EECS151 : Introduction to Digital Design and ICs

Lecture 21 – Dividers, Latches

Bora Nikolić

Pentium FDIV Bug (from Wikipedia)

The Pentium FDIV bug is a hardware bug affecting the floating-point unit (FPU) of the early Intel Pentium processors. Because of the bug, the processor would return incorrect binary floating point results when dividing certain pairs of high-precision numbers. The bug was discovered in 1994 by Thomas R. Nicely, a professor of mathematics at Lynchburg College. Missing values in a lookup table used by the FPU's floating-point division algorithm led to calculations acquiring small errors. While these errors would in most use-cases only occur rarely and result in small deviations from the correct output values, in certain circumstances the errors can occur frequently and lead to more significant deviations

Review

- Binary multipliers have three blocks:
  - Partial-product generation (NAND or Booth)
  - Partial-product compression (ripple-carry array, CSA or Wallace)
  - Final adder
- Multipliers are often pipelined
- Constant multipliers can be optimized for size/speed
- Shifters and crossbars are common building blocks in digital systems
  - Often require customization

Dividers

Pencil-And-Paper Division

```
       1512        quotient
     ______
   3| 4537         dividend
divisor 3000       divisor*q_i*10^i
     ______
      1537         partial remainder

      1500
     ______
      0037

      0030
     ______
      0007

      0006
     ______
      0001         remainder
```

Division is an iterative process:

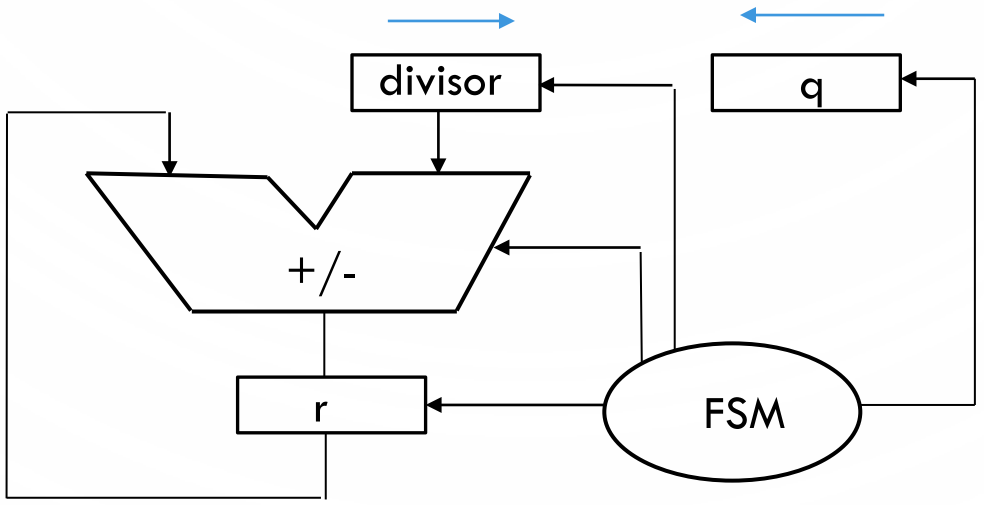
$$\boxed{r(i) = r(i+1) - q_i * D * 10^i}$$

We usually 'guess' q_i

Restoring Divider

- Assume $q_i = 1$
- Subtract divisor from r; check if r(i) >= 0
  - if r(i) > 0, guess was good ($q_i = 1$)
  - if r(i) < 0, restore the value by adding divisor, $q_i = 0$
- Shift divisor to right
- Repeat *n* times

More efficient to shift the reminder right



n shifts

n subtractions

n/2 restorations

Non-Restoring Divider

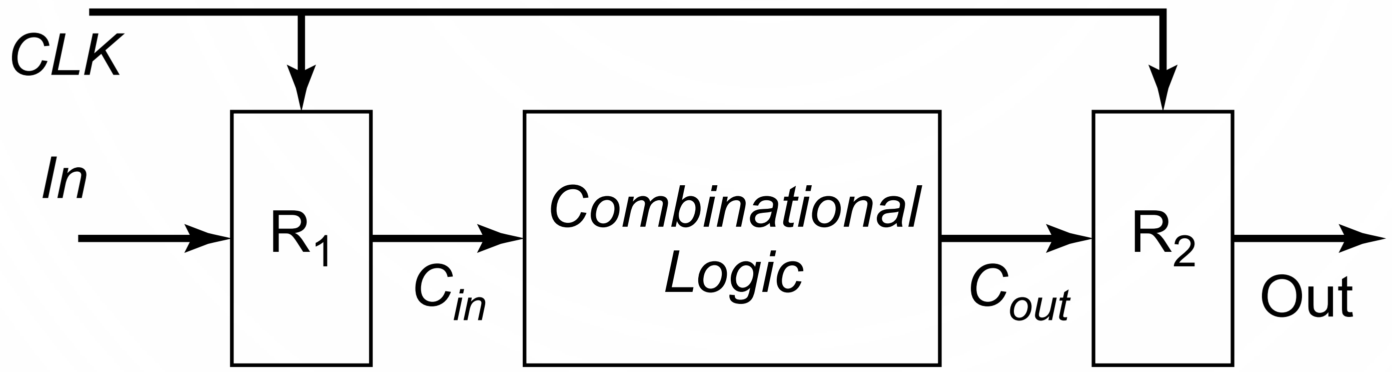
- Doesn’t restore if r(i) < 0
- Instead, adds the divisor in the next iteration
    - *n* shifts
    - *n* additions/subtractions

Faster Dividers

- Divide in a higher radix than 2 (typically 4, i.e. guess q_iq_{i+1})
- Keep the partial remainders in redundant form
- Sweeney-Robertson-Tocher (SRT) algorithm
  - Used in many processors

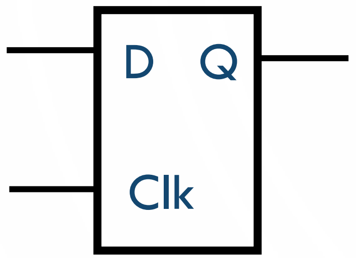
Timing

Synchronous Timing: Review

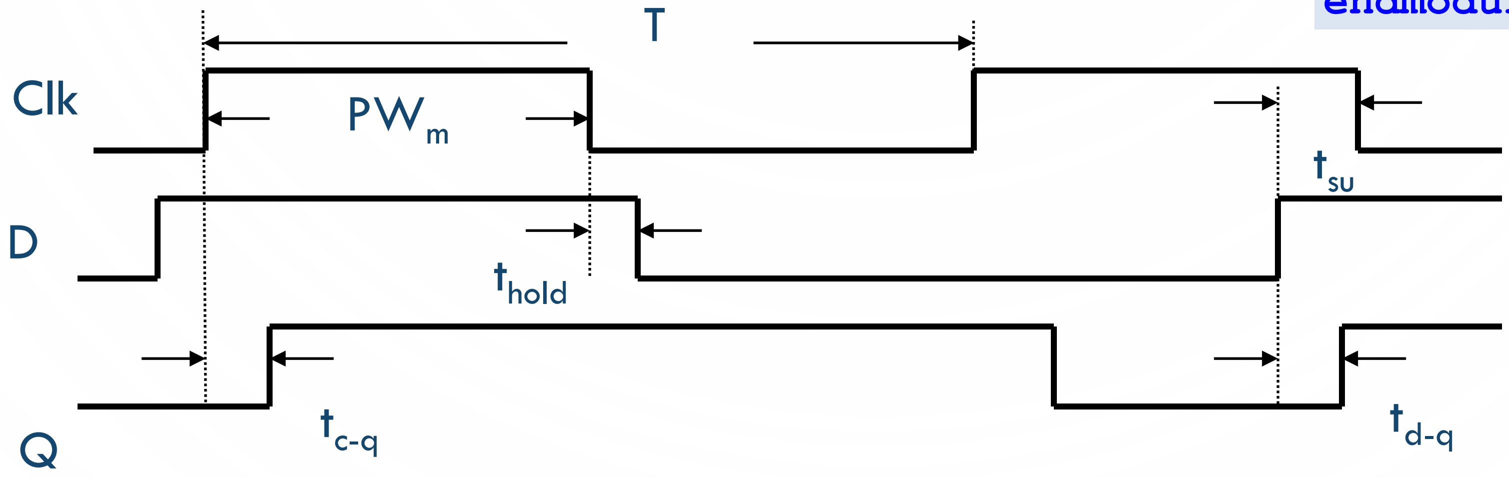


Latch Parameters

- Latch is transparent high or low



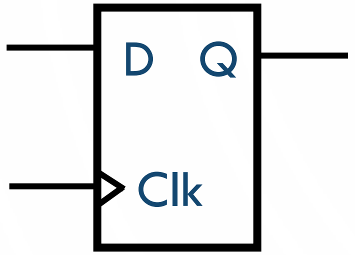
```
module latch
(
   input clk,
   input d,
   output reg q
);

  always @(clk or d)
  begin
    if ( clk )
      q <= d;
  end
endmodule
```



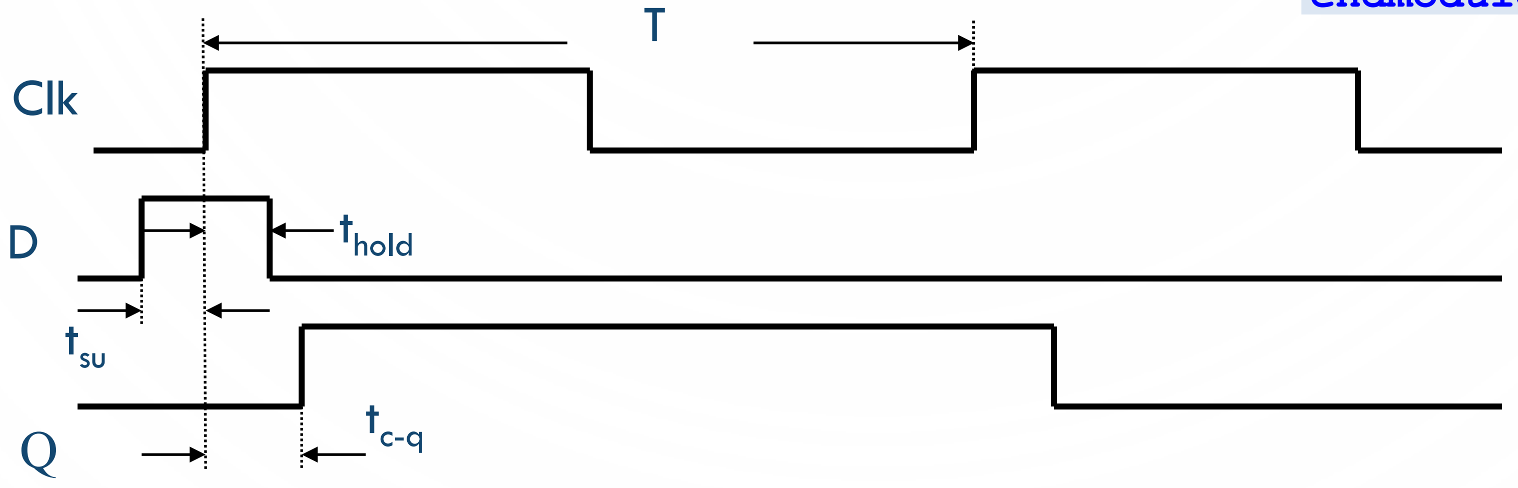
Delays can be different for rising and falling data transitions

