



EECS 2070 02 Fall 2022

Digilent Basys3 FPGA Board Part 2

黃稚存

Chih-Tsun Huang

cthuang@cs.nthu.edu.tw



國立清華大學
NATIONAL TSING HUA UNIVERSITY

資訊工程學系
Computer Science

Lecture 06



聲明

- ◎ 本課程之內容 (包括但不限於教材、影片、圖片、檔案資料等)，僅供修課學生個人合理使用，非經授課教師同意，不得以任何形式轉載、重製、散布、公開播送、出版或發行本影片內容 (例如將課程內容放置公開平台上，如 Facebook, Instagram, YouTube, Twitter, Google Drive, Dropbox 等等)。如有侵權行為，需自負法律責任。



Outline

- ① Seven-Segment Display
- ① Pushbuttons & Debounce Circuit
- ① One-Pulse Generator
- ① Summary

→ Reference: Digilent Basys3™ FPGA Board Reference Manual

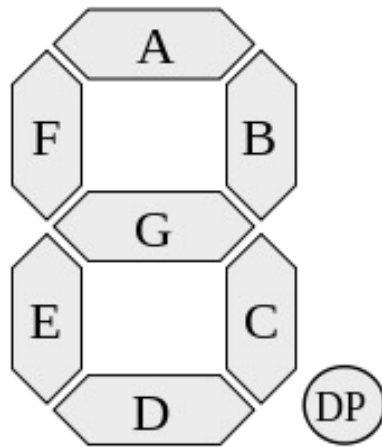


Seven-Segment Display

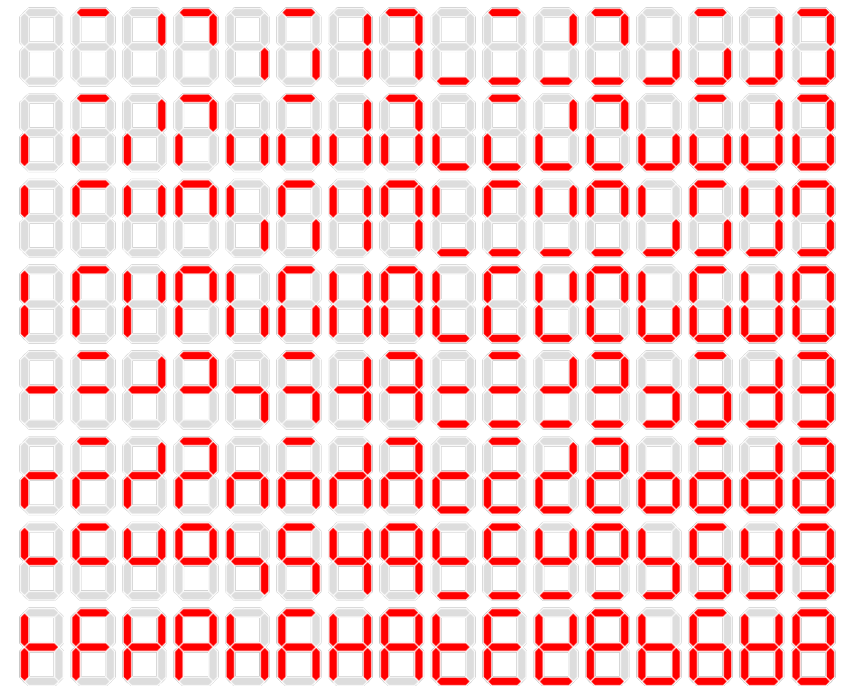


Seven Segment

- ◎ Four-digit common anode (共陽) seven-segment LED display
- ◎ 128 possible patterns on a digit
 - ◆ Ten decimal patterns are the most useful

