



Welcome to The Hardware Lab!

Fall 2018
Sequential Circuit I

Prof. Chun-Yi Lee

Department of Computer Science
National Tsing Hua University

Agenda

- **Announcement**
- Lab 1 advanced question
- Combinational circuits
- Sequential circuits
- Debounce circuit and one-pulse circuit
- 7-segment display

Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

Announcement

- Midterm exam
 - Computer based exam on **10/18/2018 (Thu)**, 3:30pm~5:30pm
- Please show up during the lab hours
 - You may request for permission to be absent **IN ADVANCE**
 - You will need a paper proof if you don't have a permission
 - Otherwise, we will deprive one of your rights each time **RANDOMLY**

Midterm quiz 少五分鐘

Midterm quiz 多考一題

Midterm quiz 考難題，時間多半小時

Final quiz 少五分鐘

Final quiz 多考一題

Final quiz 考難題，時間多半小時

每次Lecture Time都會被點名，沒到的話就再抽一次

Agenda

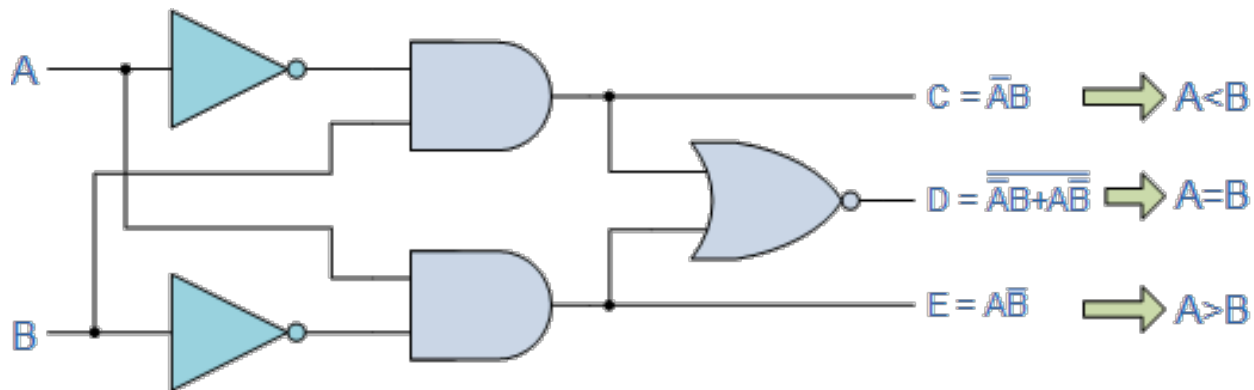
- Announcement
- **Lab 1 advanced question**
- Combinational circuits
- Sequential circuits
- Debounce circuit and one-pulse circuit
- 7-segment display

Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

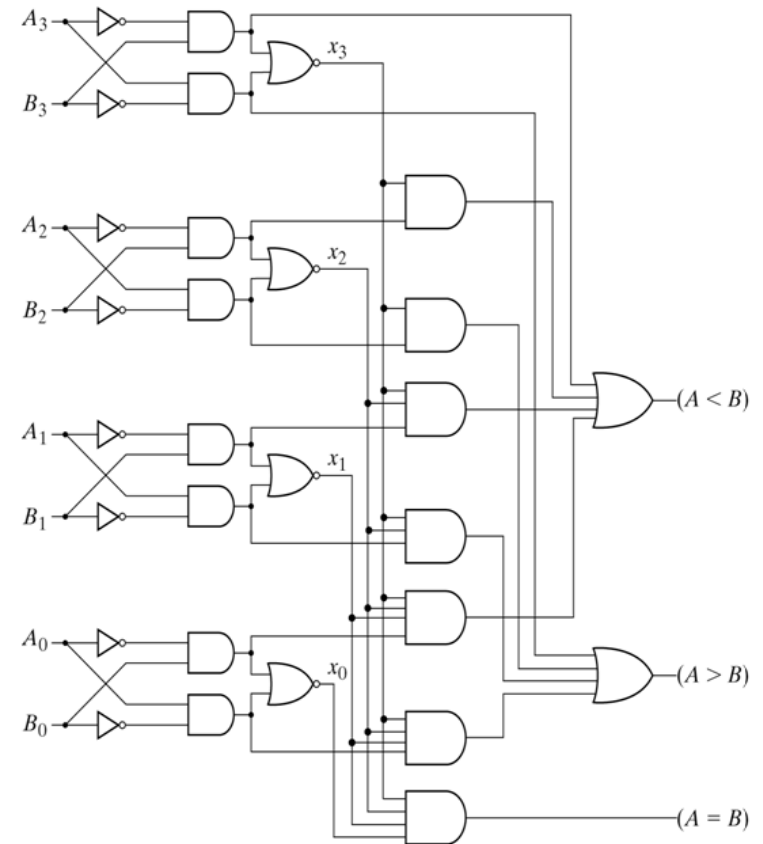
Lab 1 Advanced Question

- 3-bit comparator
- Design a 1-bit comparator first



Lab 1 Advanced Question

- Connect 1-bit comparators to form a 3-bit comparator
- Compares from **MSB** to **LSB**
 - If $A[3] > B[3]$, then $A > B$
 - If $A[3] == B[3]$, then compare $A[2]$ and $B[2]$
 - If $A[3:2] == B[3:2]$, then compare $A[1]$ and $B[1]$, and so on
 - Follow the same fashion till **LSB**



Agenda

- Announcement
- Lab 1 advanced question
- **Combinational circuits**
- Sequential circuits
- Debounce circuit and one-pulse circuit
- 7-segment display

Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

Combinational Circuits (1/2)

- You have **three ways** to model a combinational circuit
 - You will be good if you remember them

1. Gate-level modeling

- `not N1(out, in);`
- `nand N2(out, in1, in2);`
- `or N3(out, in1, in2, in3);`
- etc.

2. Continuous assignment

- Declare your **lvalues** as **wire** data type
- **assign** **lvalue** = (your logic expression);

Combinational Circuits (2/2)

3. Use always block

- Declare your **lvalues** as **reg** data type
- Only use "=" in your assignment expressions
- Avoid using "<=" in your expressions

Correct

```
reg lvalue;  
  
always @ (*)  
begin  
    lvalue = (your expression);  
end
```



Wrong

```
wire lvalue;  
  
always @ (*)  
begin  
    assign lvalue <= (your expression);  
end
```



1. lvalue should be **reg**
2. Can't use **assign**
3. Avoid "<="

Agenda

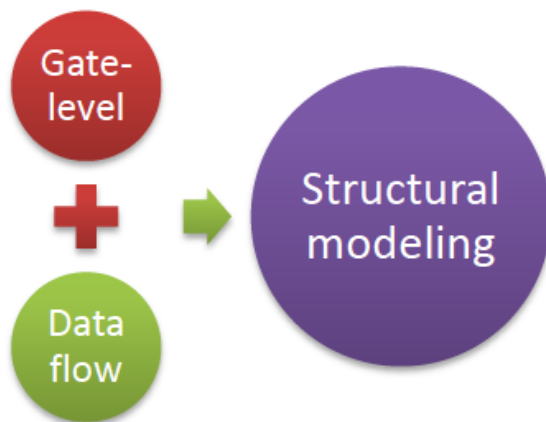
- Announcement
 - Lab 1 advanced question
 - Combinational circuits
 - **Sequential circuits**
 - Debounce circuit and one-pulse circuit
 - 7-segment display
- **Latch and flip-flop**
 - Always block usage
 - Reset

Today's class will help you:

