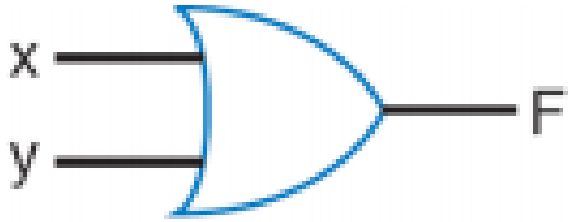
VE270 RC Week 6
Latch, FF & Verilog

VE270 TA Group

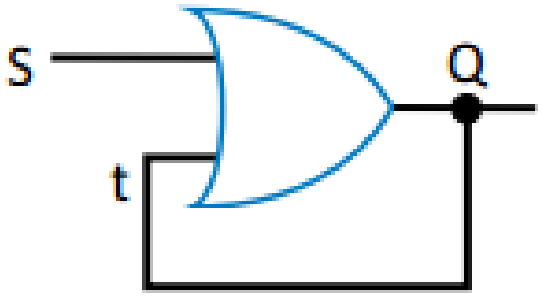
2019.10.14

Combinational Circuit vs. Sequential Circuit

- Combinational circuit:
  - Output depends only on the present combination of inputs.



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



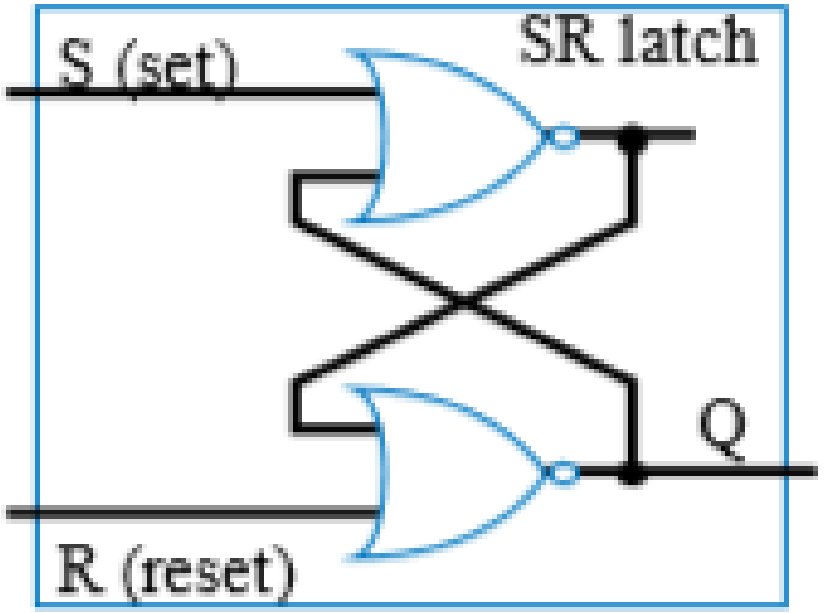
Latch & Flip-flop

- 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

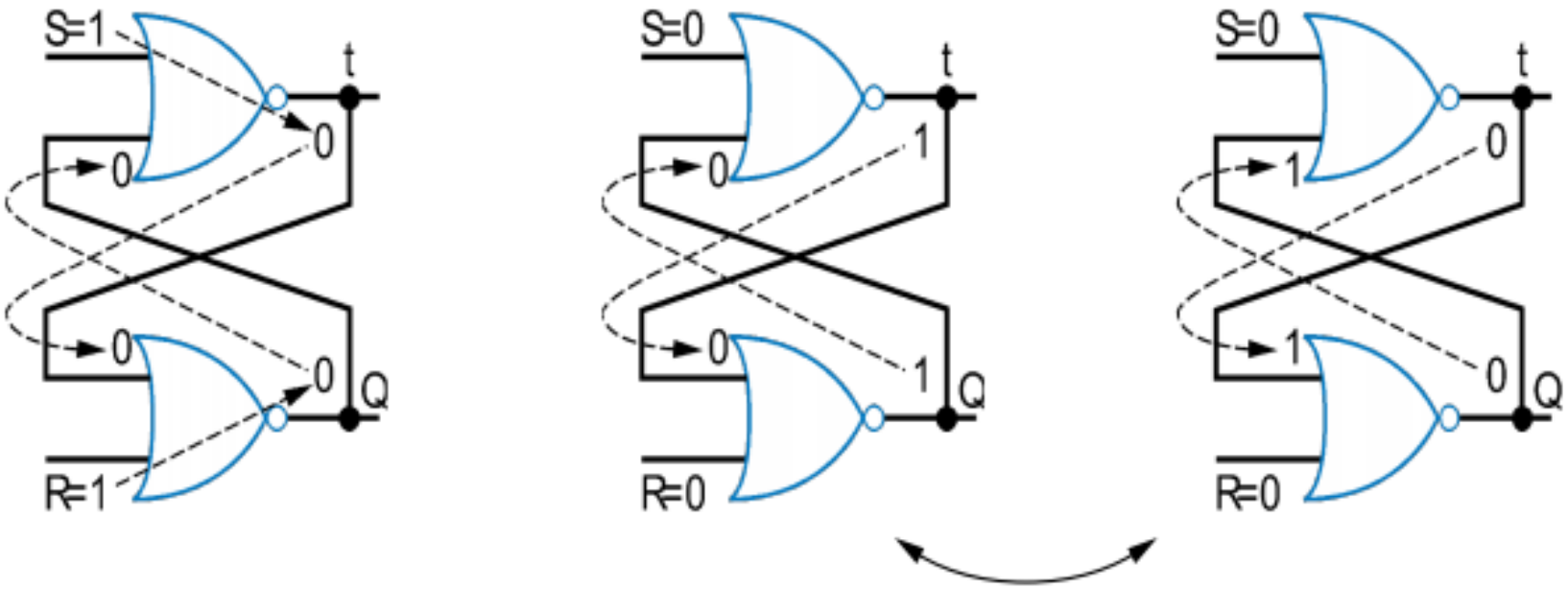
S(t)	R(t)	Q(t)	Q(t+Δ) → Q⁺	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	



