VE270 FINAL RC

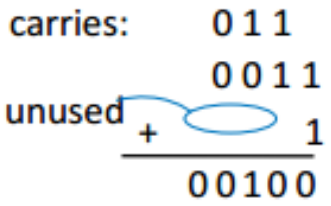
Arithmetic Components & Timing Issues

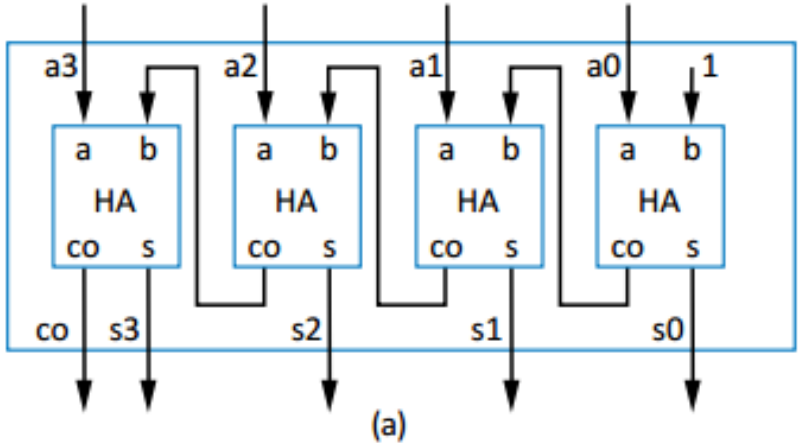
VE270 TA Group

Incrementer

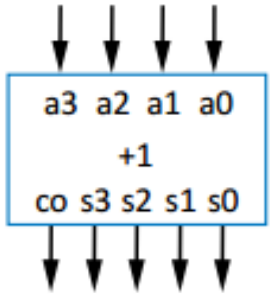
- Traditional design: Capture truth table & derive equation → truth table grows exponentially for large operand
- Alternative design: use N-bit adder with one of the inputs set to 1 → slower but simpler

Inputs				Outputs				
a3	a2	a1	a0	c0	s3	s2	s1	s0
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	1	1
0	0	1	1	0	0	1	0	0
0	1	0	0	0	0	1	0	1
0	1	0	1	0	0	1	1	0
0	1	1	0	0	0	1	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	1	0
1	0	1	0	0	1	0	1	1
1	0	1	1	0	1	1	0	0
1	1	0	0	0	1	1	0	1





(a)

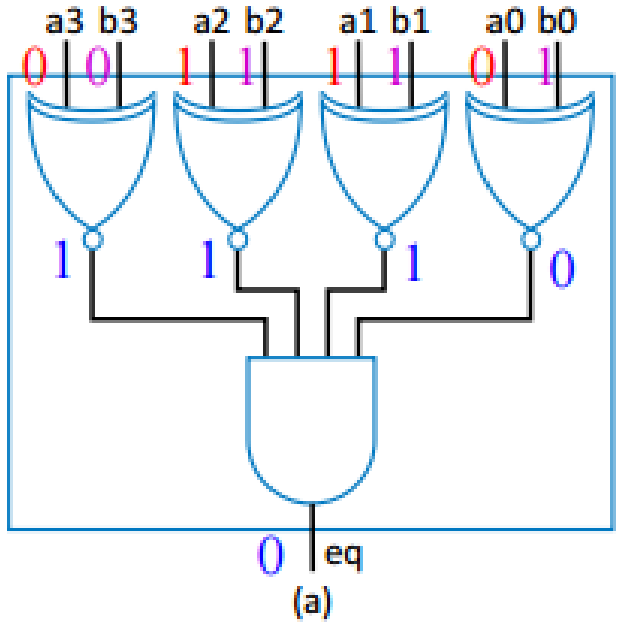


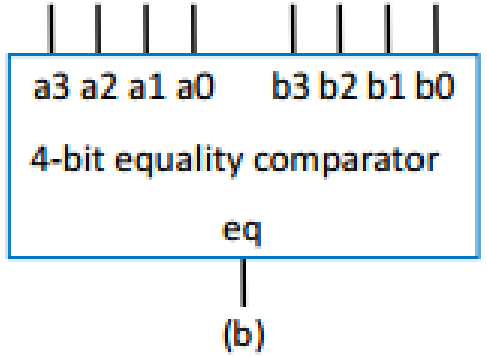
(b)

Comparator

- Outputs 1 if two N-bit numbers are equal
- Recall functionality of XNOR: outputs 1 if two operands are equal

0110 = 0111 ?