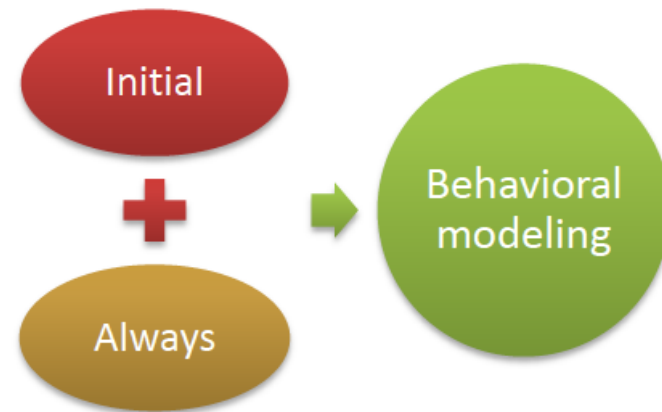
1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

Behavioral Modeling

- High level description
- Modeling a circuit by its behaviors
- Similar to C++ programming
- Behavioral modeling includes both **combinational** and **sequential** parts



Combinational only



Combinational and Sequential

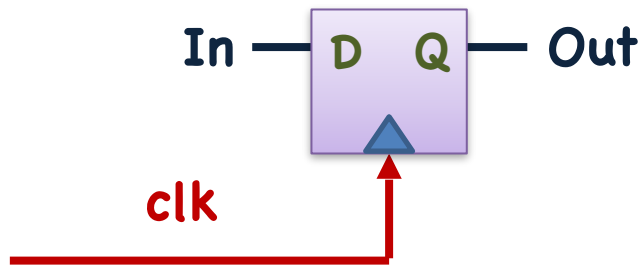
What is Sequential Circuit?

	Sequential	Combinational
Clock	Yes	No
Memory elements	Yes	No
Always block assignment	Non-blocking <=	Blocking =
Continuous assignment	Not available	assign lvalue = rvalue

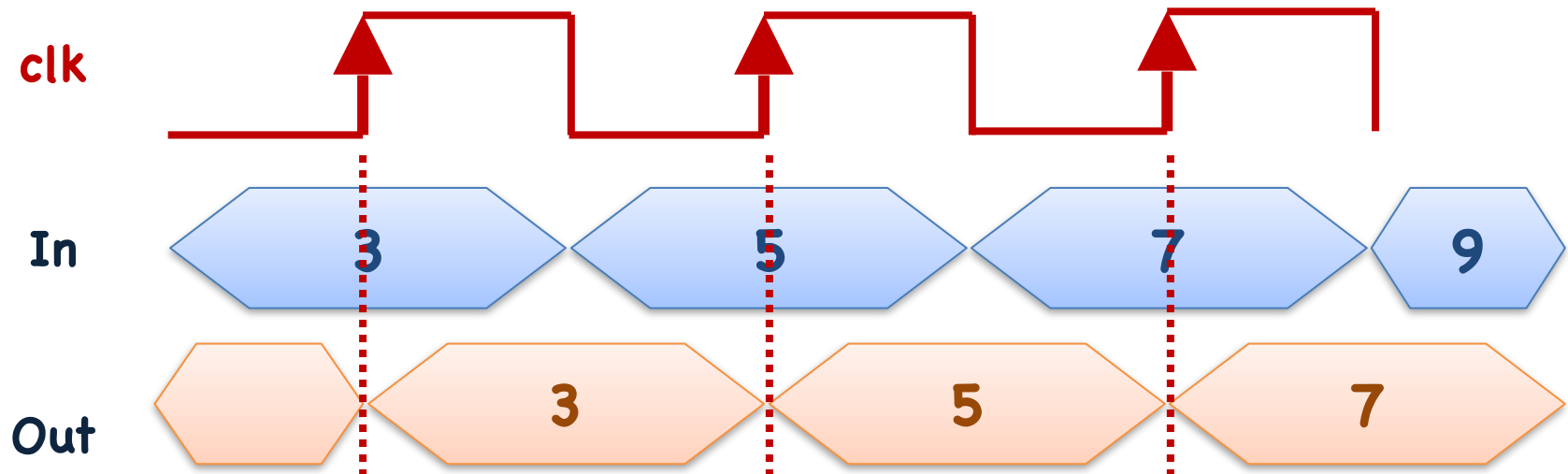
- Memory element
 - Stores "1" or "0"
 - Latch and Flip-Flop
 - Usually triggered by clock signals

Basic Sequential Circuit

D Flip-Flop

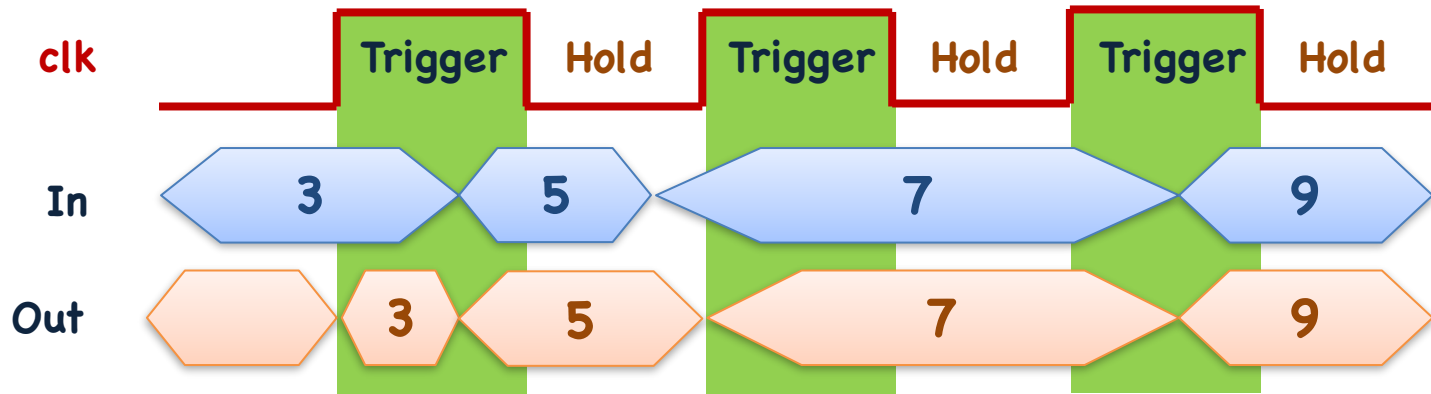


```
always @ (posedge clk)
begin
    //Non-blocking assignment
    Out <= In;
end
```

