Lab. 07

Logic Design Lab.

Spring 2023

Prof. ChangGun Lee

(cglee@snu.ac.kr)

TA. Seonghyeon Park

TA. Jihwan Kim

TA. Hoyong Lee

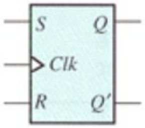
(rubis.ld.ta@gmail.com)

Contents

- **Synchronous Binary Counter**
- **Frequency Divider**
- **Implementation on Logic Design Board (Review)**
- **Two Digit Counter**

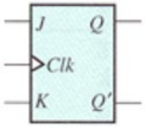
Filp Flops (Recap)

SR



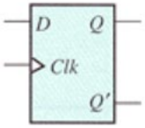
S	R	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	NA

JK



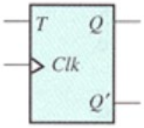
J	K	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	Q′

D

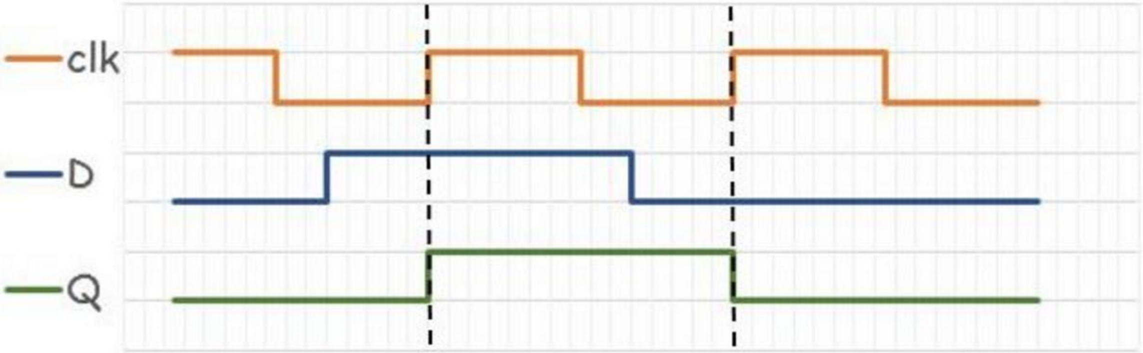


D	Q(next)
0	0
1	1

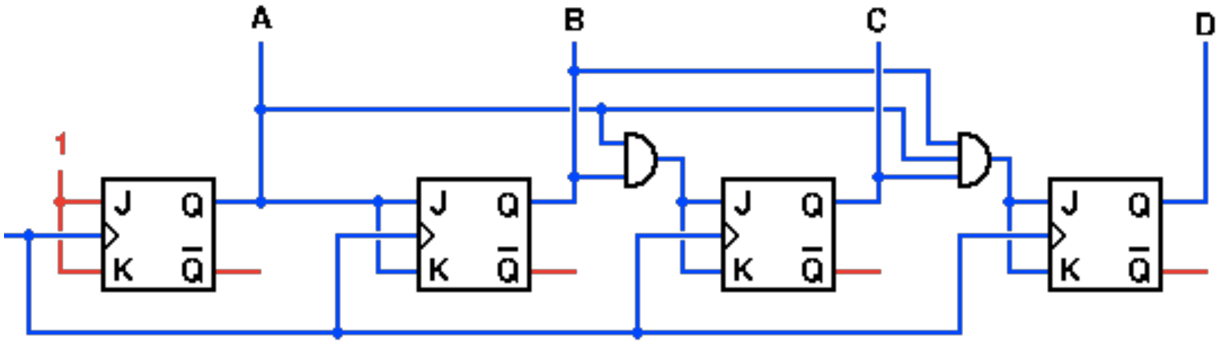
T

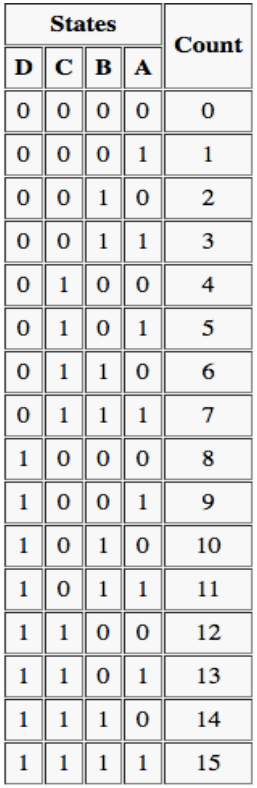


T	Q(next)
0	Q
1	Q′



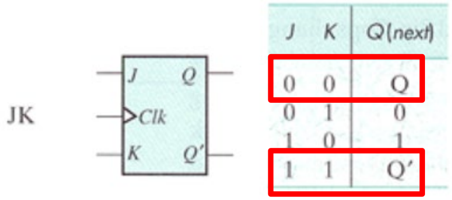
4-bit Counter – JK Filp Flop Implementation



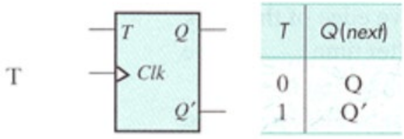


States				Count
D	C	B	A	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

- J-K flip-flop functions as a T flip-flop
  if you connect the same input to J and K.
- A will repeat 0 and 1.

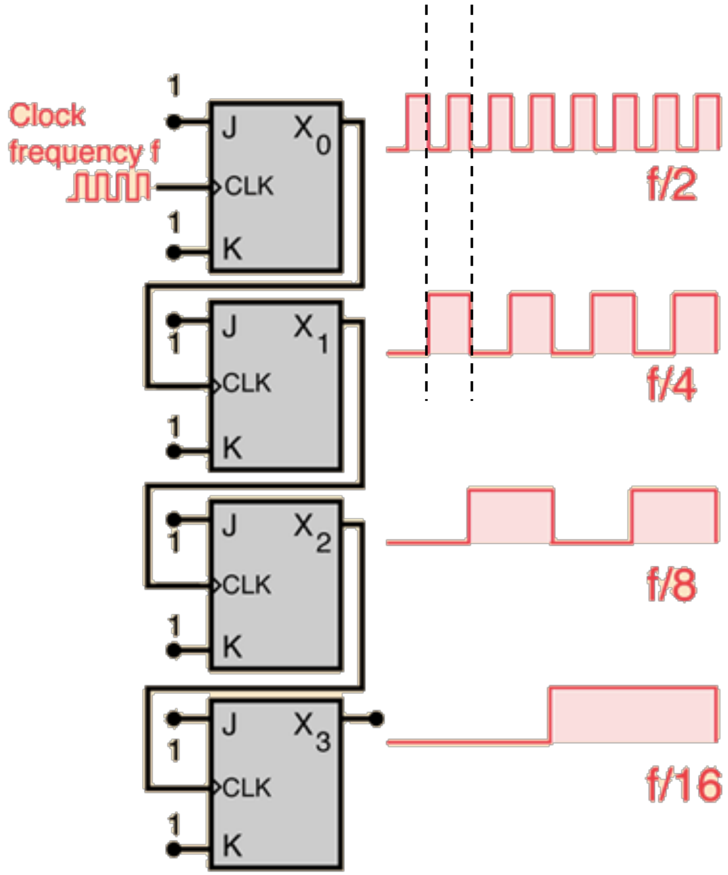


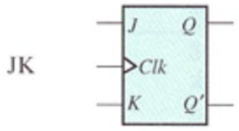
J	K	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	Q′