Magnitude Comparator

- Indicates whether A>B, A=B, or A<B, for its two N-bit inputs A and B
- Start at left, compare each bit pair, pass results to the right
- At each stage:
  - out_gt = in_gt + (in_eq * a * b')
  - out_lt = in_lt + (in_eq * a' * b)
  - out_eq = in_eq * (a XNOR b)

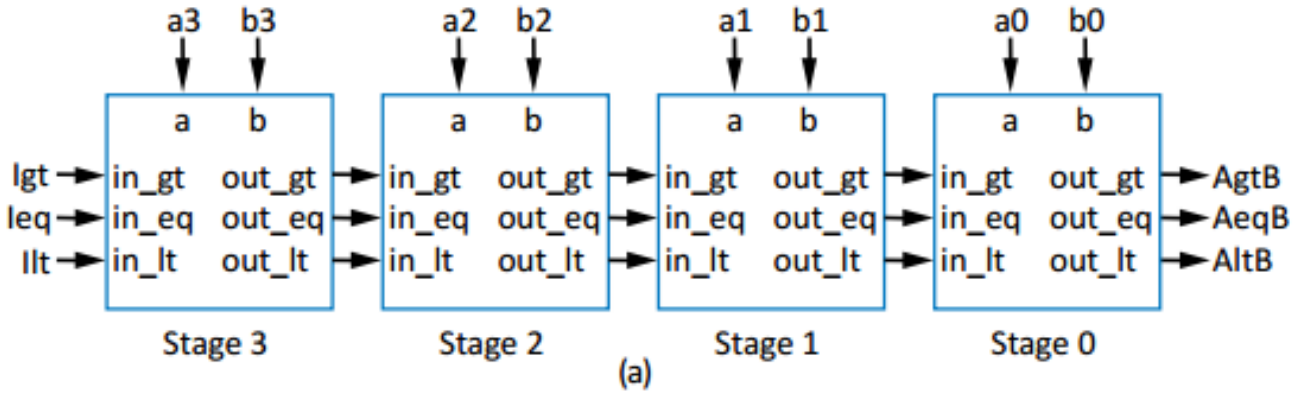
A=1011 B=1001

1011 1001 Equal

1011 1001 Equal

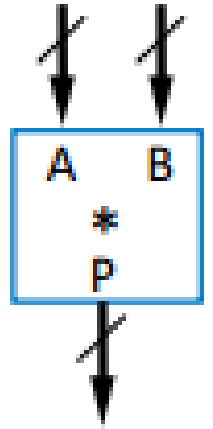
1011 1001 **Unequal**

So A > B



Multiplier

- Mimics multiplication by hand



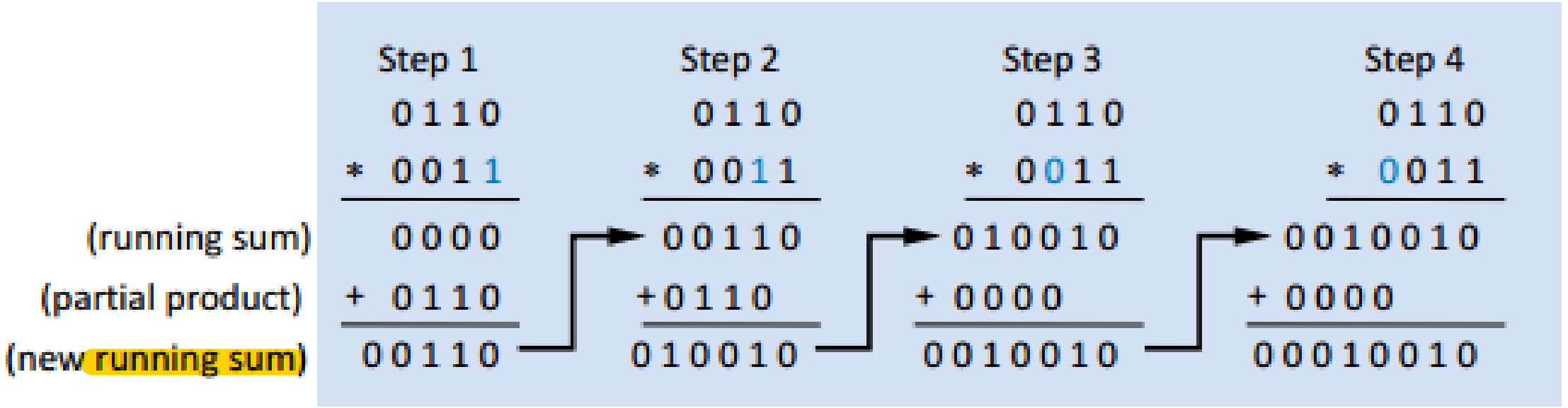
Block symbol

```
    0110                    a3   a2   a1   a0
    0011                 x  b3   b2   b1   b0
    ----    ------------------------------------
    0110                   b0a3 b0a2 b0a1 b0a0    (pp1)