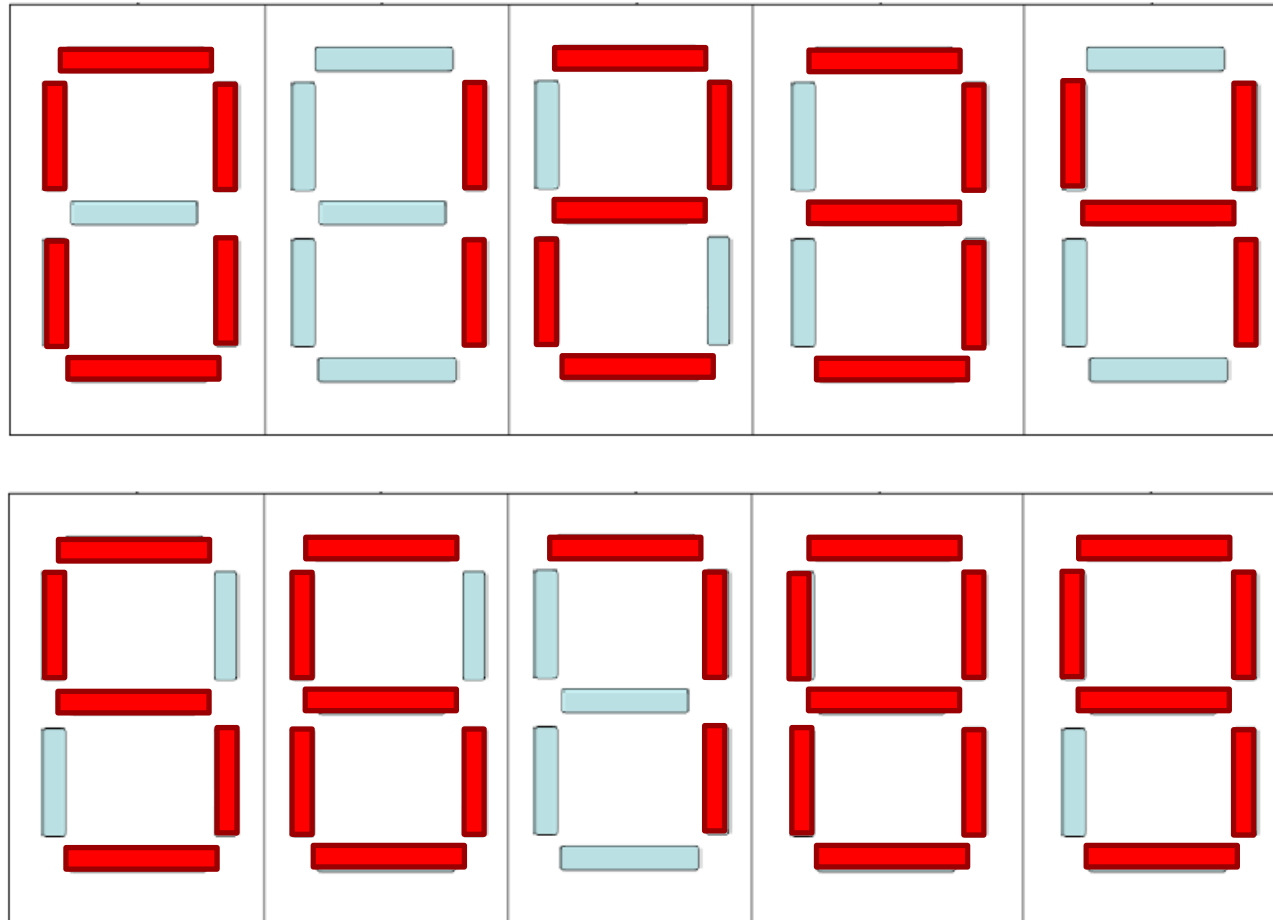


Source: Wikipedia



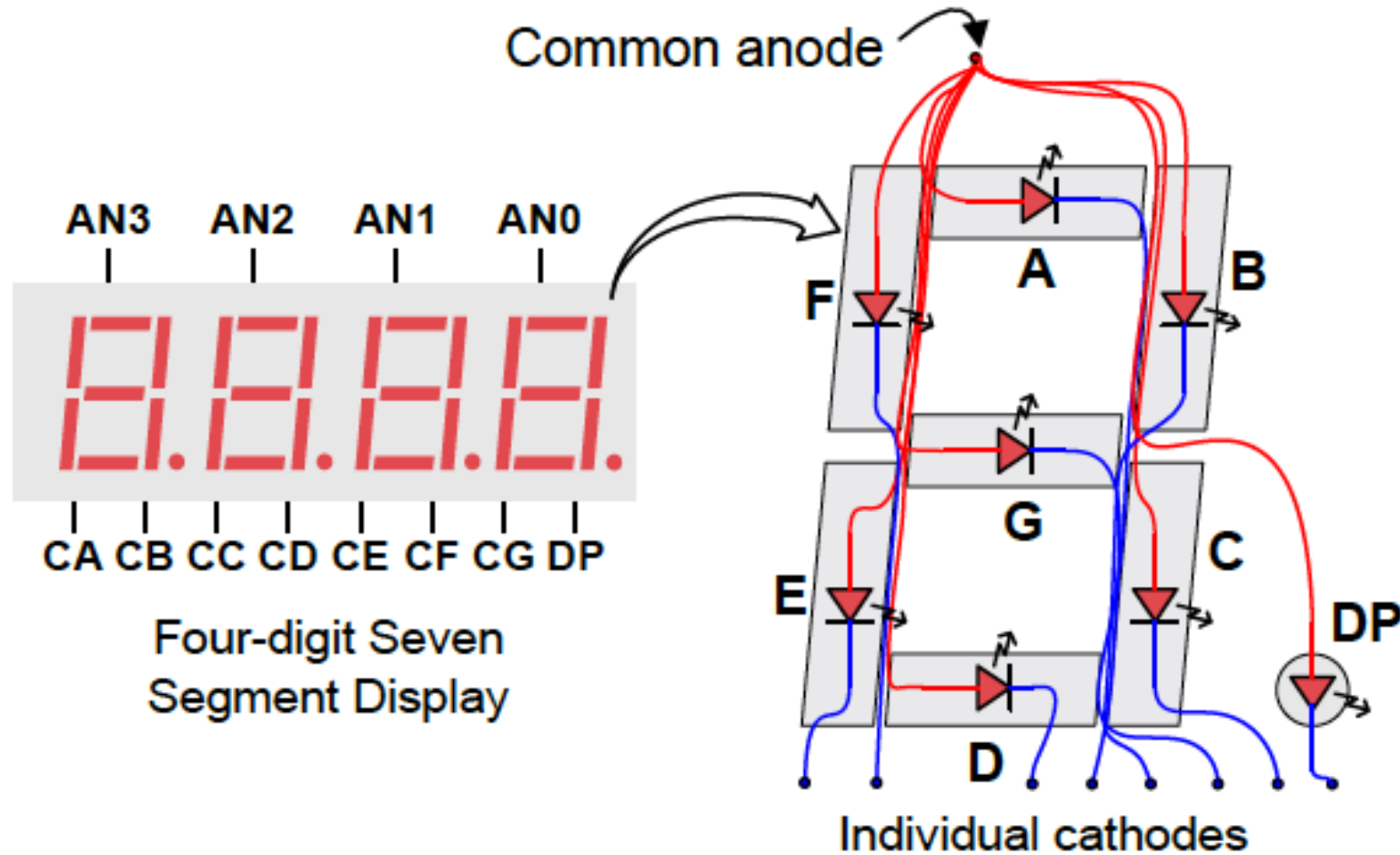


Decimal Number Code



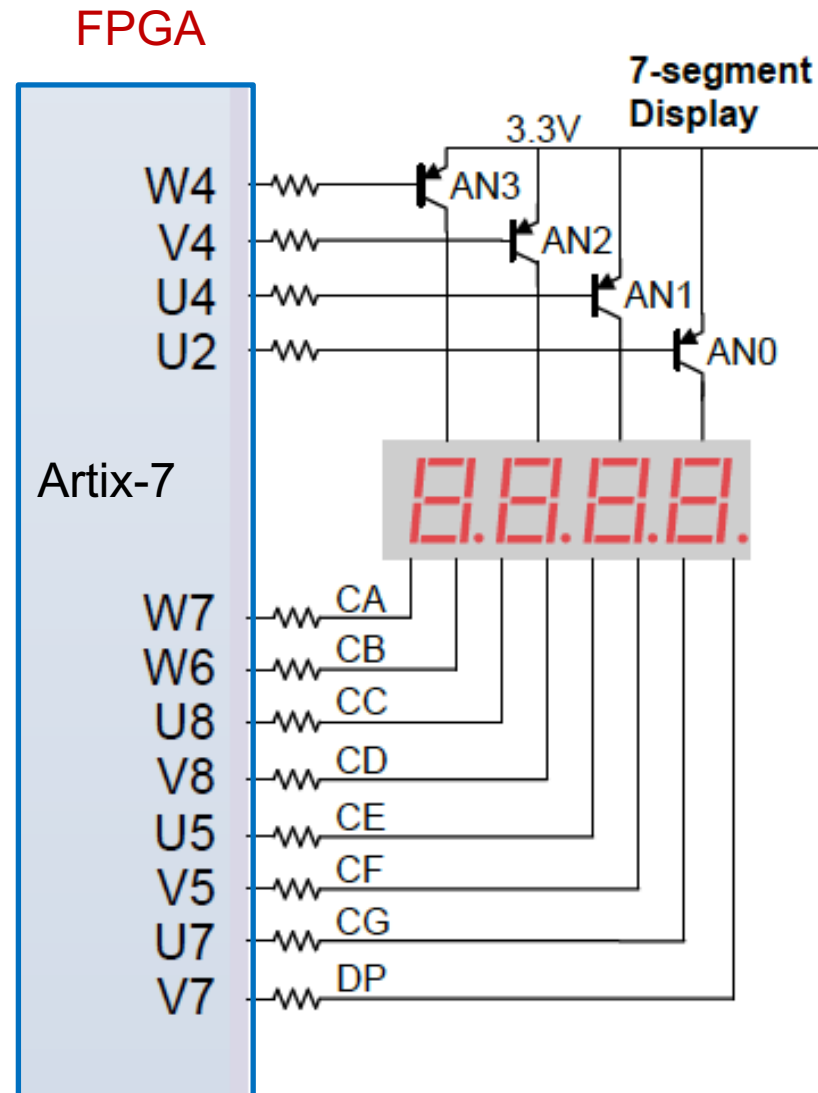


Common Anode Circuit Node





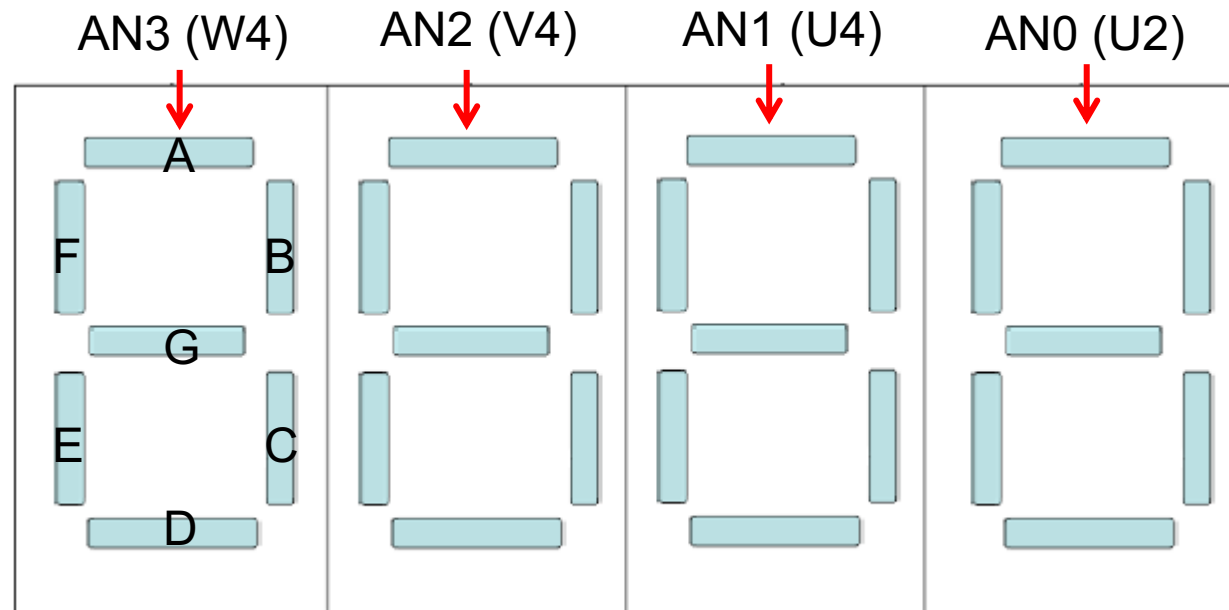
IO Mapping and Interface





Control of Seven-Segment Display (1/4)

- The cathodes of similar segments on all four displays are connected to the same FPGA pins

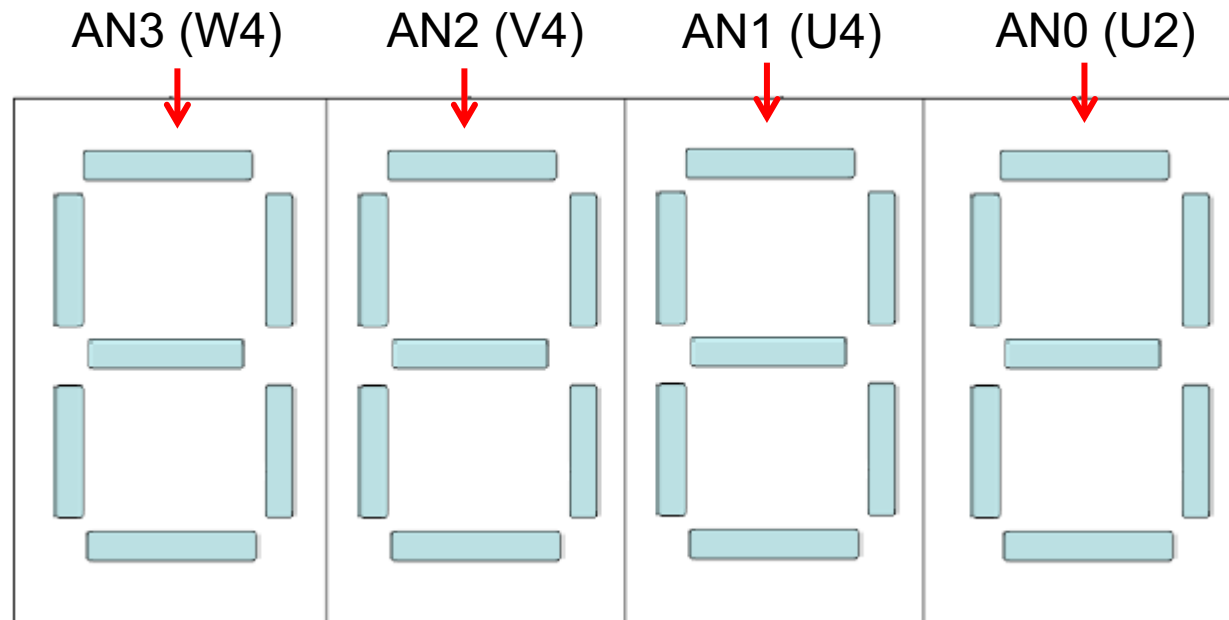


symbol	A	B	C	D	E	F	G
