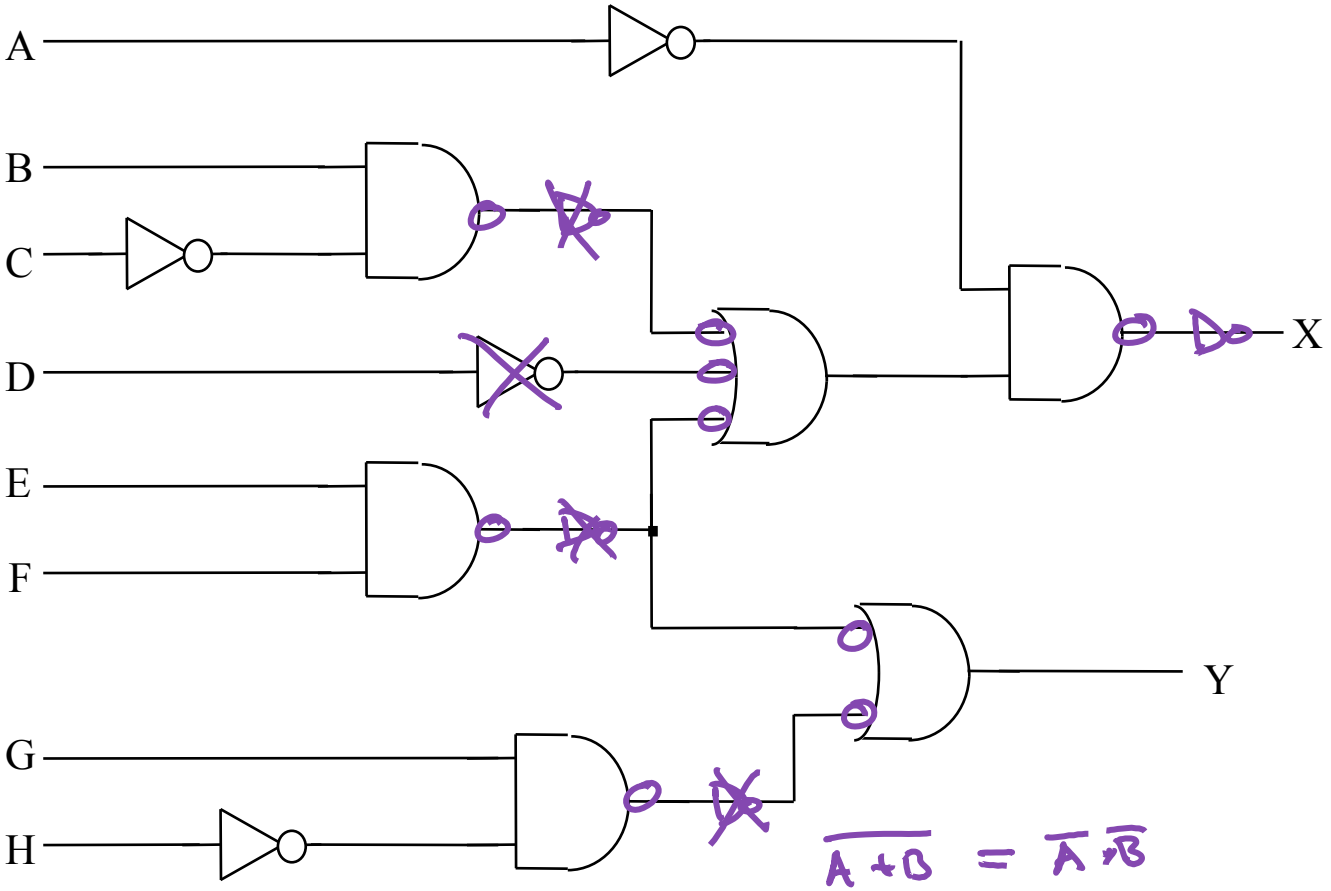
Review Problem: AND/OR to NAND

- Convert the following circuit to NAND/NOR form

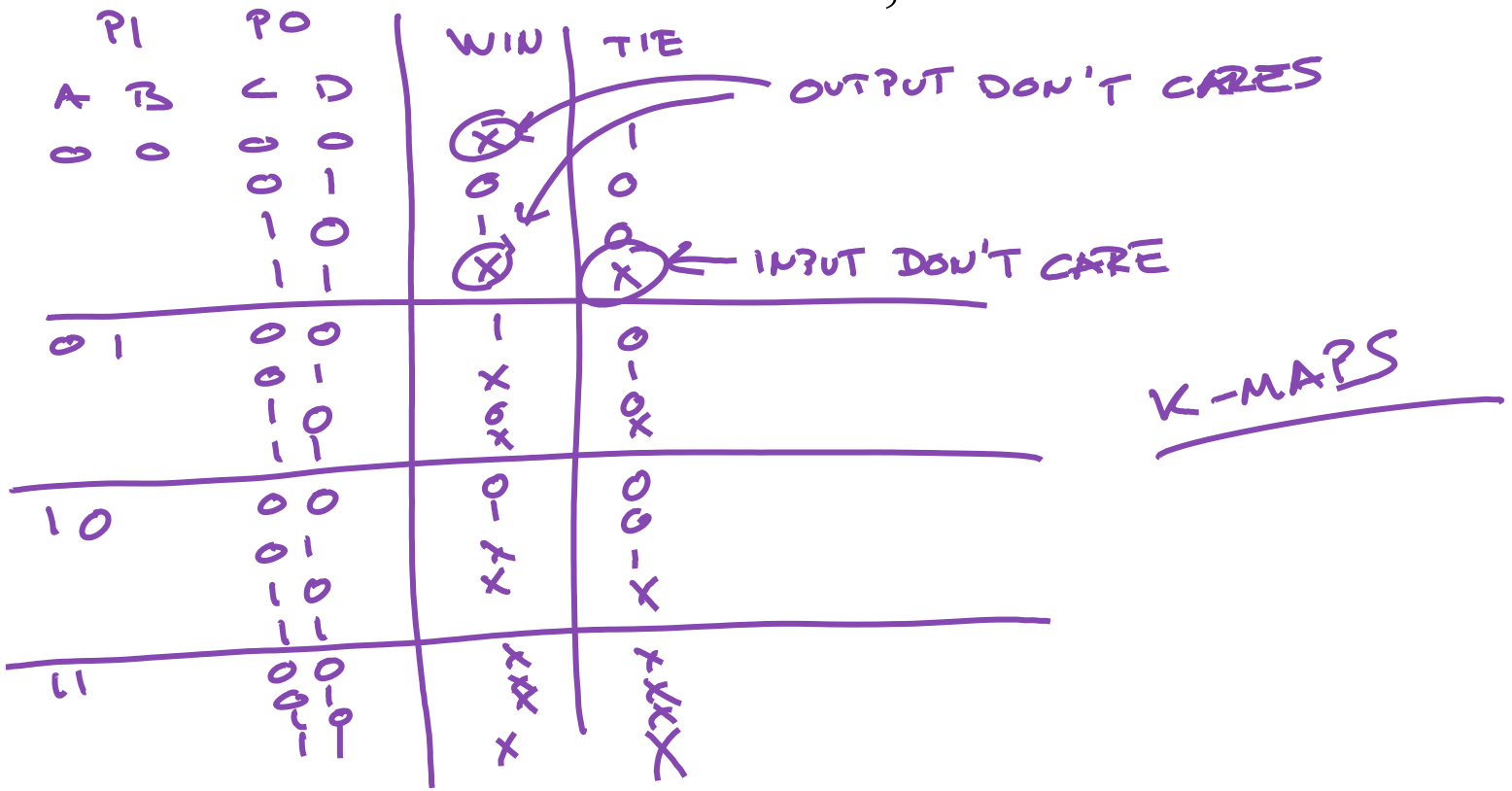


$\overline{A+B} = \overline{A}\,\overline{B}$

$\overline{AB} = \overline{A} + \overline{B}$

Review problem: circuit design

- Rock (00), Paper (01), Scissors (10) for two players.
- Output: Winner = Winner's ID (0/1)
  Tie = 1 if Tie, 0 if not

