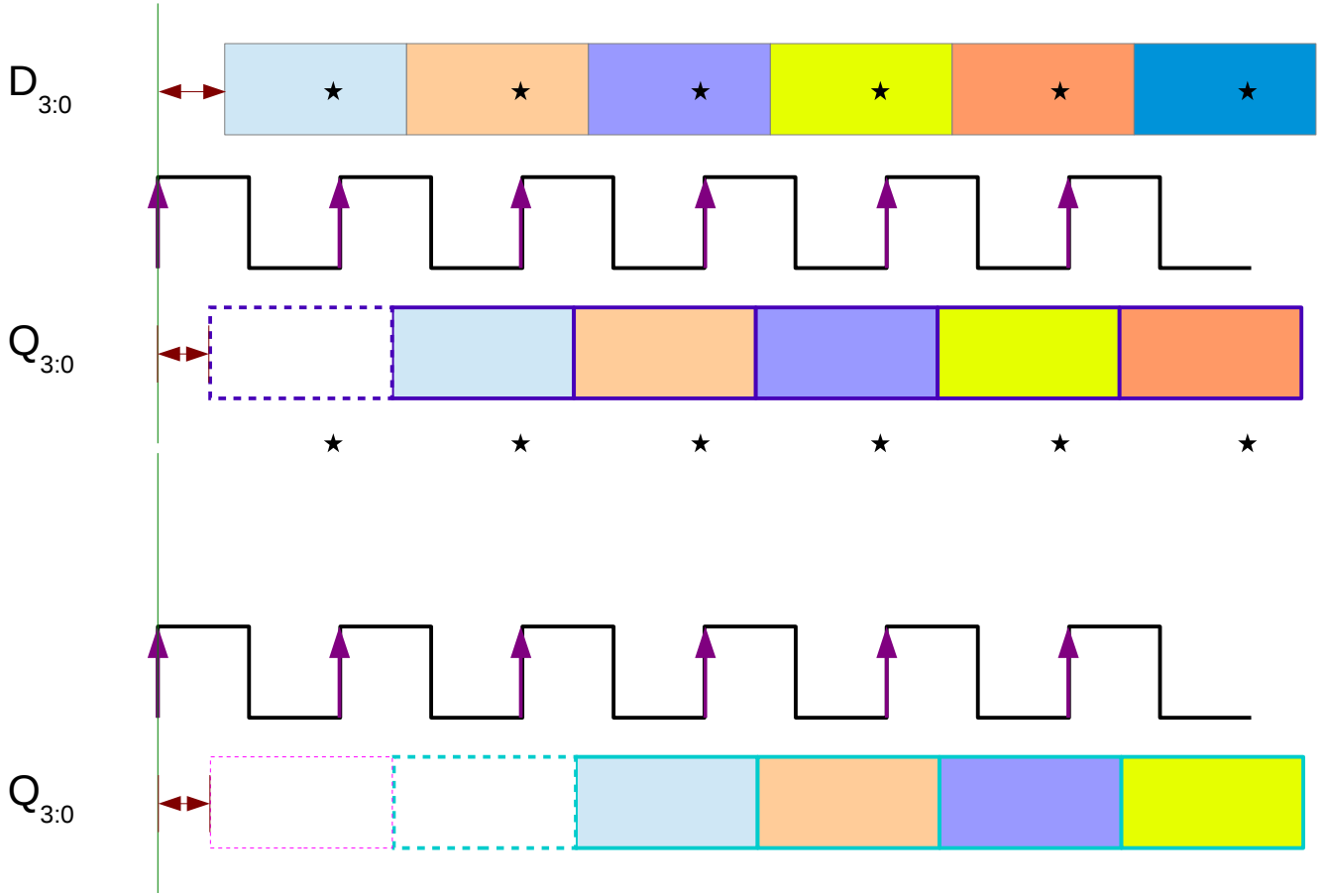
- Evaluate
- Update

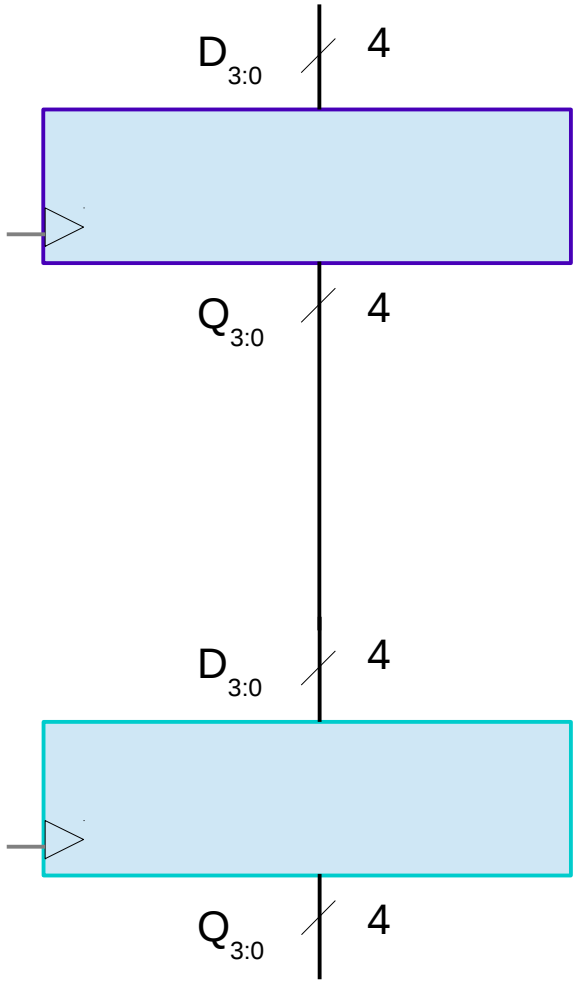
So, the input d update is done after the posedge clk