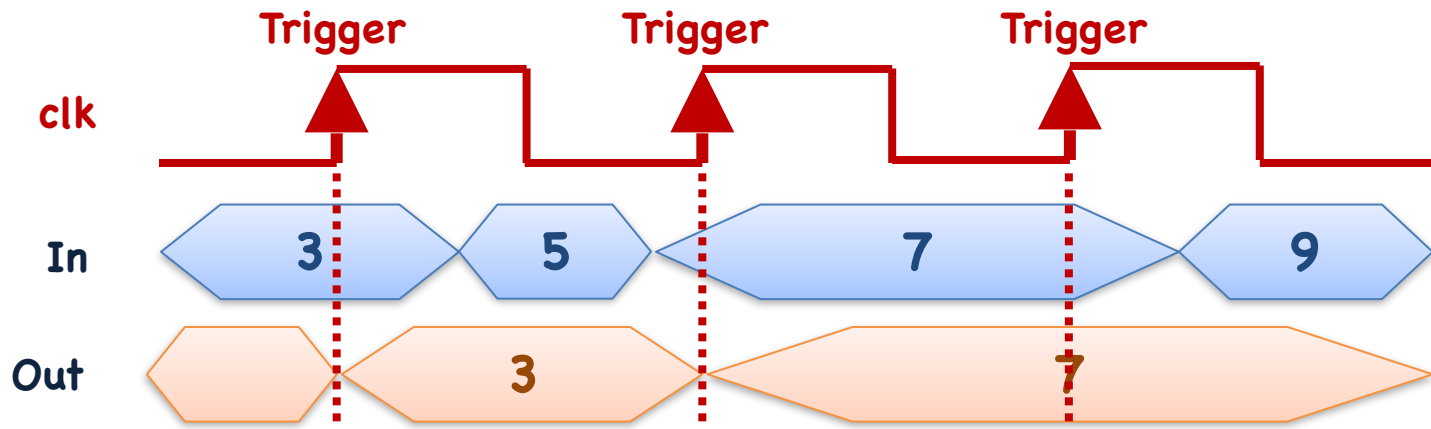


Latch and D Flip-Flop

- **Latch:** triggered by **level**



- **D Flip-Flop:** triggered by **clock edge**



D Flip Flop

- Changes its value at the clock edges

**Positive-edge triggered D Flip-Flop
Truth table**

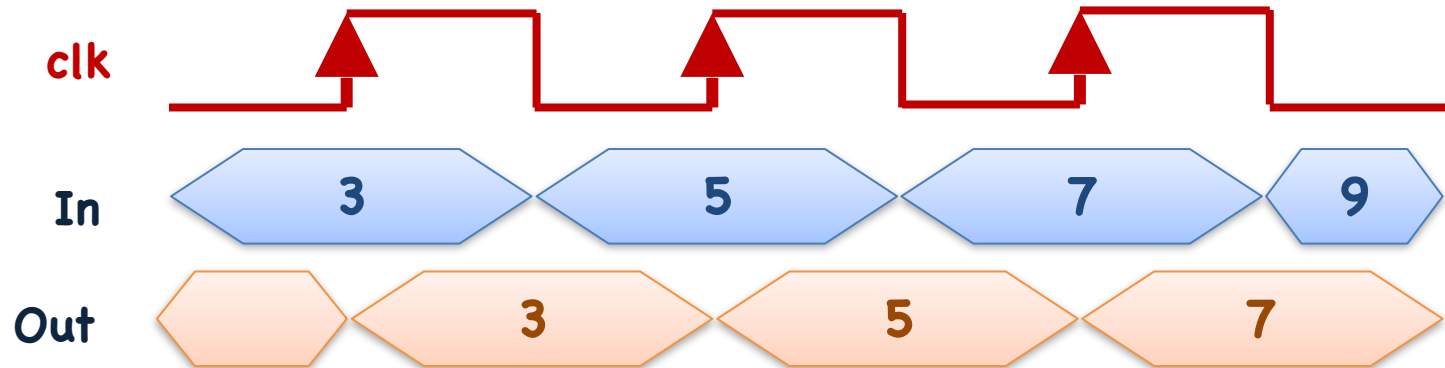
Clk	D	Q(t+1)
↑	0	0
↑	1	1
0	-	Q(t)
1	-	Q(t)

**Negative-edge triggered D Flip-Flop
Truth table**

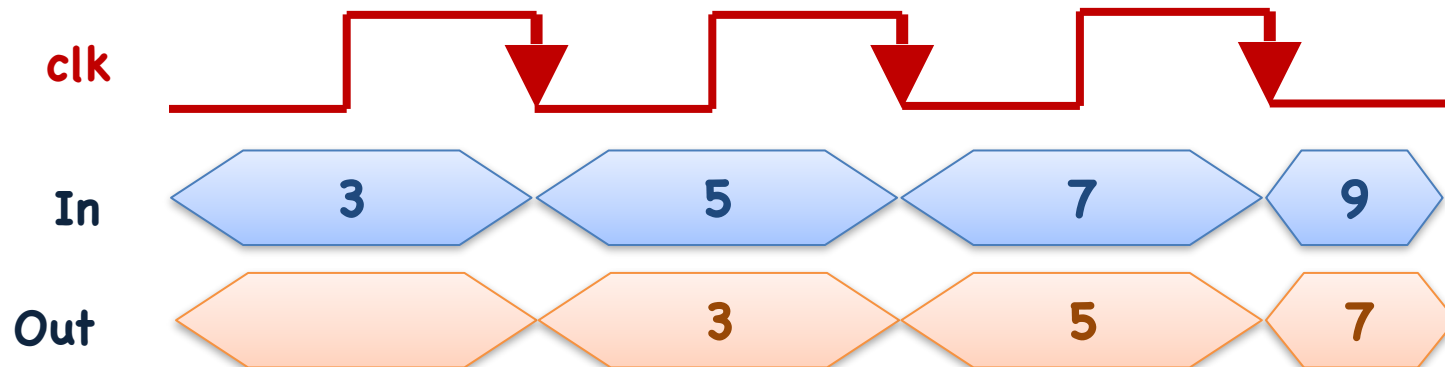
Clk	D	Q(t+1)
↓	0	0
↓	1	1
0	-	Q(t)
1	-	Q(t)