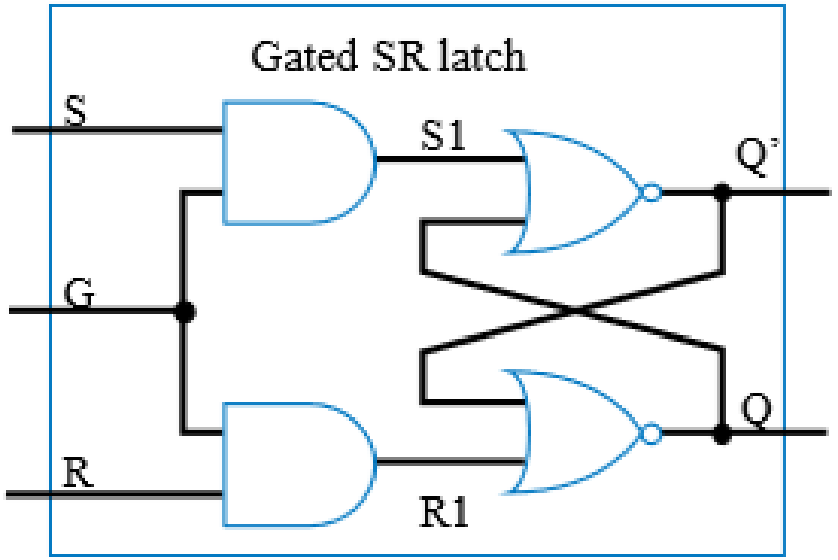
$Q^+ = S + R'Q$

SR Latch

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



Gated SR Latch

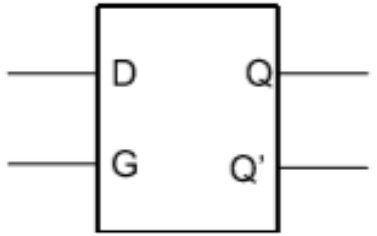


Characteristic Table

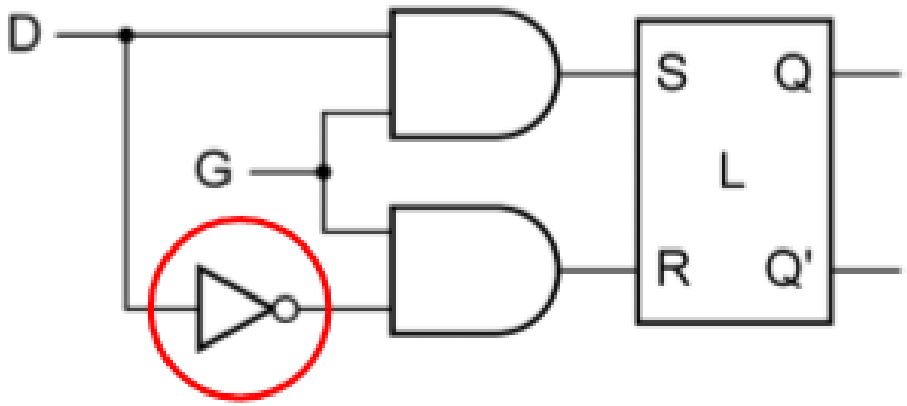
G	S	R	Q^+
0	x	x	Q; Latch locked
1	0	0	Q; Hold state
1	0	1	0; Reset state
1	1	0	1; Set state
1	1	1	not allowed