   0110                b1a3 b1a2 b1a1 b1a0   0    (pp2)
  0000             b2a3 b2a2 b2a1 b2a0   0    0    (pp3)
+0000       + b3a3 b3a2 b3a1 b3a0   0    0    0    (pp4)
--------    ------------------------------------
00010010    p7   p6   p5   p4   p3   p2   p1   p0
```

need 3 adders

Smaller Multiplier -- Sequential (Add-and-Shift) Style

- Compute one partial product at a time and maintain a running sum



pp1 pp1+pp2 pp1+pp2+pp3 pp1+pp2+pp3+pp4

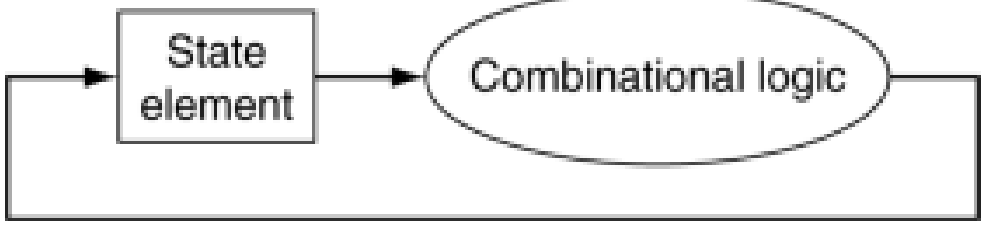
only need 1 adder, but need more time

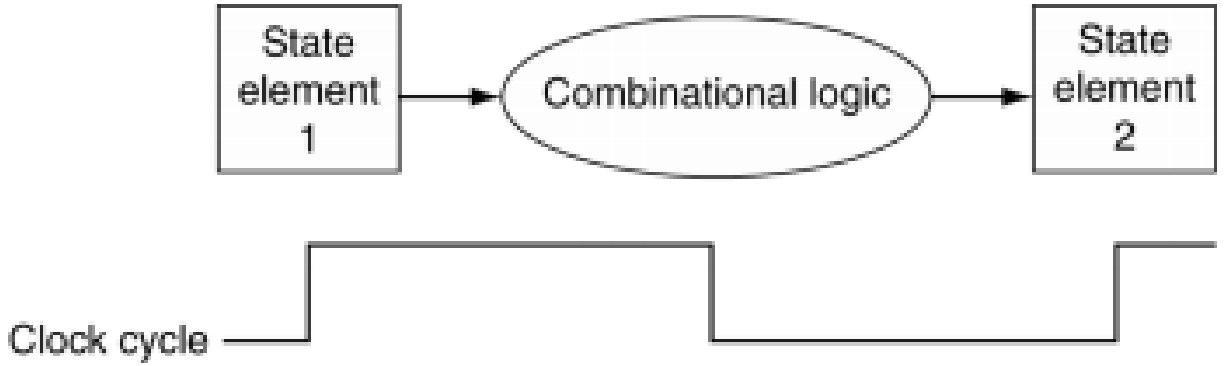
Timing issue

- Timing requirements for FSM
- Setup time and hold time
- Asynchronous input and metastability
- Synchronizer
- Switch debouncing
- Clock skew
- Hazards