Edge-Trigger in Always Block

- Positive-edge trigger: use **posedge** in your always block

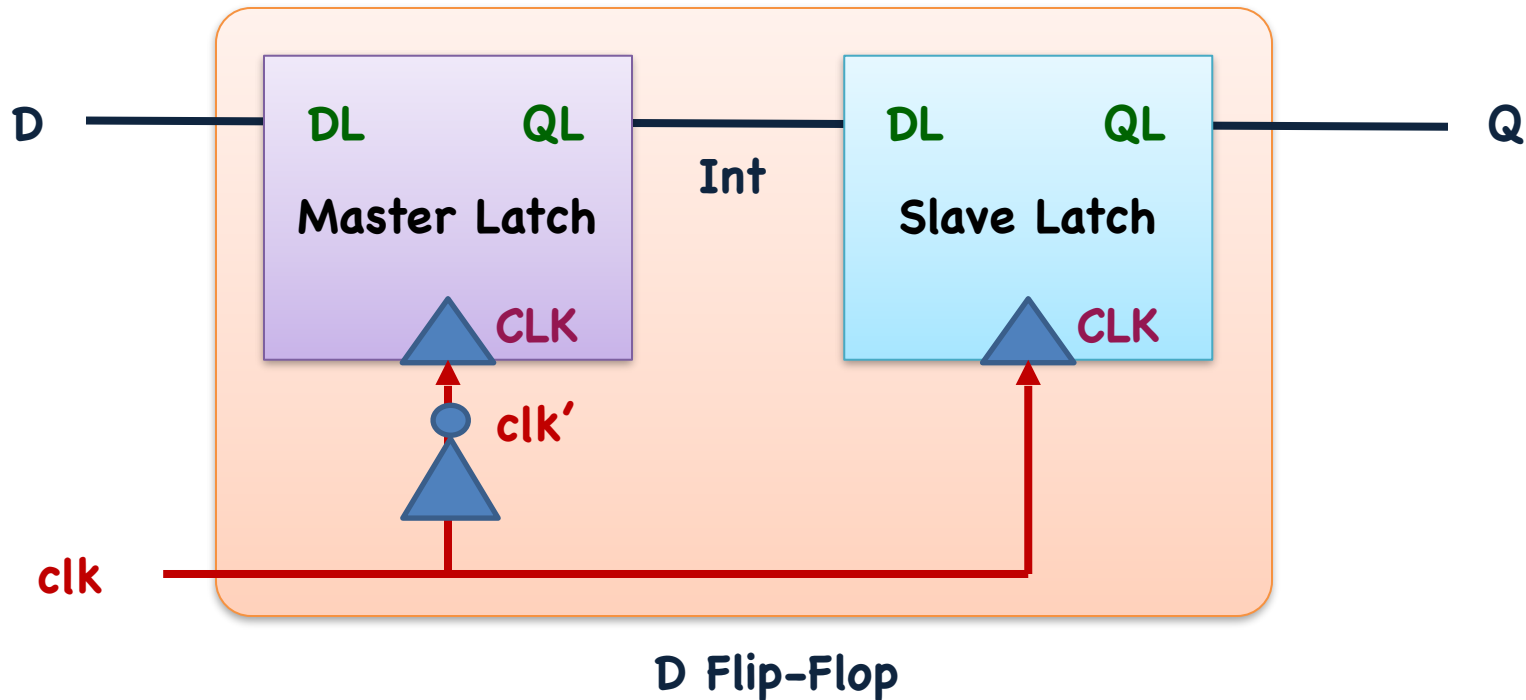


- Negative-edge trigger: use **negedge** in your always block



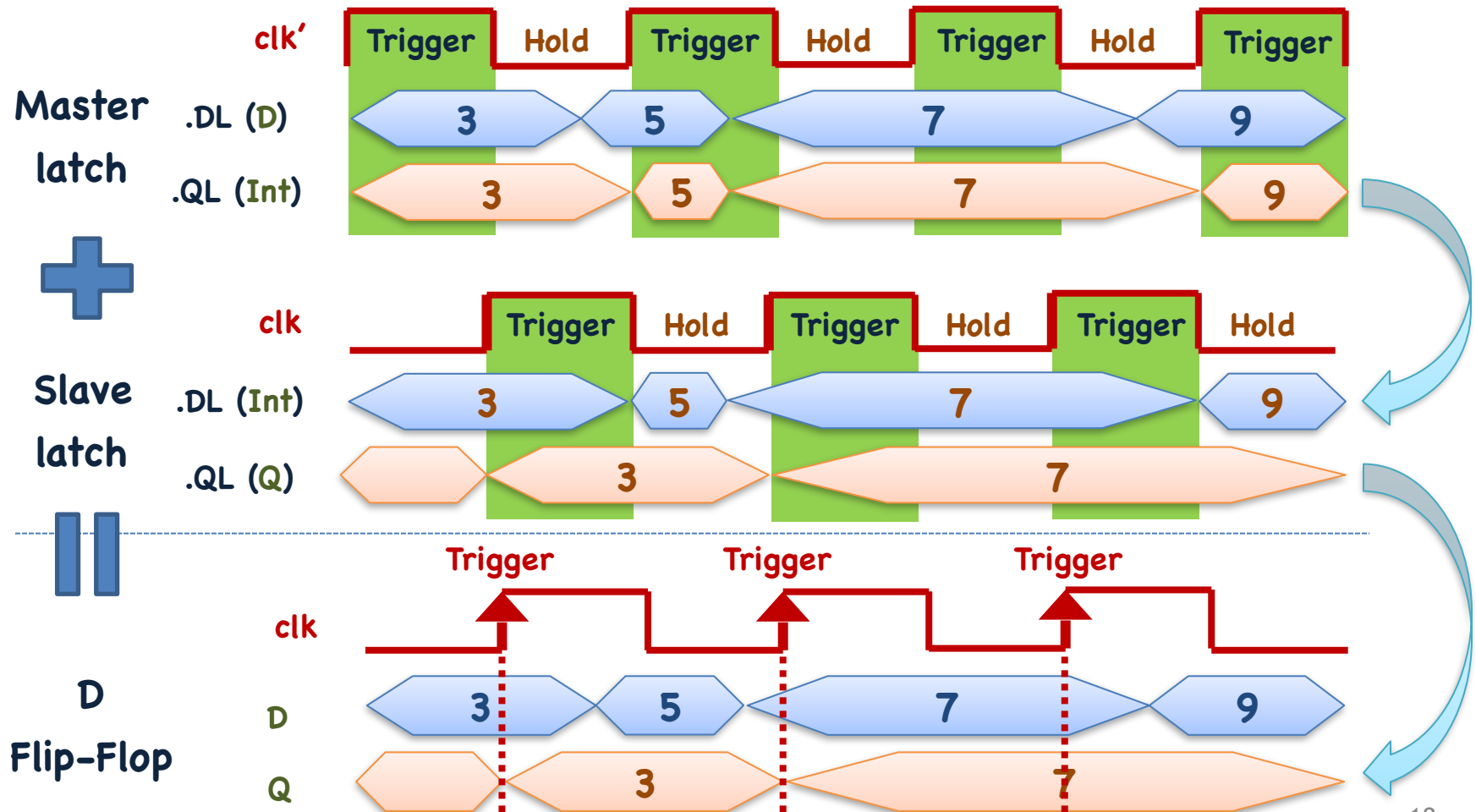
Latches in a D Flip-Flop

- A positive-edge triggered D Flip-Flop consists of two latches
 - The master latch is triggered by clk'
 - The slave latch is triggered by clk
- Two level triggered latches form an edge-triggered Flip-Flop



Flip-Flop Timing Diagram

- A timing diagram from the master latch to slave latch



Agenda

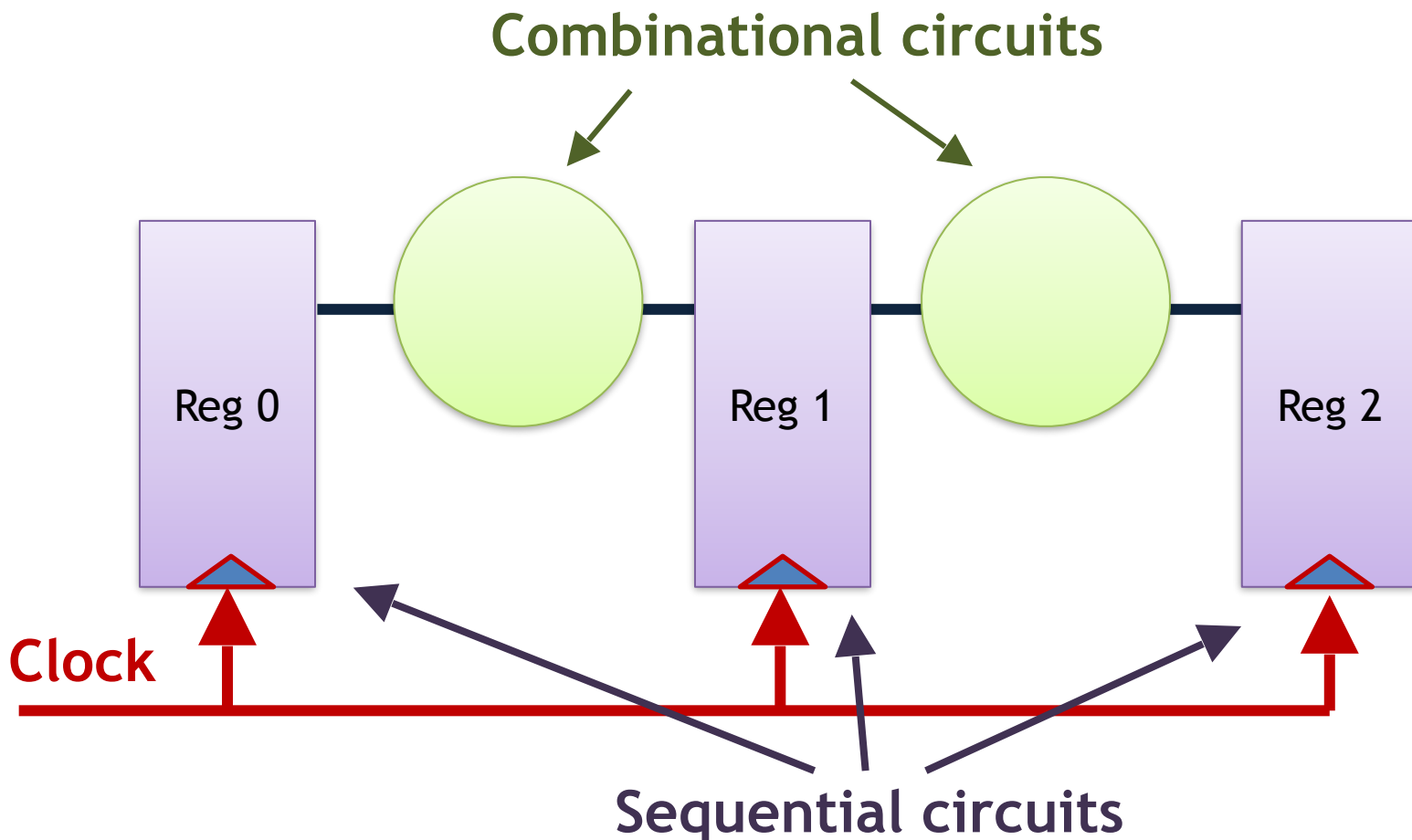
- Announcement
 - Lab 1 advanced question
 - Combinational circuits
 - **Sequential circuits**
 - Debounce circuit and one-pulse circuit
 - 7-segment display
- Latch and flip-flop
 - **Always block usage**
 - Reset

Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

Register Transfer Level

- Describes the behavior of combinational circuits between registers



Always Block

- Two types of always block

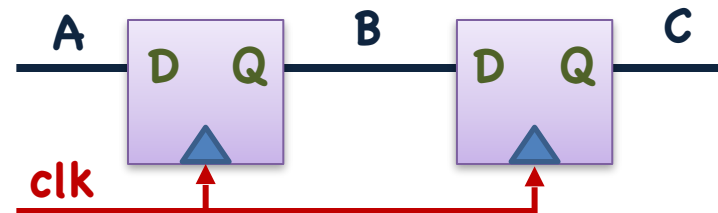
Combinational circuit

```
reg A, B, C;  
  
always @ (A or B)  
begin  
    //Blocking Assignment  
    B = A;  
    C = B;  
end
```



Sequential circuit

```
reg A, B, C;  
  
always @ (posedge clk)  
begin  
    //Non-blocking assignment  
    B <= A;  
    C <= B;  
end
```



Blocking and Non-blocking

Execute in Order

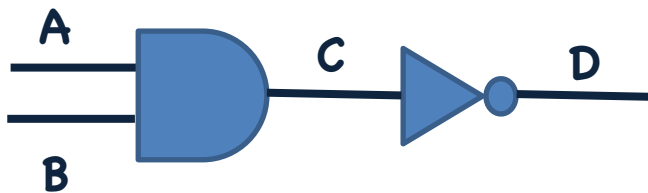
```
always @ (A or B or C)  
begin
```

```
//Blocking Assignment
```

```
C = A & B;
```

```
D = !C;
```

```
end
```



Execute in Parallel

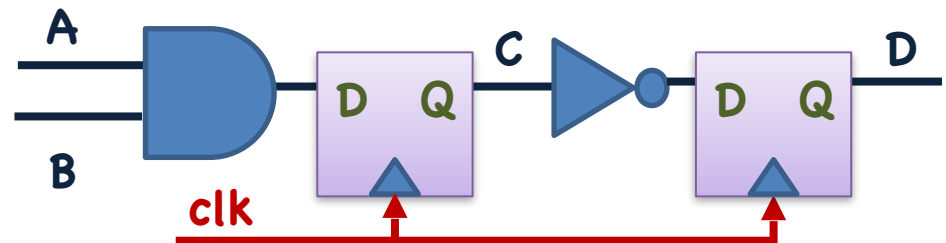
```
always @ (posedge clk)  
begin
```

```
//Non-blocking assignment
```

```
C <= A & B;
```

```
D <= !C;
```

```
end
```



- **NEVER** use blocking and non-blocking assignment in the **SAME** always block

Agenda

- Announcement
 - Lab 1 advanced question
 - Combinational circuits
 - **Sequential circuits**
 - Debounce circuit and one-pulse circuit
 - 7-segment display
- Latch and flip-flop
 - Always block usage
 - **Reset**

Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

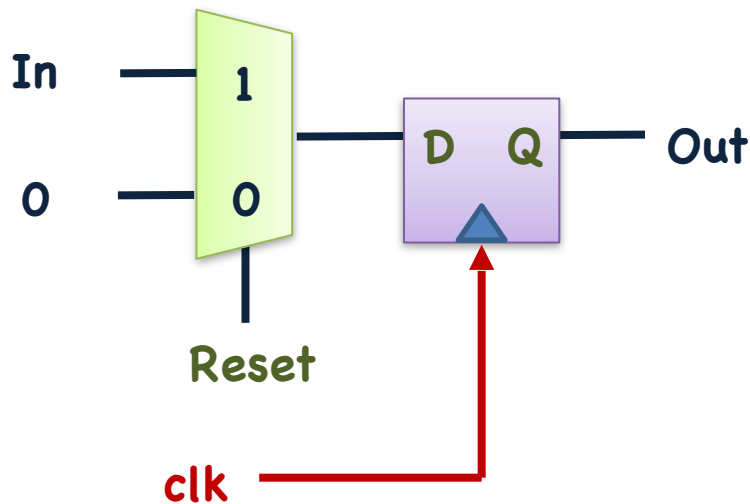
Initialize Flip-Flops

- For combinational circuits, we initialize the inputs of modules by initial blocks in the testbench
- For sequential circuits, a **reset signal** is necessary
 - Initialize the reset signal (i.e. your module) first in your testbench
- Two reset methods
 - Synchronous reset
 - Asynchronous reset (Next week)

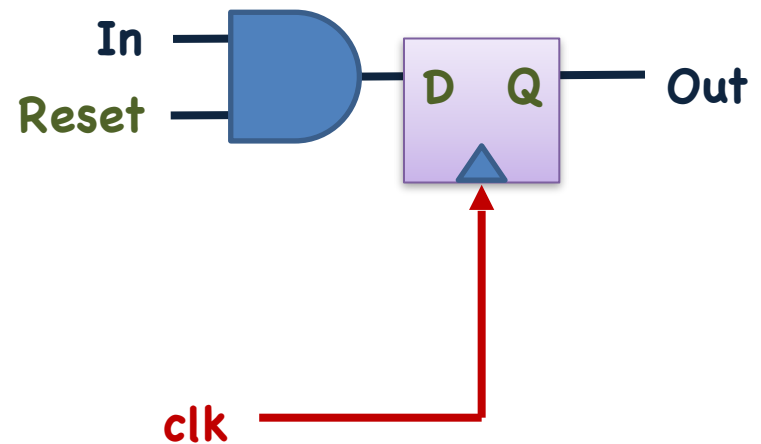
Synchronous Reset

- Reset the value stored in a D Flip-Flop to 0
- Triggered by clock edges
- Assume that $Q = 1'b0$ when Reset is set to $1'b0$

Scheme 1



Scheme 2

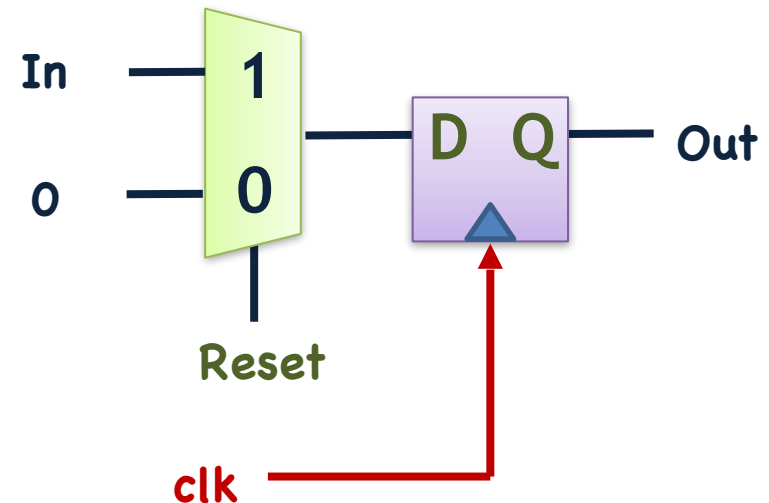


Synchronous Reset in Always Block

- **Prioritize** your reset signal over the rest of the inputs
- In your testbench, **initialize your reset signal first**
- You **shall not** use any initial block in your design modules