FPGA pin	W7	W6	U8	V8	U5	V5	U7



Control of Seven-Segment Display (2/4)

- 7 pins to control each 7-segment display
- 4 pins to choose which 7-segment to display
- Each device is negative enabled (low enabled)





Control of Seven-Segment Display (3/4)

{AN3 AN2 AN1 AN0}

Input: $D_3 D_2 D_1 D_0 = 1110$
 $A B C D E F G = 0000001$

What is the response?

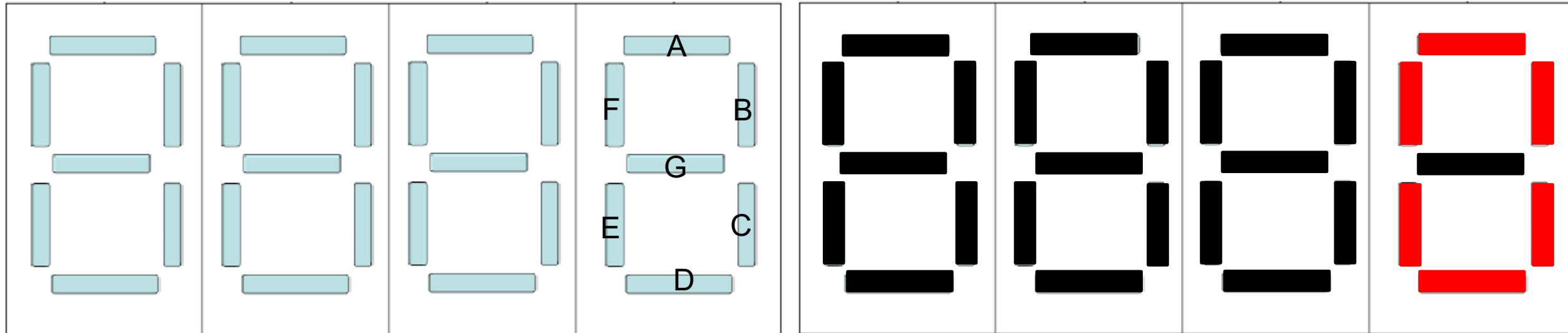


1

1

1

0

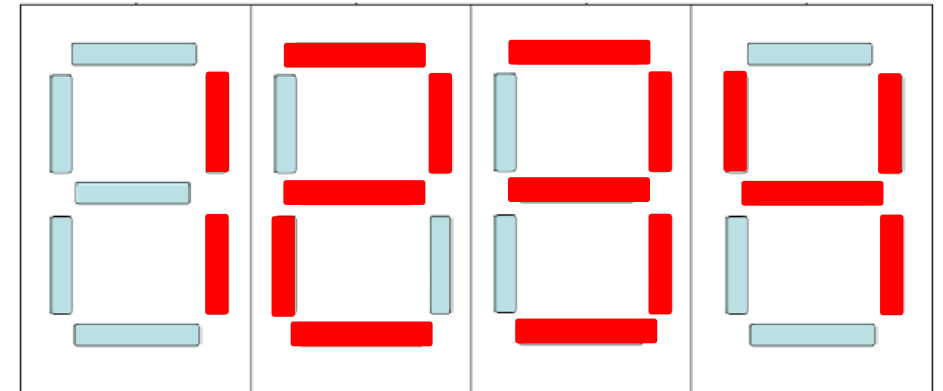




Control of Seven-Segment Display (4/4)

