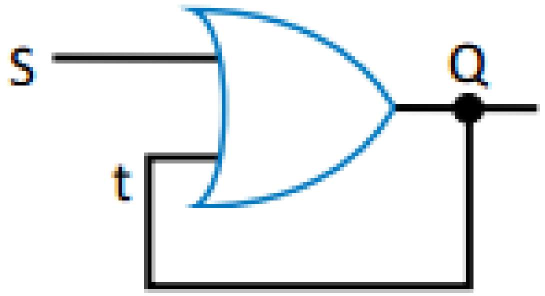
Latch & Flip flop

Sequential Circuit

- Output depends on both present inputs and history of inputs and outputs.
- It is combinational circuit with feedback



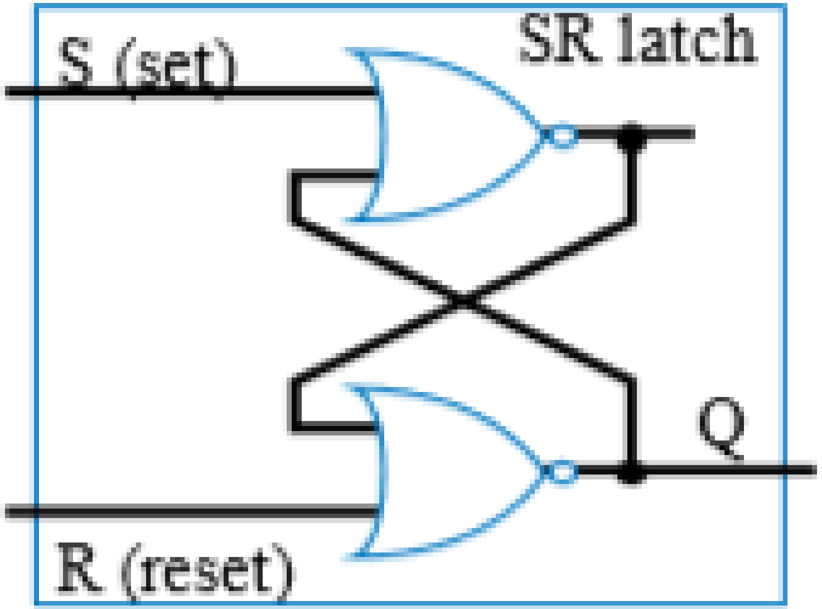
Latch & Flip-flop

- Building blocks that can store value of a bit
- Latch is level sensitive
- Filp-flop is edge sensitive

SR Latch

- S=1: set Q to 1
- R=1: reset Q to 0

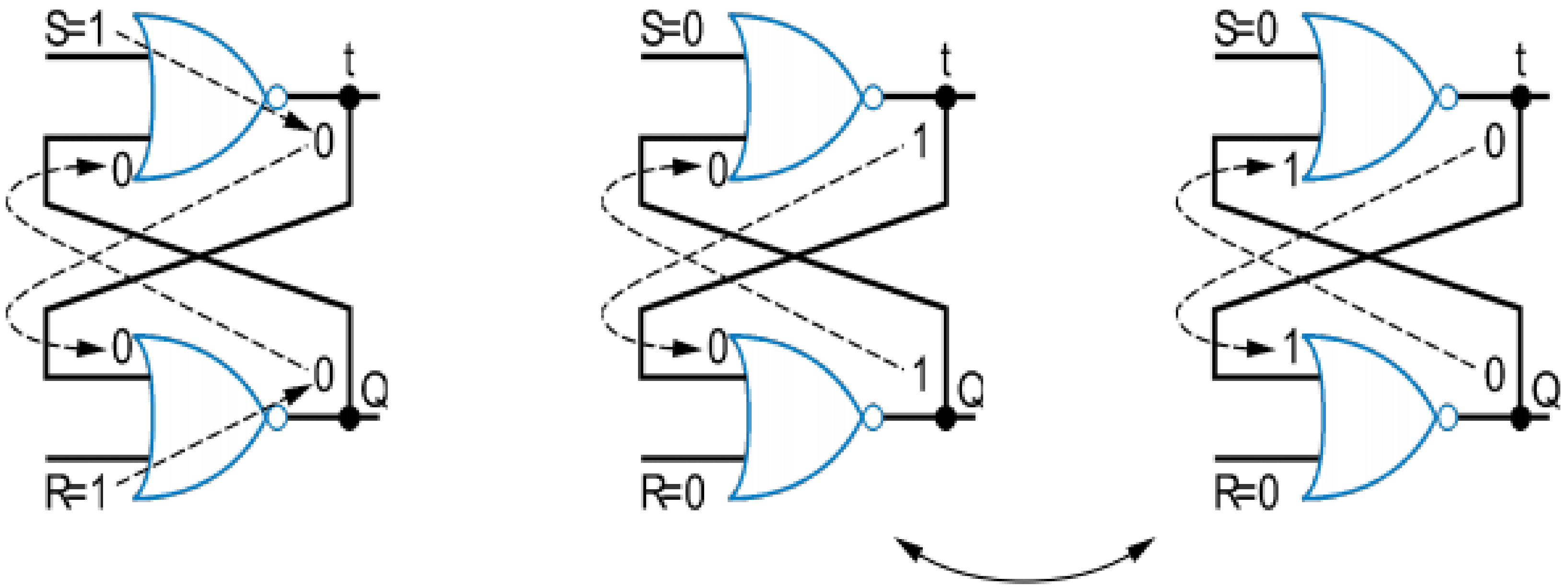
S(t)	R(t)	Q(t)	Q(t+Δ) → Q^+	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	



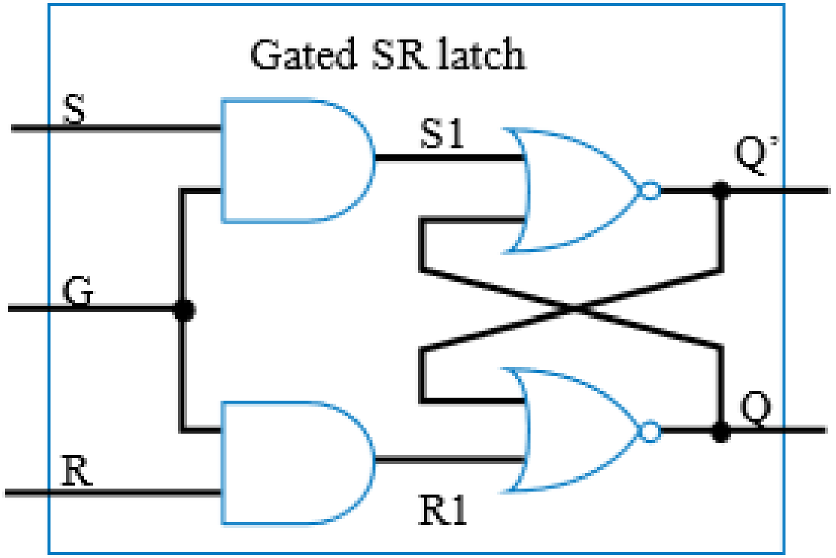
$Q^+ = S + R'Q$

SR Latch

- When S=1 & R=1, Q may oscillate when they both return to 0 simultaneously.