Flip-Flop Parameters

- Flip-flop is edge-triggered



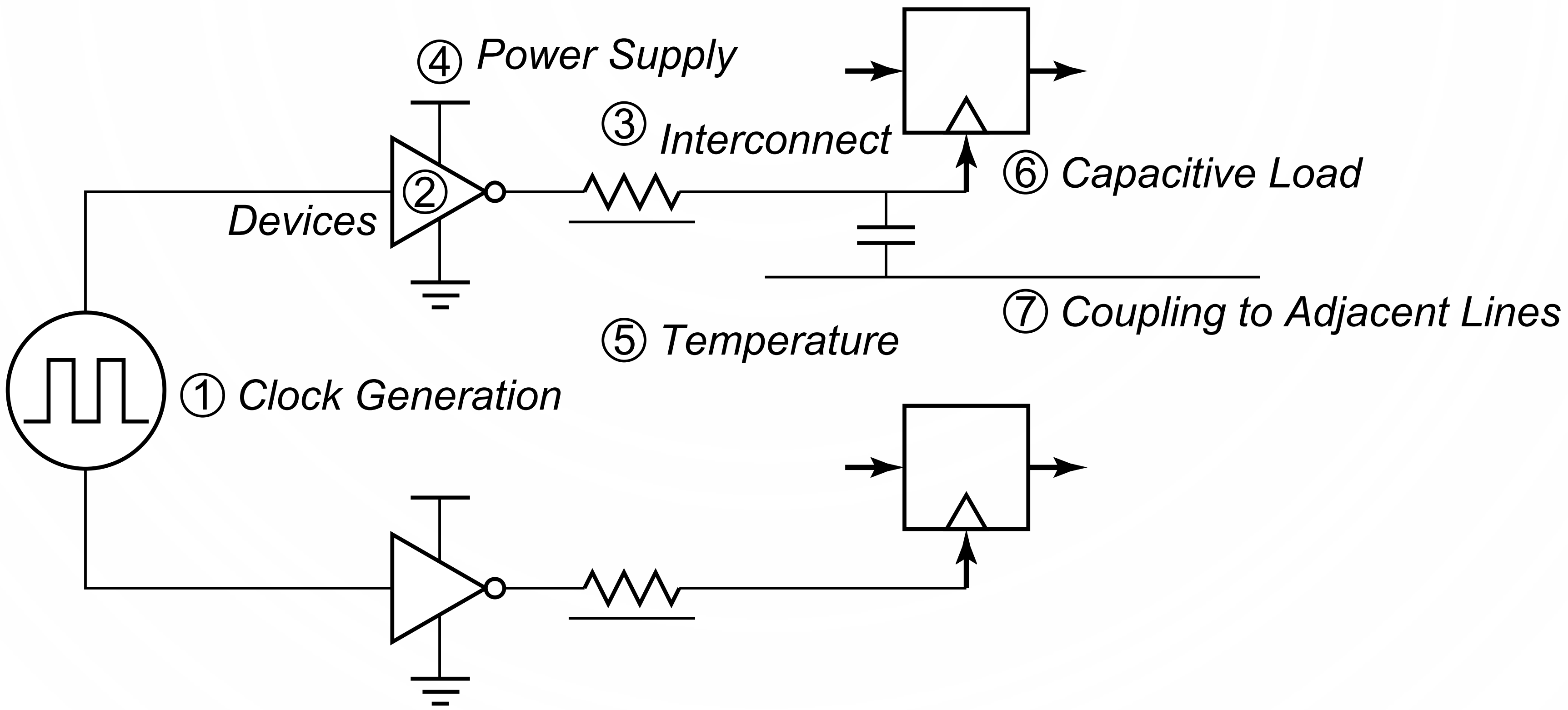
```
module flipflop
(
   input clk,
   input d,
   output reg q
);

  always @(posedge clk)
  begin
    q <= d;
  end
endmodule
```



Delays can be different for rising and falling data transitions

Clock Uncertainties

- Clock arrival time varies in space and time



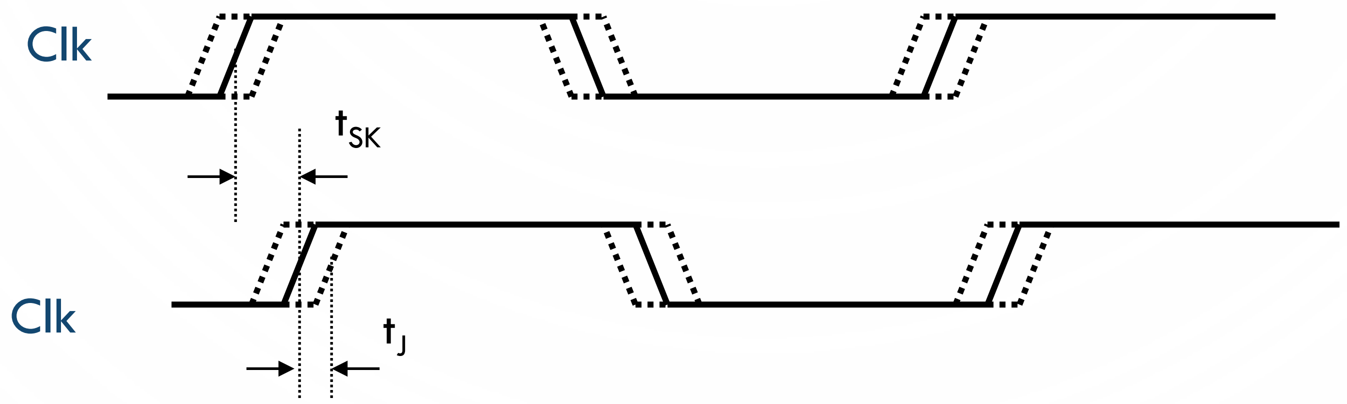
Sources of clock uncertainty

Clock Nonidealities

- Clock skew
  - Spatial variation in temporally equivalent clock edges; deterministic + random, t_{SK}
- Clock jitter
  - Temporal variations in consecutive edges of the clock signal; modulation + random noise
  - Cycle-to-cycle (short-term) t_{JS}
  - (there also exists long-term jitter t_{JL})
- Variation of the pulse width
  - Important for level sensitive clocking with latches

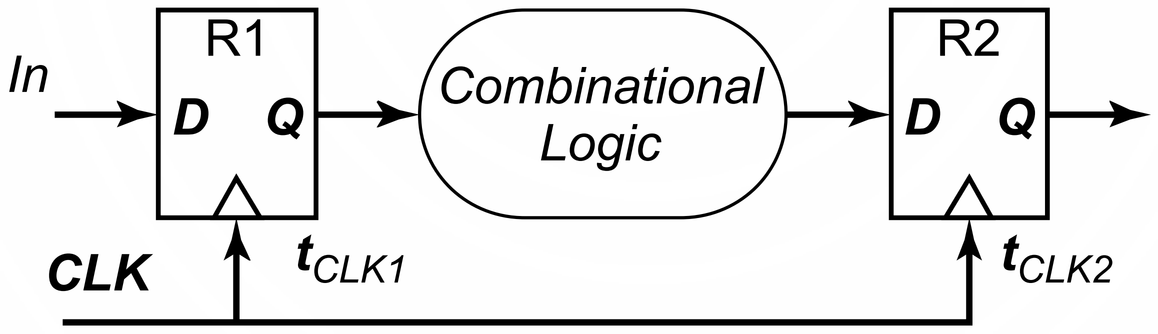
Clock Skew and Jitter

- Both skew and jitter affect the effective cycle time
- Only clock distribution skew affects the race margin



Timing Constraints

- First flip-flop launches data on the first clock edge, the second one captures on the second clock edge



$t_{clk\text{-}q}$, t_{su}, t_h

$t_{logic,max}$

$t_{logic,min}$

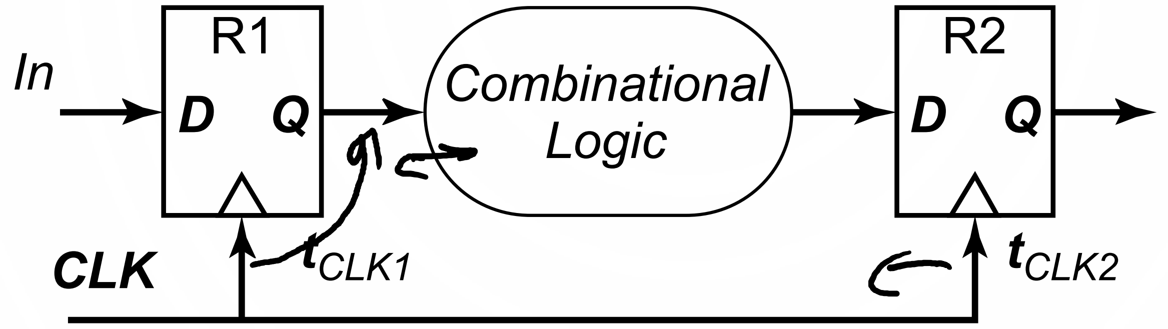
Minimum cycle time is set by the longest logic path:

$$T - t_{sk} - t_j = t_{c\text{-}q} + t_{su} + t_{logic,max}$$

Worst case is when receiving edge arrives early

Timing Constraints

- Launching flip-flop shoudn’t contaminate its own data



$t_{clk\text{-}q}$, t_{su}, t_h

$t_{logic,max}$

$t_{logic,min}$

Hold time constraint:

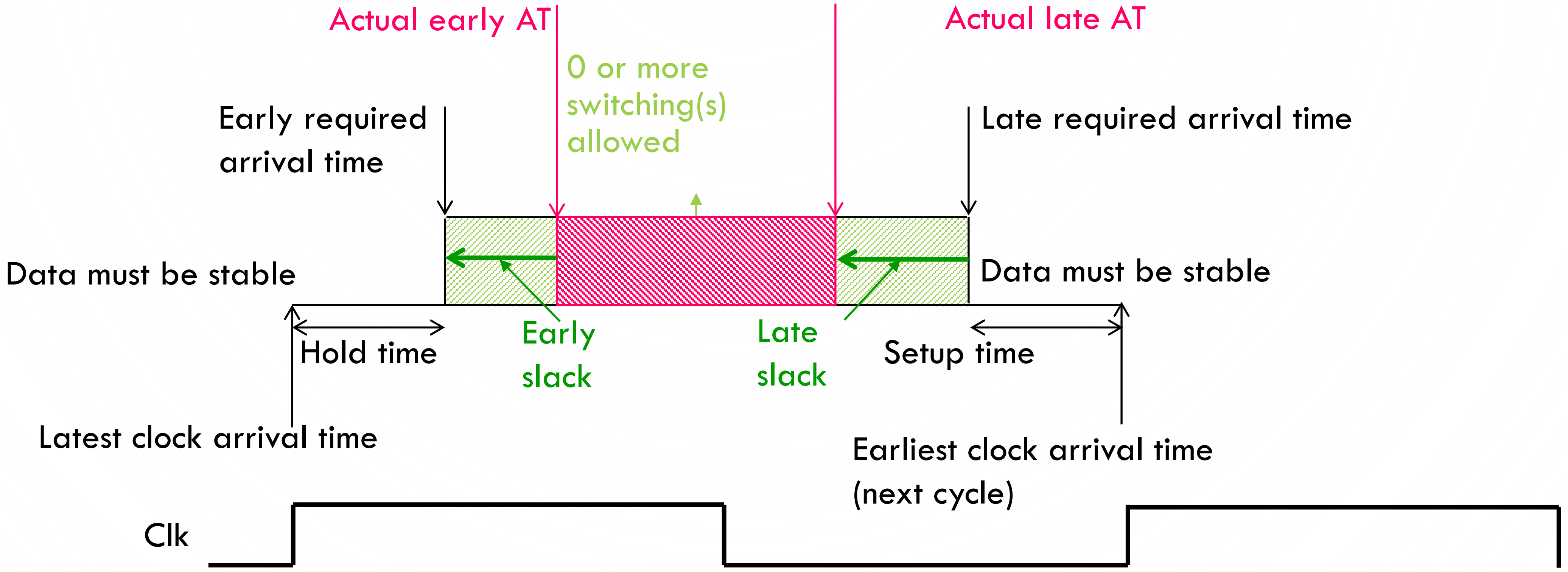
$t_{c\text{-}q} + t_{logic,\ min} > t_{hold} + t_{sk}$

Worst case is when receiving edge arrives late

Race between data and clock

Slack

- Visualizing arrival times


Actual early AT
Actual late AT
0 or more switching(s) allowed
Early required arrival time
Late required arrival time
Data must be stable
Data must be stable
Hold time
Early slack
Late slack
Setup time
Latest clock arrival time
Earliest clock arrival time (next cycle)
Clk

Timing Analysis

- Report timing

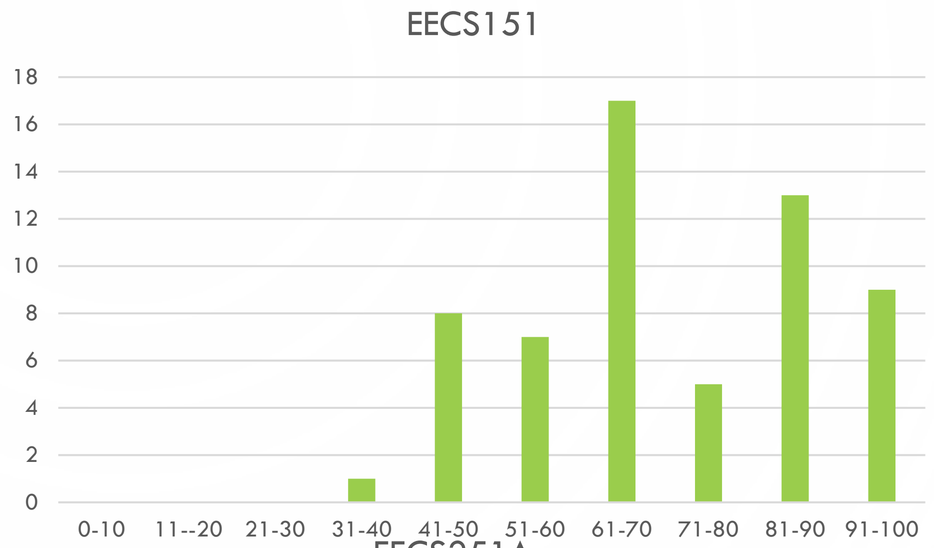
```
Startpoint: dig_agc_0/int_term_reg_0_
              (rising edge-triggered flip-flop clocked by clk_agc)
  Endpoint: dig_agc_0/int_term_reg_0_
            (rising edge-triggered flip-flop clocked by clk_agc)

Point                                              Fanout  Derate  Incr        Path       Voltage
---------------------------------------------------------------------------------------------
clock clk_agc (rise edge)                                          0.00        0.00
clock source latency                                               3.13        3.13 r
timing_control_0/C1276/Y (AND2X3)                                  0.00        3.13 r    3.00
timing_control_0/o_clk_agc (net)                 2                 0.00        3.13 r
dig_agc_0/BUFX8_G6B1I2/A (BUFX8)                                   0.00 &      3.13 r    3.00
dig_agc_0/o_clk_agc_G6B1I2 (net)                 15                0.00        3.91 r
dig_agc_0/int_term_reg_0_/CP (SDFFCQX2)                            0.01 &      3.92 r    3.00
…

clock reconvergence pessimism                                      0.00        3.92
clock uncertainty                                                      0.10        4.02
dig_agc_0/int_term_reg_0_/CP (SDFFCQX2)                            0.00        4.02 r
library hold time                                                 -0.48        3.54
data required time                                                             3.54
---------------------------------------------------------------------------------------------
data required time                                                             3.54
data arrival time                                                             -5.57
---------------------------------------------------------------------------------------------
slack (MET)                                                                    2.03
```