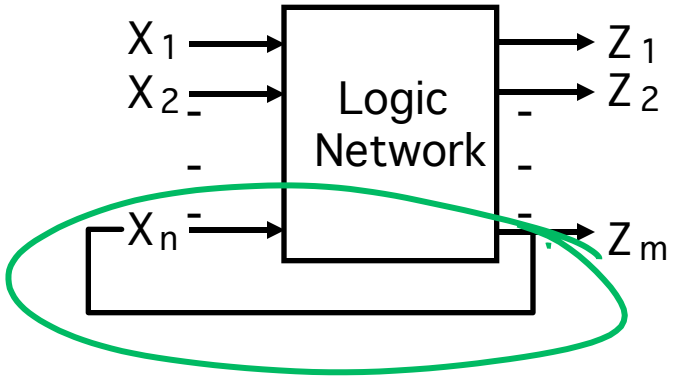


Review Problem: Basic Verilog

- Write the Verilog for a 2-input gate that is TRUE when an odd number of inputs are true.

Combinational vs. Sequential Logic

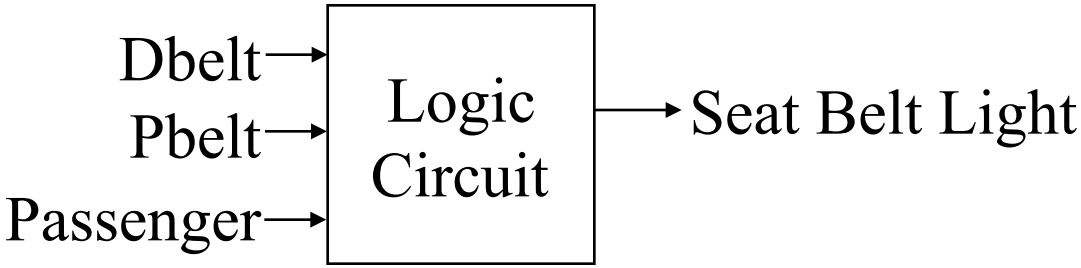
- Readings: 5-5.4.4



Network implemented from logic gates. The presence of feedback distinguishes between ***sequential*** and ***combinational*** networks.