Gated D Latch

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



D latch symbol

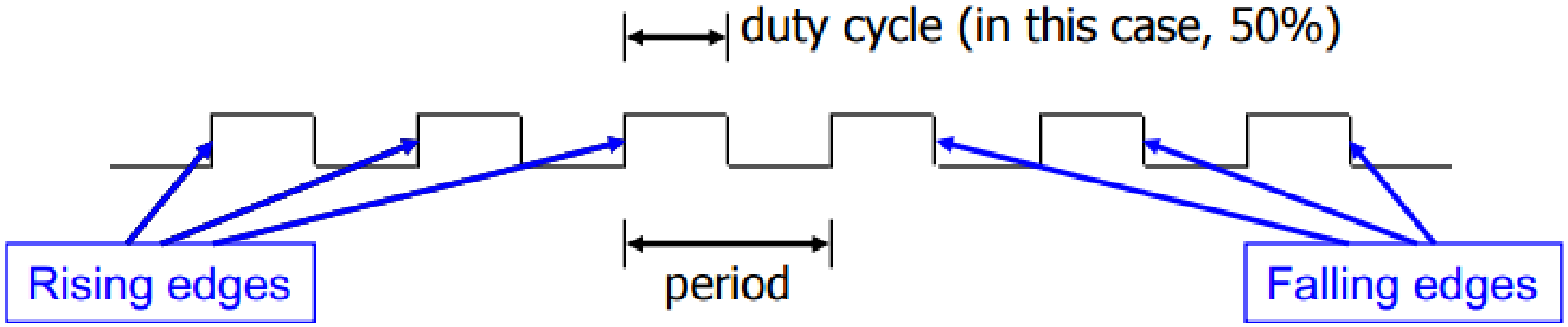


Characteristic Table

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

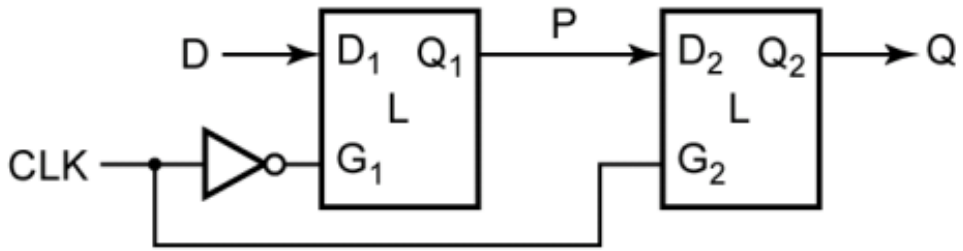
Clock Signal

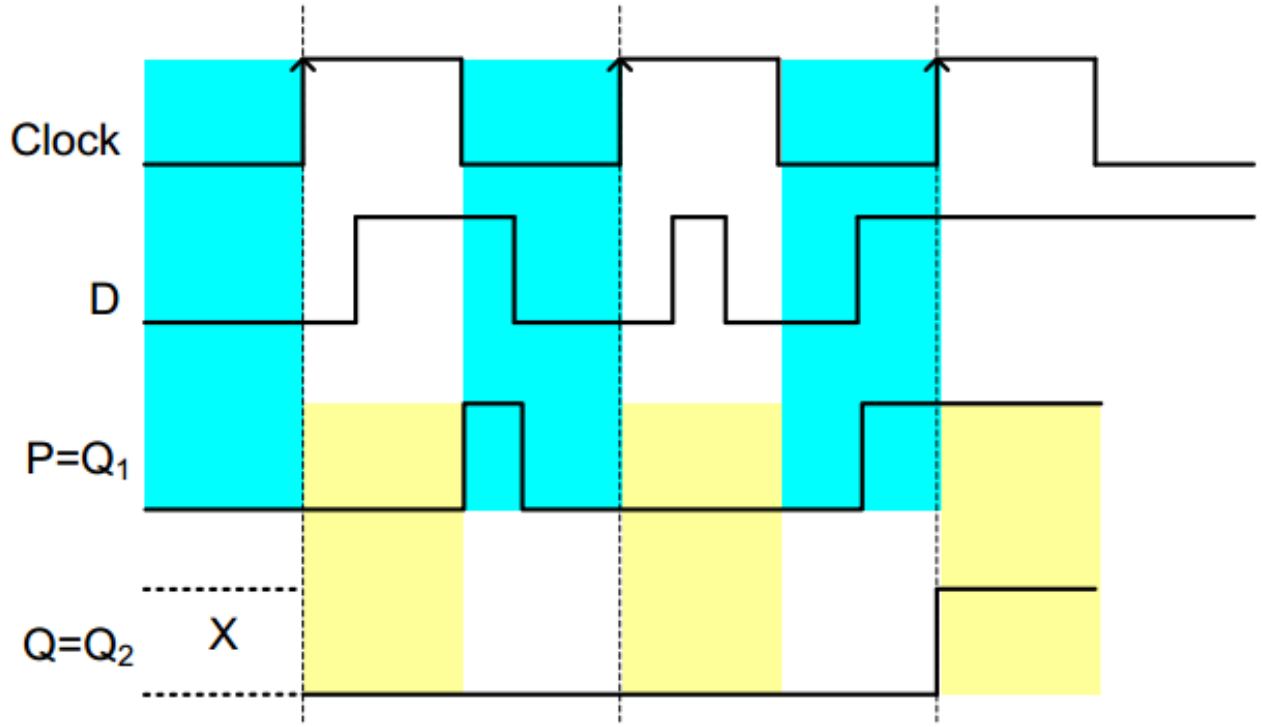
- Periodic pulse train
- Clock period: time interval between pulses



