Latches and Flip-flops

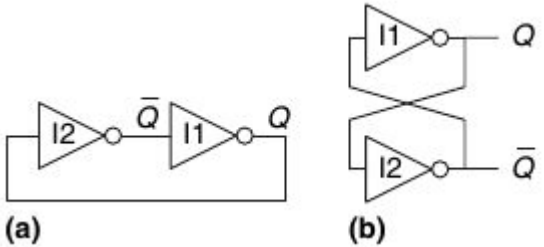
Instructor: Dr. Vinicius Prado da Fonseca (vpradodafons@online.mun.ca)

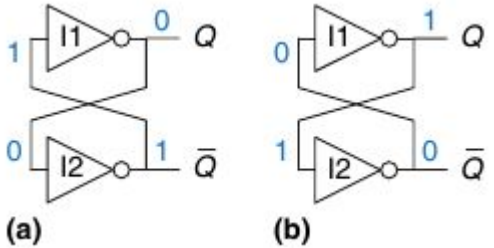
Combinational and Sequential Circuits

- Combinational logic
  - Output depends only on the current inputs
  - Truth table or boolean equation
- Sequential logic
  - Output depends on current and past inputs
  - Concept of the state
  - Current state analysis of the system to decide next state
  - Assembled to provide temporary memory
  - Discipline:
    - Block of combinational circuit;
    - with a bank of sequential circuit (Flip Flops) to keep the state.
    - Next lecture

Bi-stable Memory

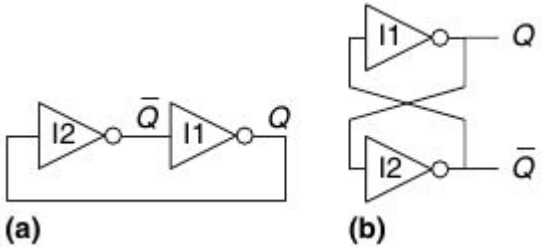
- Feedback is used to implement memory elements
  - Using two not gates to keep the same output/state
- Consider the following two NOT gates
- Output is feedback to the input
  - cross-coupled
- Same output ultil is has power

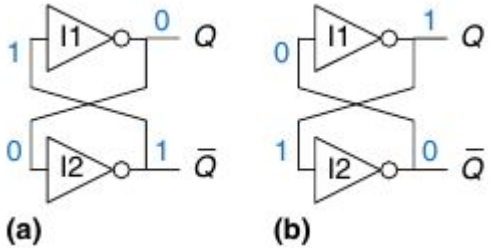




Bi-stable Memory

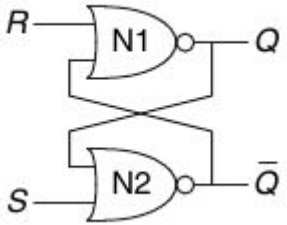
- Has no inputs, no control over the behaviour
- When power is first applied
  - The initial state is unknown
  - Usually unpredictable
  - It may differ each time the circuit is turned on





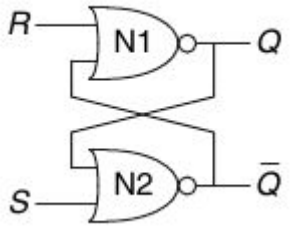
SR Latch

- The RS latches uses cross-coupled NOR gates
- NOR gate produces a FALSE output when either input is TRUE
- r stands for reset, and the s stands for set
- Two stable configurations



SR Latch

- The RS latches uses cross-coupled NOR gates
- NOR gate produces a FALSE output when either input is TRUE
- r stands for reset, and the s stands for set
- Two stable configurations




```
1 module sr_latch(output logic q, nq,
2                 input logic s, r);
3     nor (q, r, nq);
4     nor (nq, s, q);
5 endmodule
```