FF Timing – Input and Output Delays

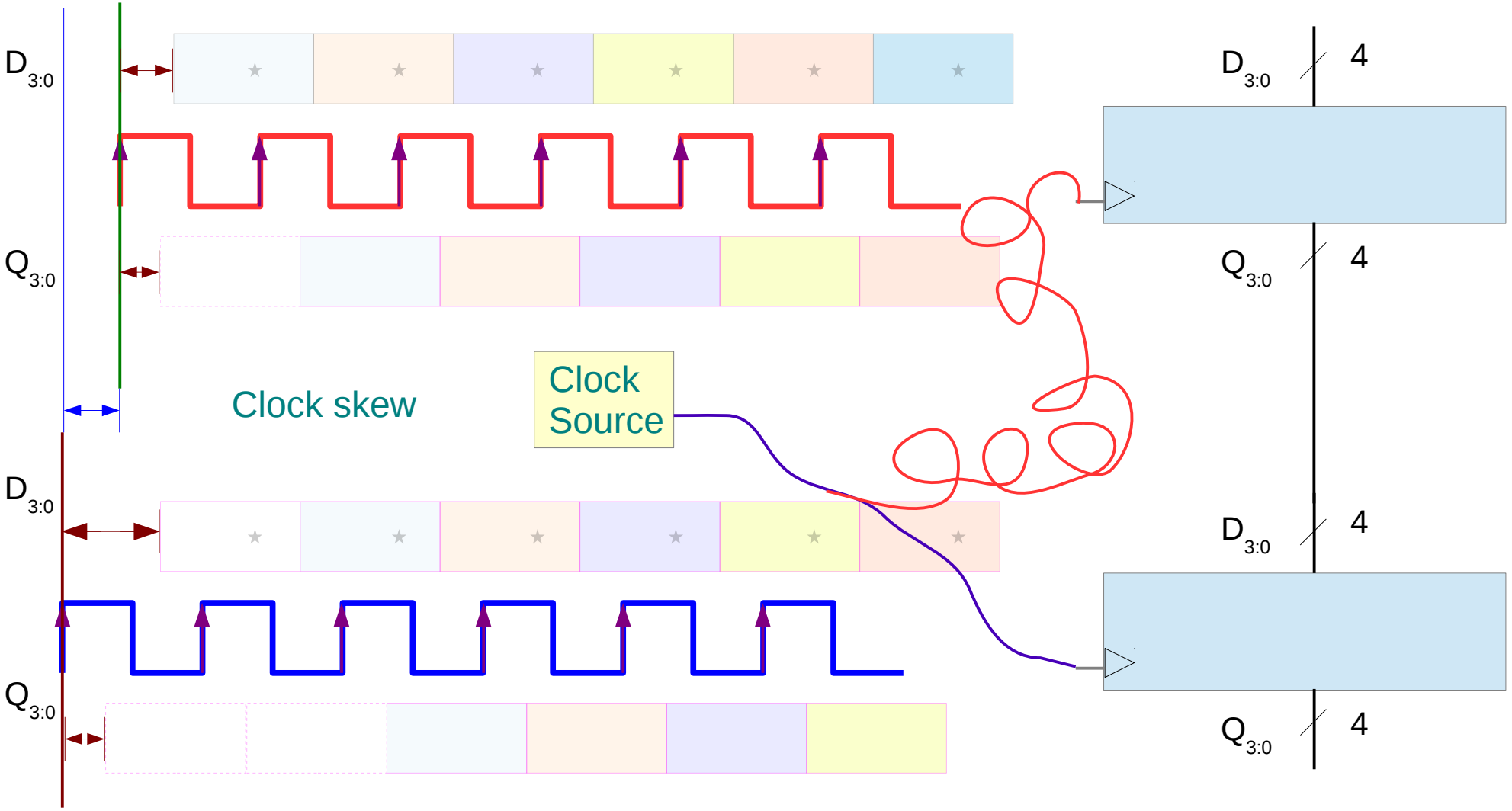


Reg to Reg Timing