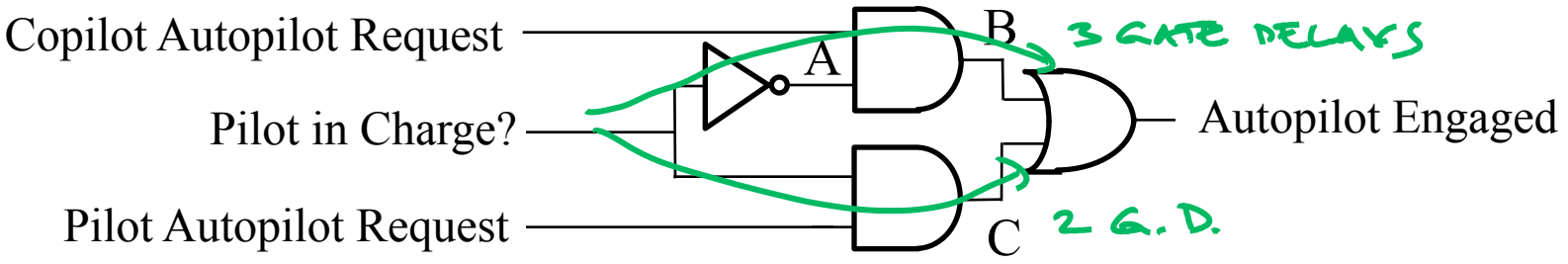
Combinational logic

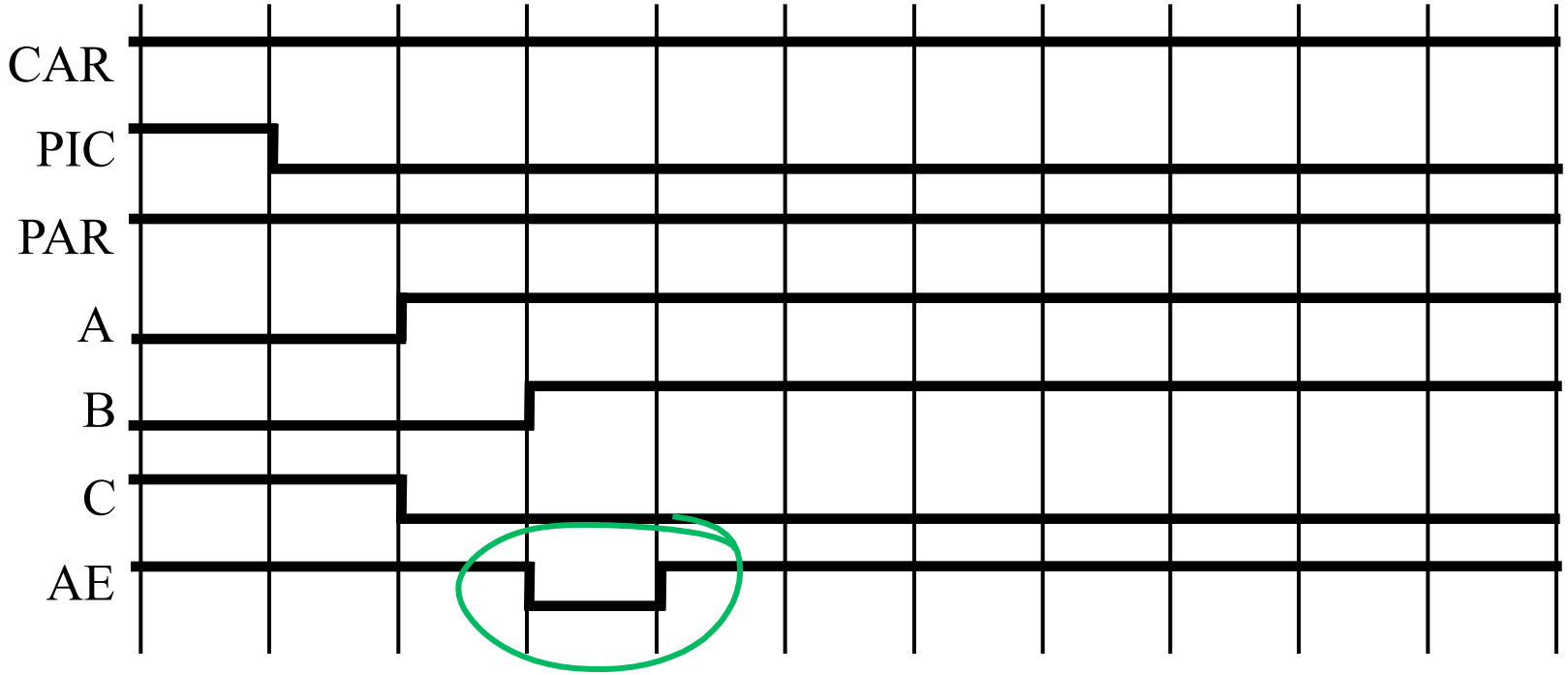
no feedback among inputs and outputs
outputs are a pure function of the inputs
e.g., seat belt light:
(Dbelt, Pbelt, Passenger) mapped into (Light)