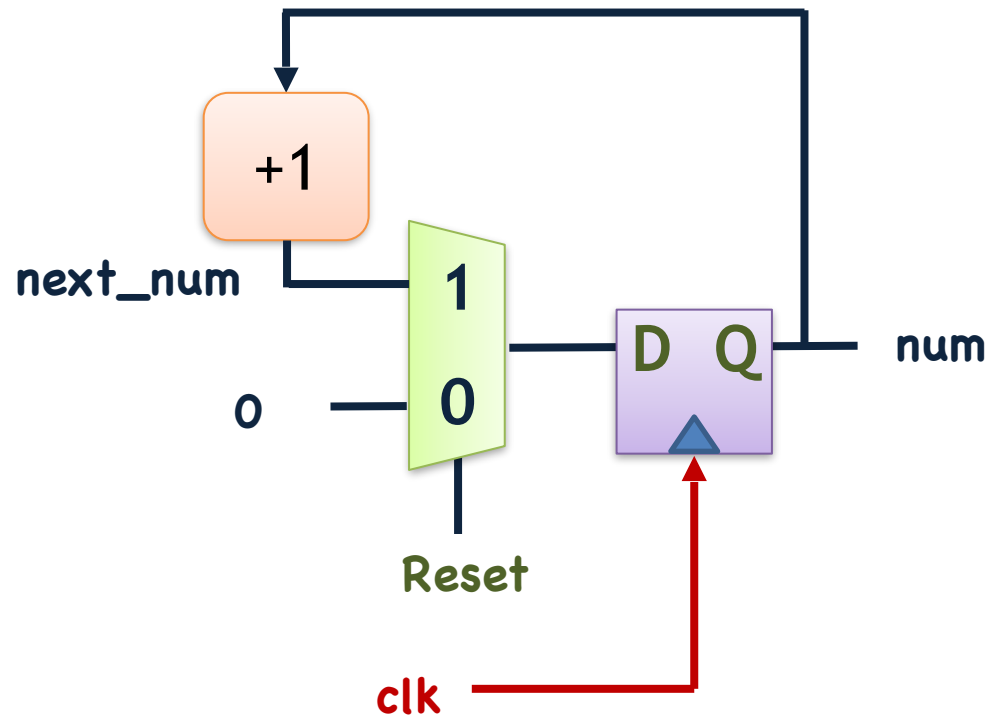
```
module My_Test_Flip_Flop (Out, In, clk, Reset);  
  input    In, clk, Reset;  
  output   Out;  
  reg      Out;  
  
  always @ (posedge clk) ← Behavior  
  begin  
    if (Reset == 1'b0)      Out <= 1'b0;  
    else                    Out <= In;  
  end  
endmodule
```

Declaration of Synchronous



Example: 3-Bit Counter

```
module Counter_3Bit(clk, Reset, num);  
input    clk, Reset;  
output   [2:0] num;  
reg      [2:0] num;  
wire     [2:0] next_num;  
  
always @(posedge clk) begin  
    if (!Reset)  
        num <= 0;  
    else  
        num <= next_num;  
    end  
  
assign    next_num = num + 1;  
  
endmodule
```



Agenda

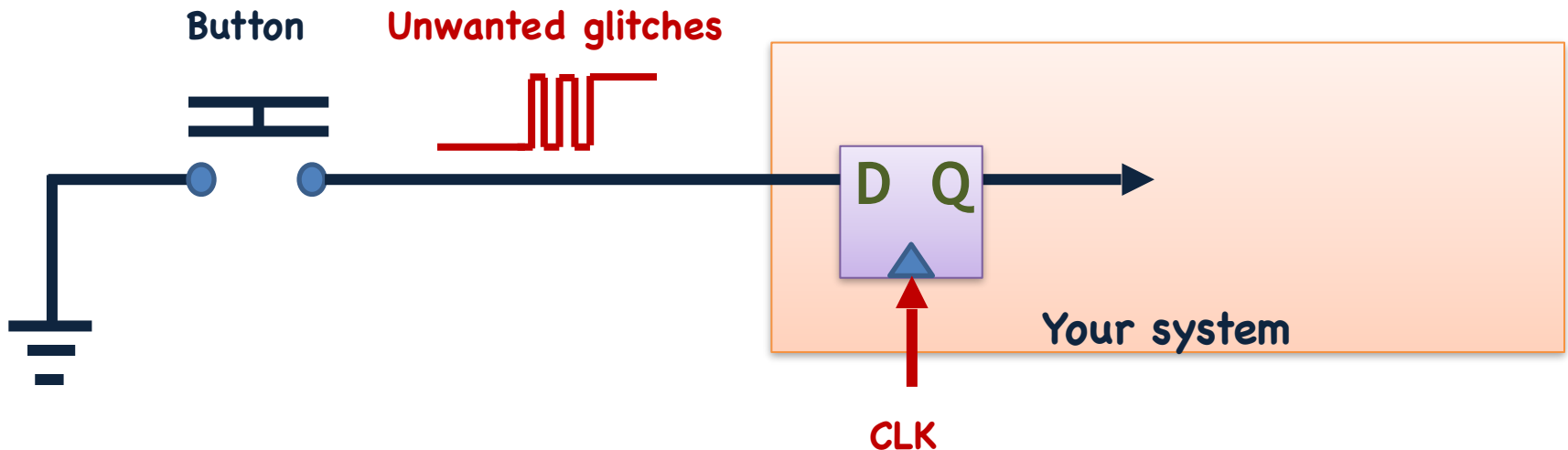
- Announcement
- Lab 1 advanced question
- Combinational circuits
- Sequential circuits
- **Debounce circuit and one-pulse circuit**
- **7-segment display**

Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

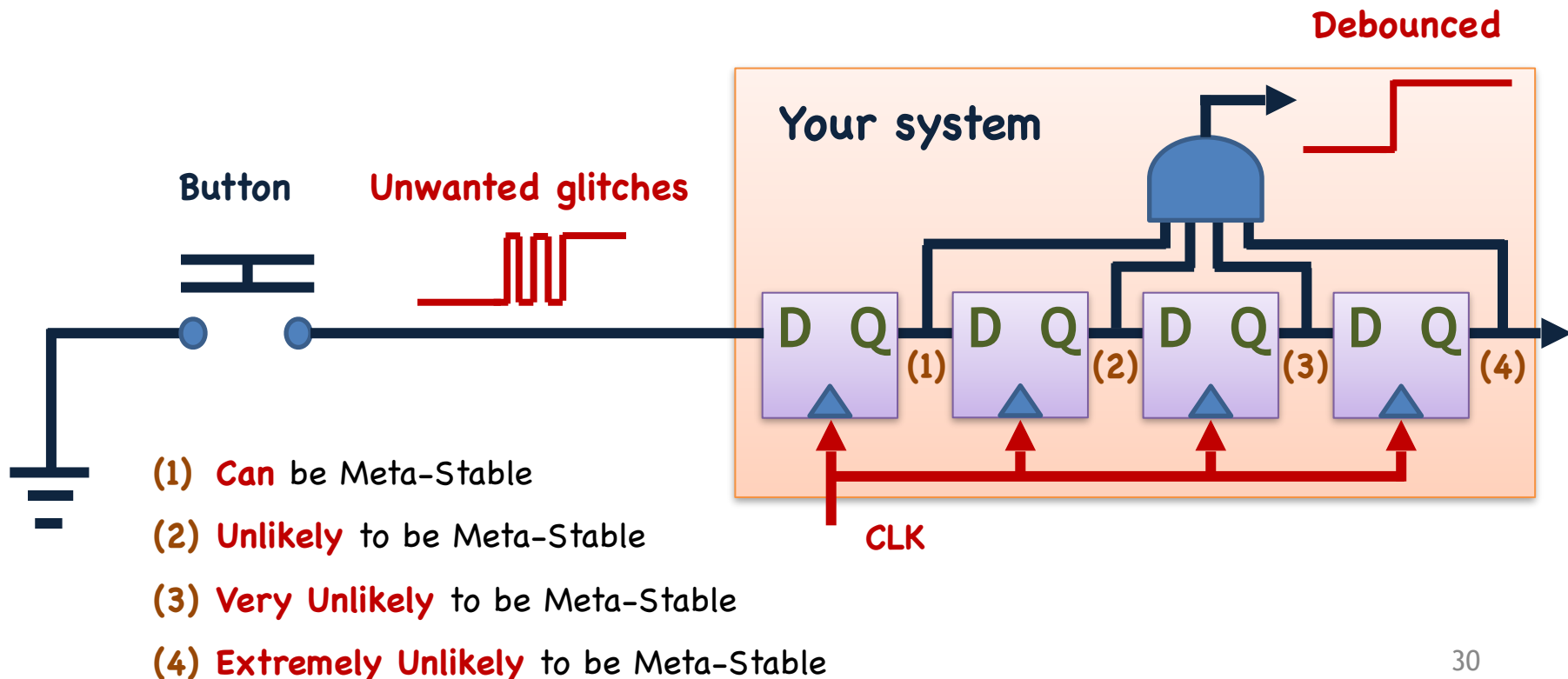
Push Buttons

- Several push buttons are available as inputs on FPGA
- Map push buttons to your inputs using .xdc file
- Need to handle **meta-stability (glitches)** problem



Handling Meta-Stability

- Debounce circuit
- Shift registers to allow time for signals to stabilize
- Detect if DFF chains are all ones or zeros



Verilog Code for Debounce Circuit

```
module debounce (pb_debounced, pb, clk);

    output    pb_debounced;    // signal of a pushbutton after being debounced
    input     pb;               // signal from a pushbutton
    input     clk;

    reg [3:0] DFF;              // use shift_reg to filter pushbutton bounce

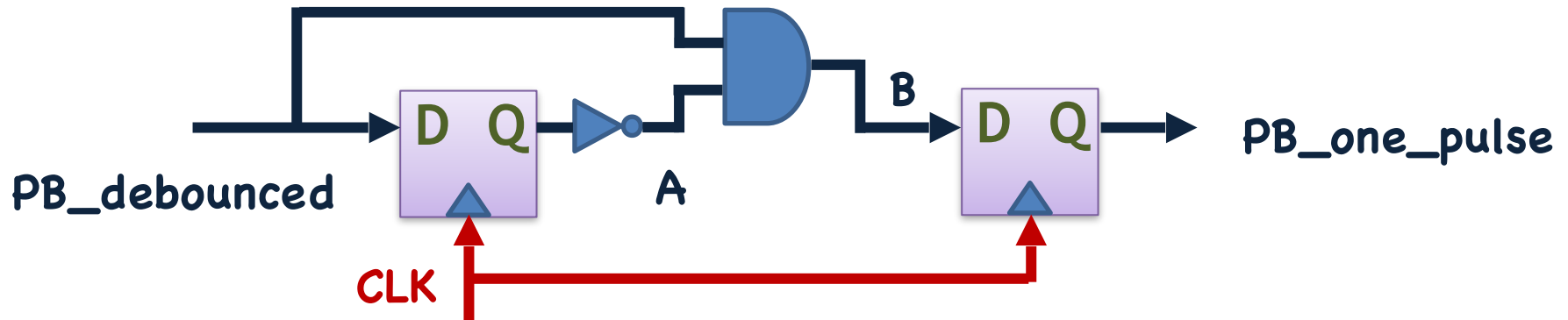
    always @(posedge clk)
    begin
        DFF[3:1] <= DFF[2:0];
        DFF[0]   <= pb;
    end

    assign pb_debounced = ((DFF == 4'b1111) ? 1'b1 : 1'b0);

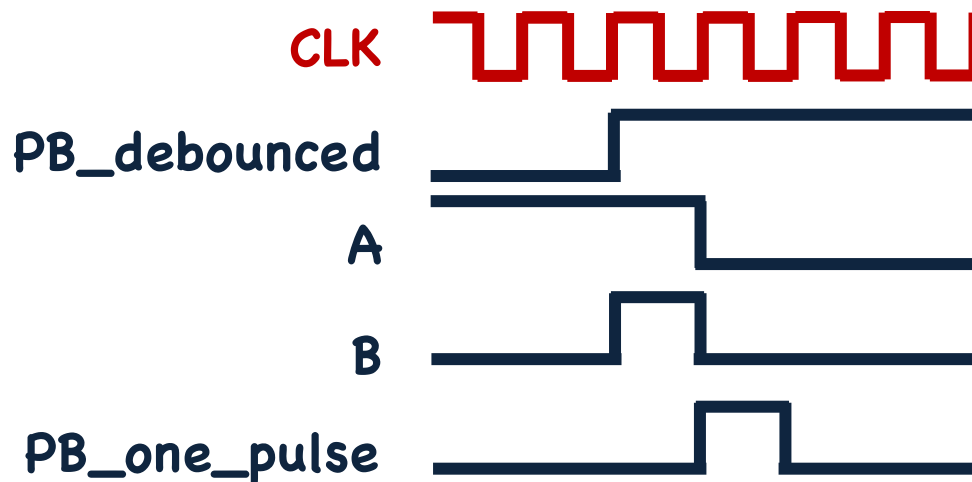
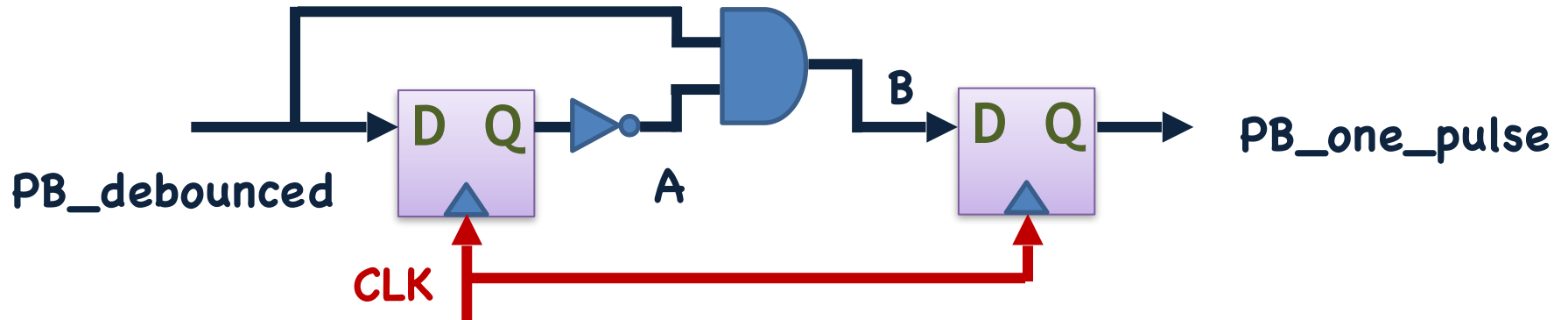
endmodule
```

One Pulse Circuit

- The one pulse circuit generates a one-clock-cycle pulse when the push button is pressed
 - A push button is usually pressed for many clock cycles
- The circuit is used when we want to trigger only once



One Pulse Circuit (Cont'd)