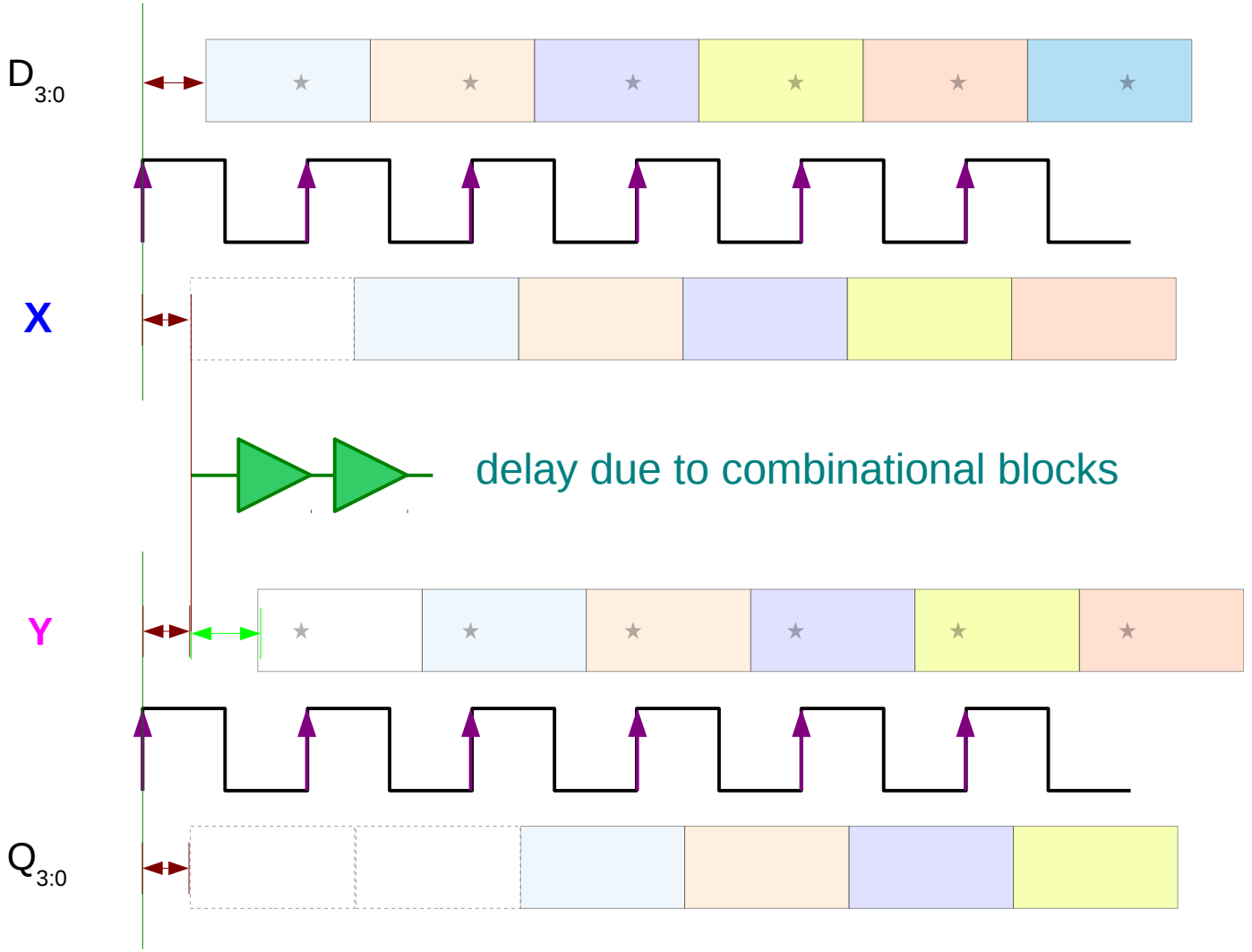


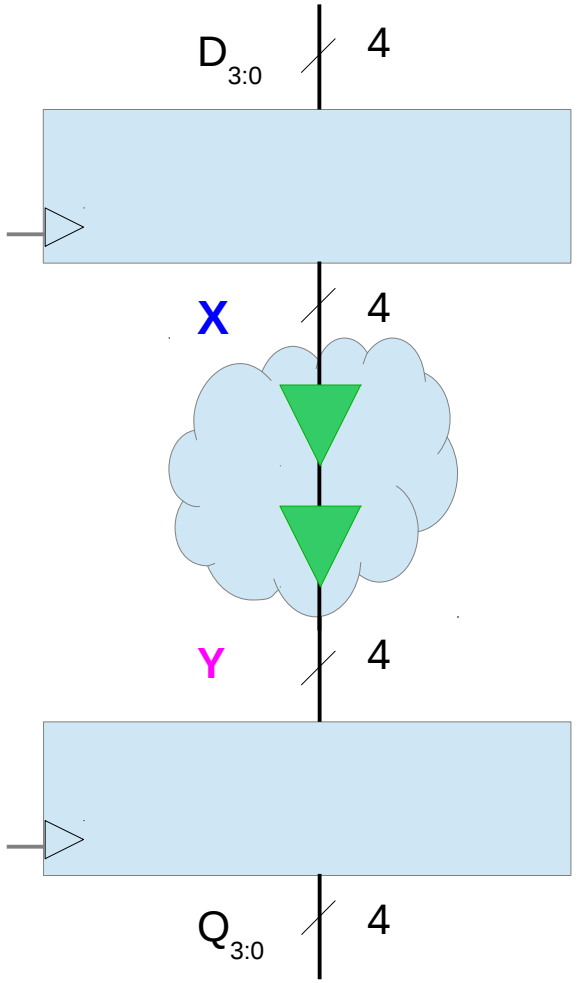