Hazards/Glitches

- Circuit can temporarily go to incorrect states

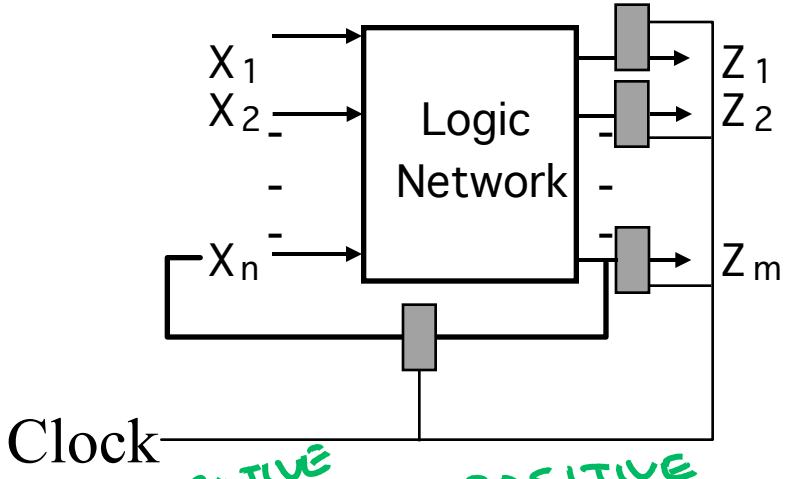


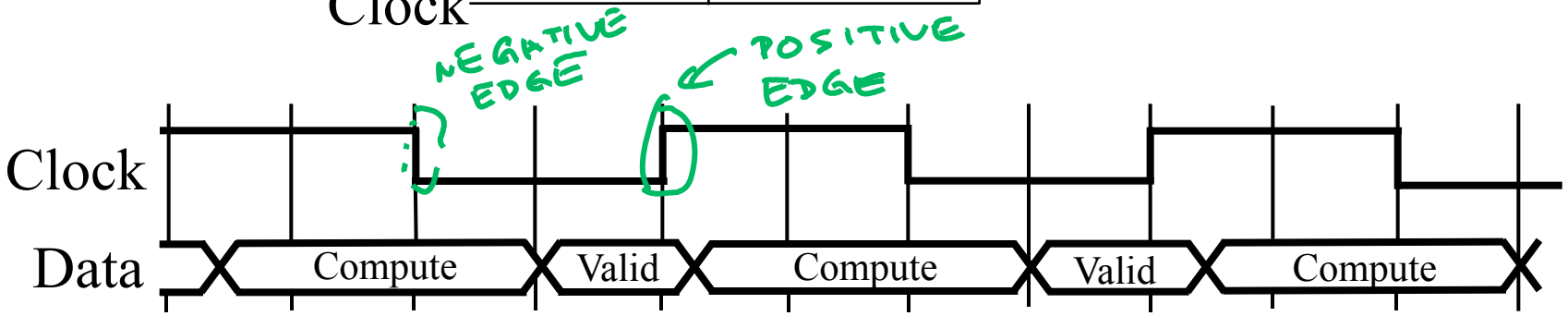


- Must filter out temporary states

Safe Sequential Circuits

- Clocked elements on feedback, perhaps outputs
  - Clock signal synchronizes operation
  - Clocked elements hide glitches/hazards





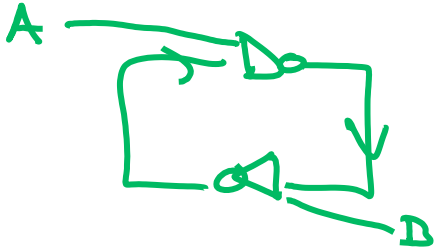
STATE HOLDING ELEMENT CHANGES VALUE AT CLOCK EDGE

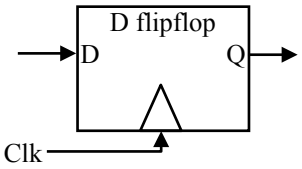
Basic D Flip Flop

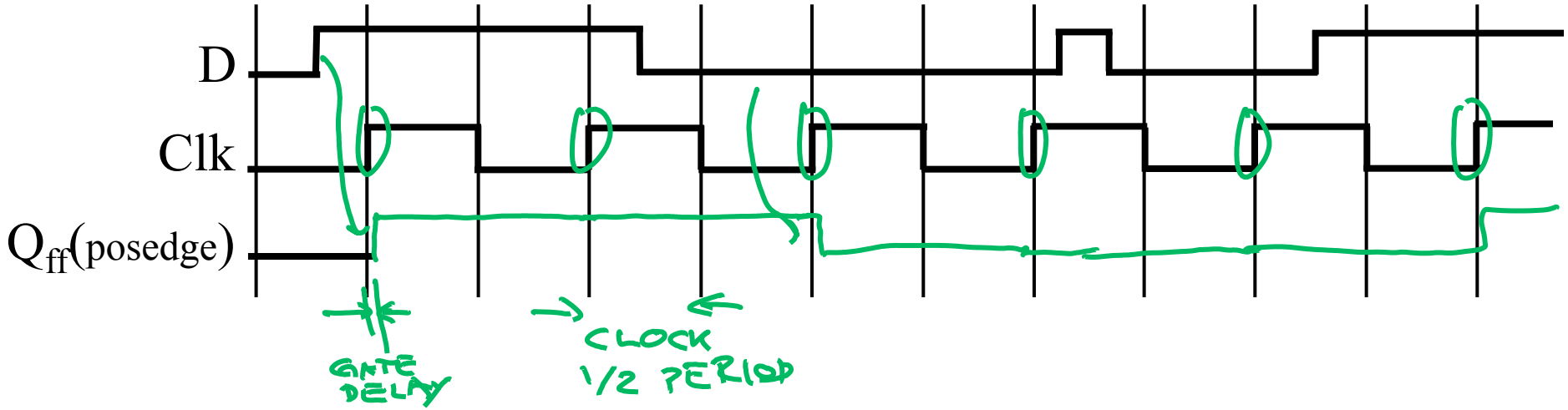
```
// Basic D flip-flop

module basic_D_FF (q, d, clk);
  output q;
  input d, clk;
  reg q; // Indicate that q is stateholding

  always @(posedge clk)
    q <= d; // ALWAYS use <= to assign to clocked elements
endmodule
```