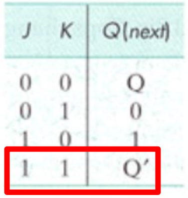


T	Q(next)
0	Q
1	Q′

Frequency Divider – JK Filp Flop Implementation





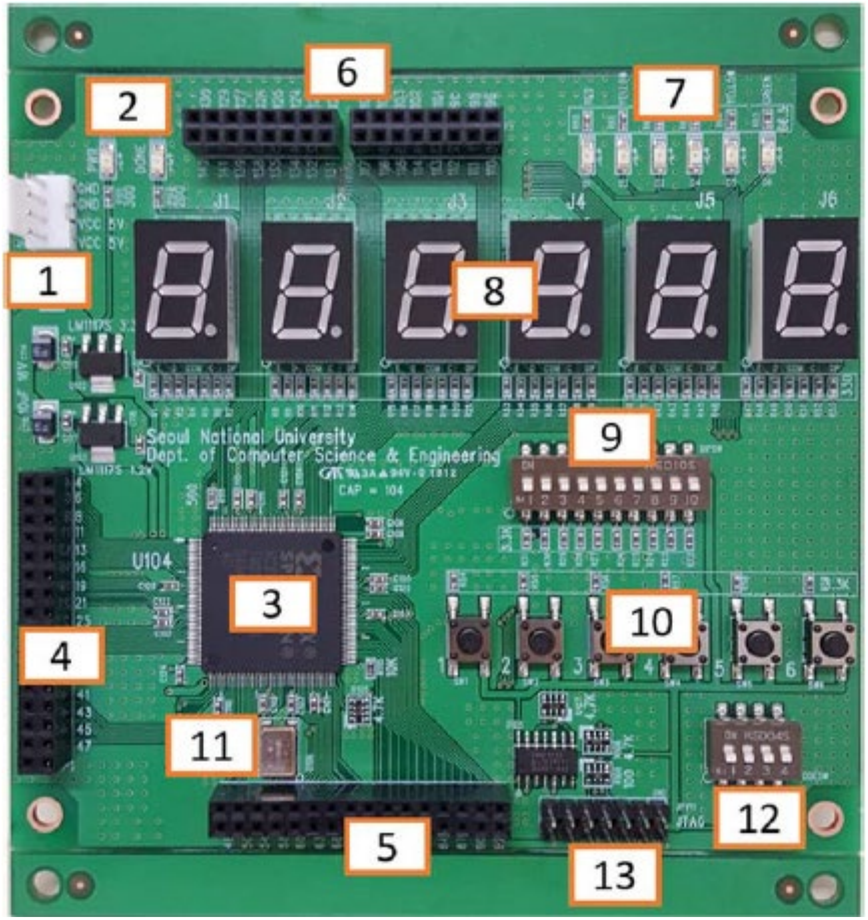


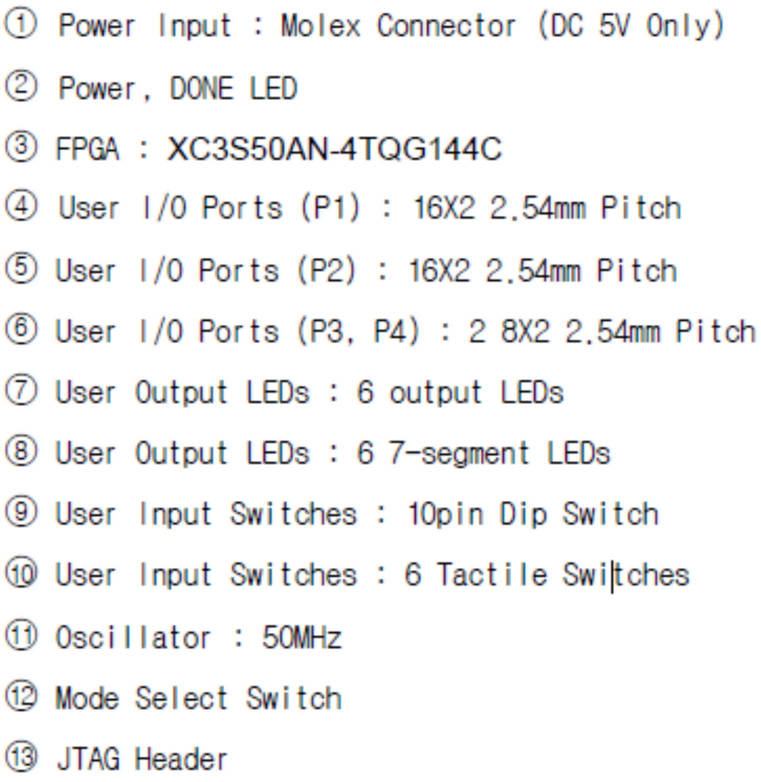
J	K	Q(next)
0	0	Q
0	1	0
1	0	1
1	1	Q′

negative edge triggered JK Flip Flop

Implementation on Logic Design Board (Review)