Clocking Methodology for FSM

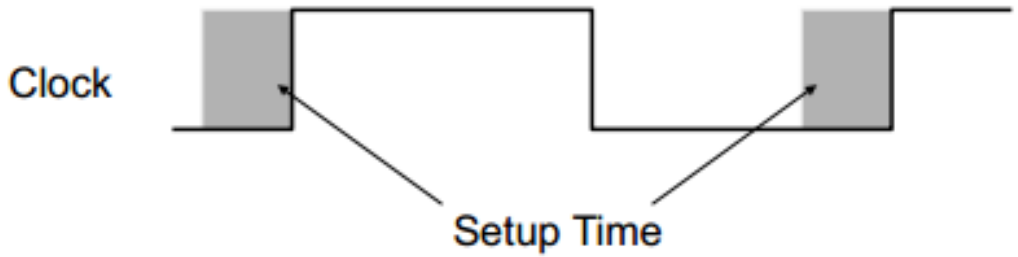
- Clock cycles should be
  - Long enough to allow combinational logic completes computation
    - Longest delay determines clock period
  - Short enough to ensure acceptable performance and to capture small changes on external inputs



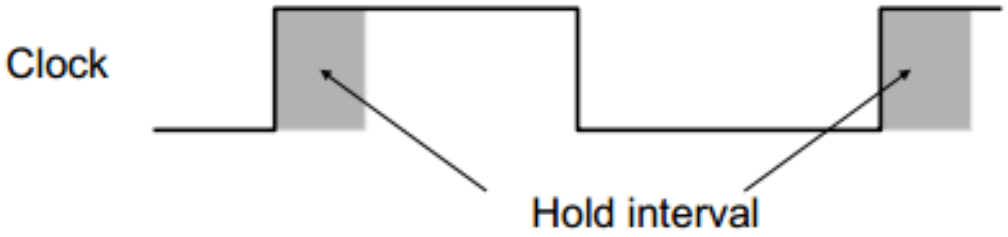


Setup time & hold time

- Setup Time
  - Minimum time that data must be stable prior to the triggering edge
  - Fix: stretch clock, optimize combinational circuit



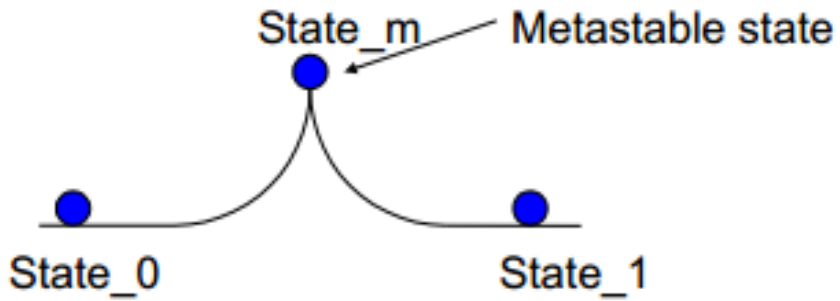
- Hold Time:
  - Minimum time that data must remain stable after the triggering edge
  - Fix: insert buffers