Rising-Edge Triggered D Filp Flop

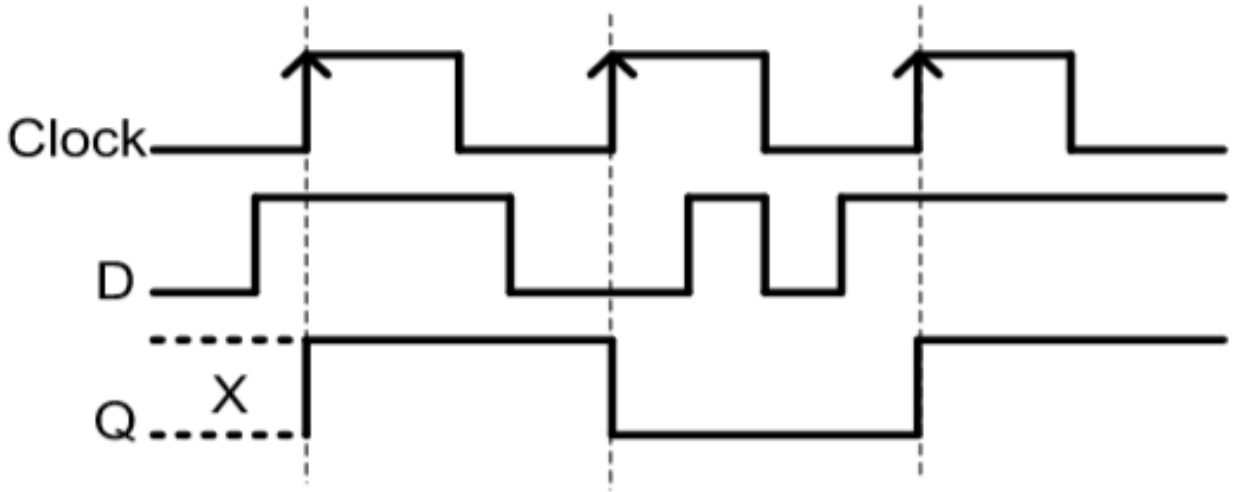
- It is made up of two gated D latch

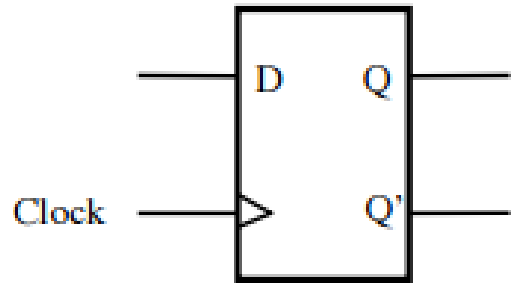




Rising-Edge Triggered D Filp Flop

- Edge sensitive

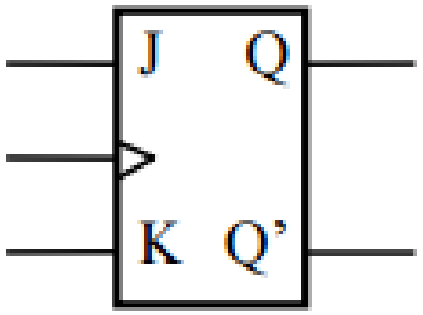




clock	D	Q^+
↑	0	0
↑	1	1
0	X	Q
1	X	Q

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

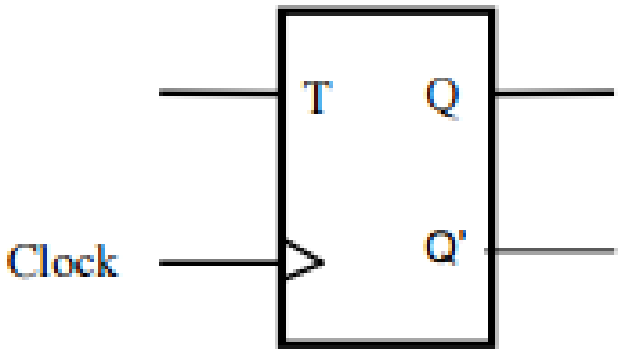