S	R	Q	$\bar{Q}$
0	0	Latch	
0	1	0	1
1	0	1	0
1	1	Metastable	

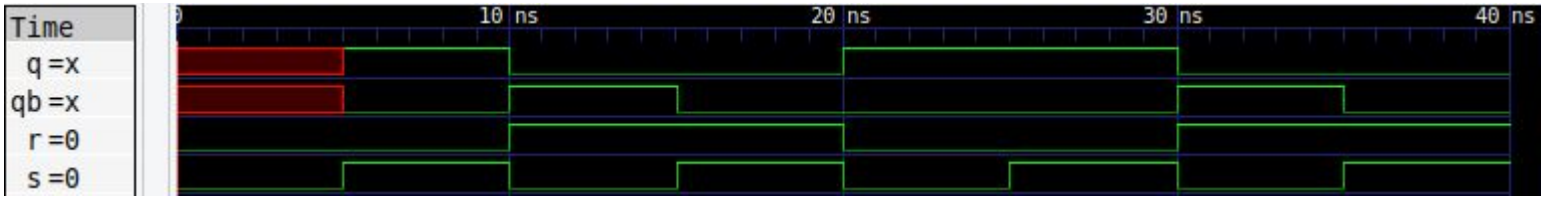
SR Latch

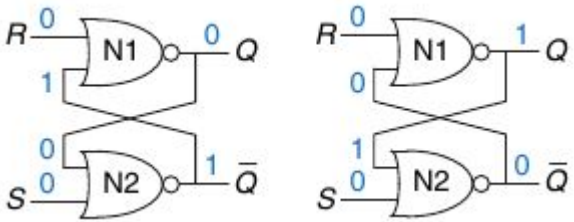
```
module sr_latch(output logic q, n_q,
input logic s, r);

    nor nor_r(q, r, n_q);

    nor nor_s(n_q, s, q);

endmodule
```

```
`timescale 1ns/1ns

module main;
  logic s, r;
        logic q, n_q;

        sr_latch dut(q, n_q, s, r);
  initial begin
   #10 $display("t=%04d sr q n_q", $time);
        s = 0;
        r = 0;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
        s = 0;
        r = 1;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
         s = 1;
        r = 0;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
        s = 0;
        r = 1;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
        s = 0;
        r = 0;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
        s = 0;
        r = 0;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
        s = 1;
        r = 1;
   #10 $display("t=%04d %b%b %b %b", $time, s, r, q, n_q);
  end
endmodule
```

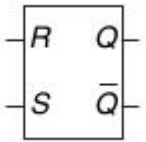
SR Latch

- sr_latch.vcd
- “Illegal” case



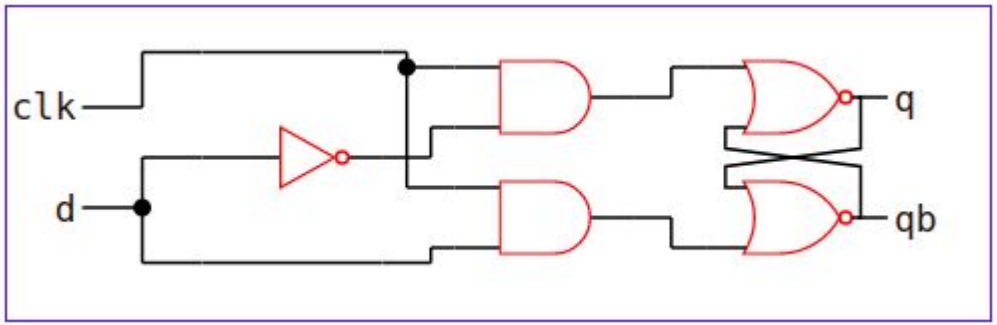


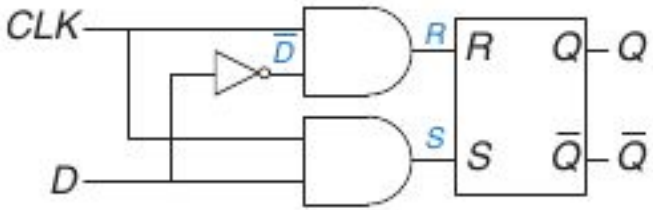
Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0