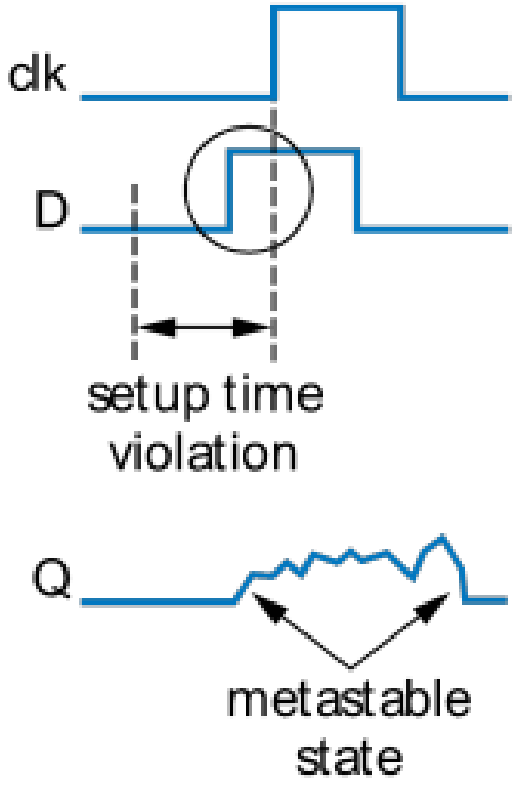


Metastable state & asynchronous inputs

- Synchronous system may be driven by asynchronous inputs which may put flip flops into the metastable state

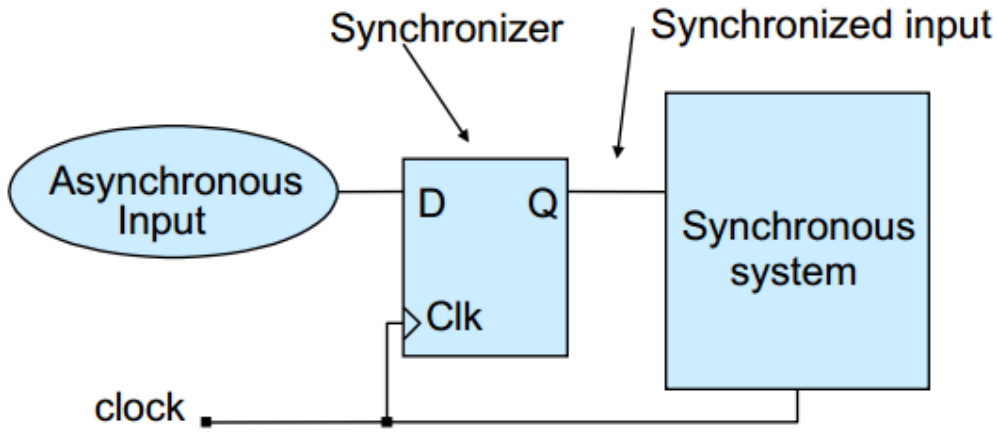
- General rule: Ensure that all inputs are synchronous, i.e. synchronize them to the clock signal. Reset and set can be exceptions.