POSITIVE EDGE TRIGGERED

IF YOU ARE WRITING AFTER always@(..) , USE <=





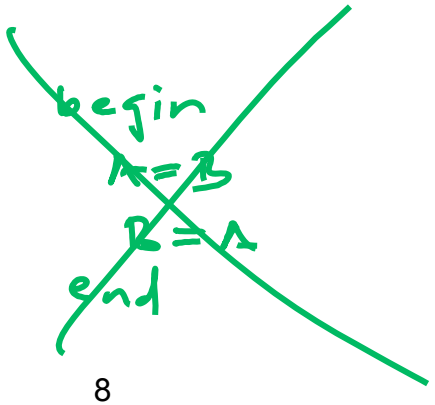


Blocking and Non-Blocking Assignments

- Blocking assignments (X=A)
  - completes the assignment before continuing on to next statement
- Non-blocking assignments (X<=A)
  - completes in zero time and doesn't change the value of the target until a blocking point (delay/wait) is encountered
- Example: swap

```
always @(posedge CLK)
    begin
        temp = B;
        B = A;
        A = temp;
    end
```

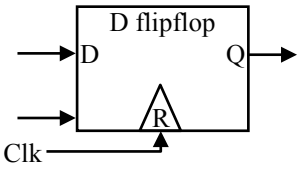
```
always @(posedge CLK)
    begin
        A <= B;
        B <= A;
    end
```



D Flip Flop w/Synchronous Reset

```
// D flip-flop w/synchronous reset

module D_FF (q, d, reset, clk);
  output q;
  input d, reset, clk;
  reg q; // Indicate that q is stateholding

  always @(posedge clk)
    if (reset)
      q <= 0;  // On reset, set to 0
    else
      q <= d; // Otherwise out = d

endmodule
```