D-latch

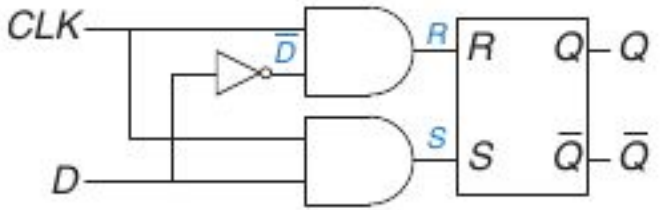
- The RS-latch can be modified to create a D-latch.
  - What should change
  - When it should change
- The D-latch (not gate) prevents illegal state.
- Sets in a specific time (clock)
  - Important for synchronous circuits





D-latch

- If CLK = 0, both S and R are FALSE
  - regardless of the value of D.
  - Keep the last value
- If CLK = 1,
  - one AND gate will produce TRUE
  - the other FALSE
  - depending on the value of D
- Q is always the complement of Q
- The D latch avoids the “illegal” case
  - R=1, S=1
- Challenge:
  - If D changes more than once during the clock=1



CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D-latch using gates

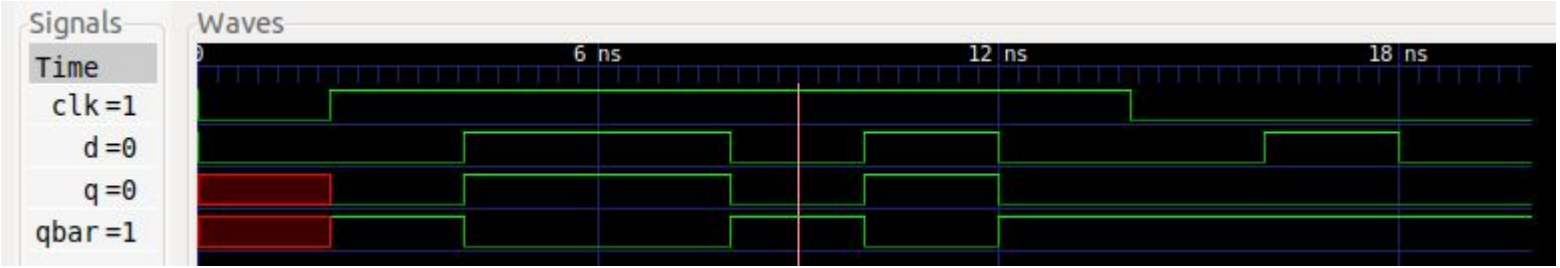
```
1  module d_latch_gate(output reg q, qbar,
2                      input logic d, clk);
3
4      wire i_nand_q, i_nand_qbar, notd;
5
6      not(notd, d);
7      and nandqb(i_nand_qbar, clk, d);
8      and nandq(i_nand_q, clk, notd);
9
10     nor (q, i_nand_q, qbar);
11     nor (qbar, i_nand_qbar, qbar);
12
13 endmodule
```

D-latch using behavioral verilog

```
1 module d_latch_behavior(output reg q, qbar,
2                         input logic d, clk);
3     always @(d, clk)
4         if (clk) begin
5             q <= d;
6             qbar <= ~d;
7         end
8 endmodule
```

D-latch

- tb_d_latch.vcd
- During one clock cycle you can have multiple values of d
  - multiple values of q and qbar

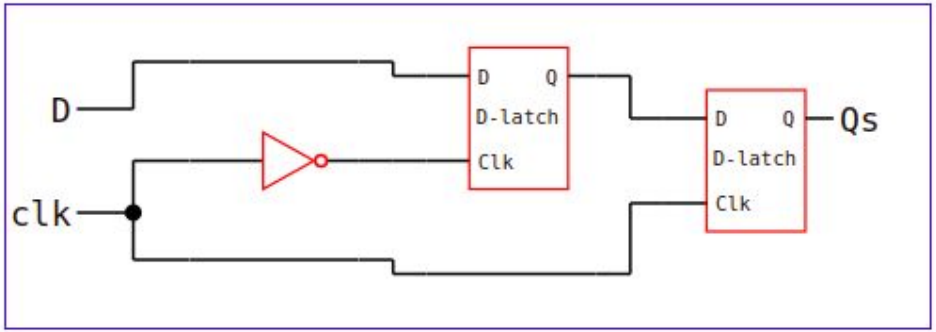


Flip-flops (D-FF)