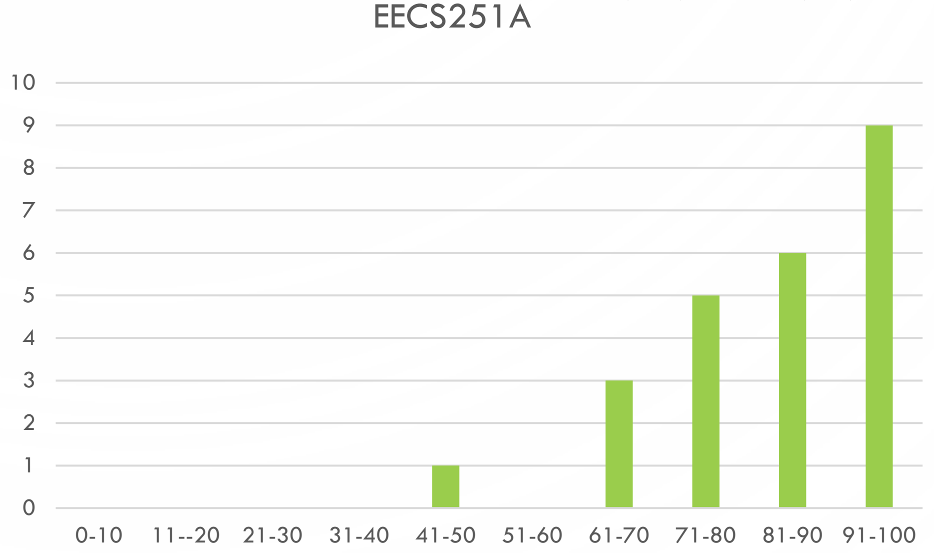
Administrivia

- Midterm 2 scores released
  - Final can clobber either midterm!
- Homework 9 posted on Friday, due 11/15
  - One more homework before Thanksgiving
- Project checkpoints #2 this week
- Thursday is a holiday (Veterans' Day)

Midterm 2

- EECS151
  - Max: 60.5/61
  - Average: 42.7/61 (70%)
- EECS251A
  - Max: 71/71
  - Average: 59.2/71 (83%)

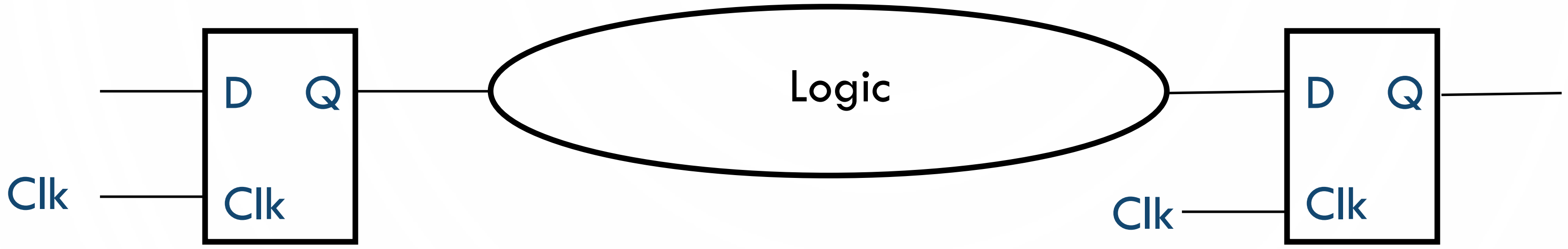




Latch-Based Timing

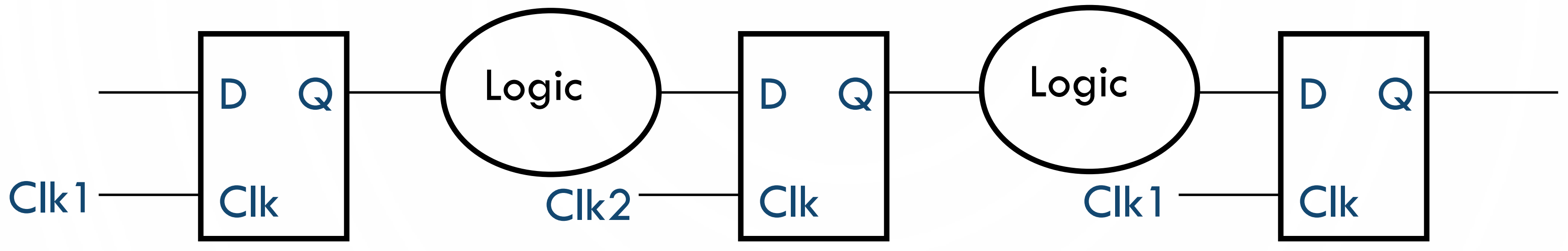
Latch-Based Timing

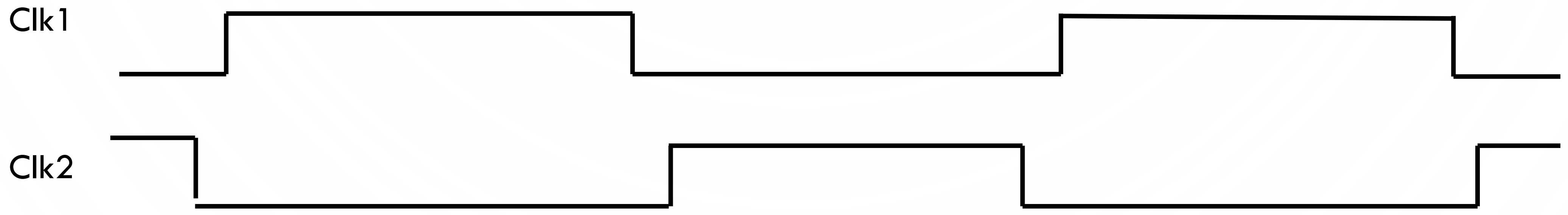
- Is there a possible timing problem in this path?



Latch-Based Timing

- Two clock phases

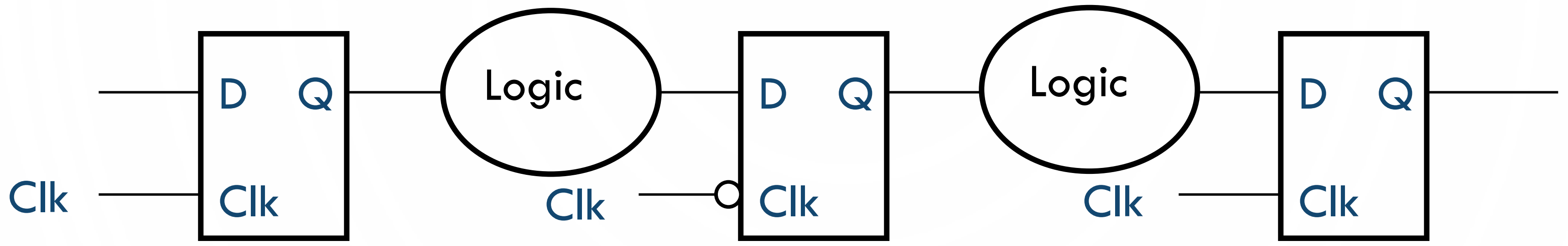


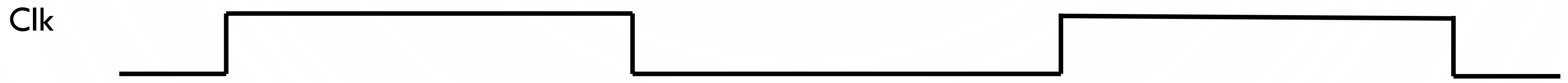


- Two consecutive latches are never transparent at the same time
  - Conservative

Latch-Based Timing

- Single clock phase



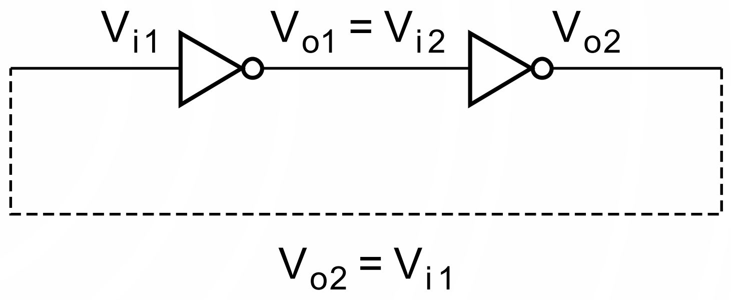


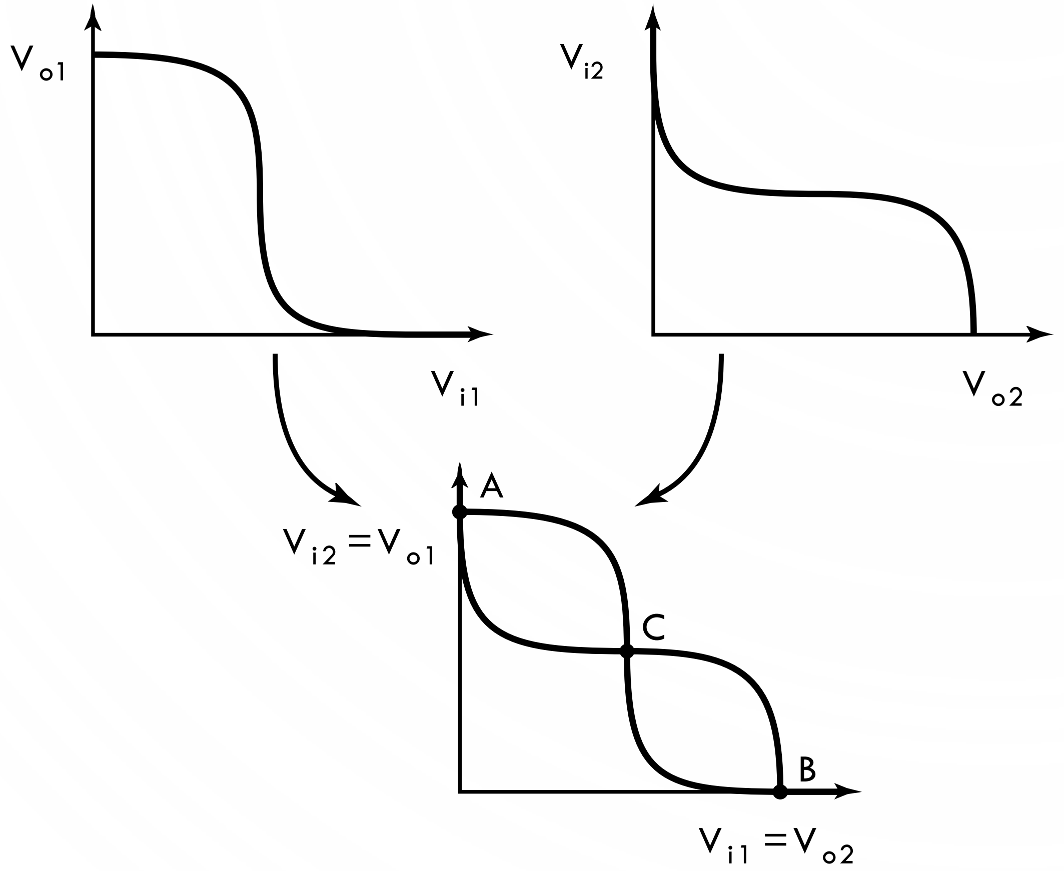
- Possibility of a race condition
  - Needs timing analysis (EE241B/EECS251B)