- Question: How to display four different numbers (e.g., **1234**)?
- Solution: **Time multiplexing**

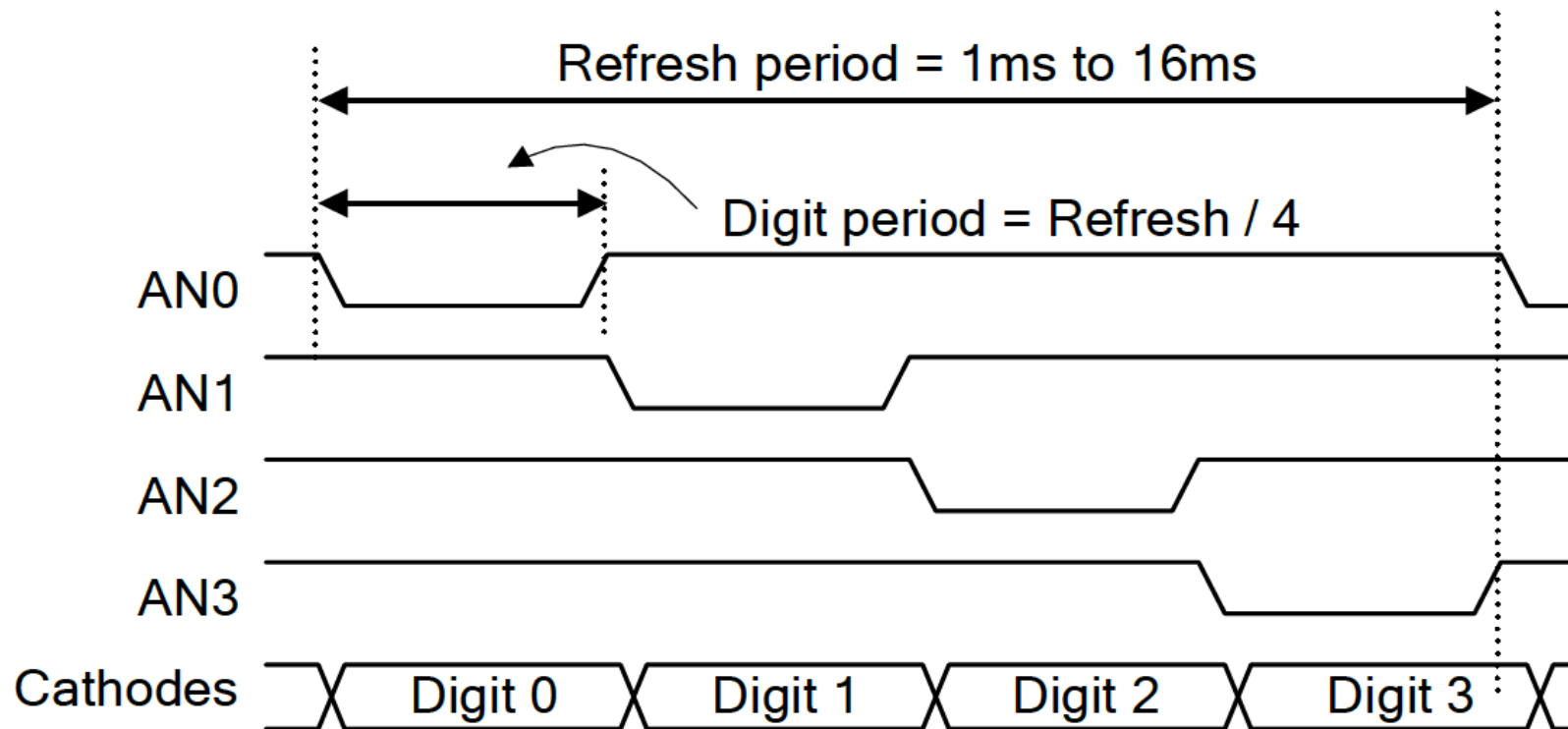
	D ₃	D ₂	D ₁	D ₀
Display '4':	1	0	0	1
Display '3':	0	0	0	1
Display '2':	0	0	1	0
Display '1':	1	0	0	1





Timing Diagram to Scan The Four Digits

- The update (or refresh) rate should be $> 45\text{Hz}$
 - ◆ Or a flicker occurs
 - ◆ Typically, the four digits should be driven once every 1 to 16ms
 - Refresh frequency of $\sim 1\text{KHz}$ to 60Hz





Control Example (Only Partial Code Segment)

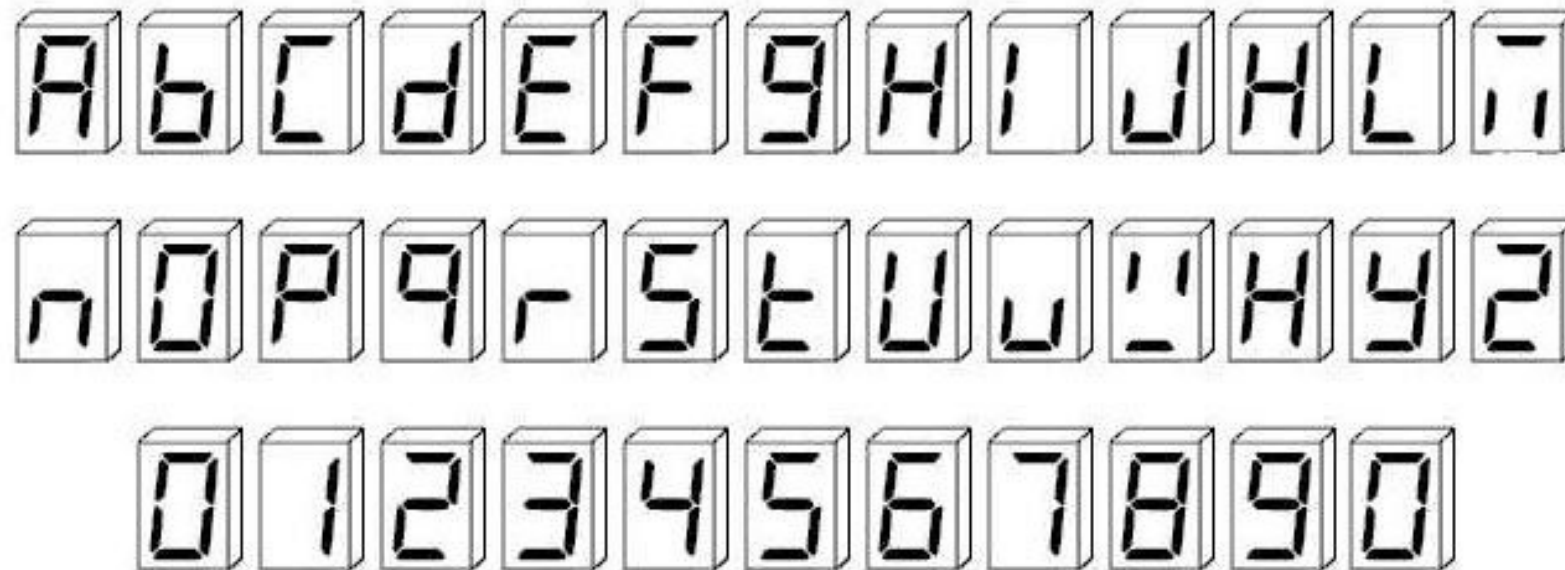
```
always @(posedge clk) begin
  case (DIGIT)
    4'b1110: begin
      value = BCD1;
      DIGIT = 4'b1101;
    end
    4'b1101: begin
      value = BCD2;
      DIGIT = 4'b1011;
    end
    4'b1011: begin
      value = BCD3;
      DIGIT = 4'b0111;
    end
    4'b0111: begin
      value = BCD0;
      DIGIT = 4'b1110;
    end
    default: begin
      value = BCD0;
      DIGIT = 4'b1110;
    end
  endcase
end
```

```
always @* begin
  case (value)
    GFE_DCBA
    4'd0: DISPLAY = 7'b100_0000;
    4'd1: DISPLAY = 7'b111_1001;
    4'd2: DISPLAY = 7'b010_0100;
    4'd3: DISPLAY = 7'b011_0000;
    4'd4: DISPLAY = 7'b001_1001;
    4'd5: DISPLAY = 7'b001_0010;
    4'd6: DISPLAY = 7'b000_0010;
    4'd7: DISPLAY = 7'b111_1000;
    4'd8: DISPLAY = 7'b000_0000;
    4'd9: DISPLAY = 7'b001_0000;
    default: DISPLAY = 7'b111_1111;
  endcase
end
```