Gated SR Latch

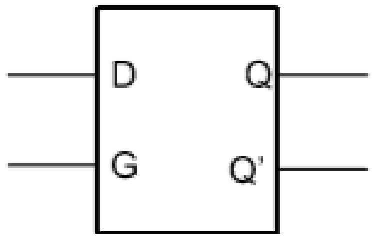


Characteristic Table

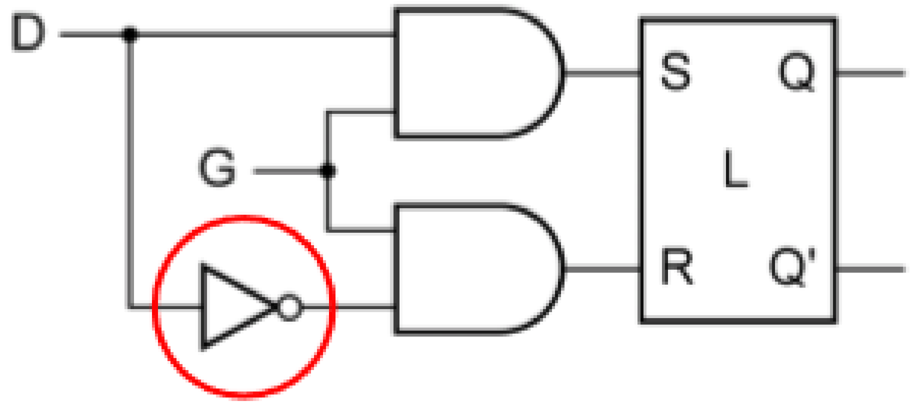
G	S	R	Q^+
0	x	x	Q; Latch locked
1	0	0	Q; Hold state
1	0	1	0; Reset state
1	1	0	1; Set state
1	1	1	not allowed

Gated D Latch

- No unstable state as in SR latch
- Q = D when G = 1
- Q holds when G = 0



D latch
symbol

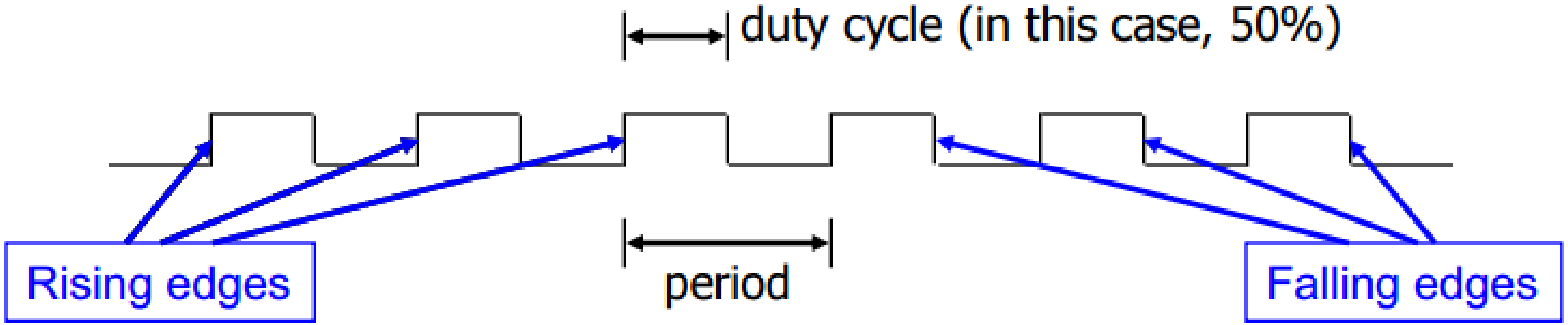


Characteristic Table

G	D	Q^+
1	0	0
1	1	1
0	X	Q

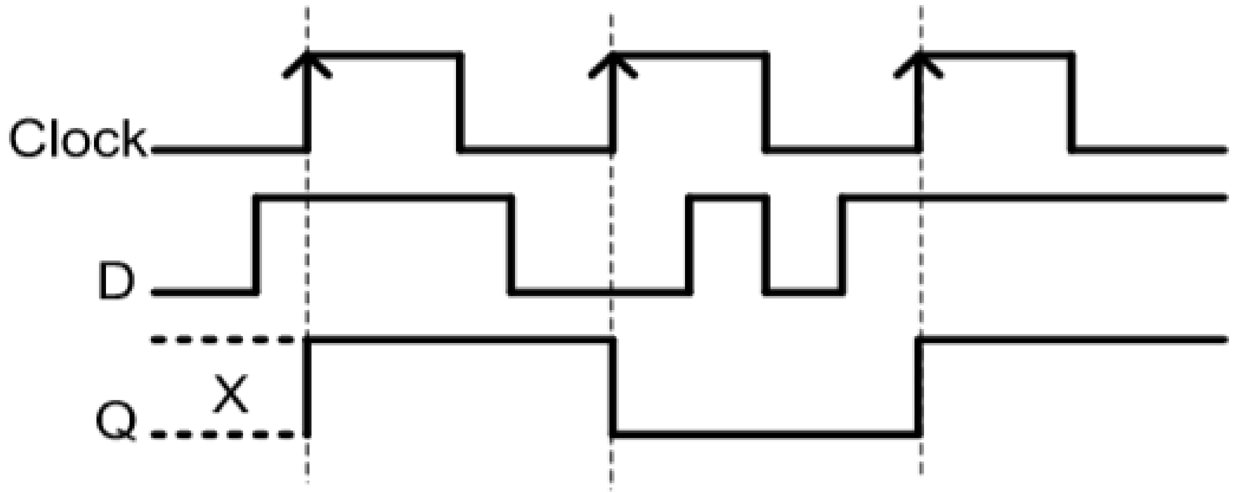
Clock Signal

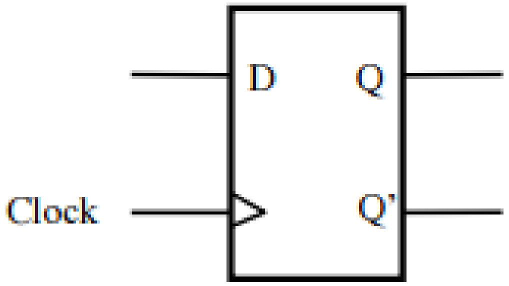
- Periodic pulse train
- Clock period: time interval between pulses



Rising-Edge Triggered D Filp Flop

- Edge sensitive

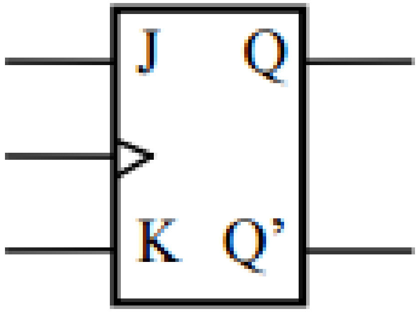




clock	D	Q^+
↑	0	0
↑	1	1
0	X	Q
1	X	Q

Characteristic equation:
$Q^+ = D$ (at active clock edges)

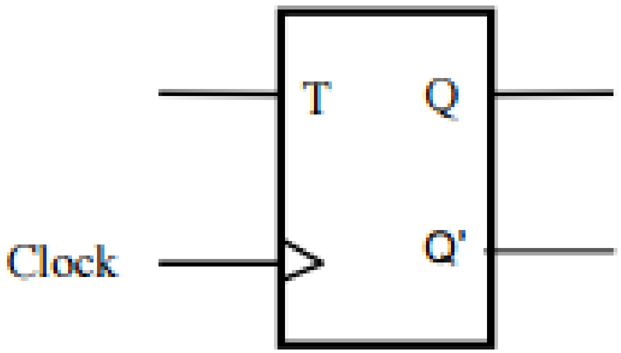
Rising-Edge Triggered J-K Filp Flop



J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q'