Latches

Cross-Coupled Inverter

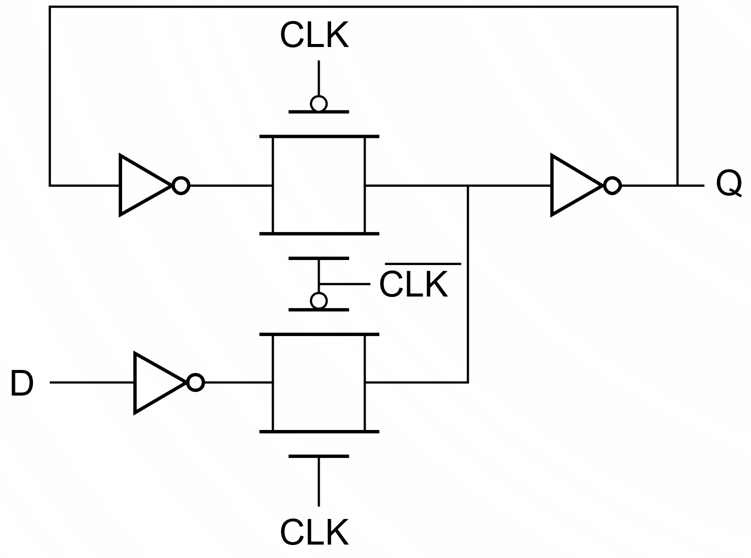
- Positive feedback stores the data



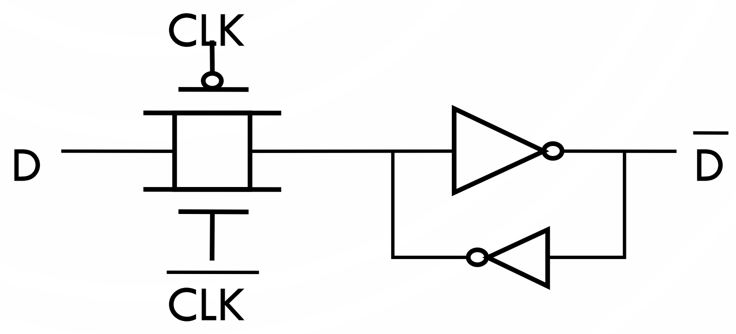


Writing into a Static Latch

Use the clock as a control signal (to break the positive feedback), that distinguishes between the transparent and opaque states



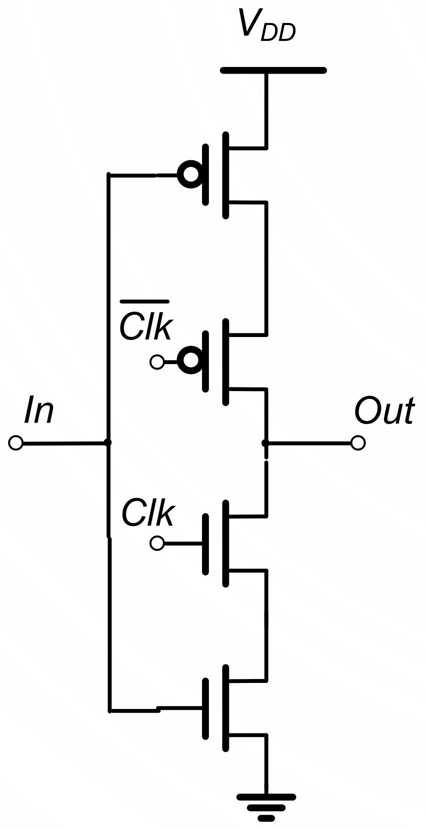
Converting into a MUX



Forcing the state (functionality depends on sizing)

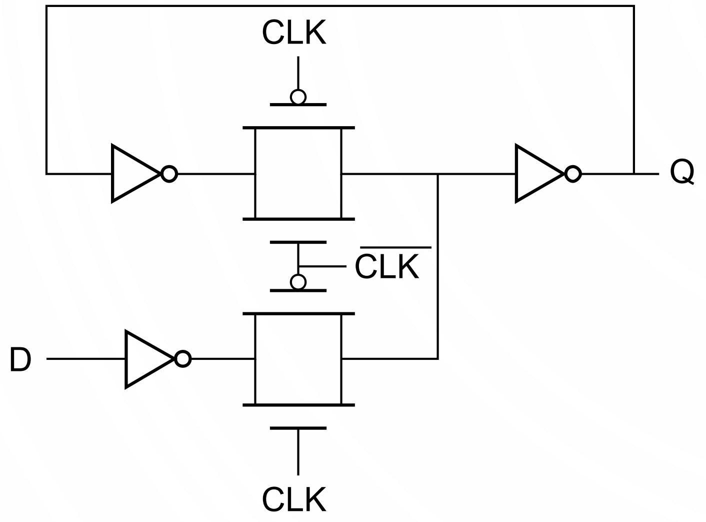
Tri-State Inverter

- Out is Z when Clk=0

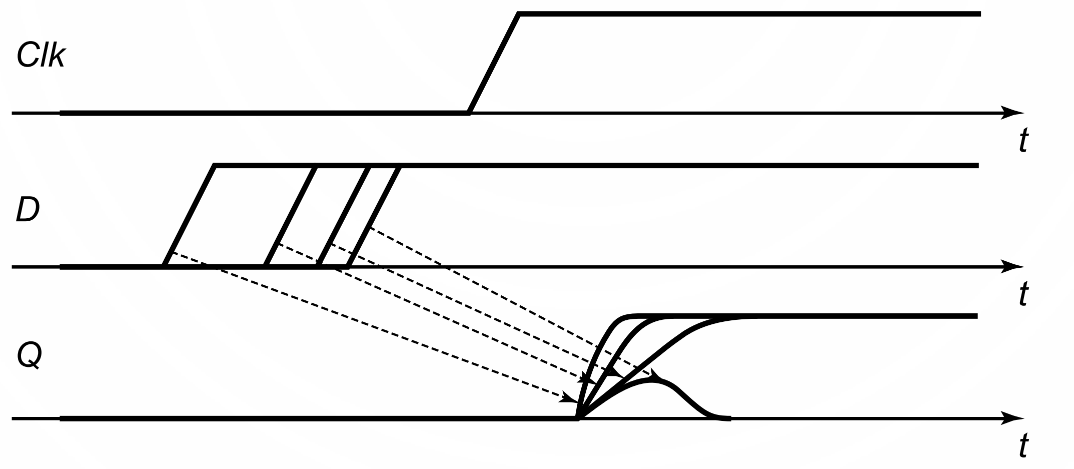


- Latch

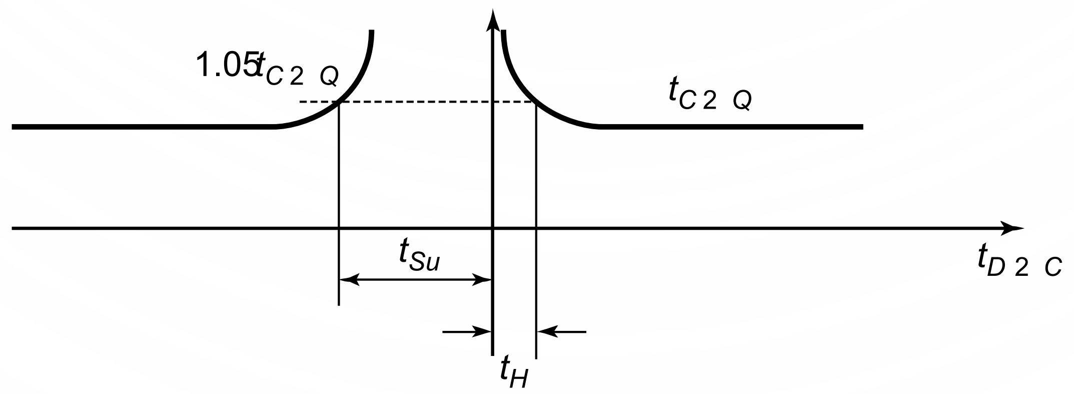
Clk-Q Delay



Setup and Hold Times



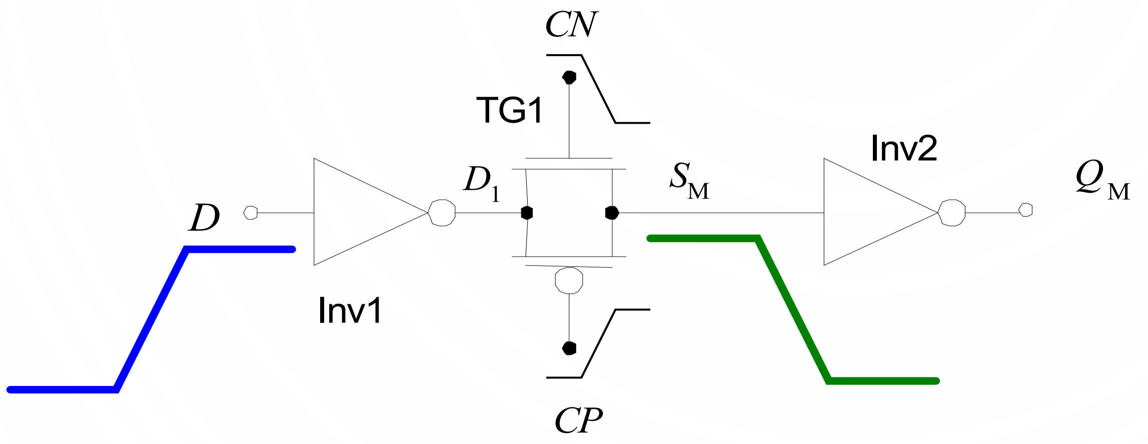
(a)

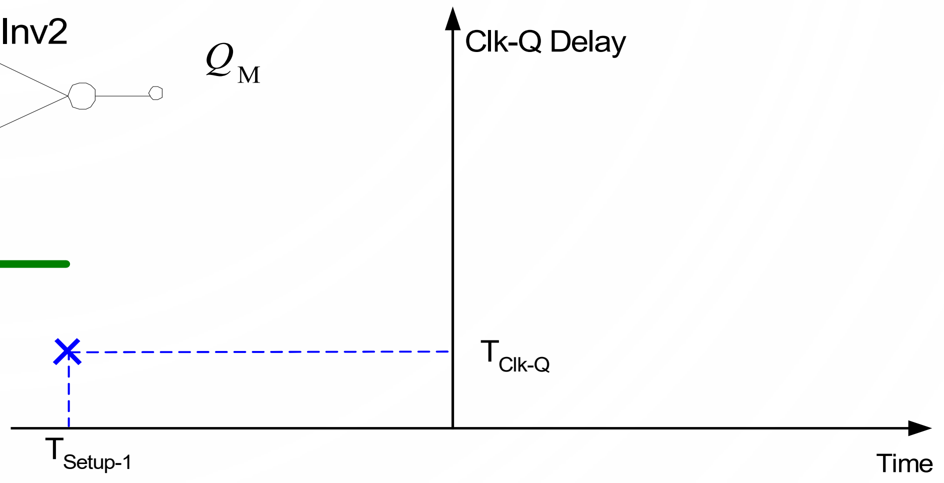


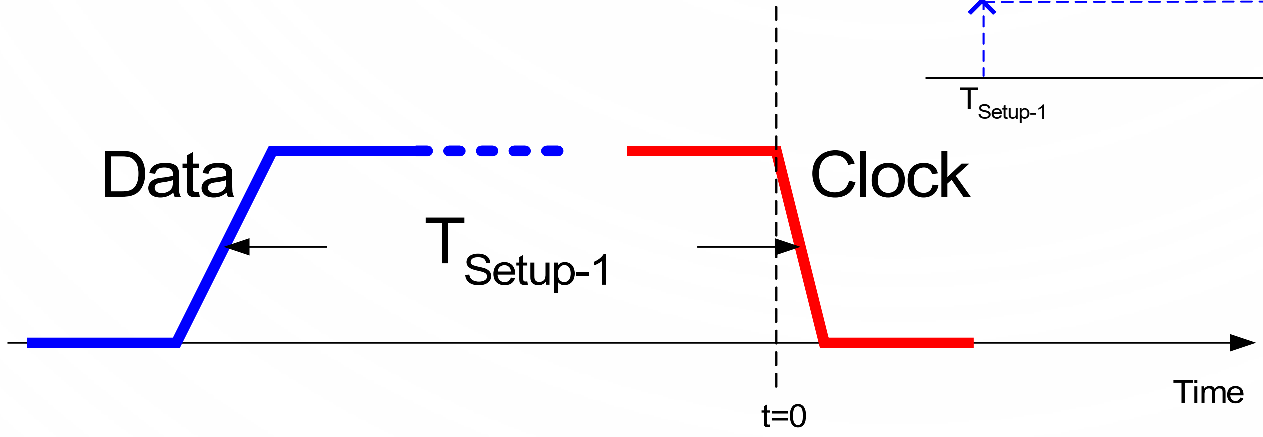
(b)

Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

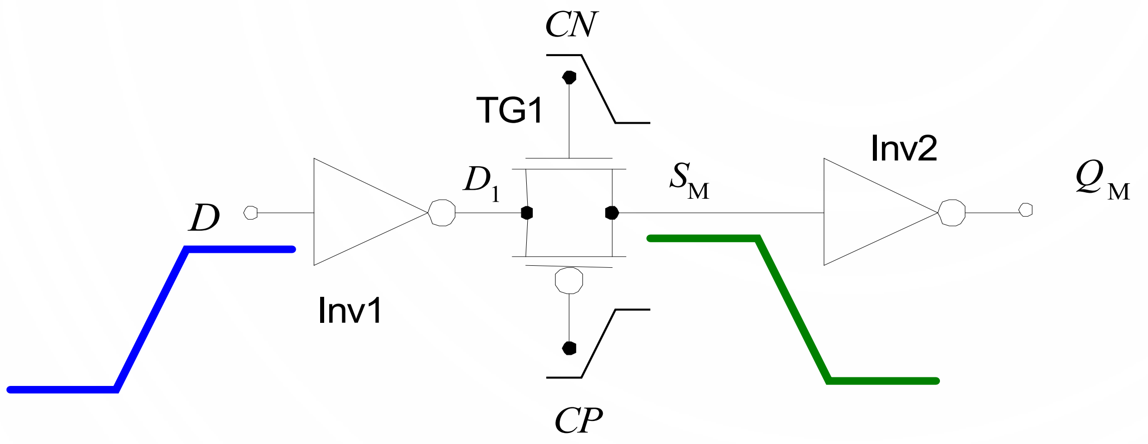


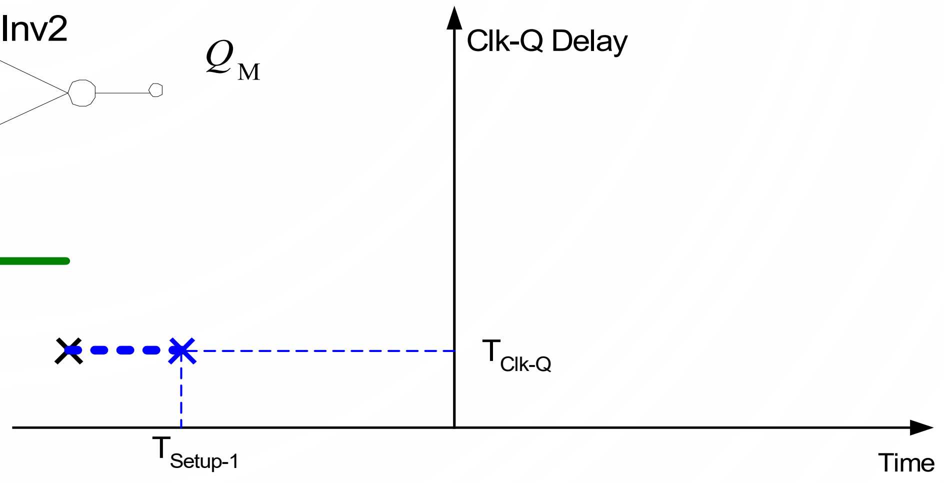


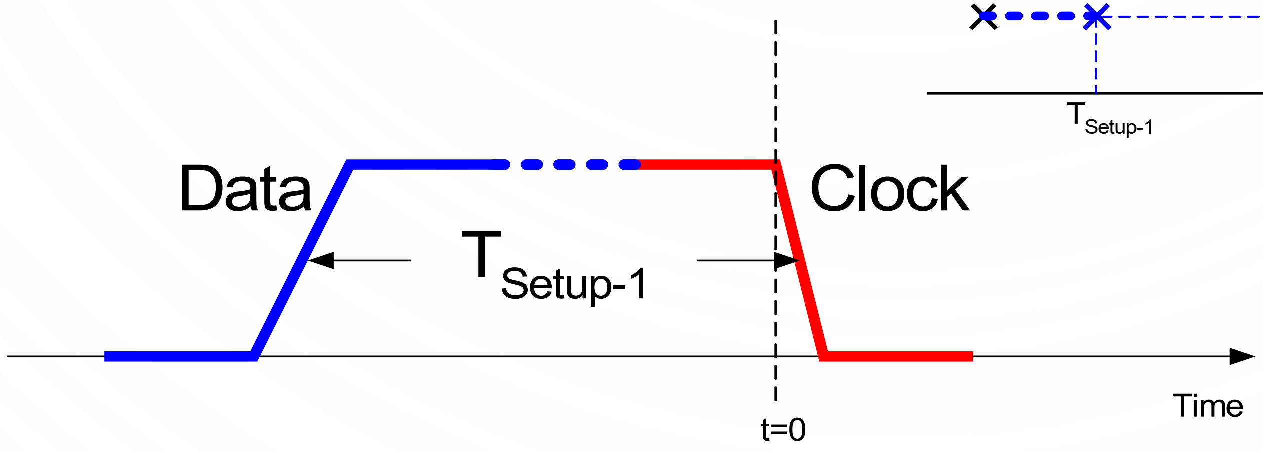


Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

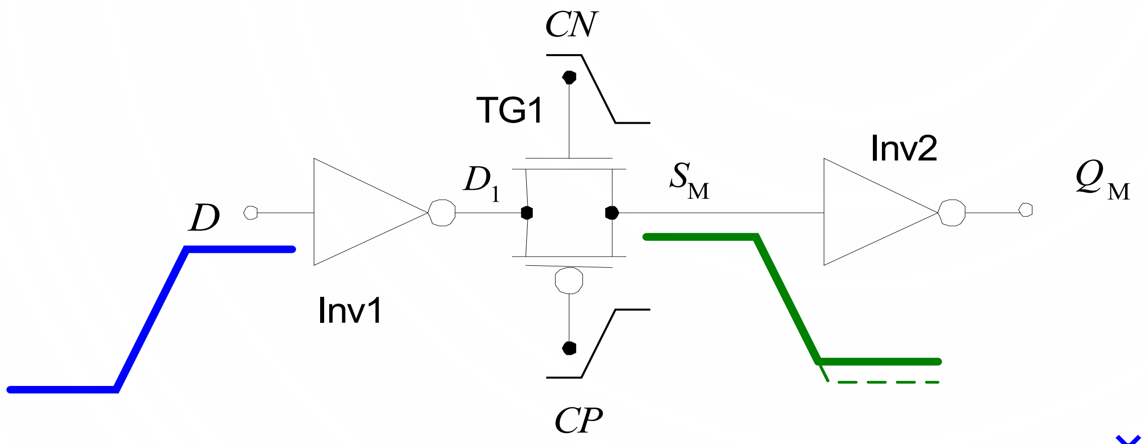


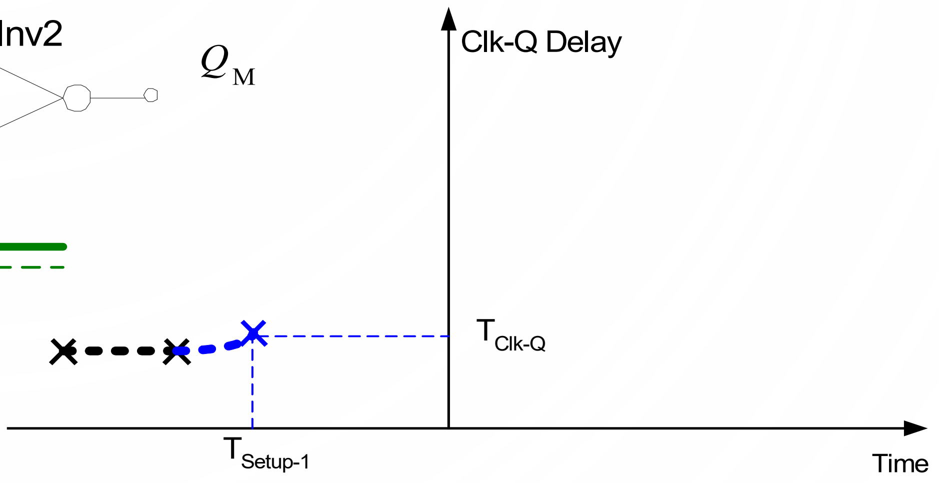


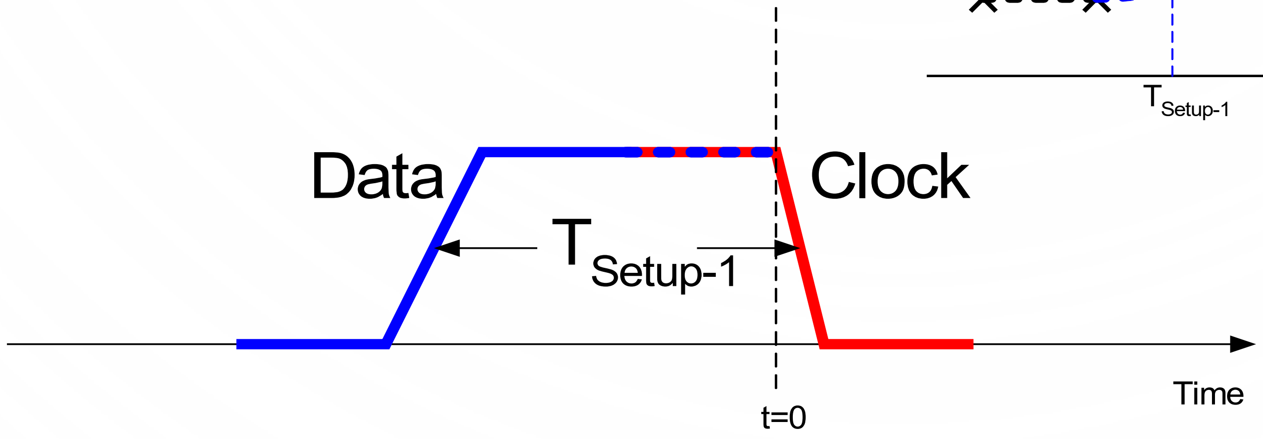


Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