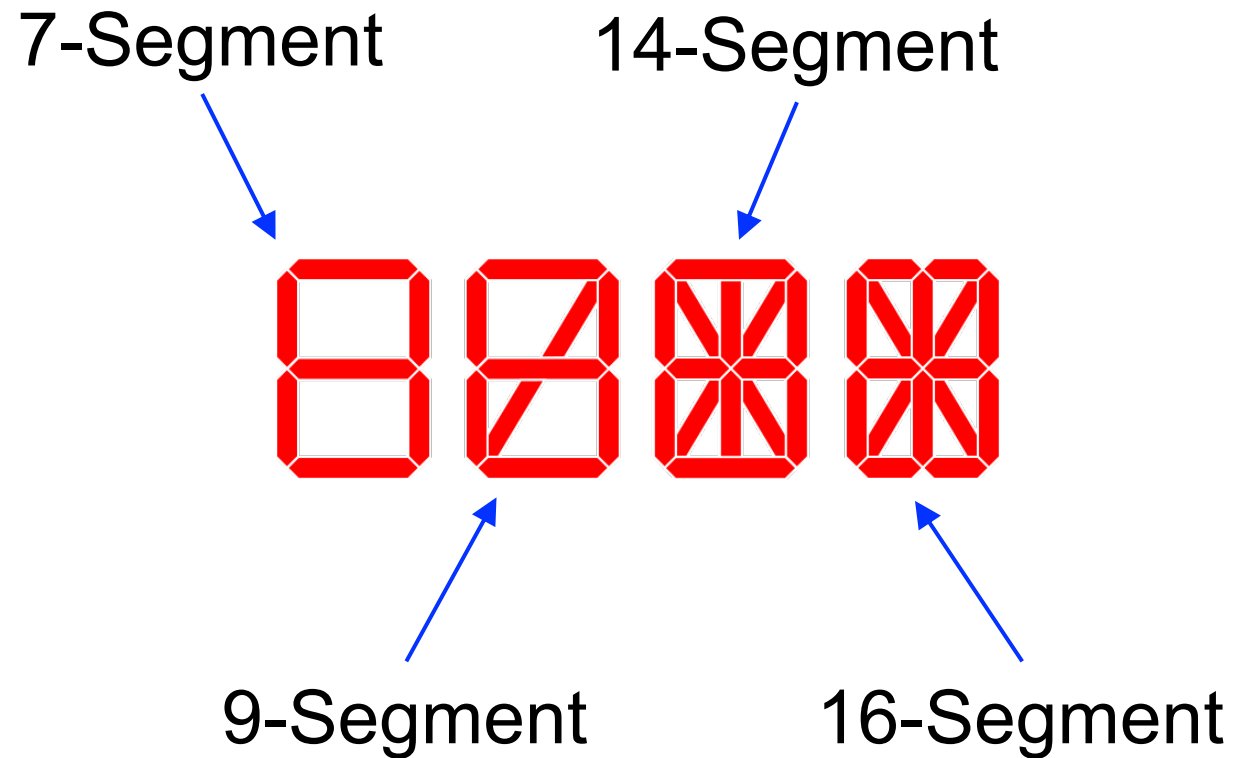
Discussion

- ⦿ The sample code is not necessary the best style
- ⦿ Do not forget the decimal point: DP (V7)
 - ◆ Can you control it on Basys 3?
- ⦿ How about the middle “colon”?
- ⦿ How about characters other than decimal digits?