Rising-Edge Triggered J-K Filp Flop



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

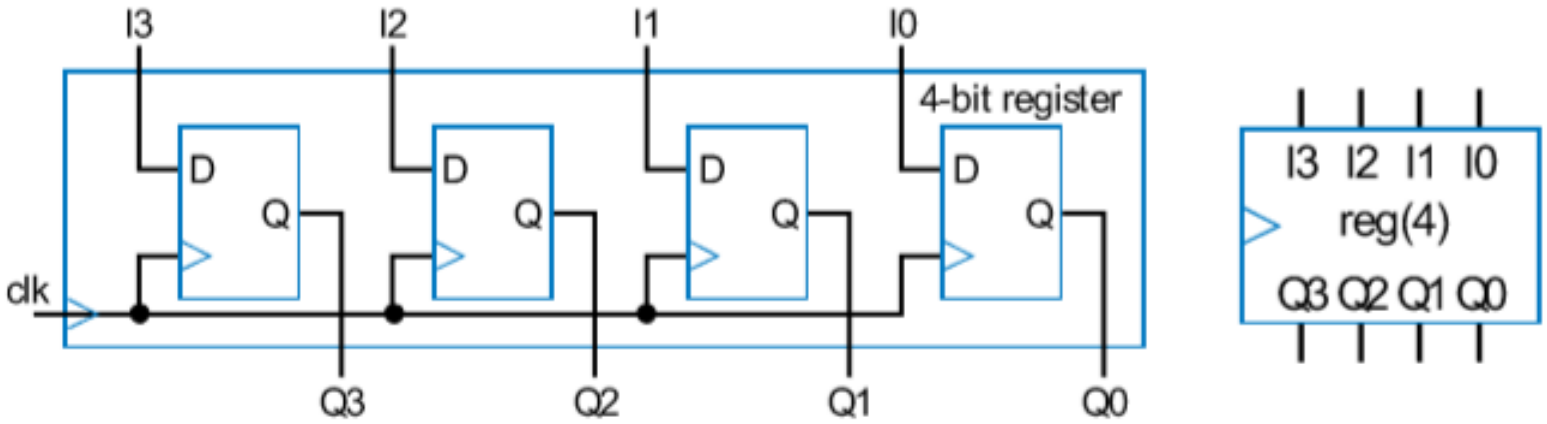


clock	T	Q^+
↑	0	Q
↑	1	Q'

Characteristic equation:
$Q^+ = T'Q + TQ' = T \oplus Q$

Register

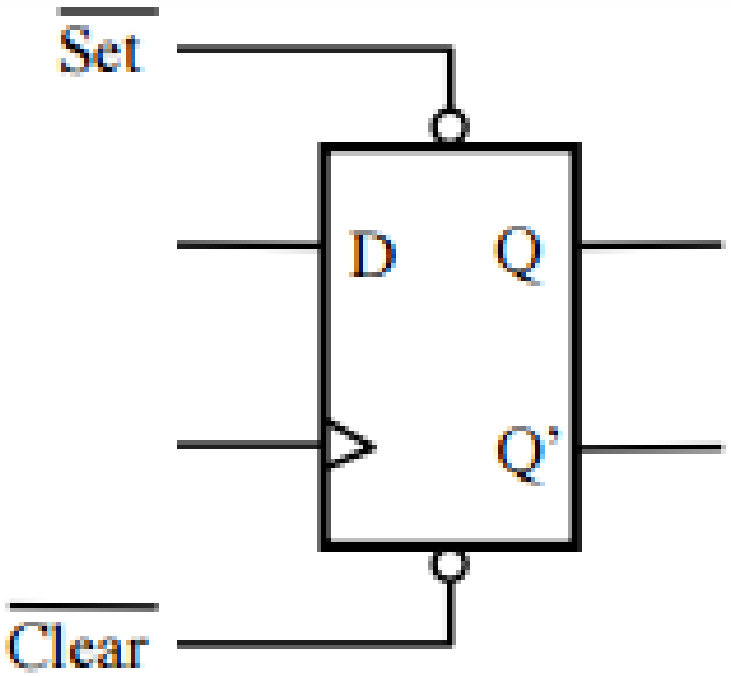
- Multiple filp-flops sharing clock signal
- Store muliple bits
- Load values simultaneously at rising edge