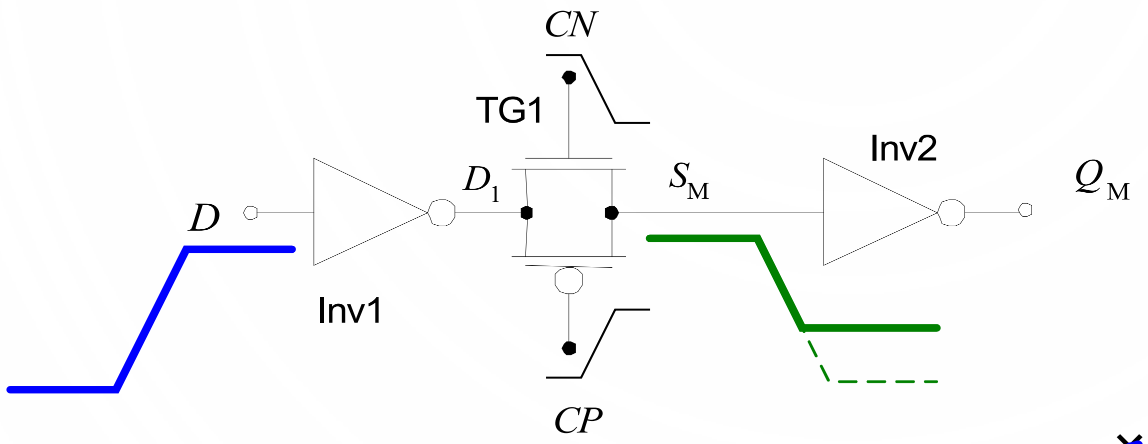


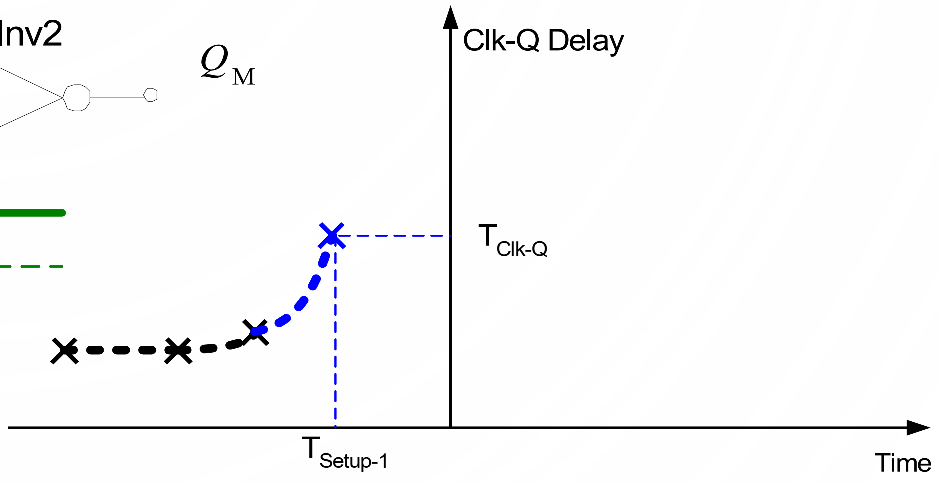


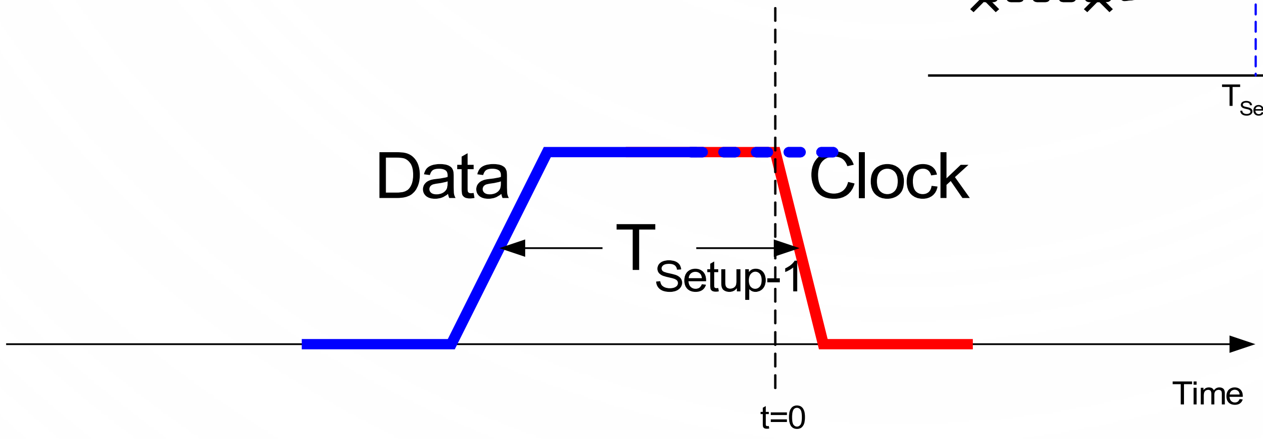


Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

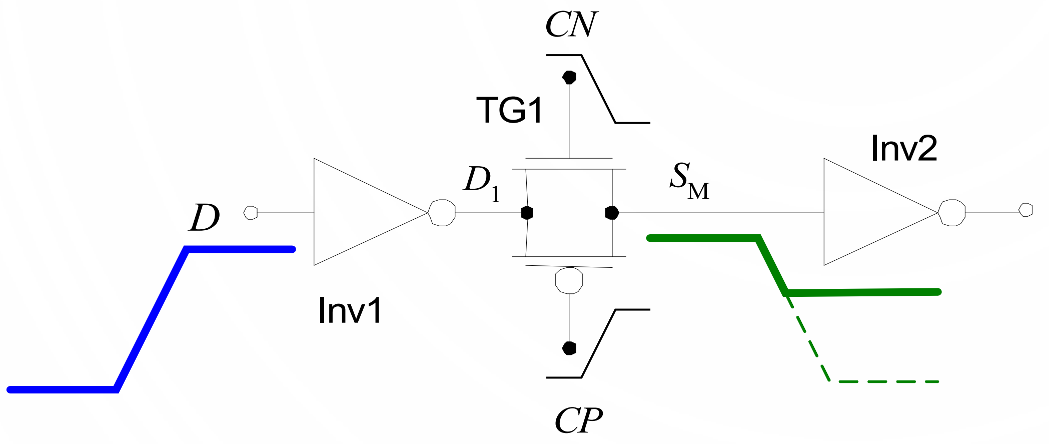


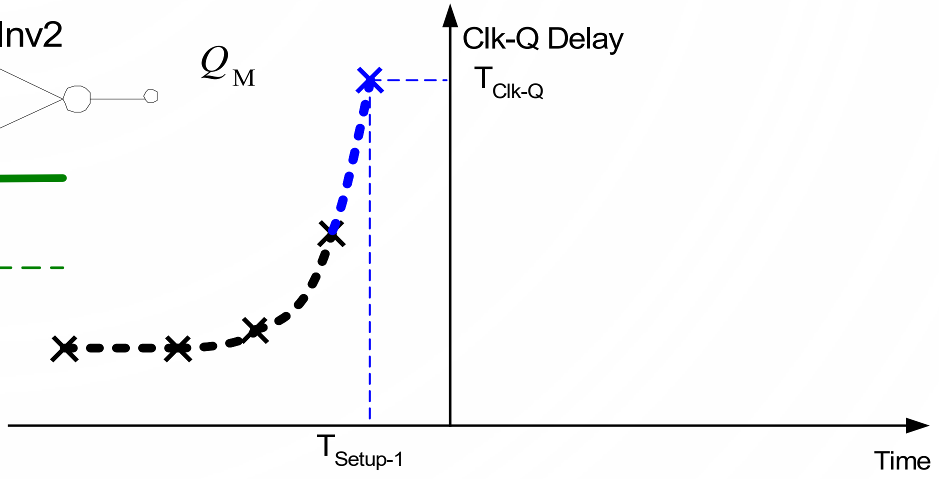


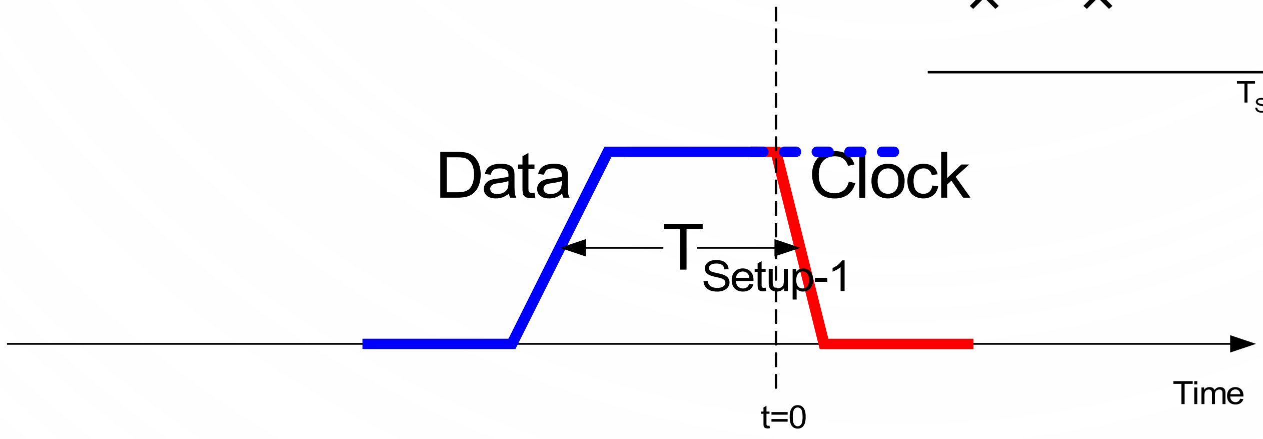


Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)