Clock Skew

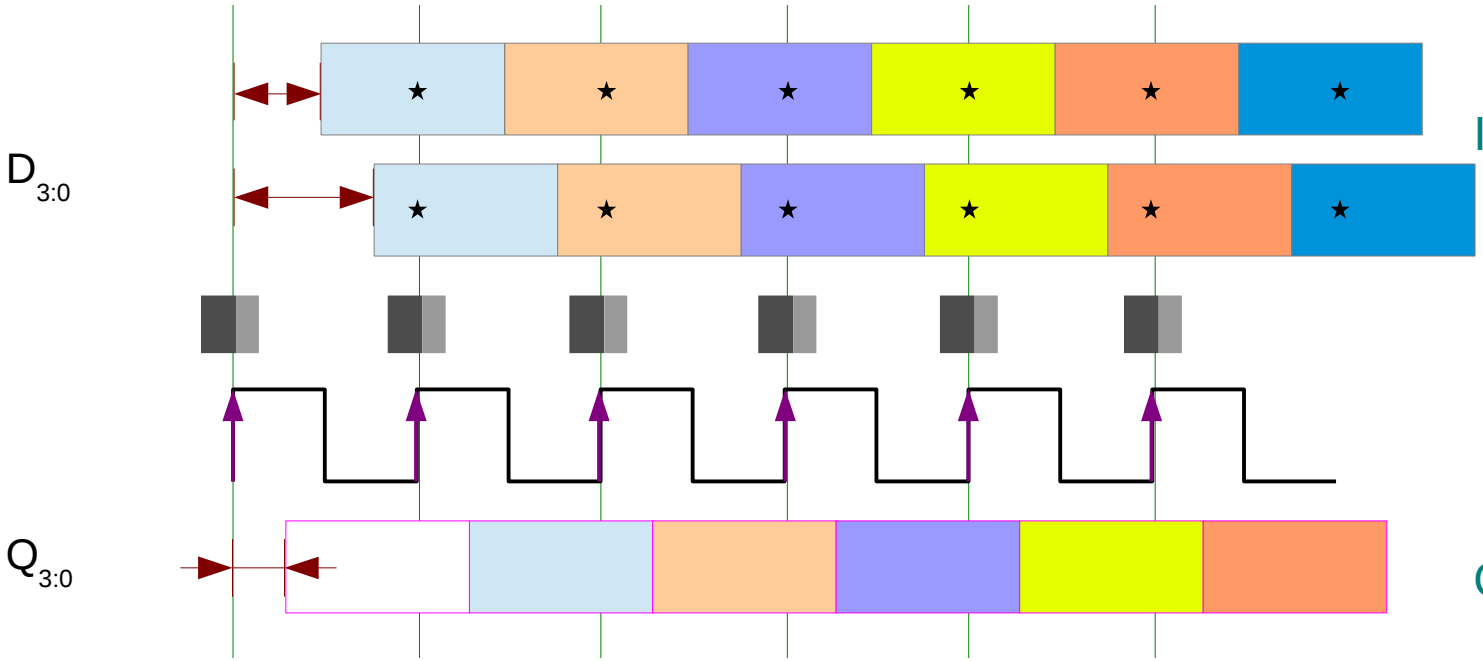


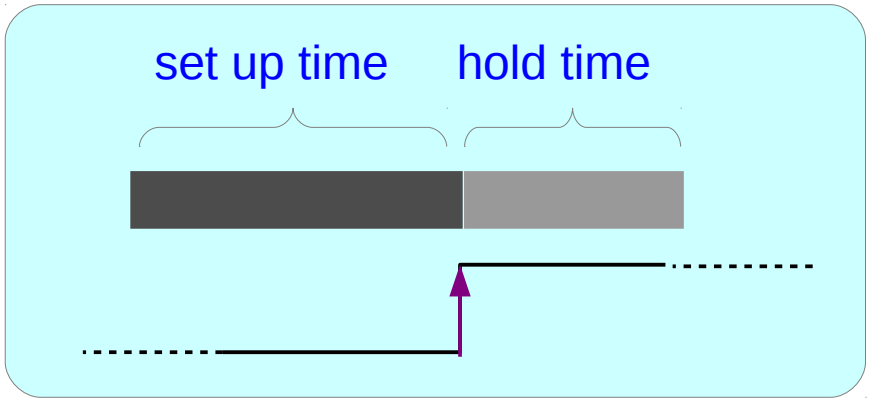