Characteristic equation:
$Q^+ = JQ' + K'Q$

Rising-Edge Triggered T Filp Flop



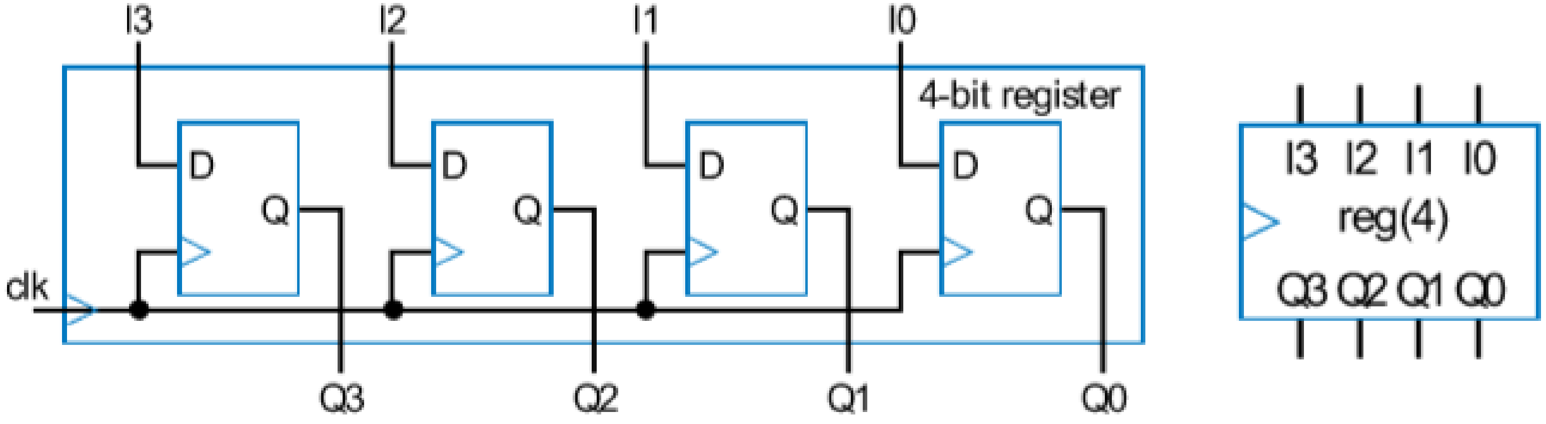
clock	T	Q^+
$\uparrow$	0	Q
$\uparrow$	1	Q'

Characteristic equation:

$$Q^+ = T'Q + TQ' = T \oplus Q$$

Register

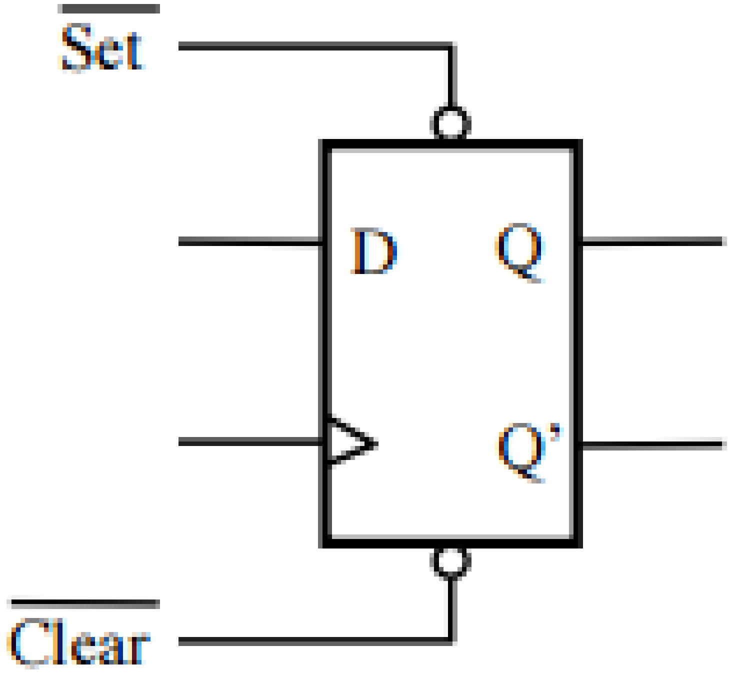
- Multiple filp-flops sharing clock signal
- Store muliple bits
- Load values simultaneously at rising edge

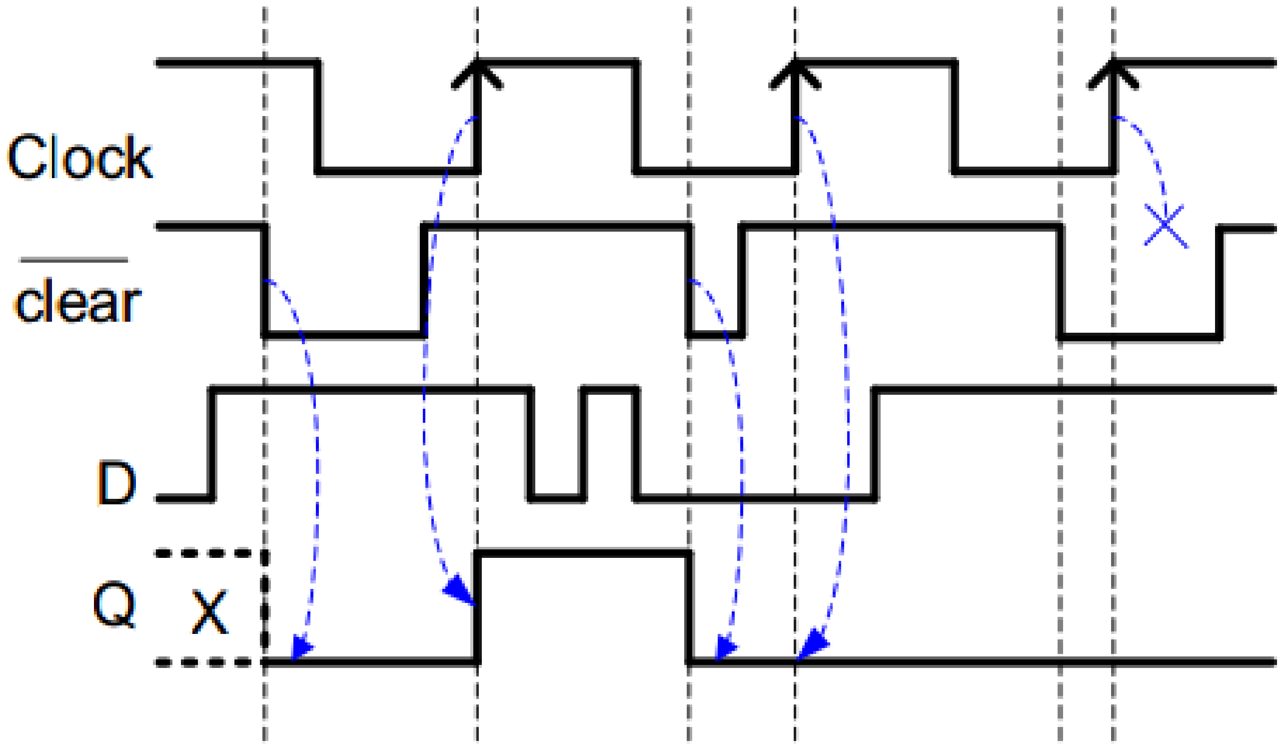


Control Inputs for Flip Flops

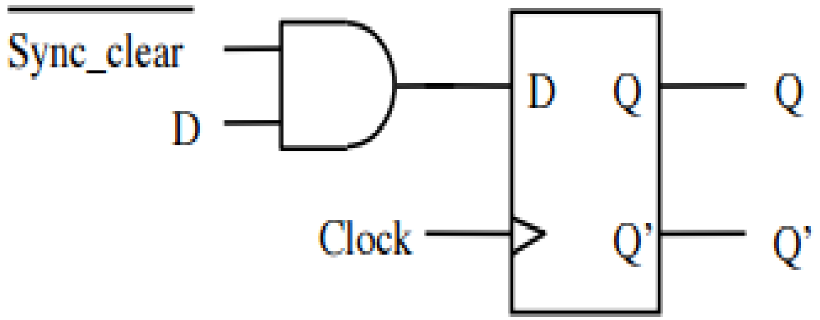
- Synchronous/asynchronous: depends on the clock signal or not
- Active low/high: controls when it’s low/high