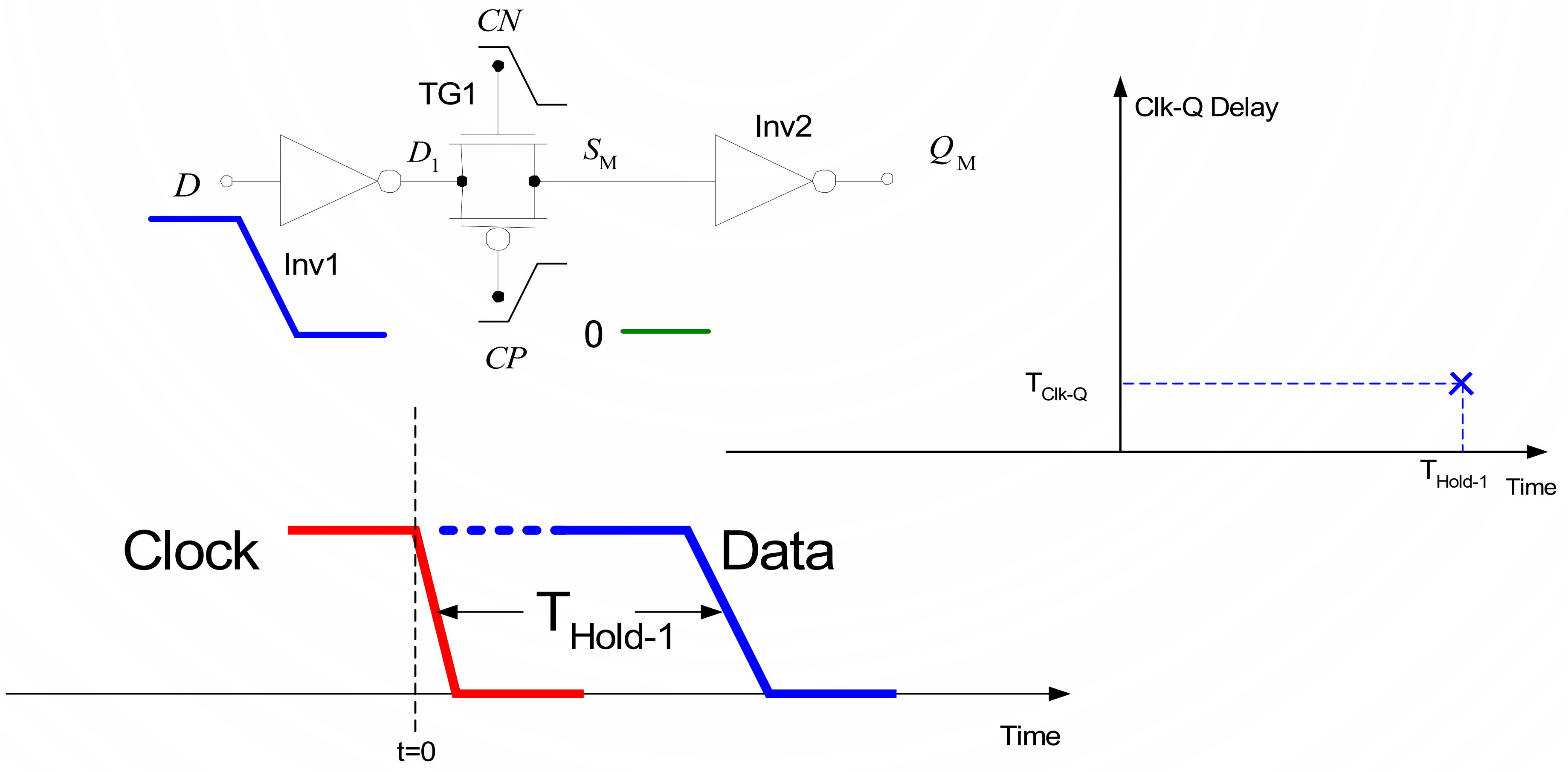






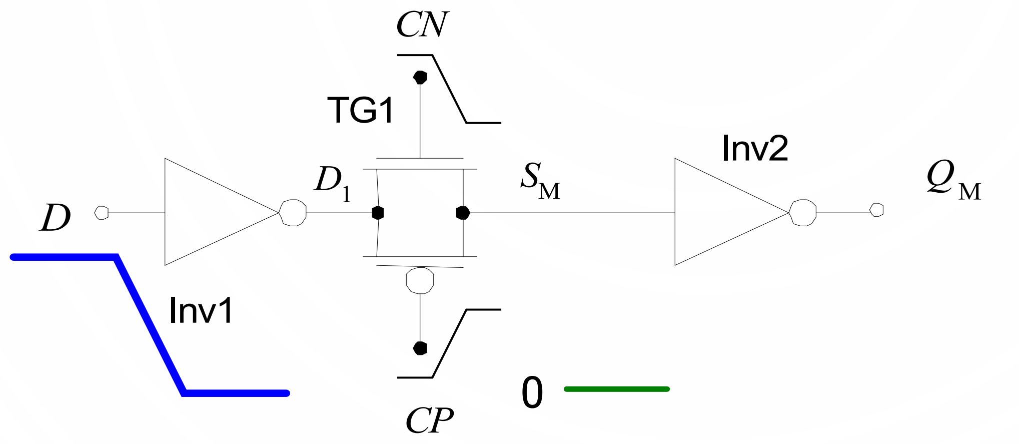
Setup-Hold Time Illustrations

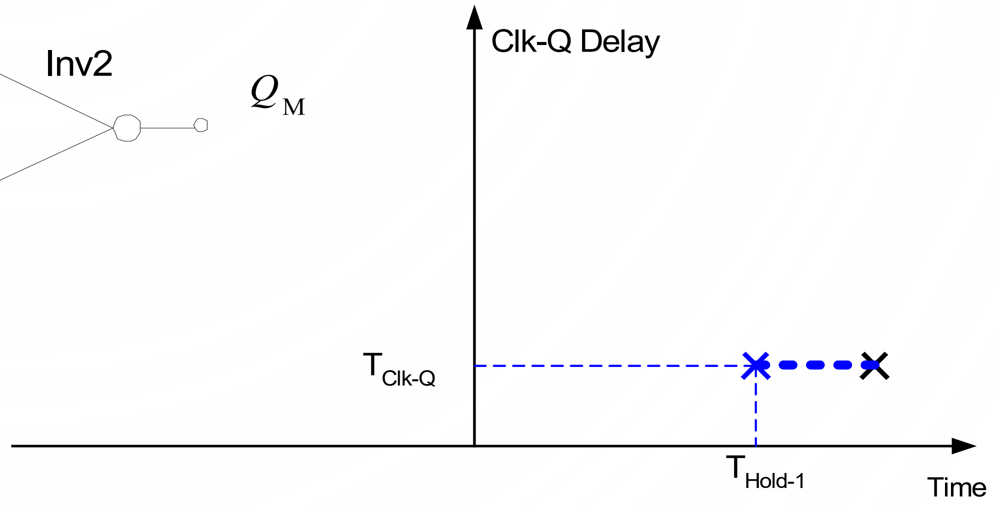
Hold-1 case

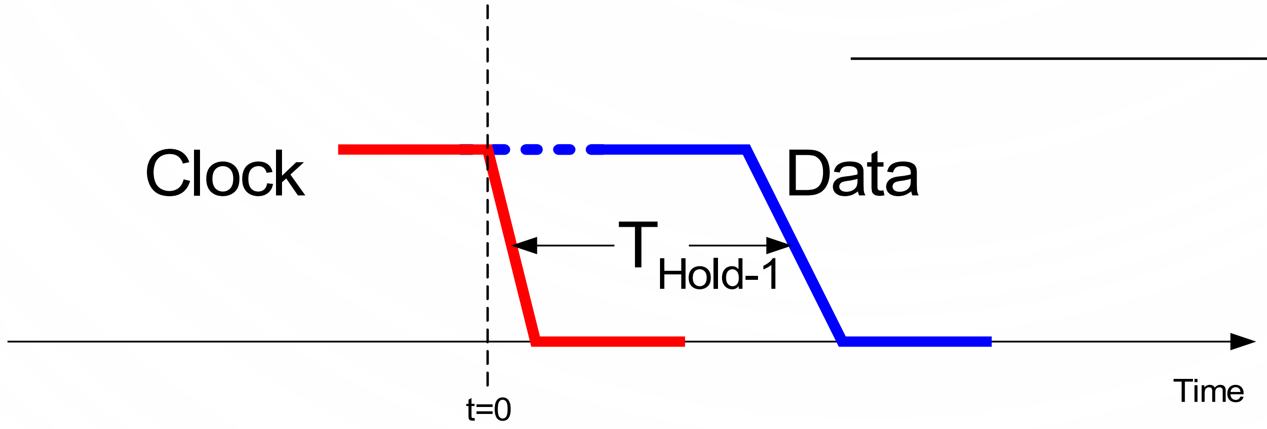


Setup-Hold Time Illustrations

Hold-1 case

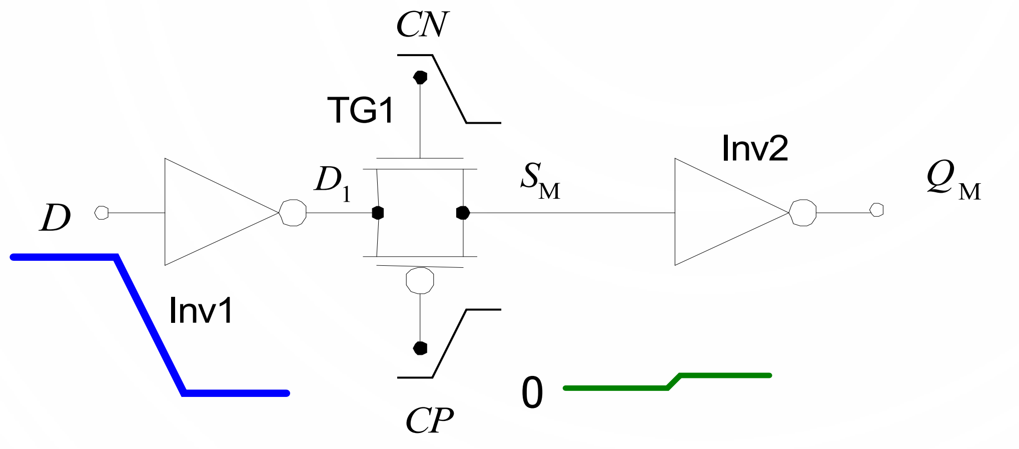


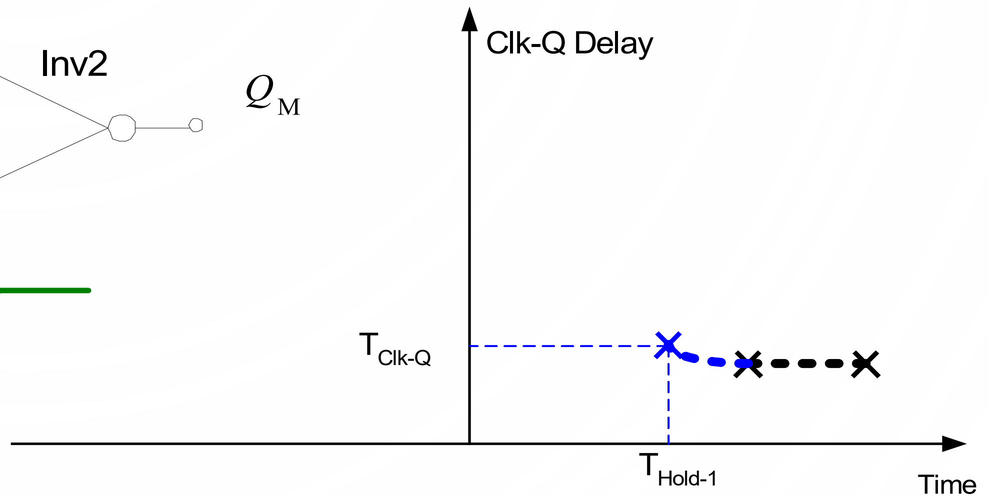


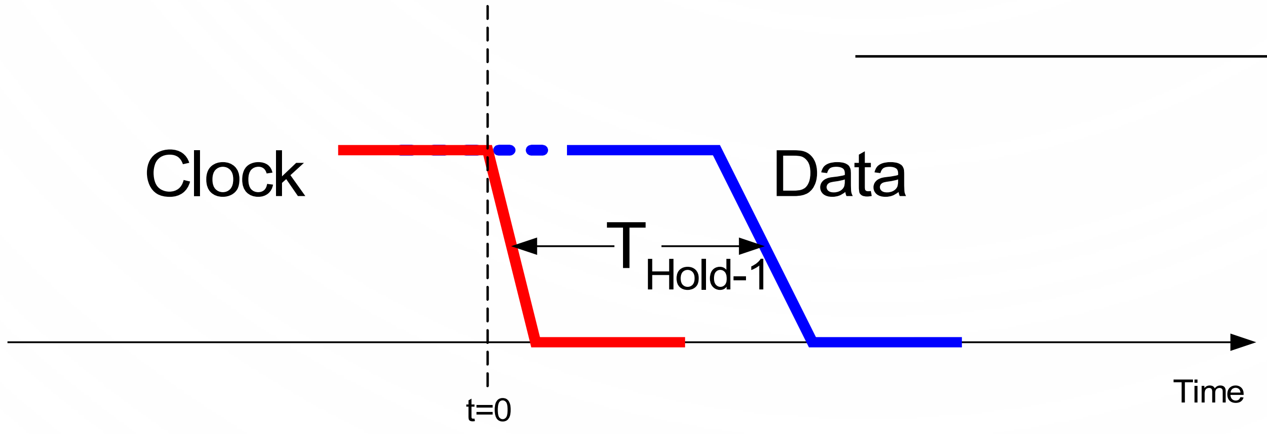


Setup-Hold Time Illustrations

Hold-1 case

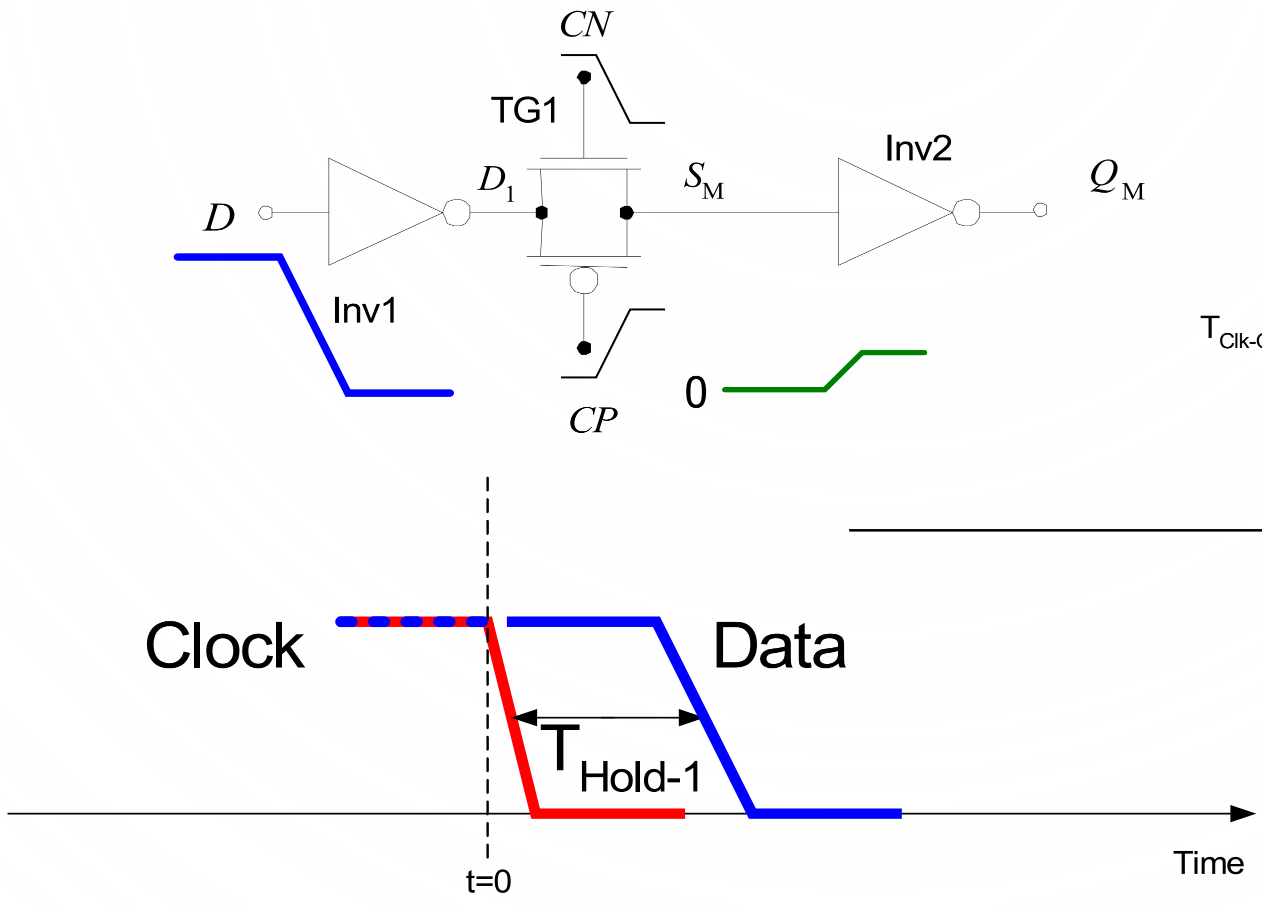


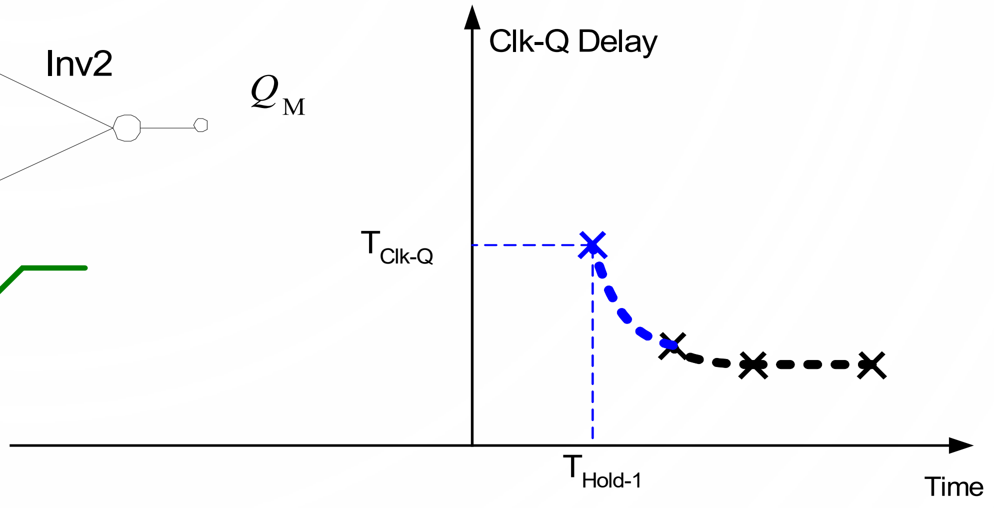




Setup-Hold Time Illustrations

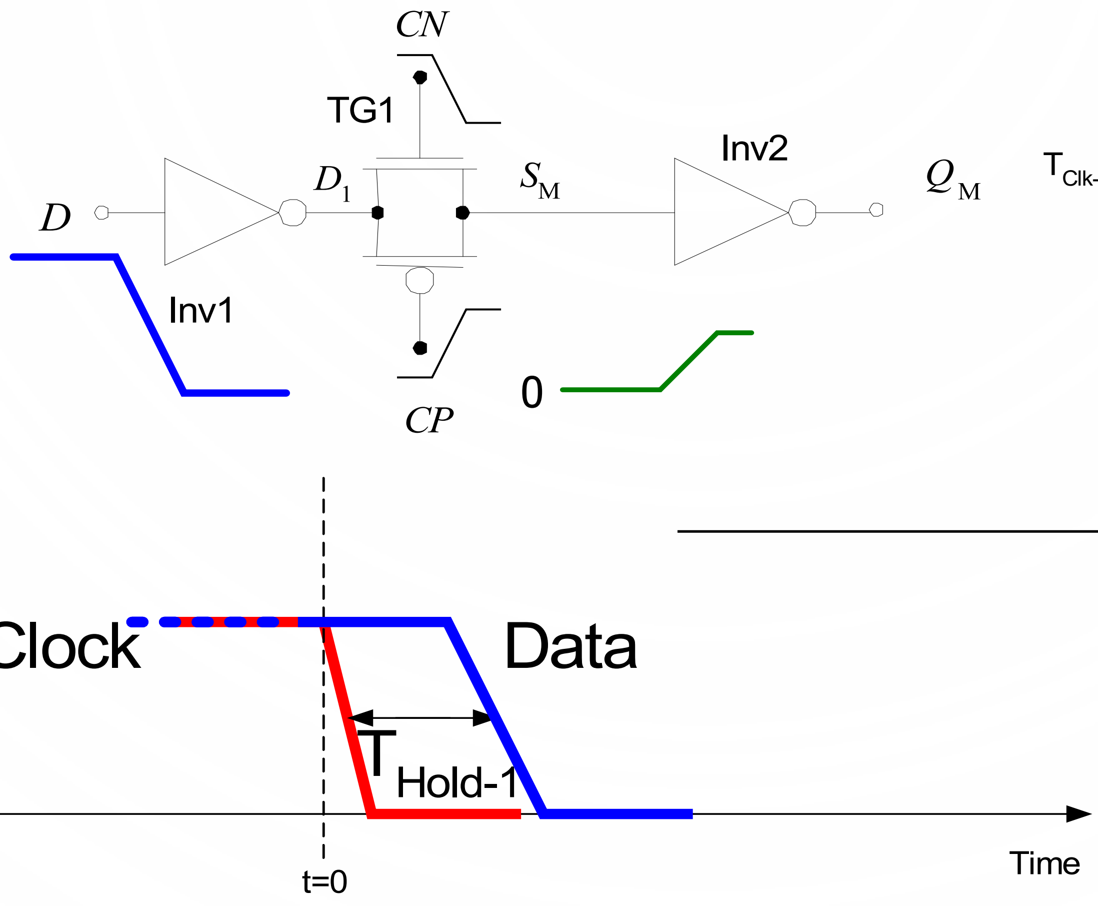
Hold-1 case



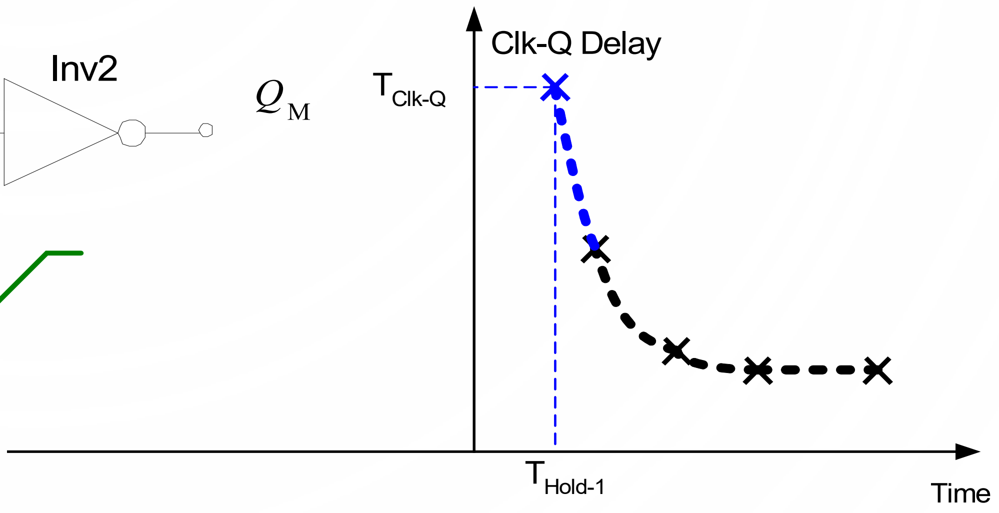


Setup-Hold Time Illustrations

Hold-1 case



⇒



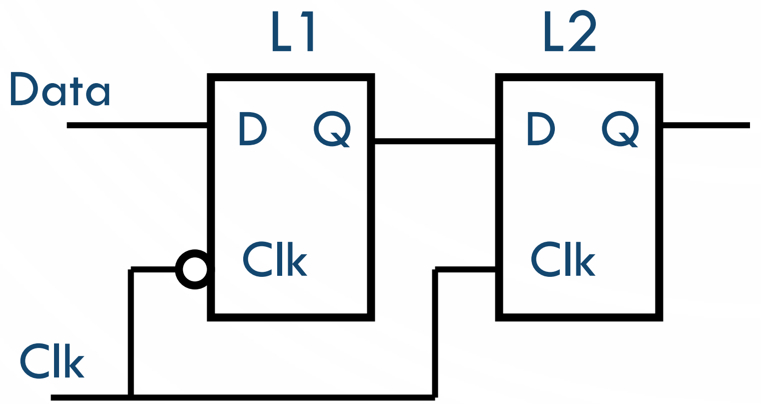
Flip-Flops

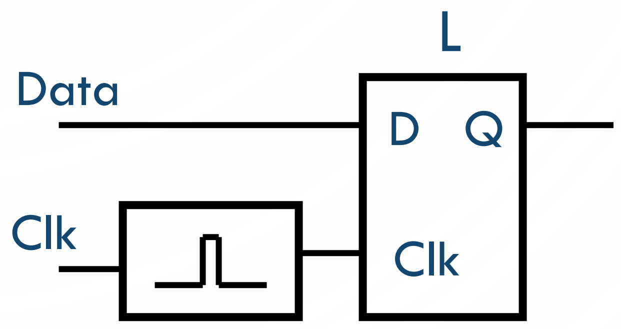
Types of Flip-Flops

Latch Pair

(Master-Slave)