- Synchronous circuits are designed to capture the input and change only with the clock changes.
- Rising edge or falling edge are used to sync the circuit.
- D-latches enable to changes during the clock cycle.
- Flip-flops avoid it

Flip-flops

- D-FF
- Back to back d-latches
- master/slave
- D flip-flop copies D to Q
- Rising edge of the clock
- Remembers its state at all other times



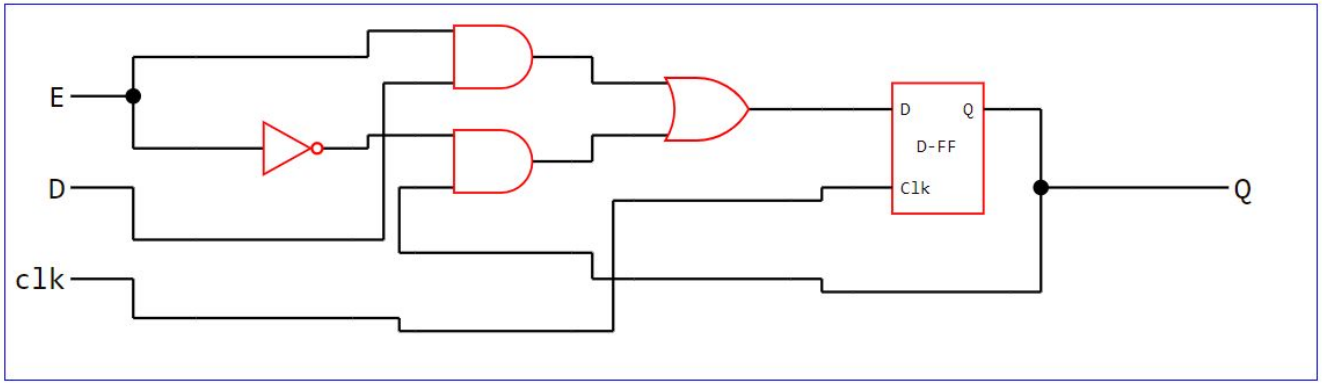
Flip-flops

- dff.vcd
- Only changes during the positive edge

```
1 module dff( output logic q, input logic d, clk);
2     always_ff @(posedge clk) begin
3         q <= #delay d;
4     end
5 endmodule
```

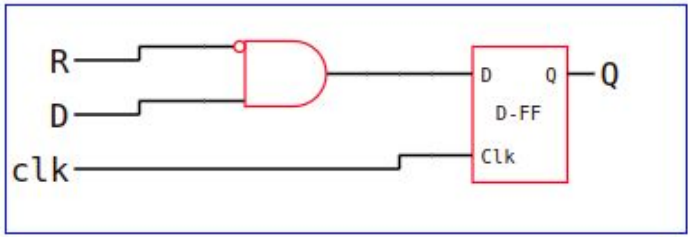
Enable Flip-flop

- Extra ports are used to add functionality.
- It is important to control when the memory is invalid the system can disable the FF.
- Example in the notes
- Implementation in the next lab



Enable and Resettable Flip-flop

- Extra ports are used to add functionality.
- It is important to control when the memory is invalid the system can reset or disable the FF.
- Example in the notes
- Implementation in the next lab



JK Flip Flop

- Not seen in this course
  - Versatile
  - Universal
  - Also has two set/reset inputs J and K
  - Inputs = 11, flips the value
  - Among other functionalities
- We will use the DFF for simplicity

Midterm material

The midterm cover topics seen in Units 1-3, from Lectures/slides 1-12 Latches and Flip-flops (included).

Questions?

- Next: 13 - Synchronous Sequential Circuits