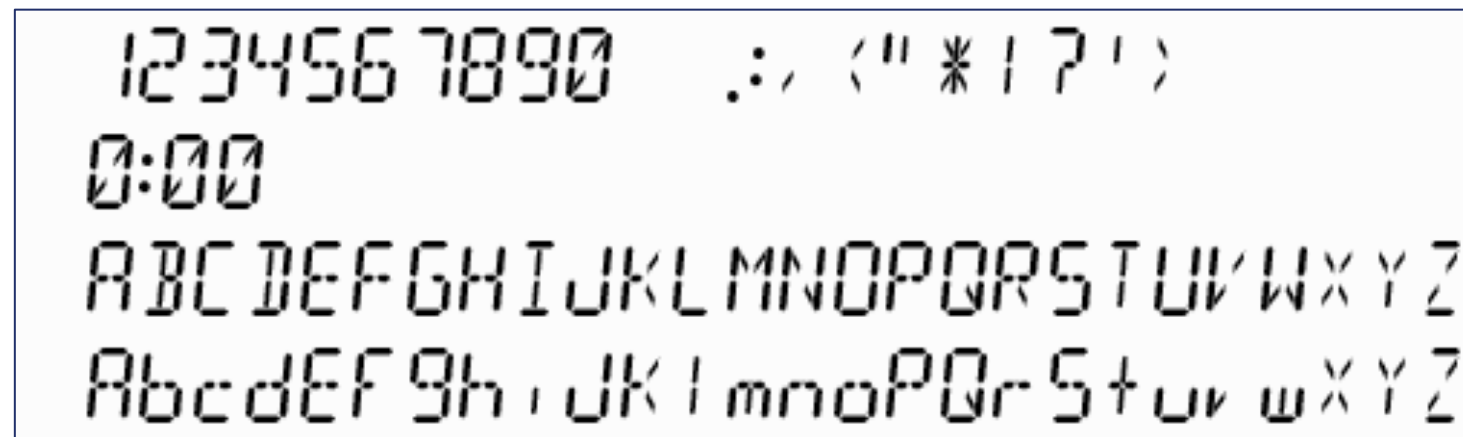


Other Variations





14-Segment Display Codes





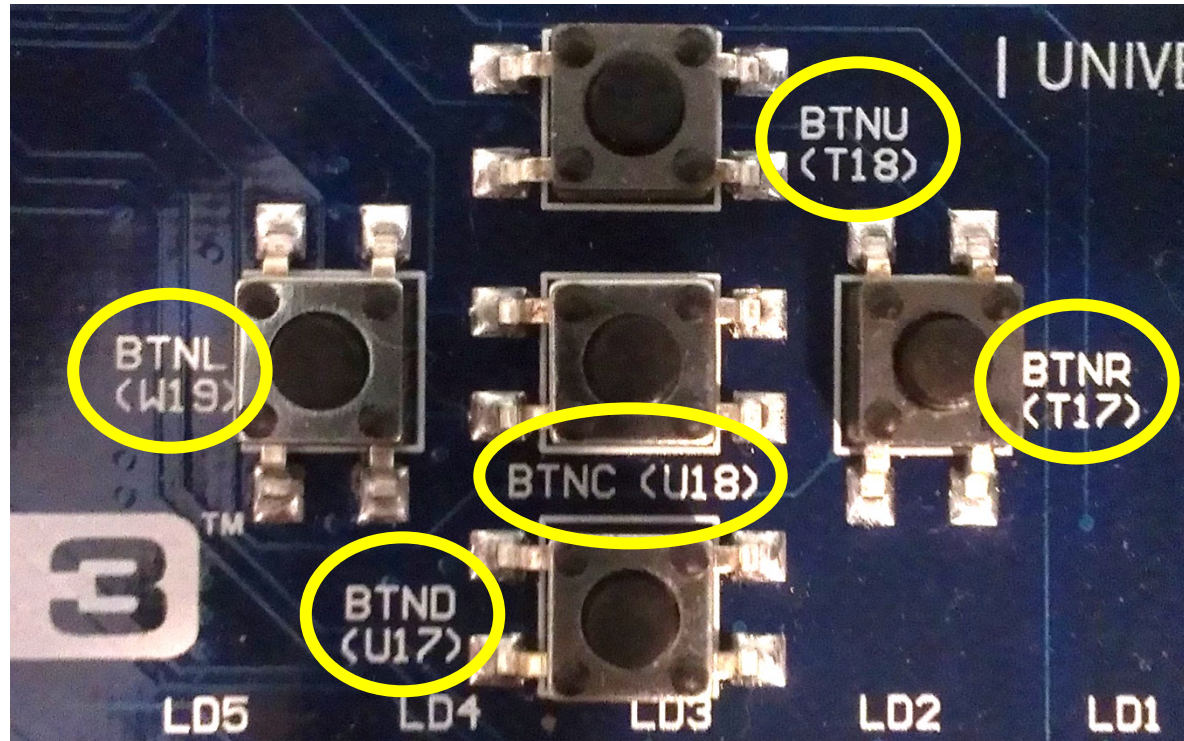
Pushbuttons & Debounced Circuit



Five Pushbuttons (1/2)

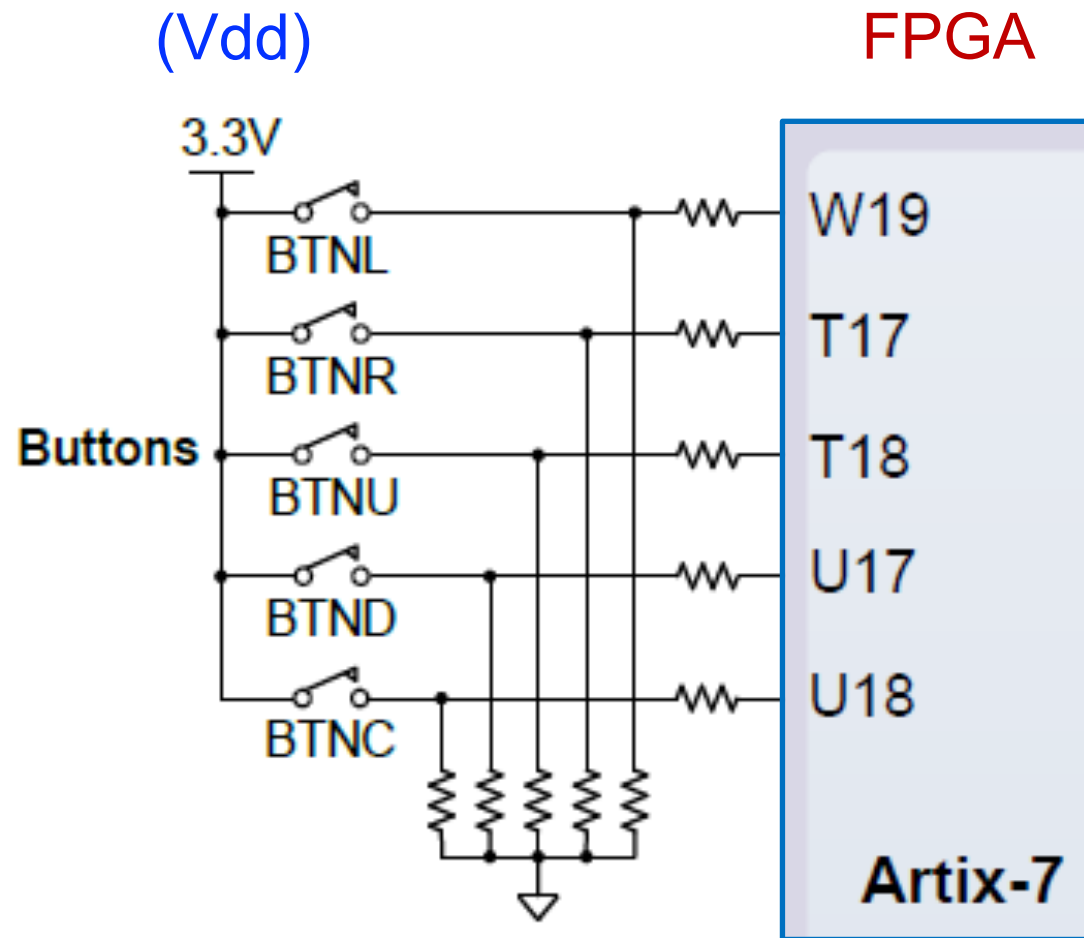
- Arranged in a plus-sign configuration in the demo board
- Momentary** switches
 - Normally generate a **low (0)** output when they are at rest
 - High (1)** output only when they are pressed

Symbol	Pin
BTNU	T18
BTND	U17
BTNL	W19
BTNR	T17
BTNC	U18





Five Pushbuttons (2/2)