SNU Logic Design Board


6
2
7
1
8
9
Seoul National University
Dept. of Computer Science & Engineering
U104
3
4
10
11
12
5
13



① Power Input : Molex Connector (DC 5V Only)

② Power, DONE LED

③ FPGA : XC3S50AN-4TQG144C

④ User I/O Ports (P1) : 16X2 2.54mm Pitch

⑤ User I/O Ports (P2) : 16X2 2.54mm Pitch

⑥ User I/O Ports (P3, P4) : 2 8X2 2.54mm Pitch

⑦ User Output LEDs : 6 output LEDs

⑧ User Output LEDs : 6 7-segment LEDs

⑨ User Input Switches : 10pin Dip Switch

⑩ User Input Switches : 6 Tactile Switches

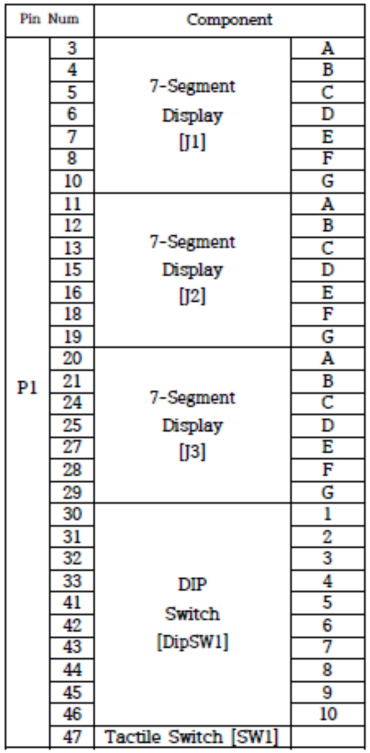
⑪ Oscillator : 50MHz

⑫ Mode Select Switch

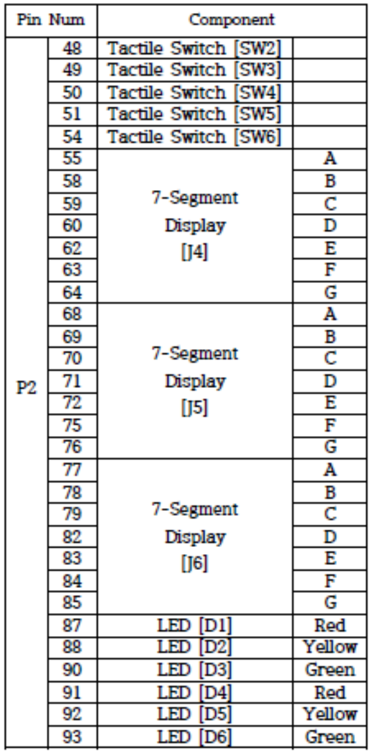
⑬ JTAG Header

SNU Logic Design Board

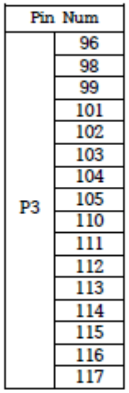
- **Pin Number Mapping**



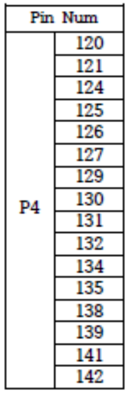
Pin Num		Component	
P1	3	7-Segment Display [J1]	A
	4		B
	5		C
	6		D
	7		E
	8		F
	10		G
	11	7-Segment Display [J2]	A
	12		B
	13		C
	15		D
	16		E
	18		F
	19		G
	20	7-Segment Display [J3]	A
	21		B
	24		C
	25		D
	27		E
	28		F
	29		G
	30	DIP Switch [DipSW1]	1
	31		2
	32		3
	33		4
	41		5
	42		6
	43		7
	44		8
	45		9
	46		10
	47	Tactile Switch [SW1]	