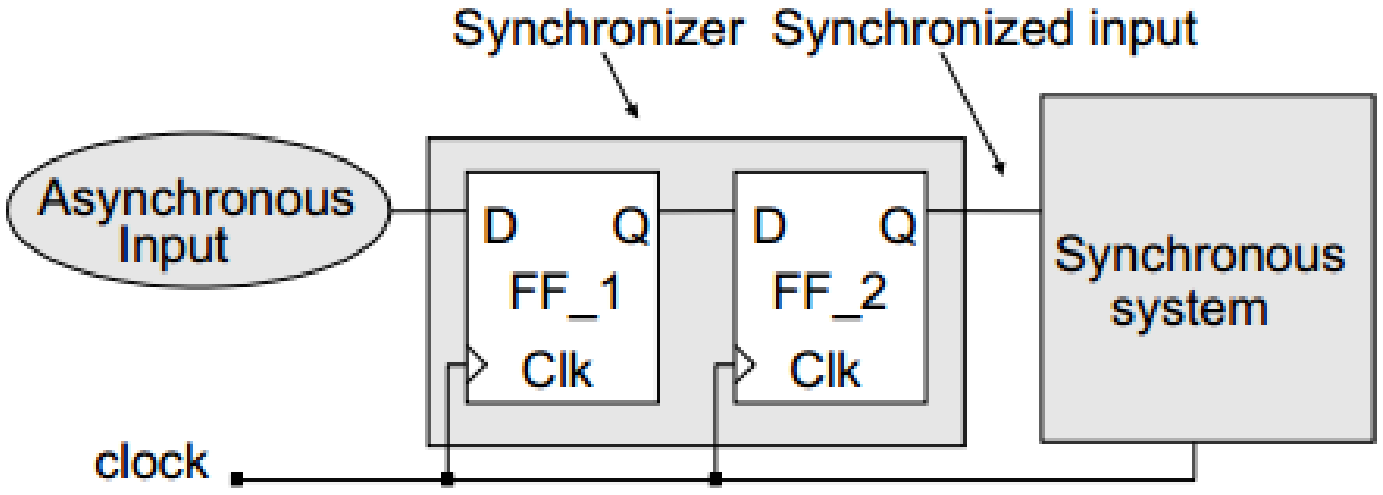
Synchronizer

- Use a D-type flip flop to synchronize the input.
- Synchronizer failure: The flip-flop may be in metastable state, then its state will be nondeterministic, its output may enter a metastable state and is not synchronized anymore



Reduce possibility of synchronizer failure

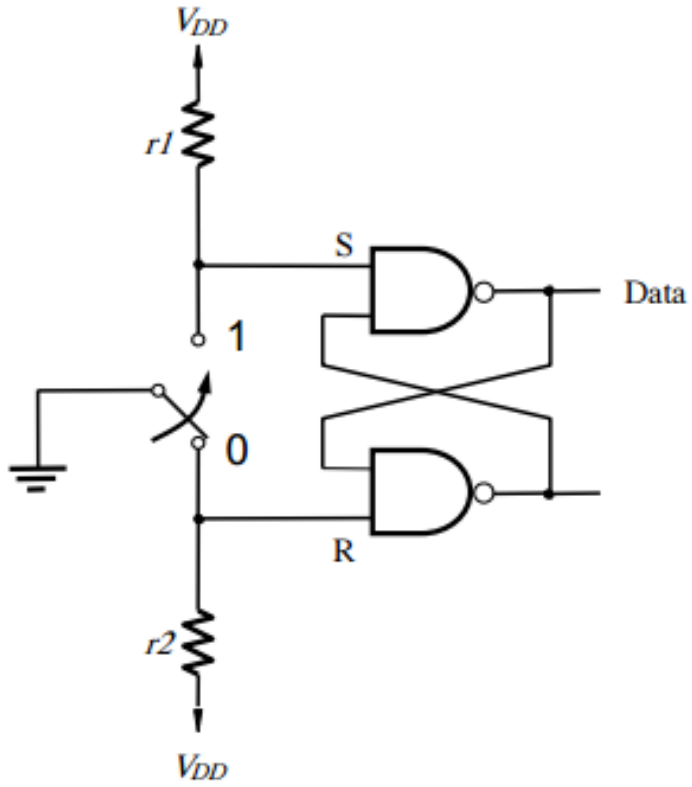
- Stretch the clock period to allow more time for the circuit to recover → less synchronizer failure, but slower circuit
- Double DFF synchronizer: The first flip-flop might enter a metastable state, but has a clock period to recover. Both flip-flop must enter a metastable state to have a synchronizer failure.



Switch debouncing

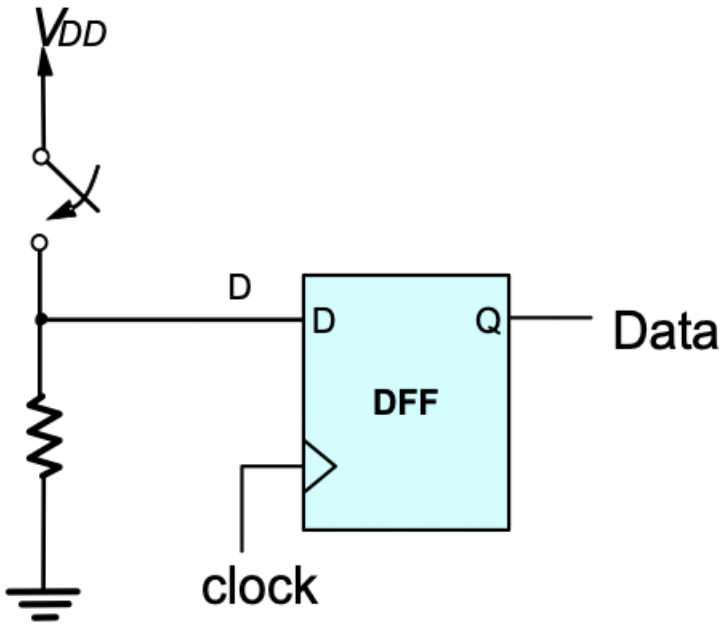
- SR Latch
  - When throw is at position 0, S = 1, R = 0, Data = 0
  - When throw changes to position 1 R = 1, S = (0⟷1) Data = 1