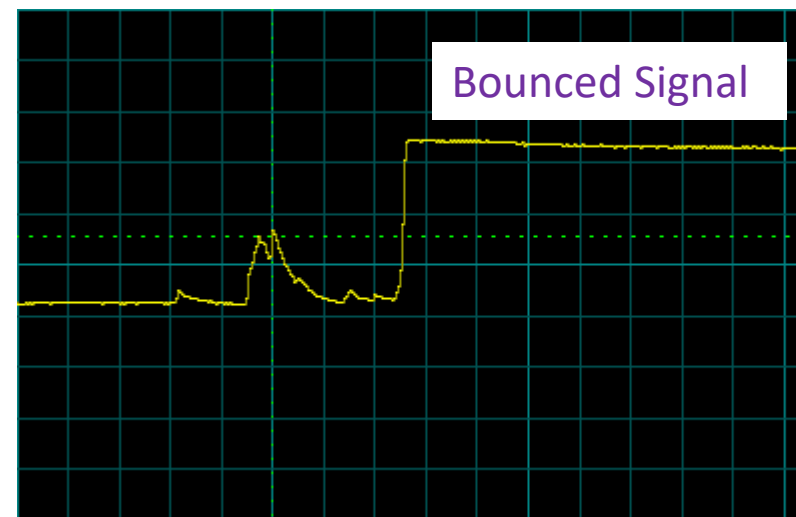
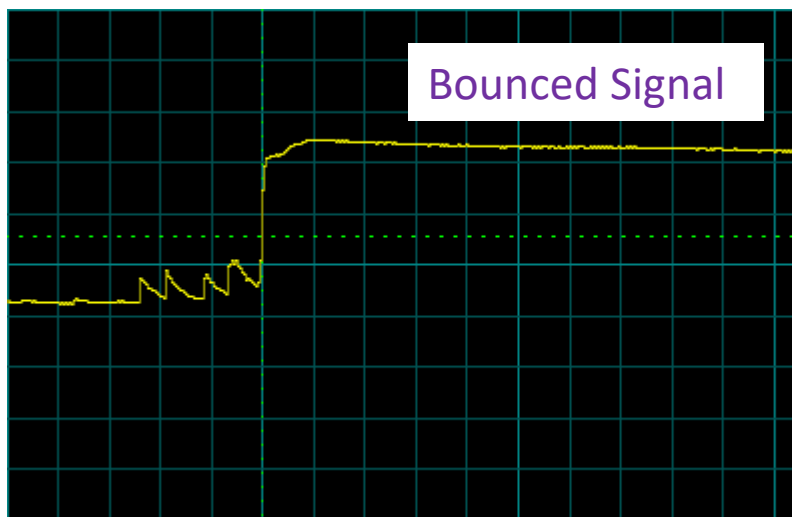
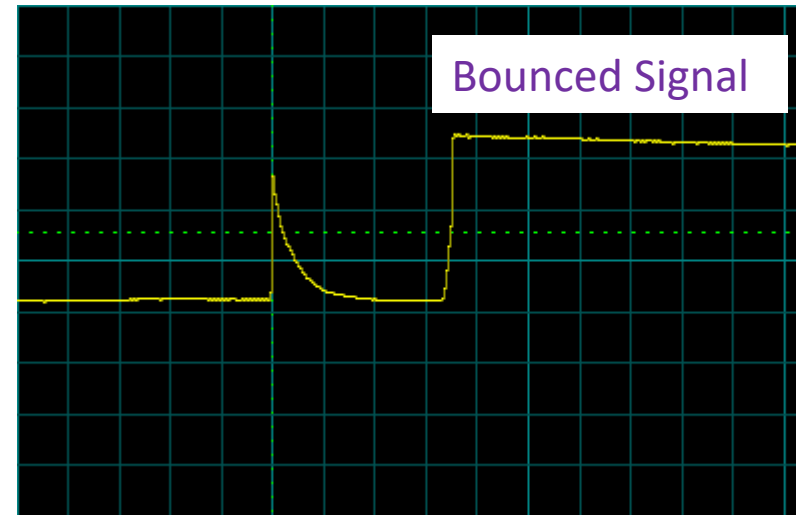
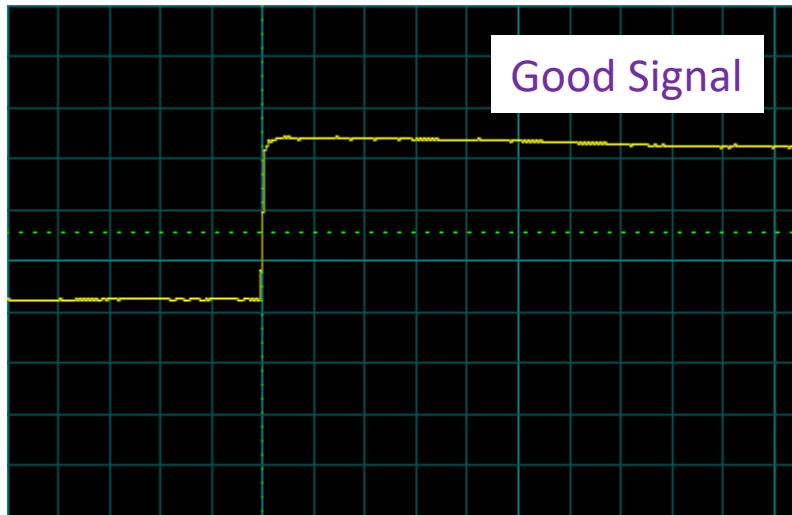


Switch Contact Bounce

- ◎ Pushbutton contains a metal spring
 - ◆ When it is pressed and released, the switch contact will bounce several times before stabilizing
 - ◆ A random number of unwanted signal pulses
 - ◆ They are in the μs range
 - ◆ But the FPGA is sensitive to pulses down to ns range

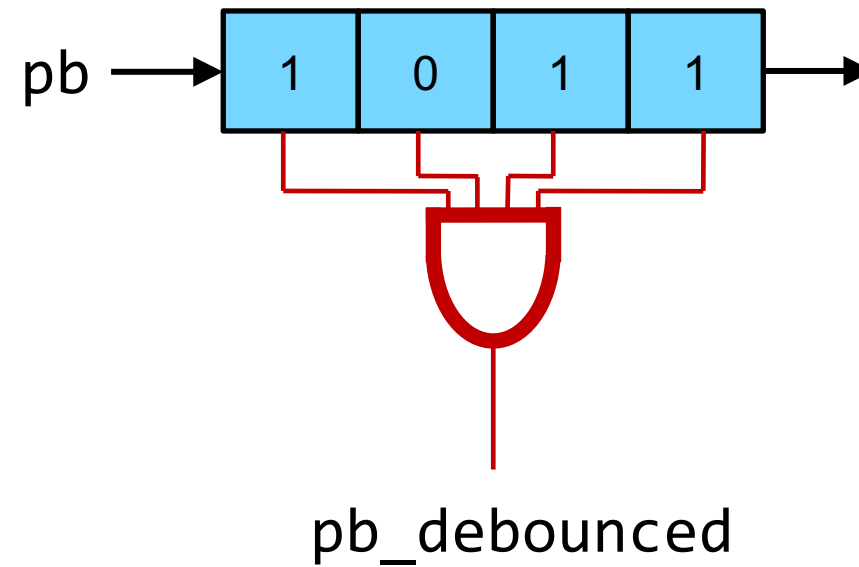


Possible Ground Bounces When Pushing A Button





Pushbutton Debounced





Verilog Code of Debounced Circuit

```
module debounce (  
    output pb_debounced, // output after being debounced  
    input  pb,            // input from a pushbutton  
    input  clk  
);  
    reg [3:0] shift_reg; // use shift_reg to filter the bounce  
  
    always @(posedge clk)  
        begin  
            shift_reg[3:1] <= shift_reg[2:0];  
            shift_reg[0] <= pb;  
        end  
  
    assign pb_debounced = ((shift_reg == 4'b1111) ? 1'b1 : 1'b0);  
  
endmodule
```