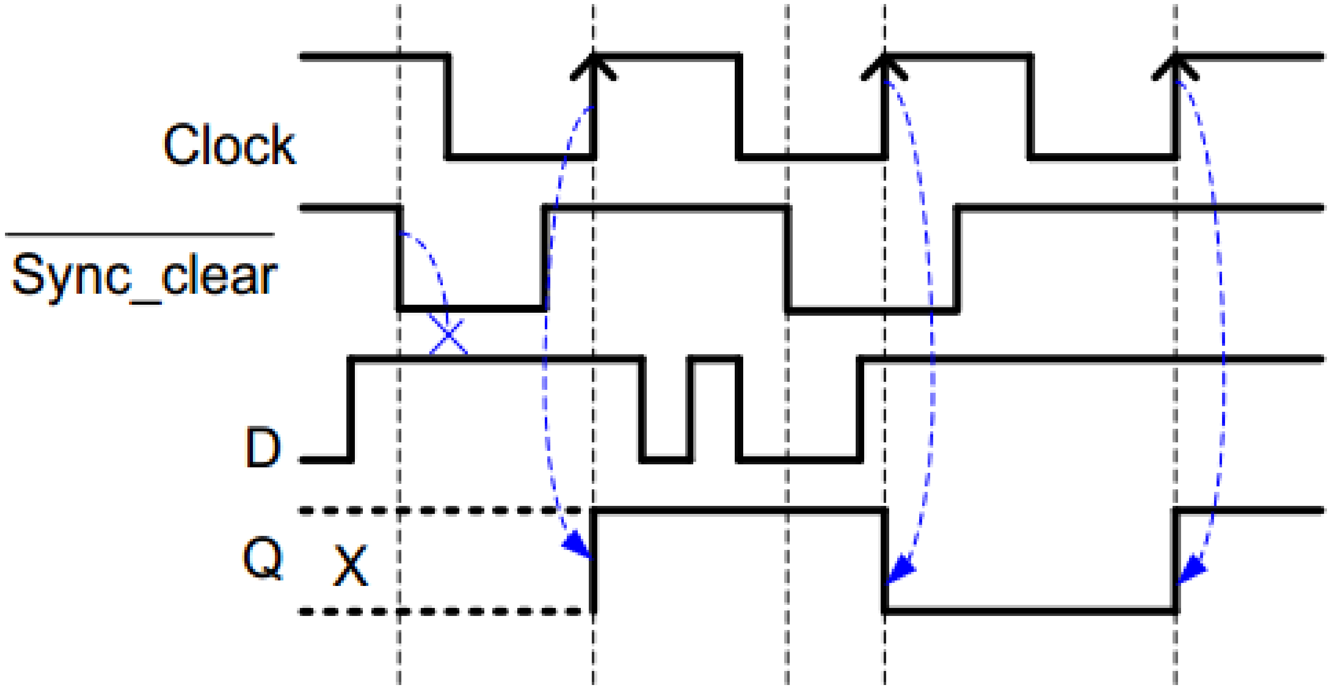
D flip flop with active low asynchronous Clear





D flip flop with active low synchronous Clear





Verilog

Verilog HDL

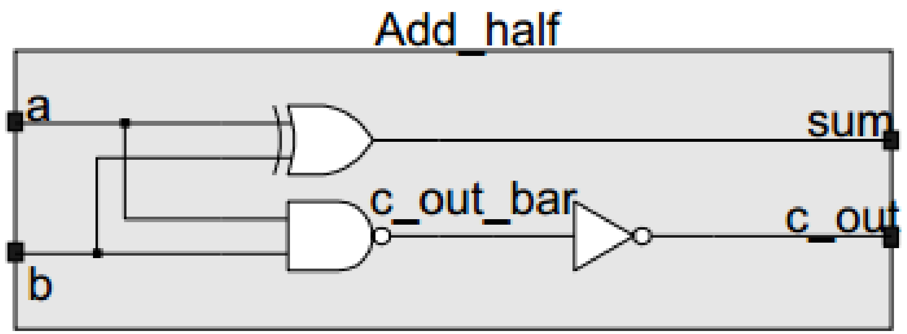
Syntax:

- module
- always
- if…else…
- switch
- assignment

Tips:

- [N-1:0] represents N bits
- Destination variables inside *always* must be *reg* type

Module



module name

module ports

```
module Add_half (sum, c_out, a, b);
  input     a, b;
  output    sum, c_out;

  wire      c_out_bar;

  xor       (sum, a, b);
  nand      (c_out_bar, a, b);
  not       (c_out, c_out_bar);
endmodule
```

declaration of port modes

declaration of internal signal

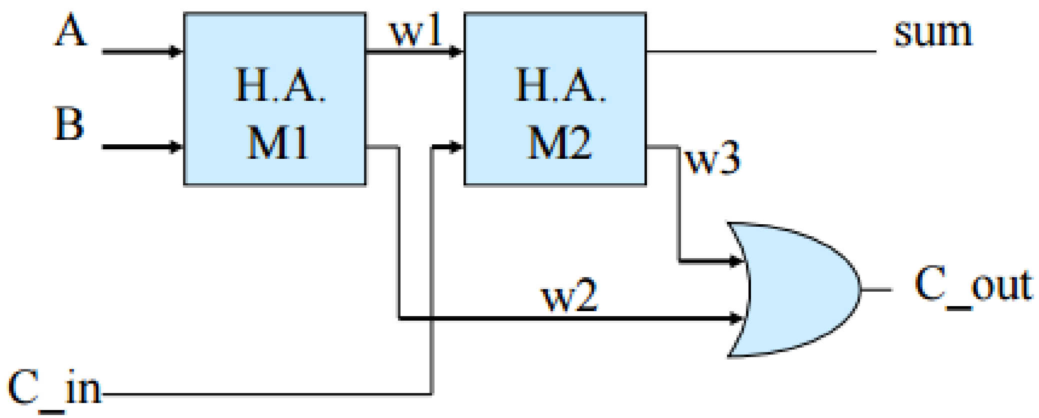
instantiation of pre-defined primitive gates

Same variable indicates connection

Module

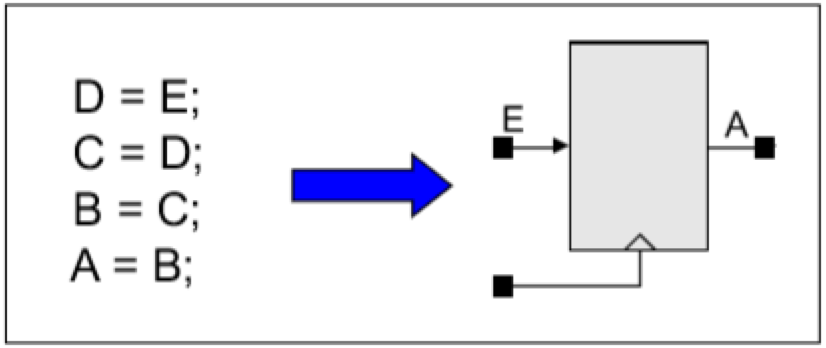
```
module Add_full (sum, c_out, a, b, c_in); // parent module
 input   a, b, c_in;
 output  c_out, sum;
 wire    w1, w2, w3;

  Add_half M1 (w1, w2, a, b);      // child module
  Add_half M2 (sum, w3, w1, c_in); // child module
  or (c_out, w2, w3);              // primitive instantiation
endmodule
```