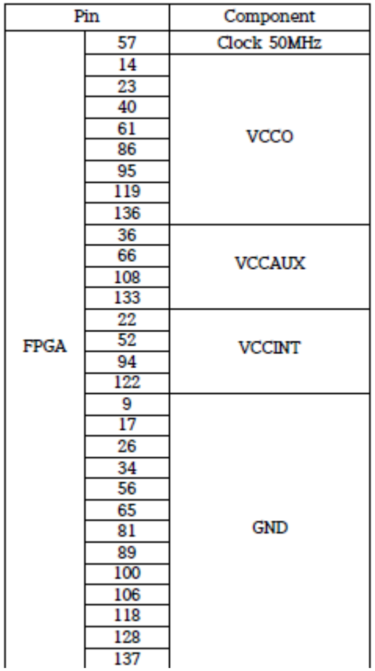
Pin Num		Component	
P2	48	Tactile Switch [SW2]	
	49	Tactile Switch [SW3]	
	50	Tactile Switch [SW4]	
	51	Tactile Switch [SW5]	
	54	Tactile Switch [SW6]	
	55	7-Segment Display [J4]	A
	58		B
	59		C
	60		D
	62		E
	63		F
	64		G
	68	7-Segment Display [J5]	A
	69		B
	70		C
	71		D
	72		E
	75		F
	76		G
	77	7-Segment Display [J6]	A
	78		B
	79		C
	82		D
	83		E
	84		F
	85		G
	87	LED [D1]	Red
	88	LED [D2]	Yellow
	90	LED [D3]	Green
	91	LED [D4]	Red
	92	LED [D5]	Yellow
	93	LED [D6]	Green



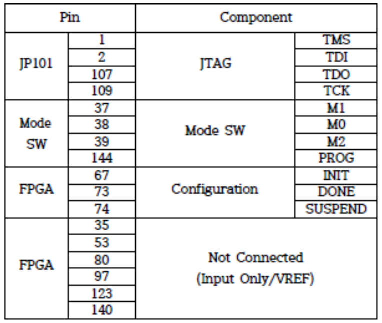
Pin Num	
P3	96
	98
	99
	101
	102
	103
	104
	105
	110
	111
	112
	113
	114
	115
	116
	117



Pin Num	
P4	120
	121
	124
	125
	126
	127
	129
	130
	131
	132
	134
	135
	138
	139
	141
	142



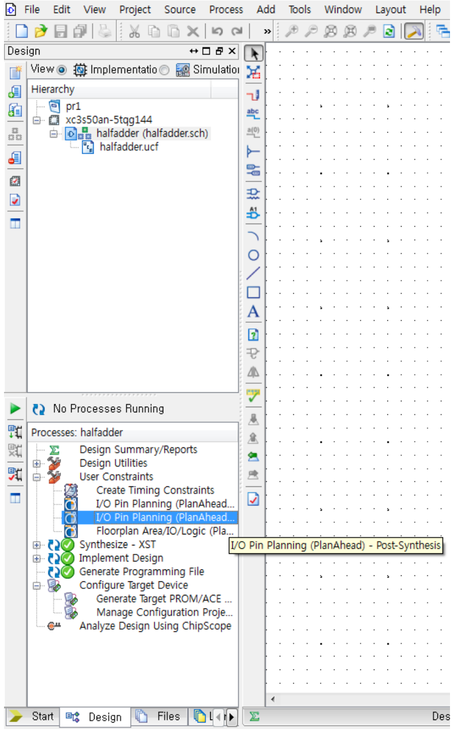
Pin		Component
FPGA	57	Clock 50MHz
	14	VCCO
	23	
	40	
	61	
	86	
	95	
	119	
	136	
	36	VCCAUX
	66	
	108	
	133	
	22	VCCINT
	52	
	94	
	122	
	9	GND
	17	
	26	
	34	
	56	
	65	
	81	
	89	
	100	
	106	
	118	
	128	
	137	