Path Delay



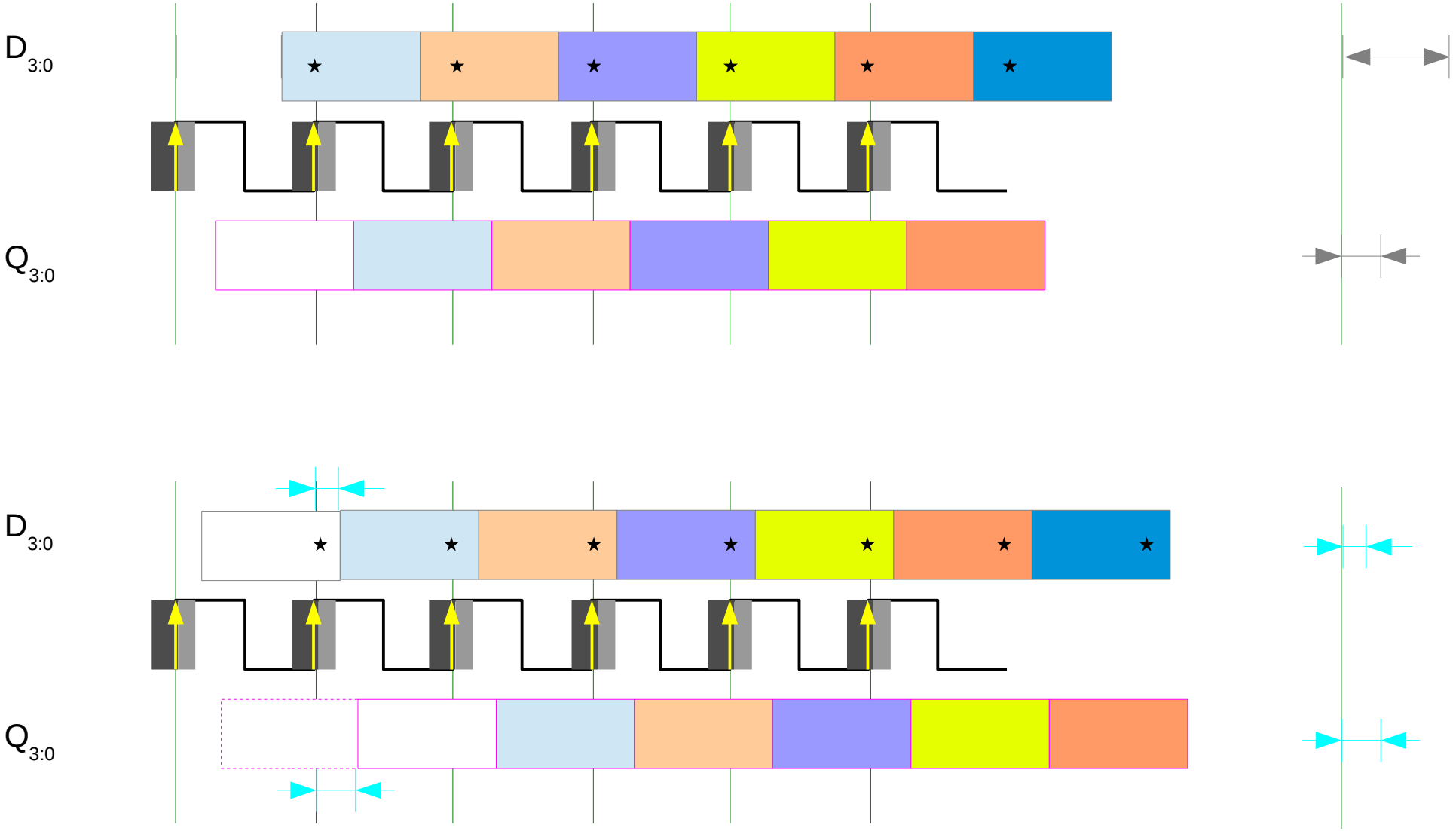


Setup & Hold Time (1)