Module instance name

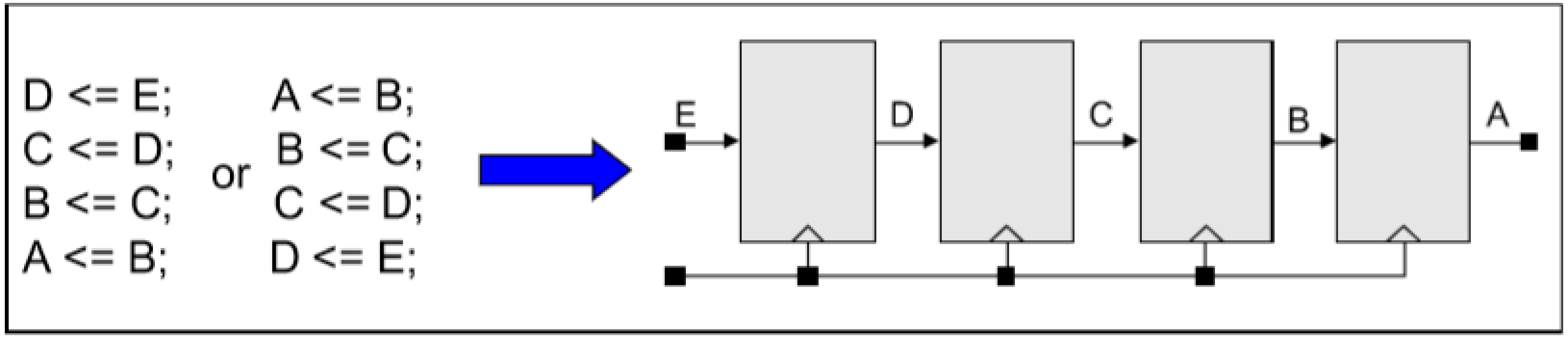


Assignment

- =
  Assign one statement by one statement



- <=
  Assign simultaneously



Testbench

Outputs of the module

Inputs of the module

Module to be tested

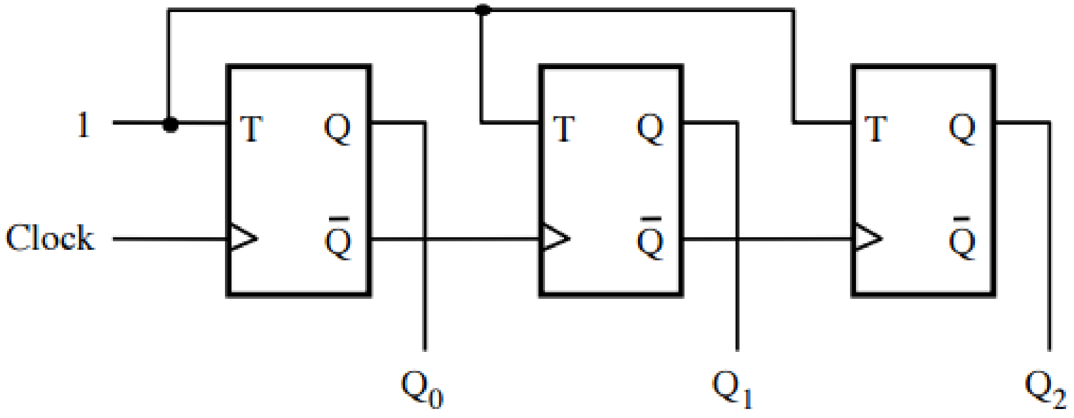
Changes of inputs

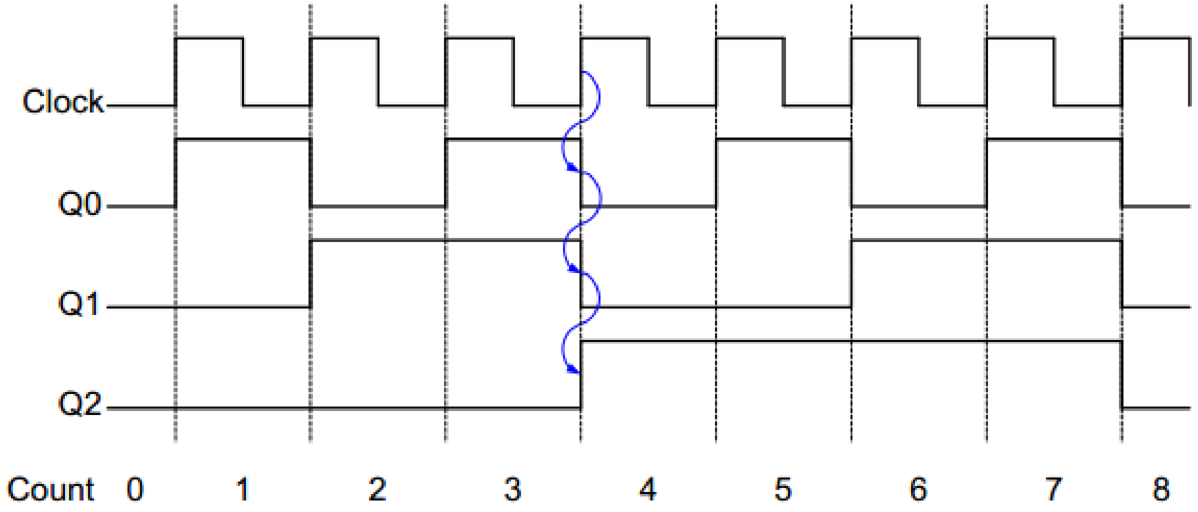
```
module Test_Banch;
  parameter half_period = 50;
  parameter counter_size = 4;

  wire [counter_size-1:0] Q;
  reg  [counter_size-1:0] Din;
  reg  clock, load, reset, CE;

  counter_N_bit #(counter_size) UUT (clock,reset,load,CE,Din,Q,CEO);

  initial begin
    #0    clock = 0; Din = 0; load = 0; CE = 1; reset = 1;
   #100  reset = 0;
   #200  Din = 8; load = 1;
   #100  load = 0;
   #300  CE = 0;
   #200  CE = 1;
  end

  always #half_period clock = ~clock;

  initial #2000 $stop;
endmodule
```