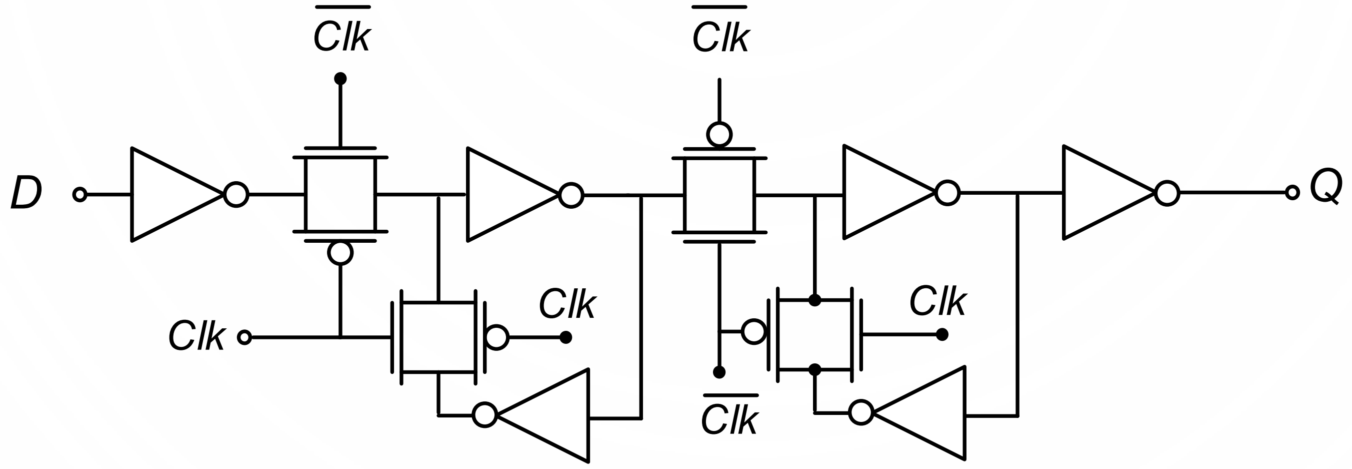
Pulse-Triggered Latch





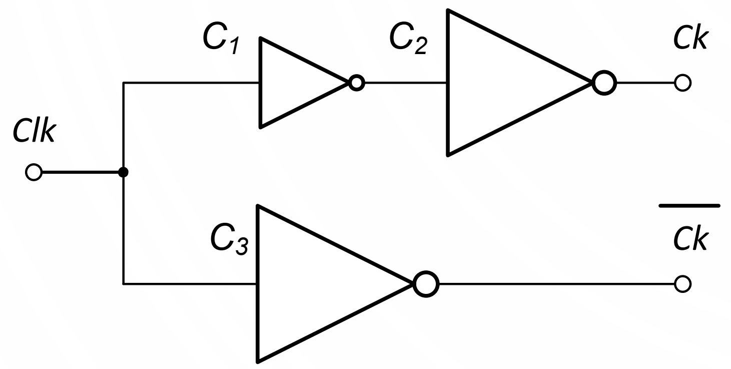
Transmission Gate Flip-Flop

- Two back-to-back latches

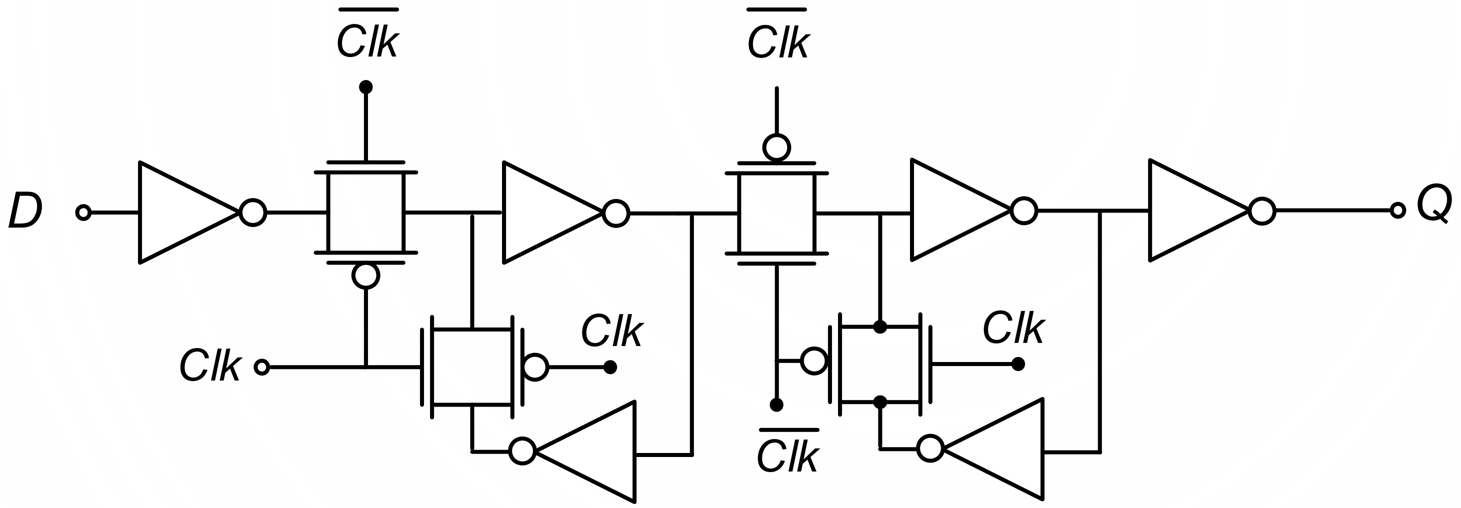


Aside: Inverter Fork

- Often found in flip-flops: equalize Ck, Ckb delays
  - Logical effort = ?



Clk-Q, Setup and Hold Times



Review

- Binary division is a slow, iterative process
- Non-restoring division speeds it up
- SRT divider, higher radix, redundant number representation
- Timing analysis for early and late signal arrivals
- Flip-flop-based pipelines are a lot easier to analyze than latch-based ones
- Latches are based on positive feedback
- Clk-Q delay calculated similarly to combinational logic
- Setup, hold defined as D-Clk times that correspond to Clk-Q delay increases
- Flip-flop is typically a latch pair