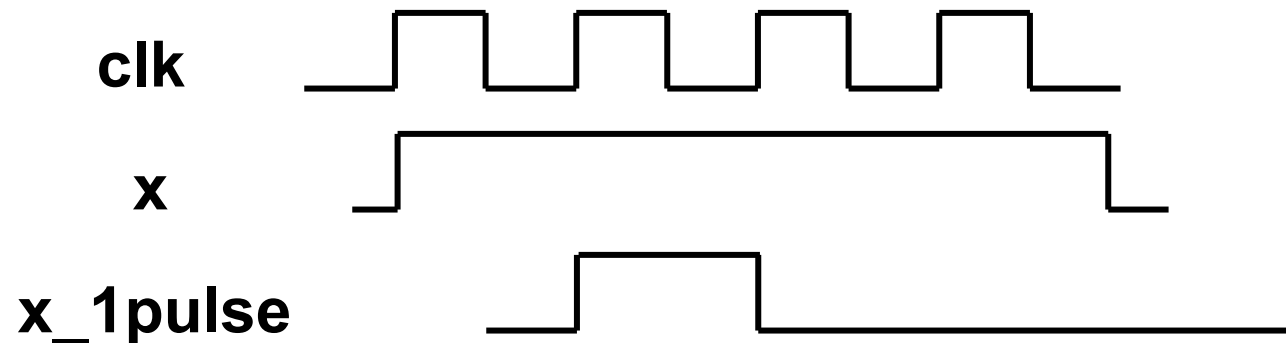



One-Pulse Generator



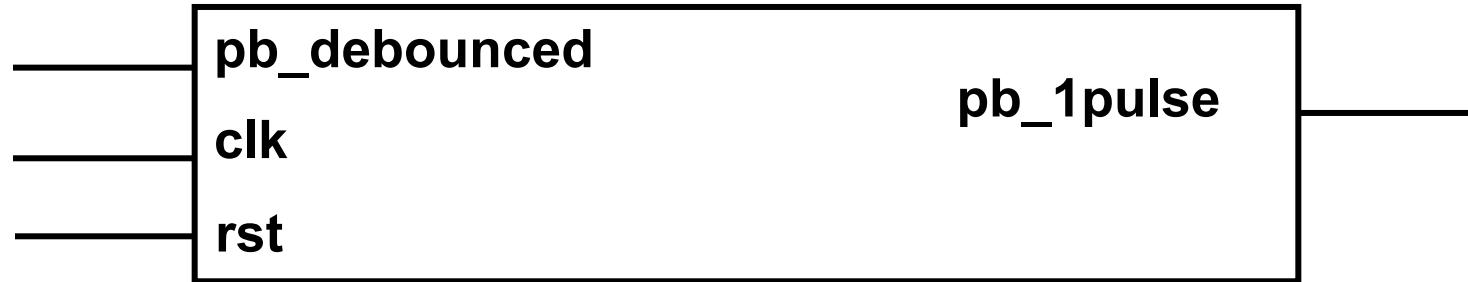
One-Pulse Generator

- When a pushbutton is pressed for a short moment
 - ◆ The time to turn on the switch is usually much longer (in ms range) than one clock period (in μs or ns range)
 - ◆ The circuit will “see” that we are supplying a string of ones as the input
- A one-pulse circuit generates only a one-clock-period-long pulse every time a pushbutton is pressed
 - ◆ Independent of the actual time period that the pushbutton was pressed





Block Diagram of One-Pulse Generator



Inputs

- ◆ pb_debounced: debounced pushbutton signal
- ◆ clk: the same clock to the circuit
- ◆ rst: reset

Output

- ◆ pb_1pulse: high for only one clock cycle when the pushbutton is pressed



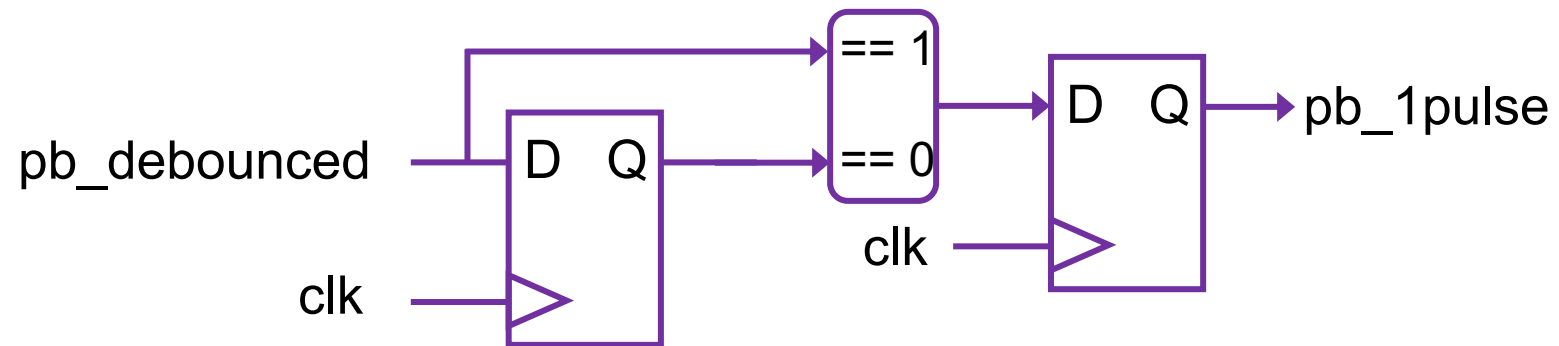
Block Diagram and Waveforms

pb_debounced

pb_debounced_delay

\sim pb_debounced_delay

pb_1pulse



pb_debounced
pb_1pulse
clk





Sample Verilog Code

(Do you need the reset signal?)

```
module onepulse (  
    input wire pb_debounced,  
    input wire clk,  
    output reg pb_1pulse  
);  
    reg pb_debounced_delay;  
  
    always @(posedge clk) begin  
        if (pb_debounced == 1'b1 & pb_debounced_delay == 1'b0)  
            pb_1pulse <= 1'b1;  
        else  
            pb_1pulse <= 1'b0;  
  
        pb_debounced_delay <= pb_debounced;  
    end  
endmodule
```



Alternative One-Pulse Generator

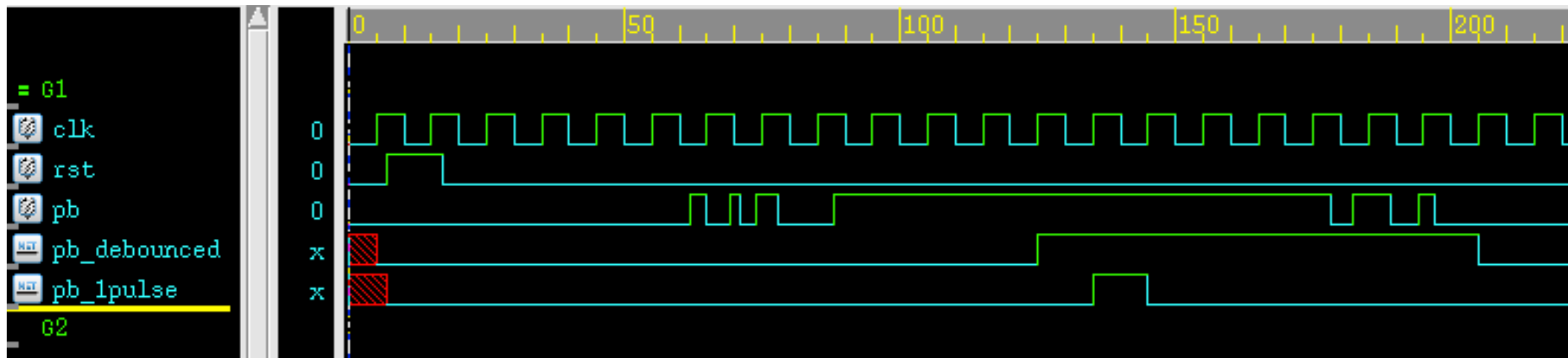
```
module onepulse (  
    input wire rst,  
    input wire clk,  
    input wire pb_debounced,  
    output reg pb_1pulse  
);  
    // internal registers  
    reg pb_1pulse_next;  
    reg pb_debounced_delay;  
  
    always @* begin  
        pb_1pulse_next = pb_debounced  
            & ~pb_debounced_delay;  
    end
```

```
    always @(posedge clk, posedge rst)  
    begin  
        if (rst == 1'b1) begin  
            pb_1pulse <= 1'b0;  
            pb_debounced_delay <= 1'b0;  
        end else begin  
            pb_1pulse <= pb_1pulse_next;  
            pb_debounced_delay <= pb_debounced;  
        end  
    end  
endmodule
```



Simulation of One-Pulse Generator

- You may observe the waveform timing
 - ◆ Sequential behavior:
A DFF introduces one-cycle delay





Summary

- Refresh rate of the 7-segment → smoothness of the display
- Every input from the pushbutton
 - ◆ Should be properly processed with the debouncing **and/or** one-pulse converters
 - ◆ Except the reset signal (why?)
- Again, clock rate of the debounce/on-pulse circuits → smoothness of the input response