Verilog Code

```
// One-pulse circuit
module onepulse (PB_debounced, clock, PB_one_pulse);
  input          PB_debounced;
  input          CLK;
  output         PB_one_pulse;
  reg            PB_one_pulse;
  reg            PB_debounced_delay;

  always @(posedge CLK) begin
    PB_one_pulse <= PB_debounced & (! PB_debounced_delay);

    PB_debounced_delay <= PB_debounced;
  end
endmodule
```

Agenda

- Announcement
- Lab 1 advanced question
- Combinational circuits
- Sequential circuits
- Debounce circuit and one-pulse circuit
- **7-segment display**

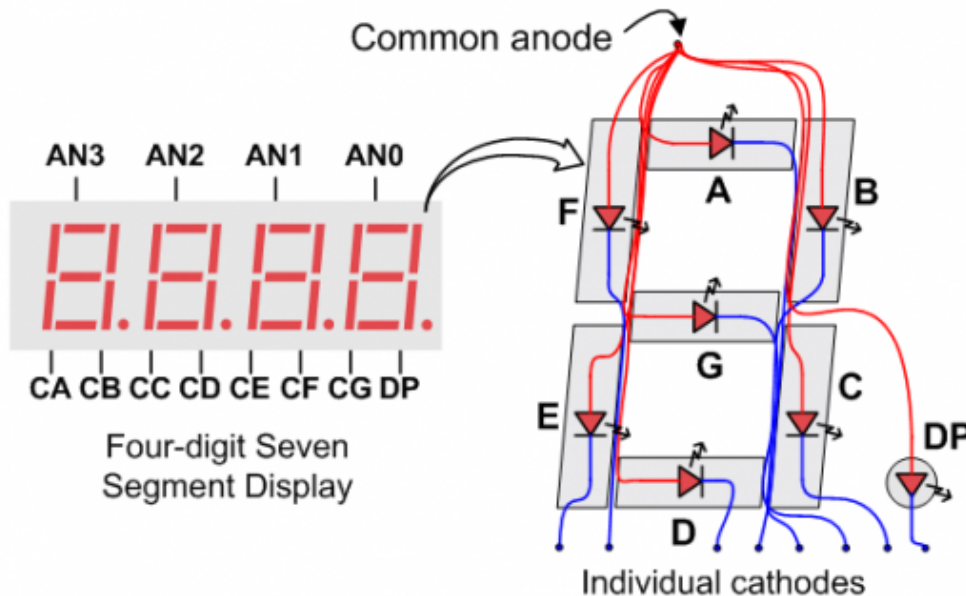
Today's class will help you:

1. Understand the difference between latch and flip-flop
2. Understand the timing diagram of latch and flip-flop
3. Understand synchronous reset
4. Understand the usage of push buttons, debounce circuit, and one-pulse circuit
5. Understand how to display four different digits on a 7-segment display

7-Segment Display Control (1/2)

- Four enable signals for the four display units
 - **AN[3:0]** correspond to **the four digits**
 - **AN** is used to **ENABLE** one of **the four digits**
 - Set one bit of AN[3:0] to **LOW** to illuminate one digit

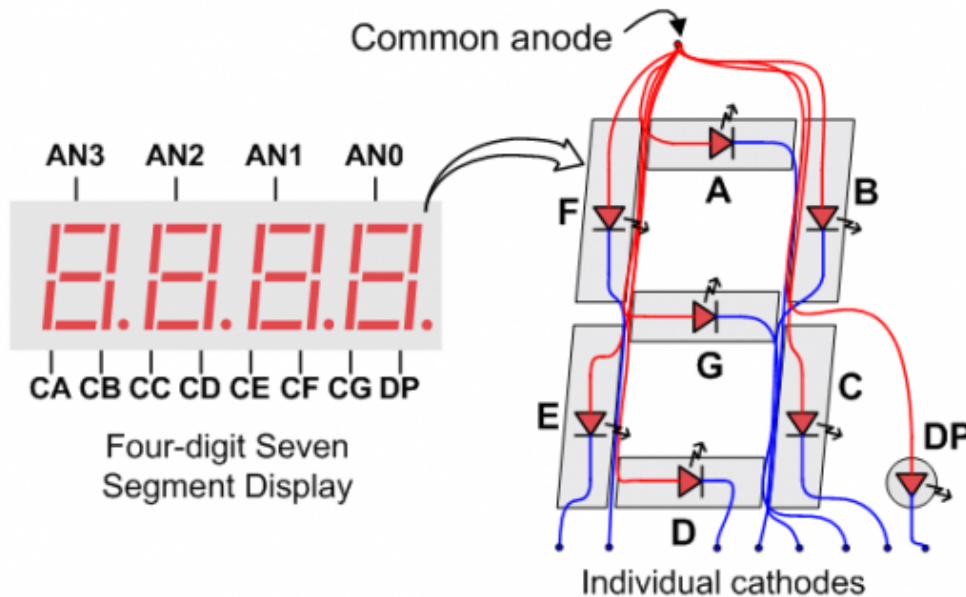
AN Bit	FPGA Pin
3	W4
2	V4
1	U4
0	U2



Digit	AN[3:0]
3	0111
2	1011
1	1101
0	1110

7-Segment Display Control (2/2)

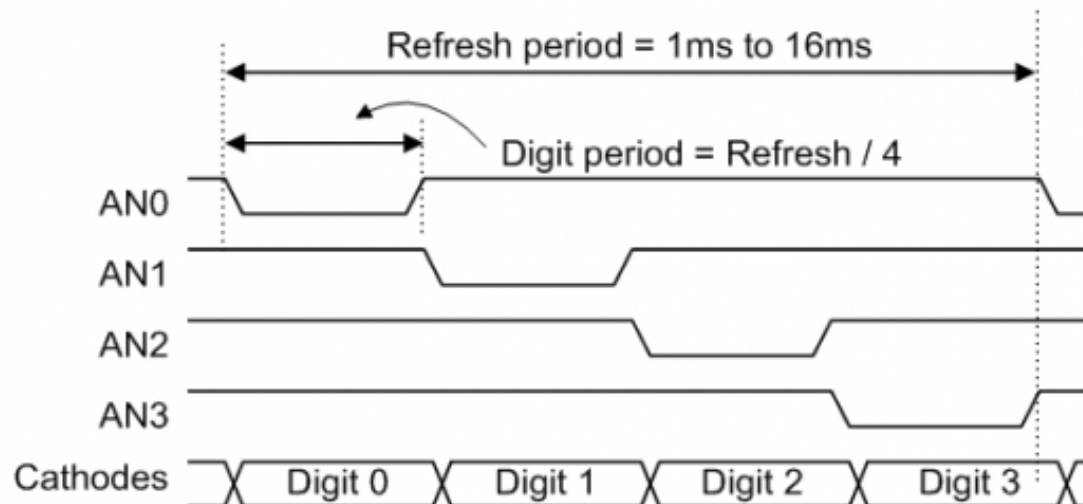
- Eight signals for the segments and dot
 - **seg[0:6]** in the XDC file correspond to **segments A~G** shown below
 - **dp** in the XDC file corresponding to **DP (dot)**
 - To illuminate a segment, set its value to **LOW**



Symbol	FPGA Pin
CA	W7
CB	W6
CC	U8
CD	V8
CE	U5
CF	V5
CG	U7
DP	V7

Display Four Digits Concurrently

- Time multiplexing
- Use a **mux** and **clock divider**
 - **Mux** to select the digit to display (i.e. set which **AN** bit to low)
 - **Clock divider** to display at the right refresh frequency (**1ms to 16ms**)
- The clock provided by Basys 3 is **100MHz**
 - That is, **10ns** per clock cycle
 - Generate a clock with a frequency of **$1/2^{17}$ clk**



*Pictures cited from www.xilinx.com for education purpose only

Thank you for your attention!



*Taken at Yosemite Valley Village at Yosemite, California, USA.
This picture is taken by Chun-Yi Lee himself, who is also a fan of photography