Counter

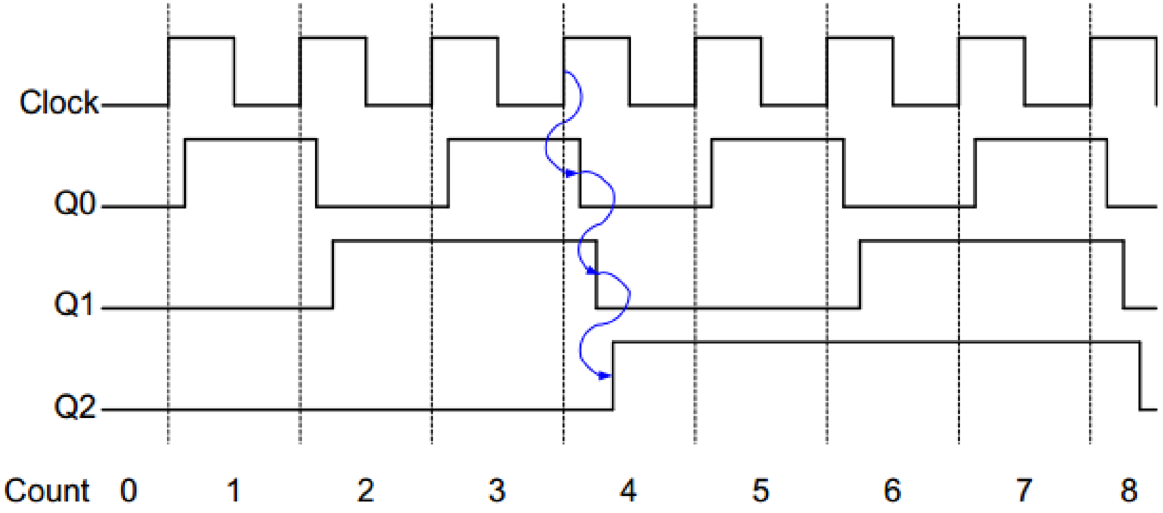
Asynchronous Counter

- Flip flops are cascaded together
- Delays → asynchronous output





Ideal

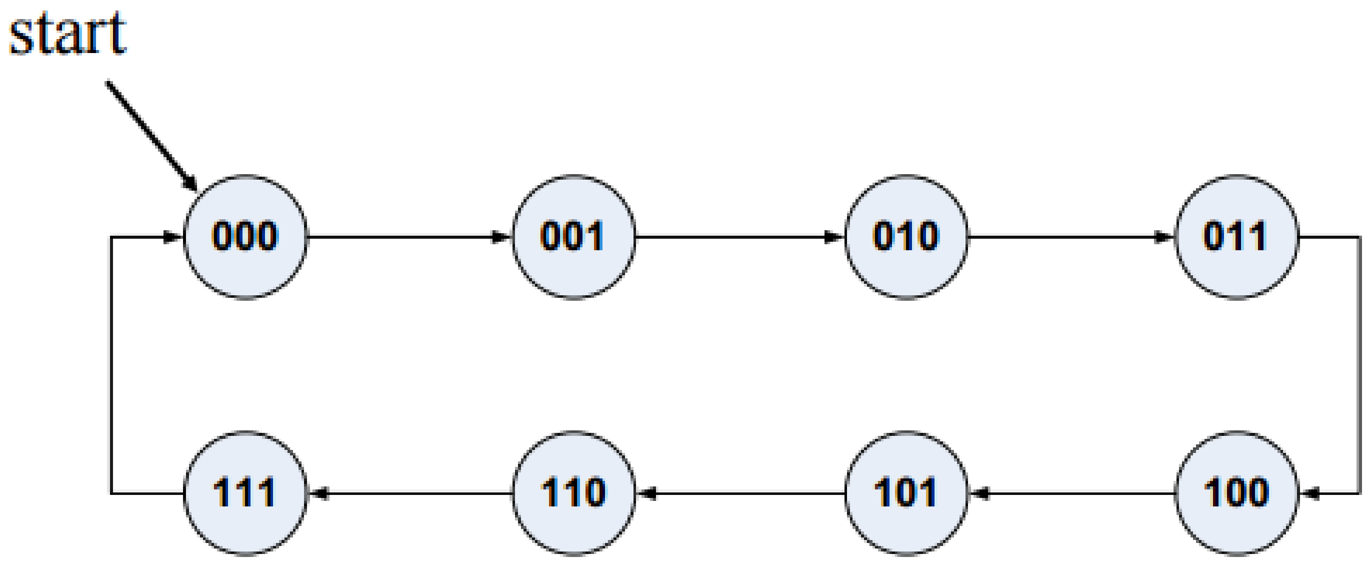


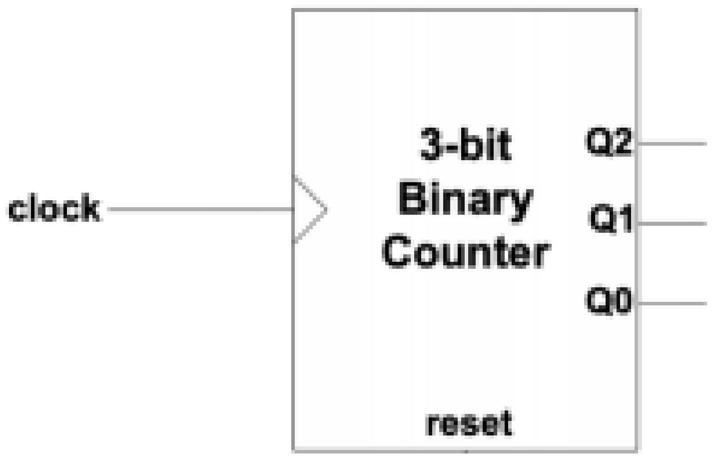
Practical

Synchronous Counter

- All the filp-flops are triggered by the same clock
- May be implemented by different type of filp flops

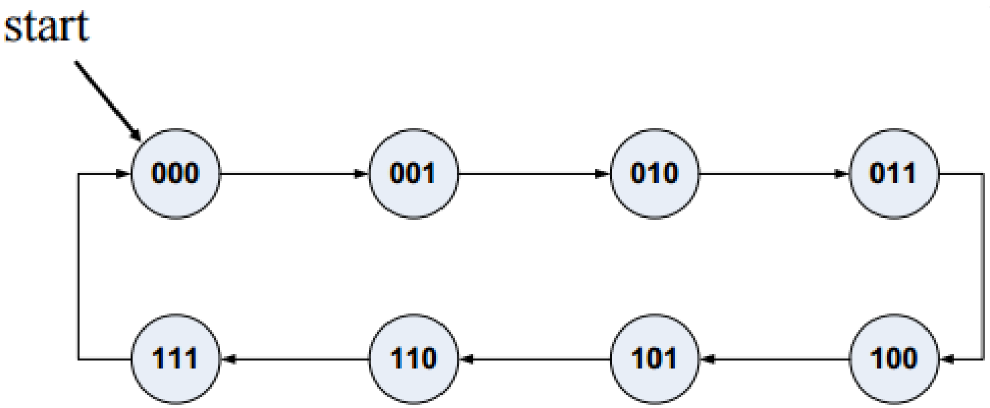
Synchronous 3-bit Binary Counter

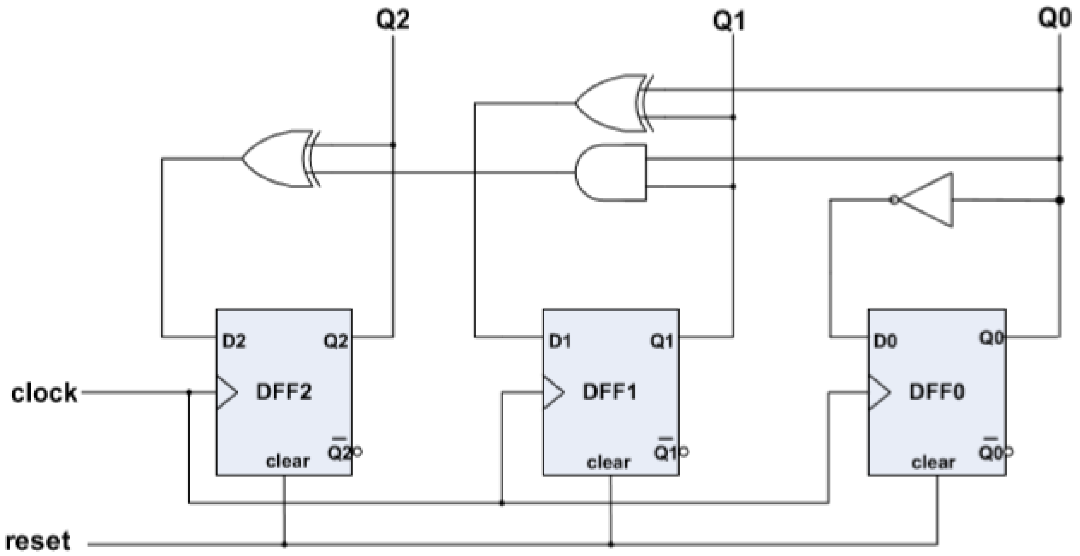




How to design a synchronous counter given a sequence of numbers?