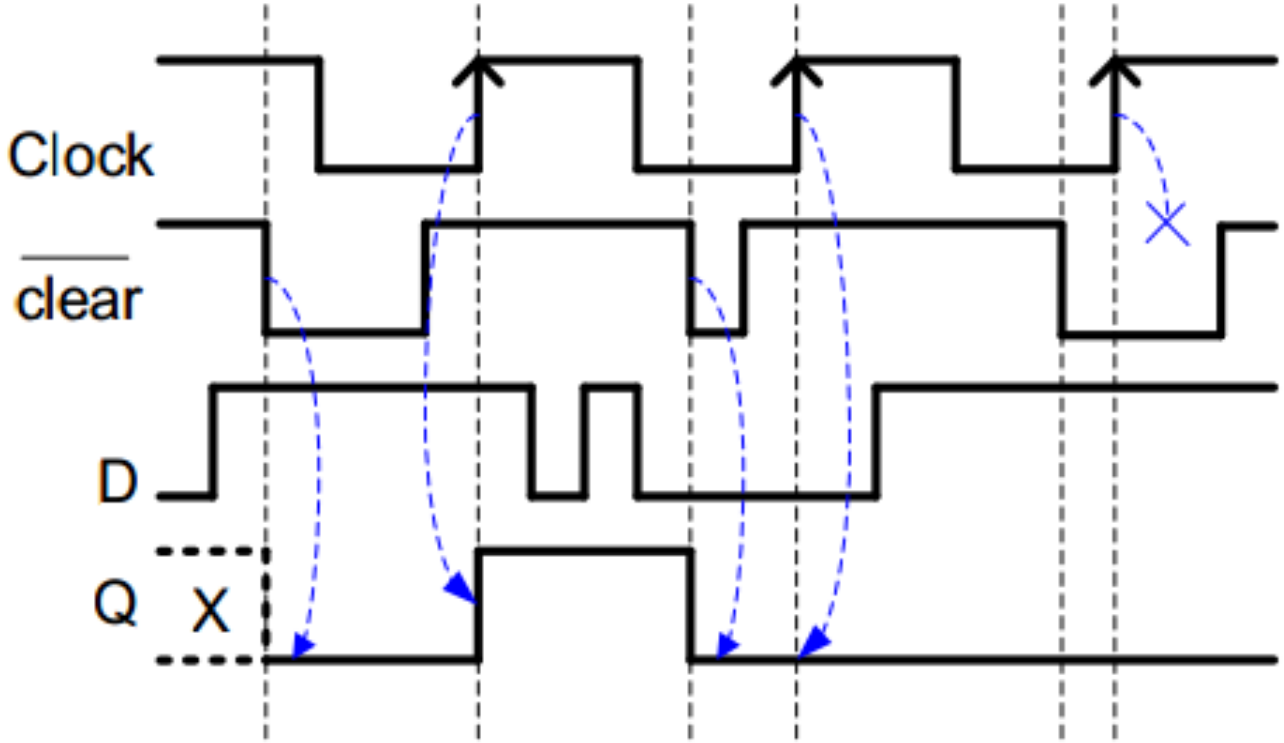


Control Inputs for Flip Flops

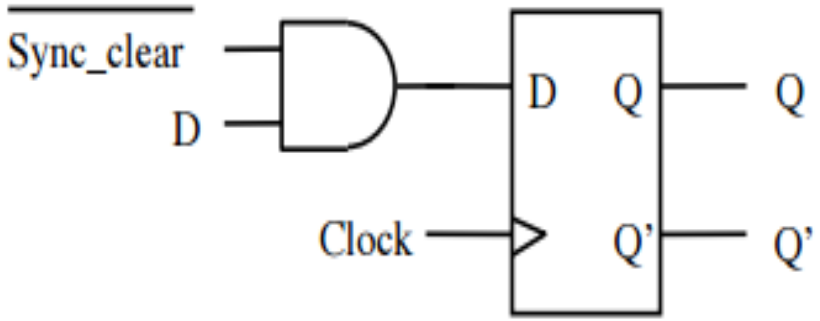
- Synchronous/asynchronous: depends on the clock signal or not
- Active low/high: controls when it’s low/high