WITH CLOCK EDGE

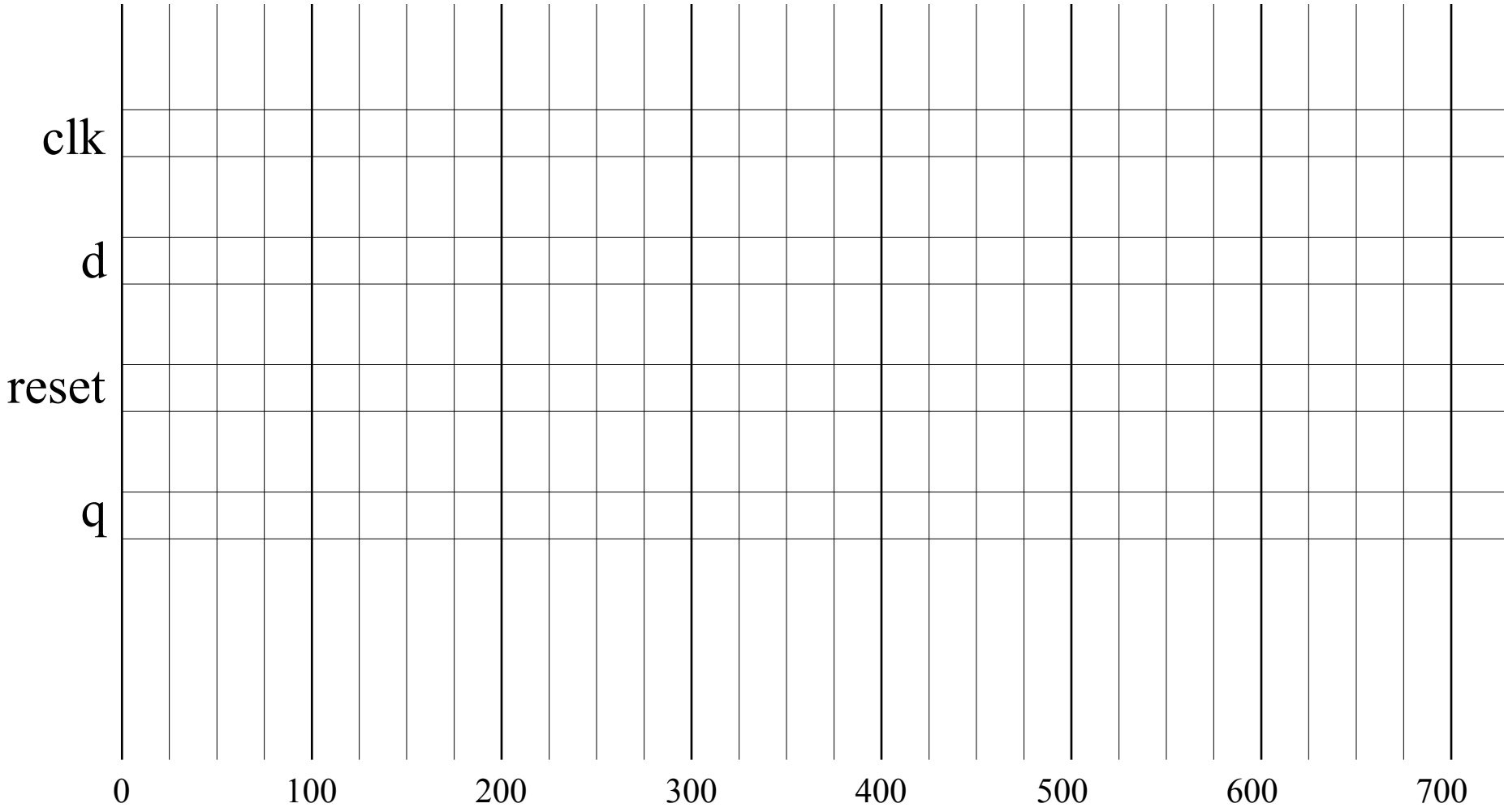


Verilog Testbench

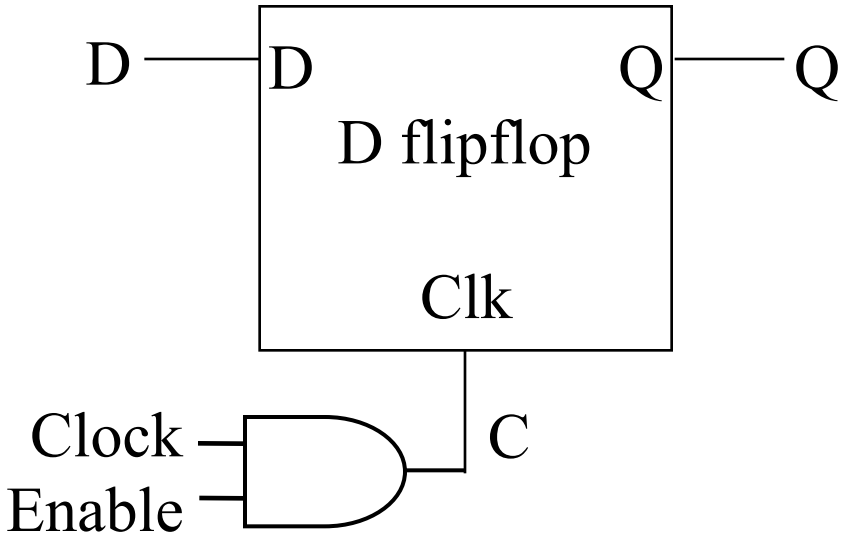
```
module stimulus;
  reg clk, reset, d;
  wire q;

  parameter ClockDelay = 100;

  D_FF dut (.q, .d, .reset, .clk); // Instantiate the D FF

  initial clk <= 0;// Set up the clock
  always #(ClockDelay/2) clk <= ~clk; // Toggle clock every 50 time units

  initial // Set up the reset signal
  begin
    d <= 0; reset <= 1; @(posedge clk);
            reset <= 0; @(posedge clk);
    d <= 1;             @(posedge clk);
    d <= 0;             @(posedge clk);
                        @(posedge clk);
    $stop(); // end the simulation
  end

endmodule
```

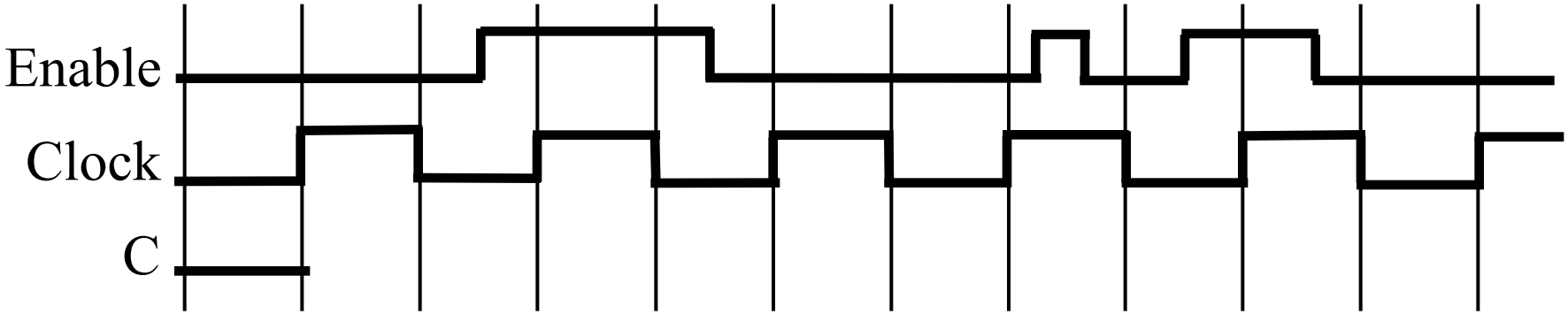
TIME	D	R	C	Q
0	0	1	0	X
50	0	0	1	0
100	0	0	0	0
150	1	0	1	0
200	1	0	0	0
250	0	0	1	1
300				
400				
450				