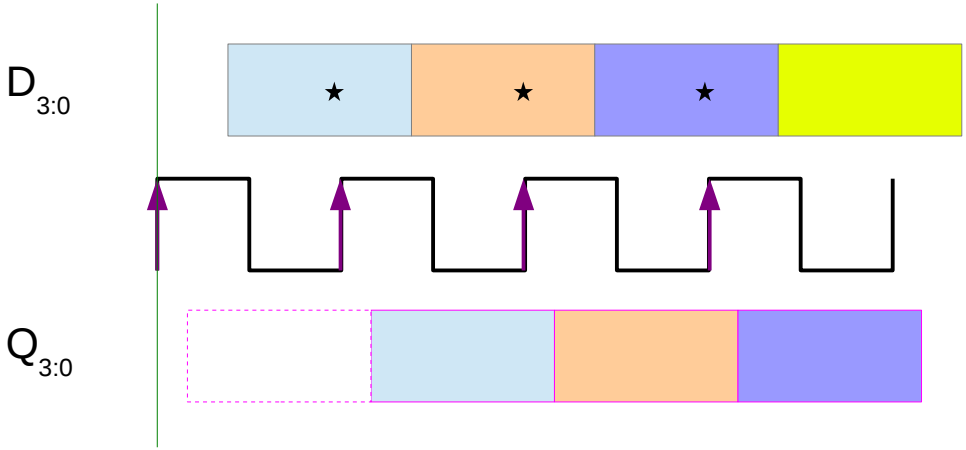


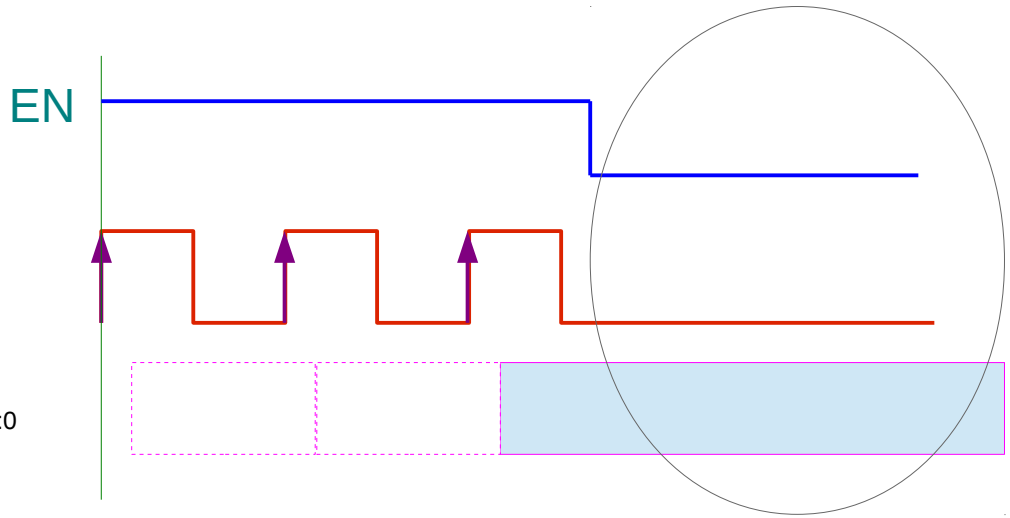