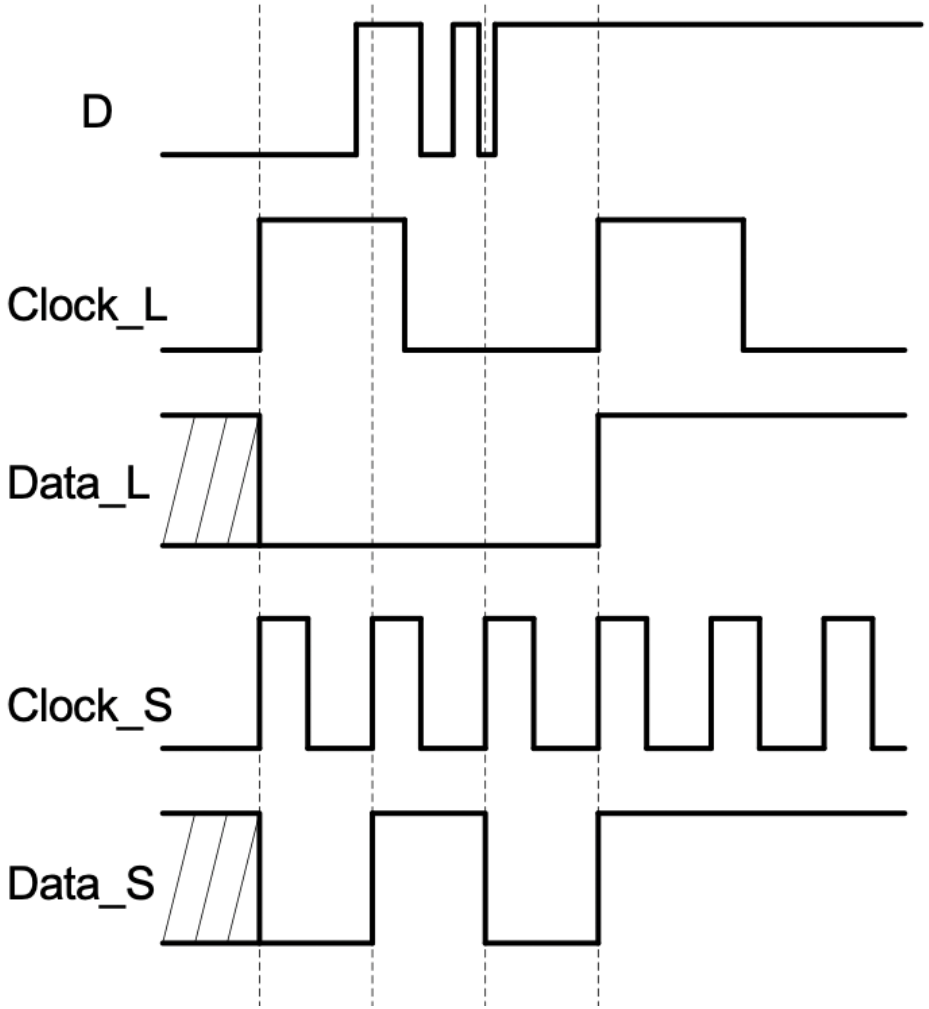
S	R	Q	Q^+	
0	0	0	X	not allowed
0	0	1	X	
0	1	0	1	set
0	1	1	1	
1	0	0	0	reset
1	0	1	0	
1	1	0	0	hold
1	1	1	1	