Testbench Waveforms



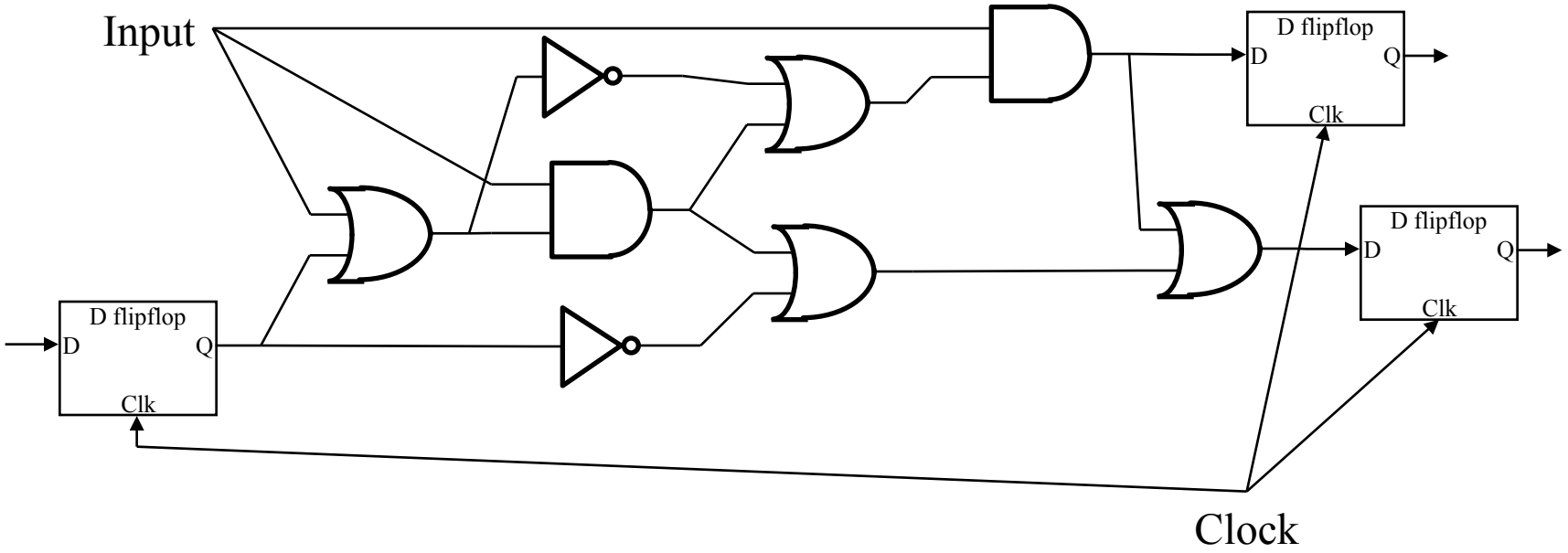
Flipflop Realities 1: Gating the Clock



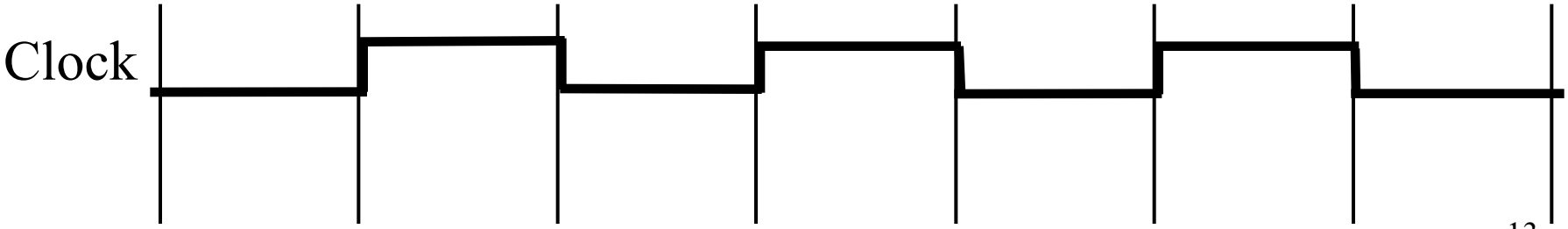


- **NEVER put a logic gate between the clock and DFF's CLK input.**

Flipflop Realities 2: Clock Period, Applying Stimulus

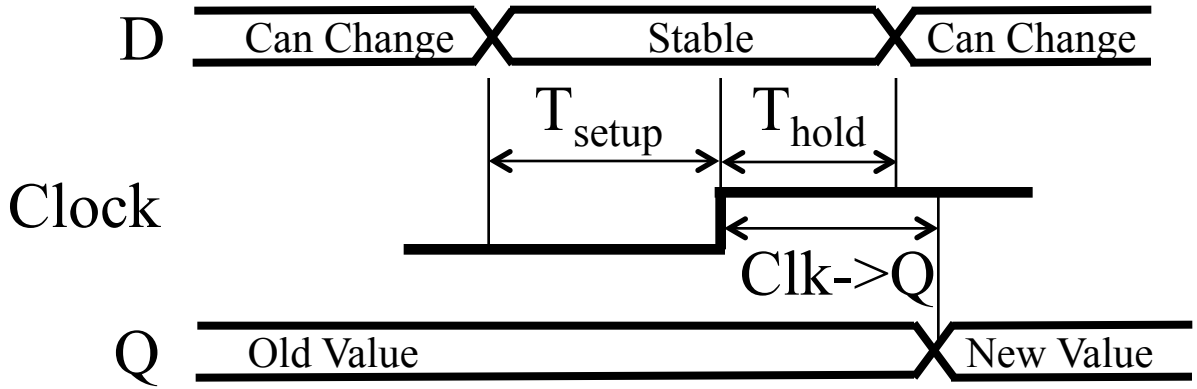


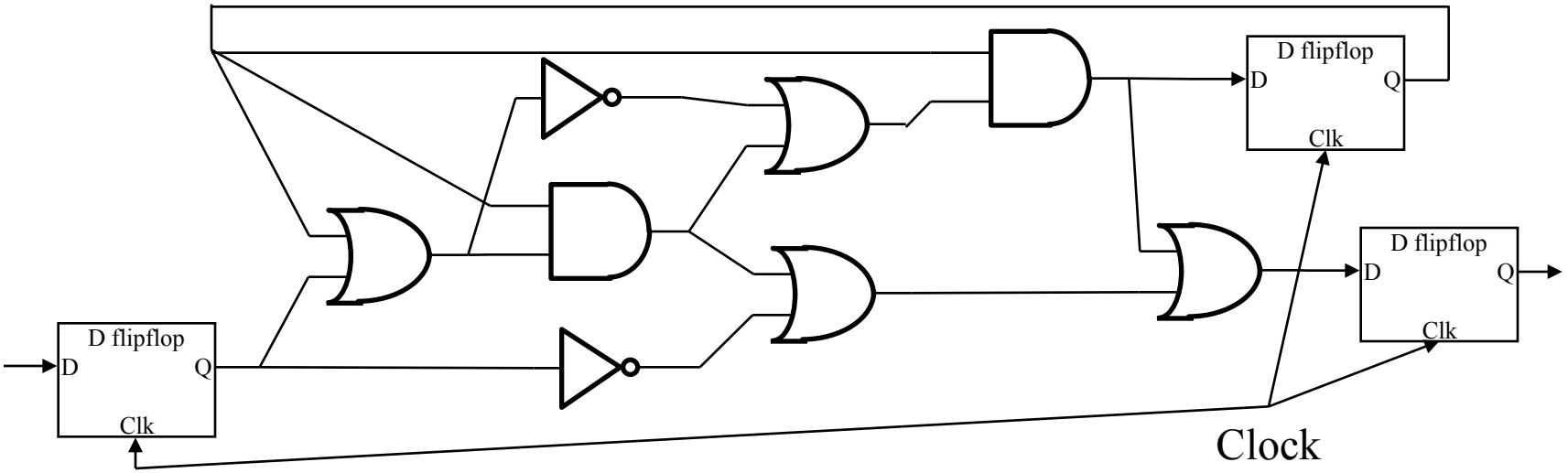
- Clock Period?
- Apply Inputs when?



T_{setup}, T_{hold}, Clk->Q

- Flipflops require their inputs be stable for time period around clock edge



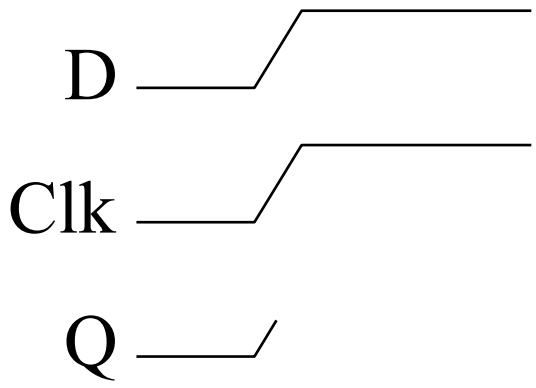


Timing Definitions

- T_{setup}: Time D must be stable BEFORE clock edge
  - Adds to critical path delay
- Clk->Q: Time from clock edge to Q changing
  - Adds to critical path delay
- T_{hold}: Time D must be stable AFTER clock edge
  - Sets minimum path from Q of one DFF to D of another

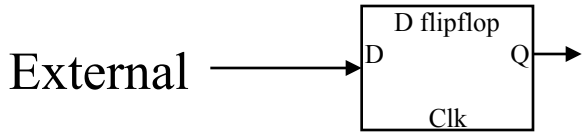
Flipflop Realities 3: External Inputs

- External inputs aren't synchronized to the clock

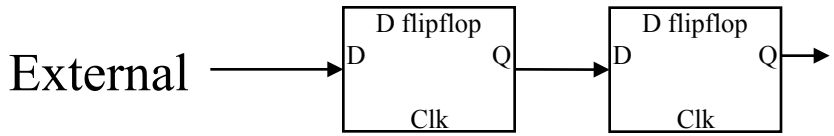


Dealing with Metastability

- Single DFF



- 2 DFFs in series



- 2 DFFs in parallel