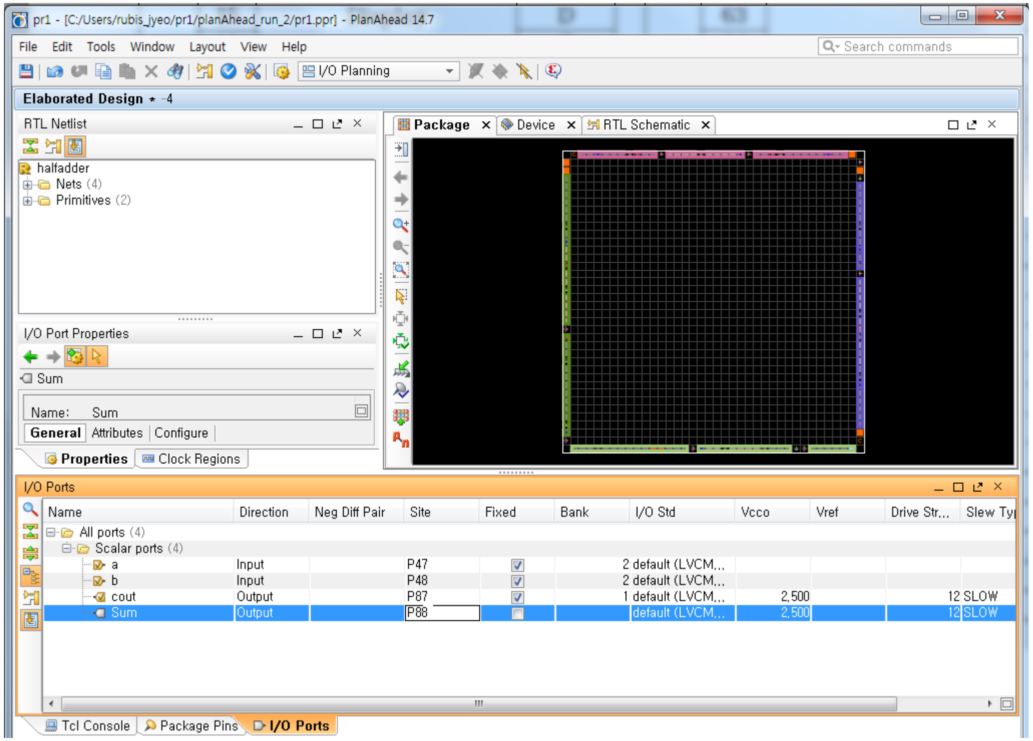


Pin		Component	
JP101	1	JTAG	TMS
	2		TDI
	107		TDO
	109		TCK
Mode SW	37	Mode SW	M1
	38		M0
	39		M2
	144		PROG
FPGA	67	Configuration	INIT
	73		DONE
	74		SUSPEND
FPGA	35	Not Connected (Input Only/VREF)	
	53		
	80		
	97		
	123		
	140		