Switch debouncing

- D Flip Flop: clock needs careful design



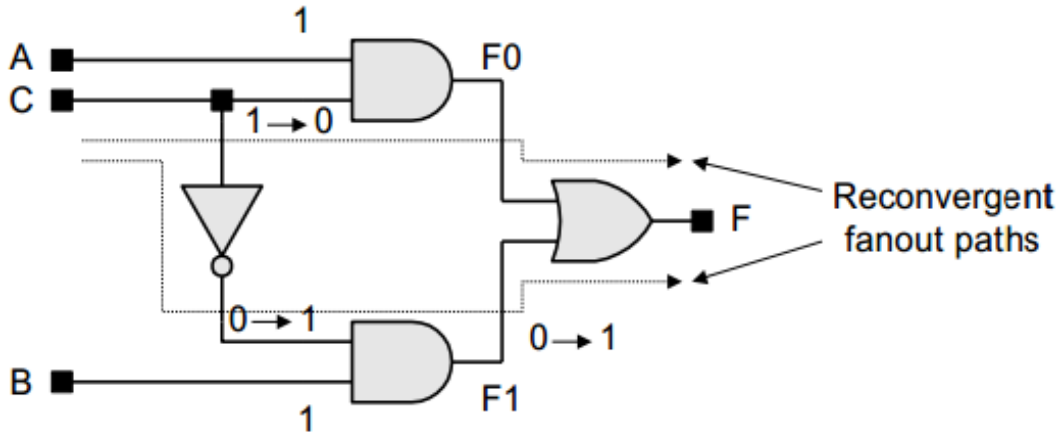


Clock skew

- The situation in which the clock signal arrives at different flip flops at different times is known as clock skew
- Clock skew may cause synchronous circuit behaves asynchronously

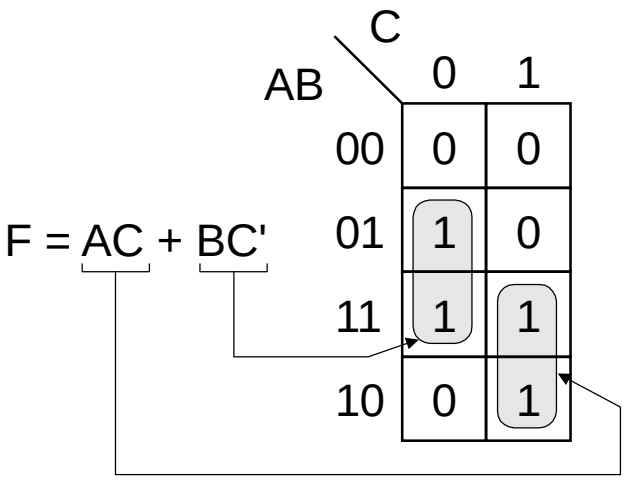
Static hazards

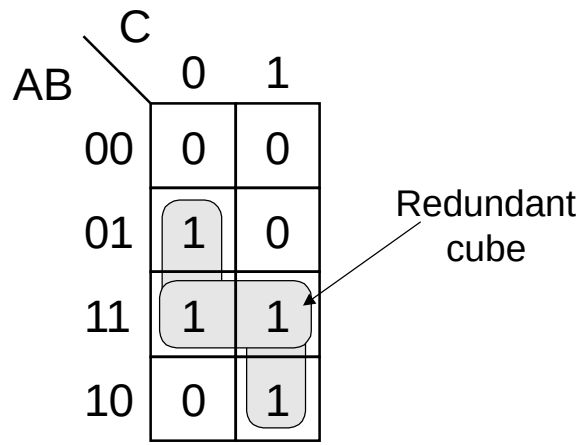
If transition is within the same group in the K-map, no hazard



- static 1-hazard

- static 0-hazard





F = AC + BC’ + AB

Dynamic hazards

- output changes more than once as a result of a single output change
- Elimination of all static hazards eliminates dynamic hazards.