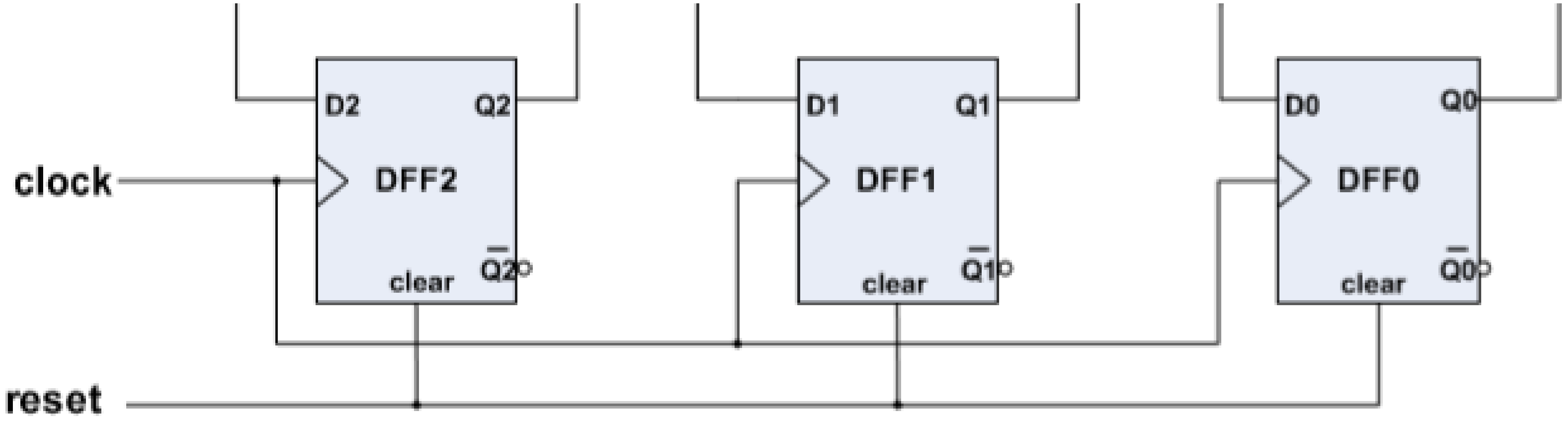
- Combine combinational circuit and sequential circuit
- Steps:
  1. Decide what kinds of flip flops to use and how many
  2. Draw truth table
  3. Derive logic equations
  4. Draw combinational circuit



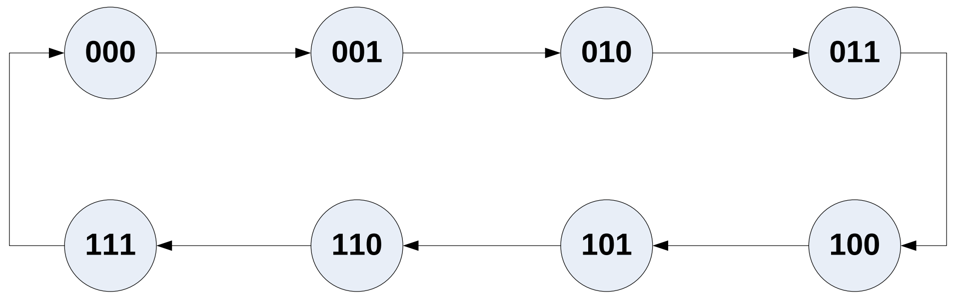


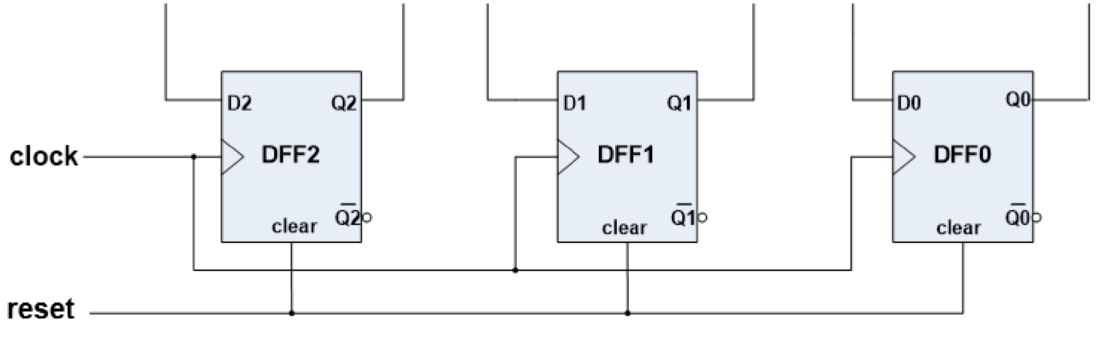
Step 1: FF

- 3-bit counter → 3 flip flops
- Here we choose D filp flops



Step 2: Draw Truth Table





Q^+ = D upon active edge

Present State			Next State			D flip flop input		
Q2	Q1	Q0	$Q2^+$	$Q1^+$	$Q0^+$	D2	D1	D0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Step 3: Derive Logic Equations

Present State			Next State			D flip flop input		
Q2	Q1	Q0	$Q2^+$	$Q1^+$	$Q0^+$	D2	D1	D0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

$$
\begin{aligned}
D2 &= Q2Q1' + Q2Q0' + Q2'Q1Q0 \\
&= Q2(Q1' + Q0') + Q2'Q1Q0 \\
&= Q2(Q1Q0)' + Q2'(Q1Q0) \\
&= Q2 \oplus (Q1Q0)
\end{aligned}
$$