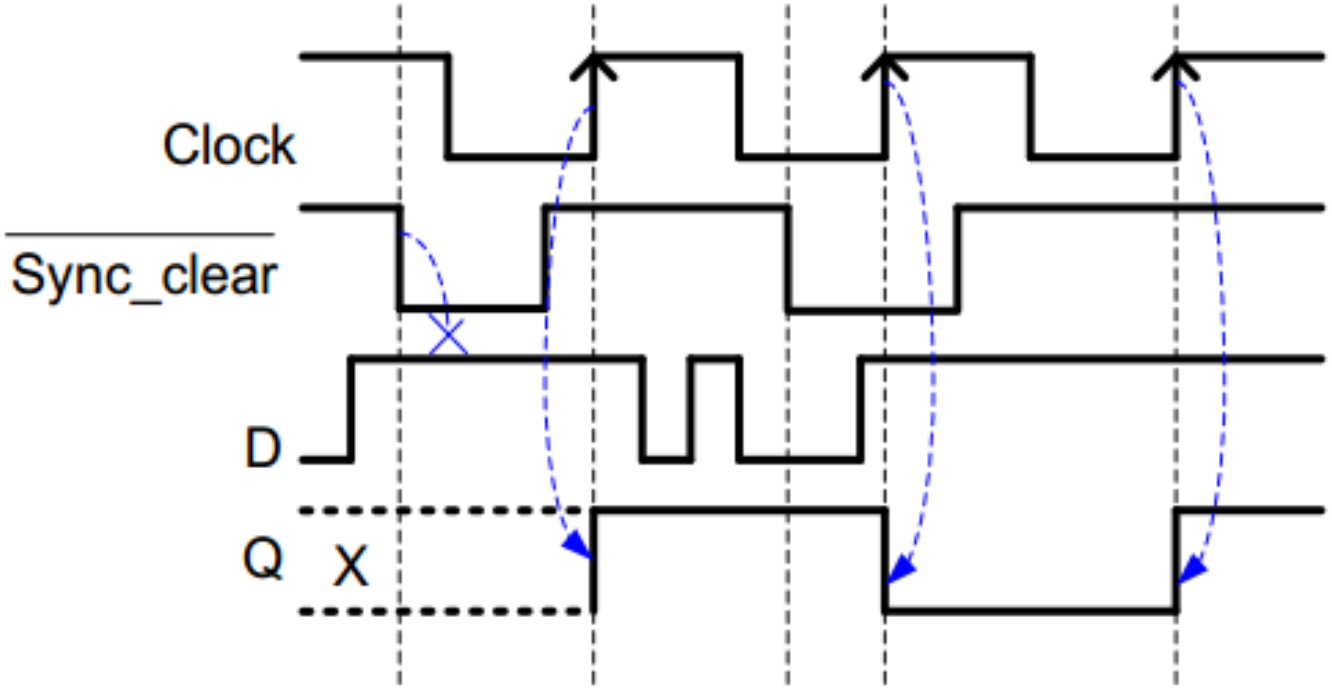
D flip flop with active low asynchronous Clear