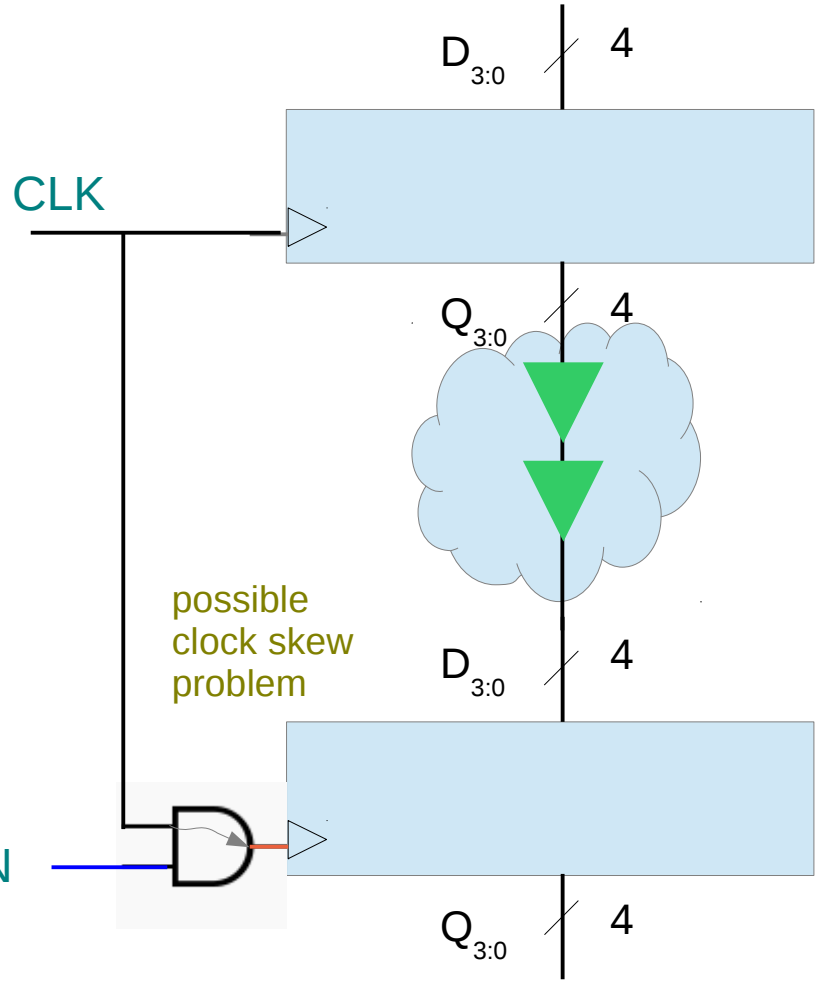
Setup & Hold Time (2)