$$
\begin{aligned}
D1 &= Q1'Q0 + Q1Q0' \\
&= Q1 \oplus Q0
\end{aligned}
$$

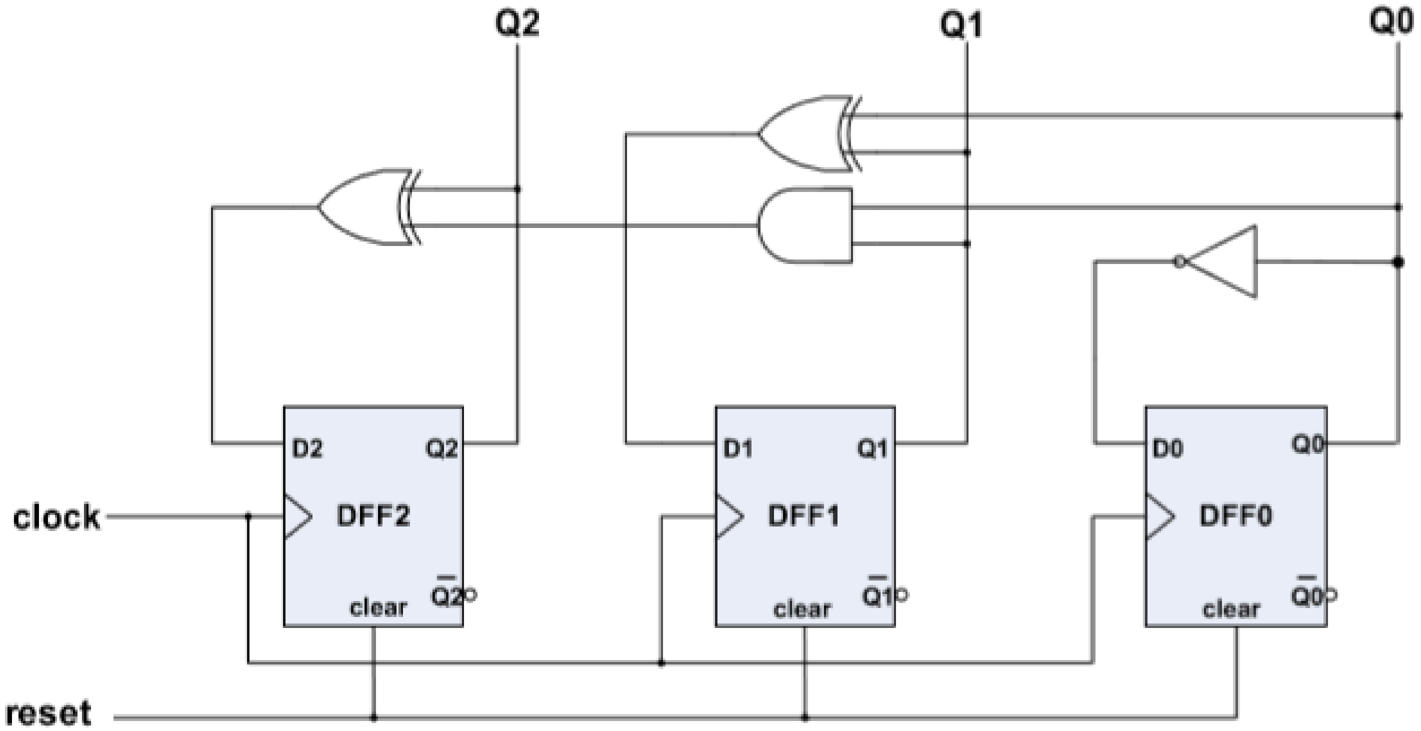
$$
D0 = Q0'
$$

Step 3: Draw Combinational Circuit

$$
\begin{aligned}
D2 &= Q2Q1'+Q2Q0'+Q2'Q1Q0 \\
&= Q2(Q1'+Q0')+Q2'Q1Q0 \\
&= Q2(Q1Q0)'+Q2'(Q1Q0) \\
&= Q2 \oplus (Q1Q0)
\end{aligned}
$$

$$
\begin{aligned}
D1 &= Q1'Q0+Q1Q0' \\
&= Q1 \oplus Q0
\end{aligned}
$$

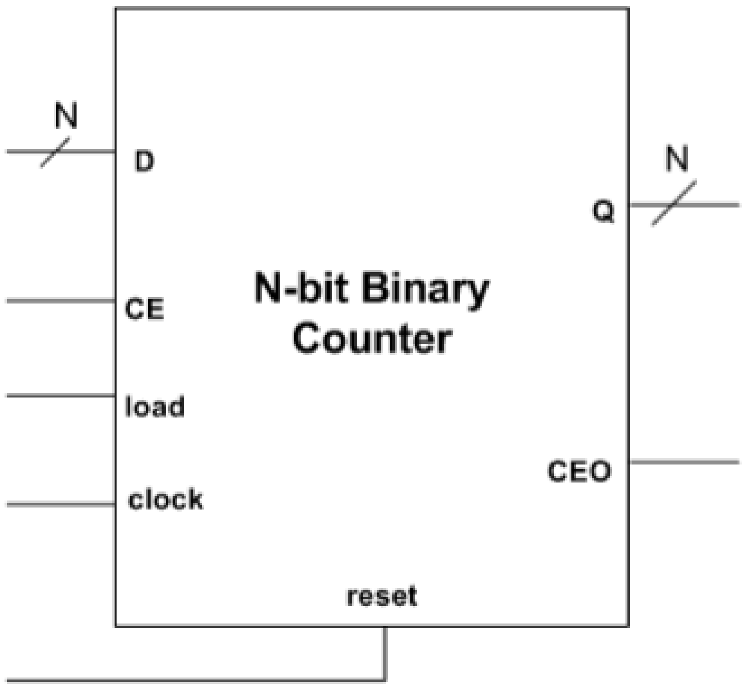
$$D0 = Q0'$$



Control Signals

Priority of External Control Signals:

reset	load	CE	Action on the rising clock edge
1	X	X	Clear (Qn <= 0)
0	1	X	Load (Qn <= Dn)
0	0	1	Count
0	0	0	Hold (No Change)



Count Enable Output:

$$CEO = CE \cdot Q_{N-1} \cdot Q_{N-2} \cdot \ldots \cdot Q_0$$

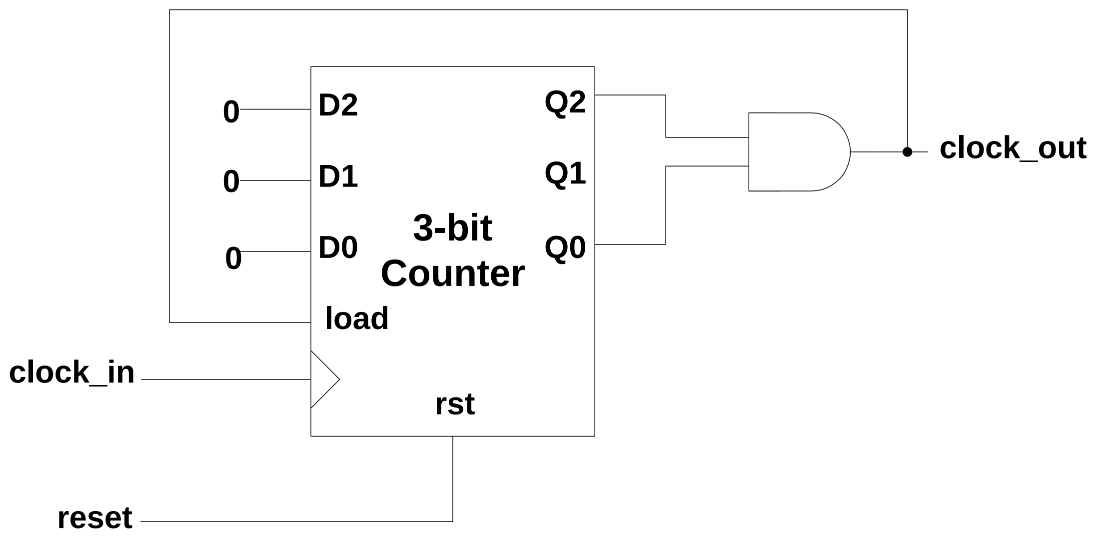
Clock Divider

Divide by N:

- $2^{n-1} < N < 2^n$, where n is the size of counter
- N-1 is the upper bound of the counter's sequence
- AND the bits that are 1 in the upper bound to the load

Divide by 6:

- $2^2 < 6 < 2^3 \rightarrow$ 3-bit counter
- Countering sequence: 000→001→010→011→100→101→000
- Upper bound: 101 → load = Q0 AND A2



GOOD LUCK