D flip flop with active low synchronous Clear

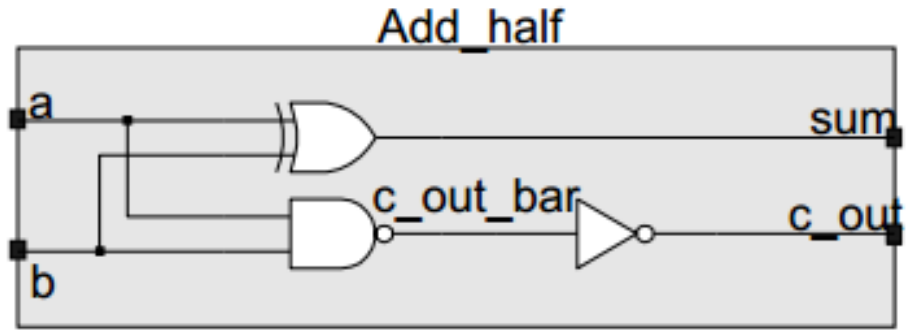




Verilog HDL

- Lower-level than C
- Describes hardwares

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

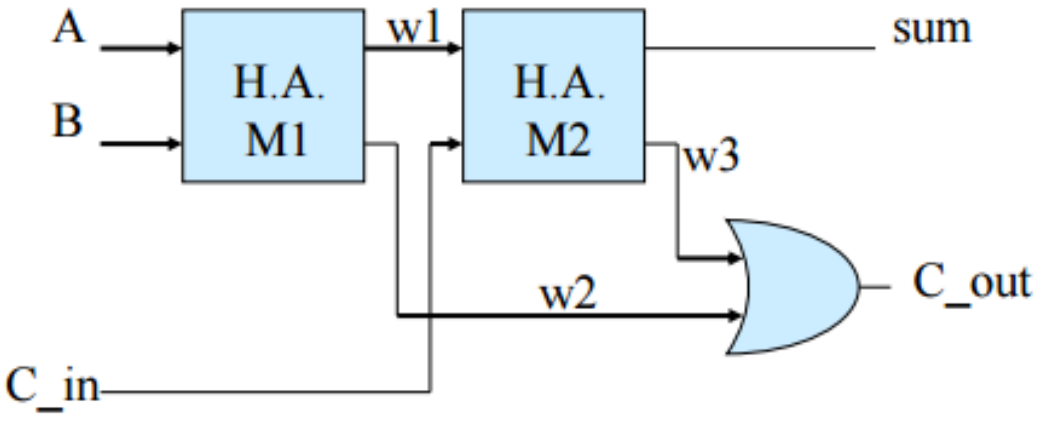
Same variable indicates connection

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



Synchronous Control Input

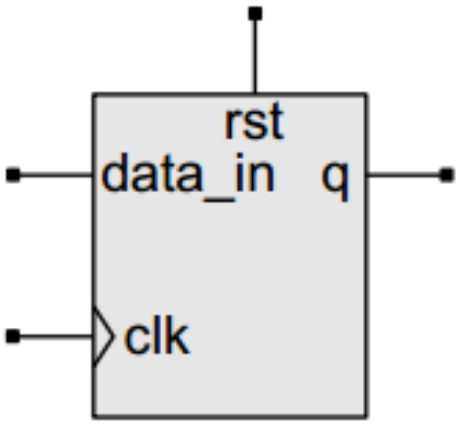
If you want to change the value of "q" in the always block, you should state "**reg q;**" beforehand.

```
module D_ff (q, data_in, clk, syn_rst);
 input data_in, clk, syn_rst;
 output q;

 reg q;

 always @ (posedge clk)
 begin
   if (syn_rst == 1) q <= 0;
   else q <= data_in;
 end
endmodule
```

Synchronous reset



Asynchronous Control Input

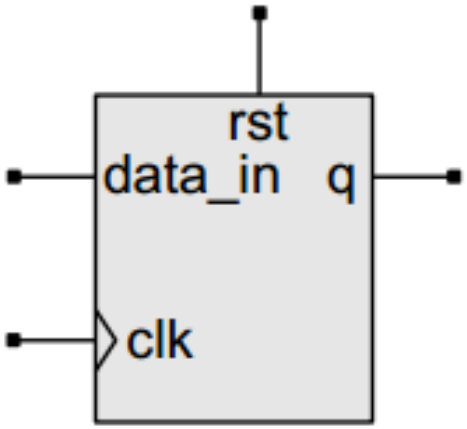
Pay attention to what you put in the condition of always

```
module D_ff (q, data_in, clk, asyn_rst);
 input data_in, clk, asyn_rst;
 output q;

 reg q;

 always @ (posedge clk or posedge asyn_rst)
 begin
   if (asyn_rst == 1) q <= 0;
   else q <= data_in;
 end
endmodule
```

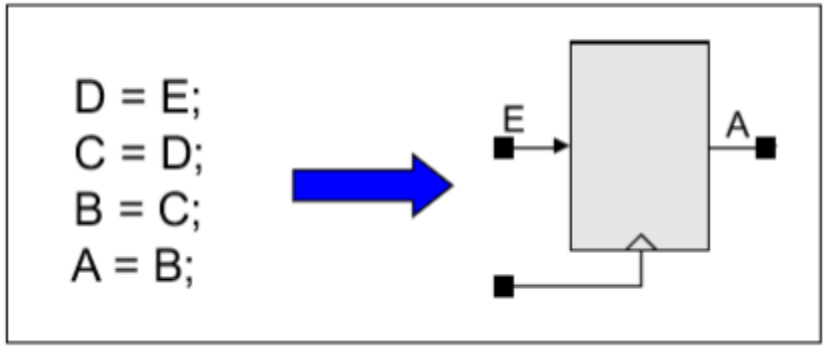
Asynchronous active high reset



Assignment

- =

  Assign one statement by one statement



- <=

  Assign simultaneously