Clock Gating

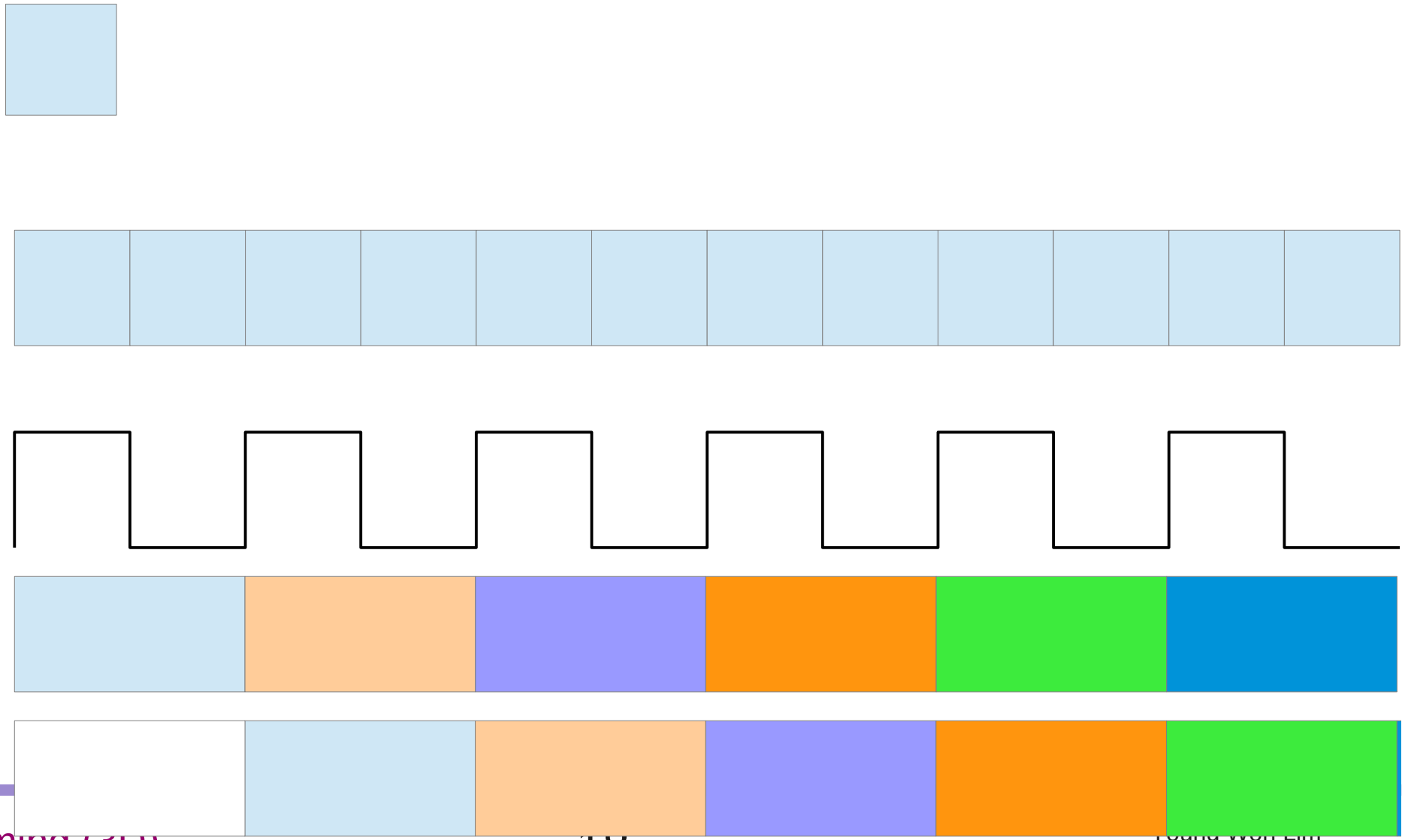






Register Timing

NOR-based SR Latch



References

[1] http://en.wikipedia.org/
[2] http://planetmath.org/
[3] M.L. Boas, “Mathematical Methods in the Physical Sciences”