SNU Logic Design Board

- **Assign I/O pins for the half adder**

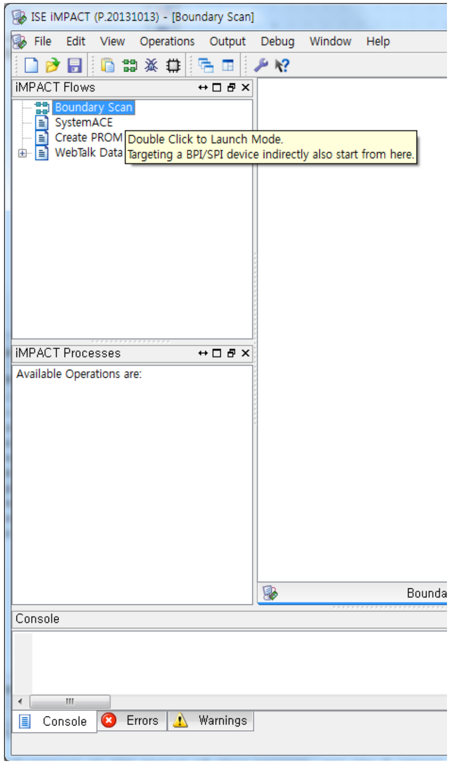

File Edit View Project Source Process Add Tools Window Layout Help
Design
View Implementation Simulation
Hierarchy
pr1
xc3s50an-5tqg144
halfadder (halfadder.sch)
halfadder.ucf
No Processes Running
Processes: halfadder
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead...
I/O Pin Planning (PlanAhead...
Floorplan Area/IO/Logic (Pla...
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE ...
Manage Configuration Proje...
Analyze Design Using ChipScope
I/O Pin Planning (PlanAhead) - Post-Synthesis
Start
Design
Files



Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Str...	Slew Typ
a	Input		P47	☑	2	default (LVCM...				
b	Input		P48	☑	2	default (LVCM...				
cout	Output		P87	☑	1	default (LVCM...	2,500		12	SLOW
Sum	Output		P88	□		default (LVCM...	2,500		12	SLOW

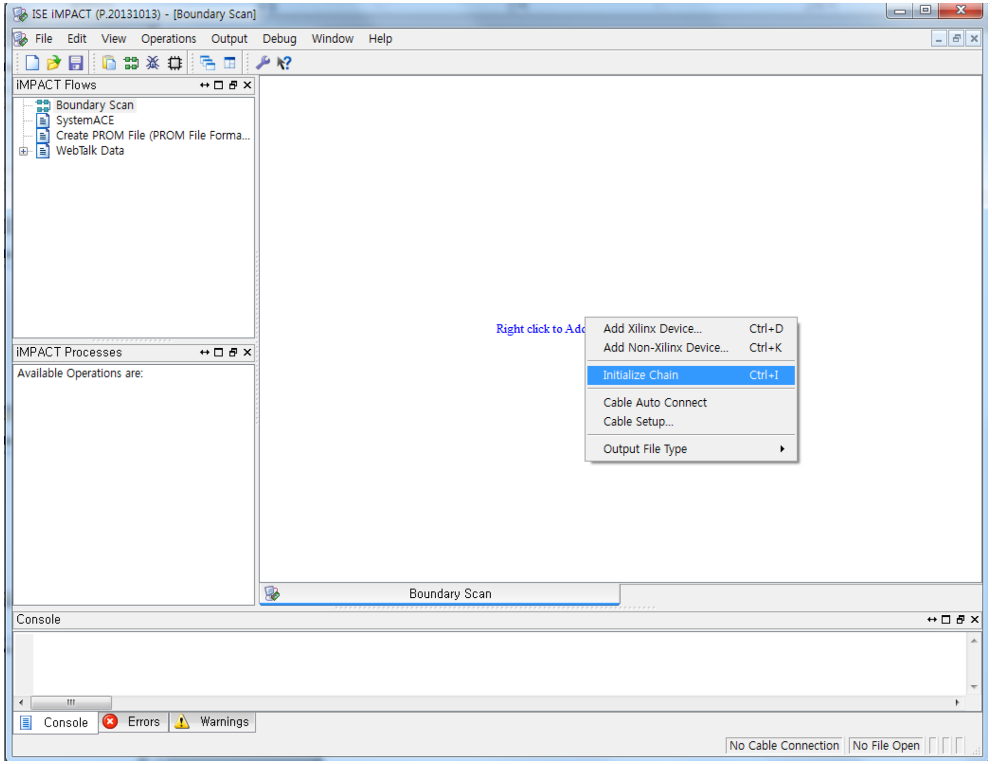
SNU Logic Design Board

- **Boundary Scan > Initialize Chain**



Double Click to Launch Mode.
Targeting a BPI/SPI device indirectly also start from here.

Available Operations are:

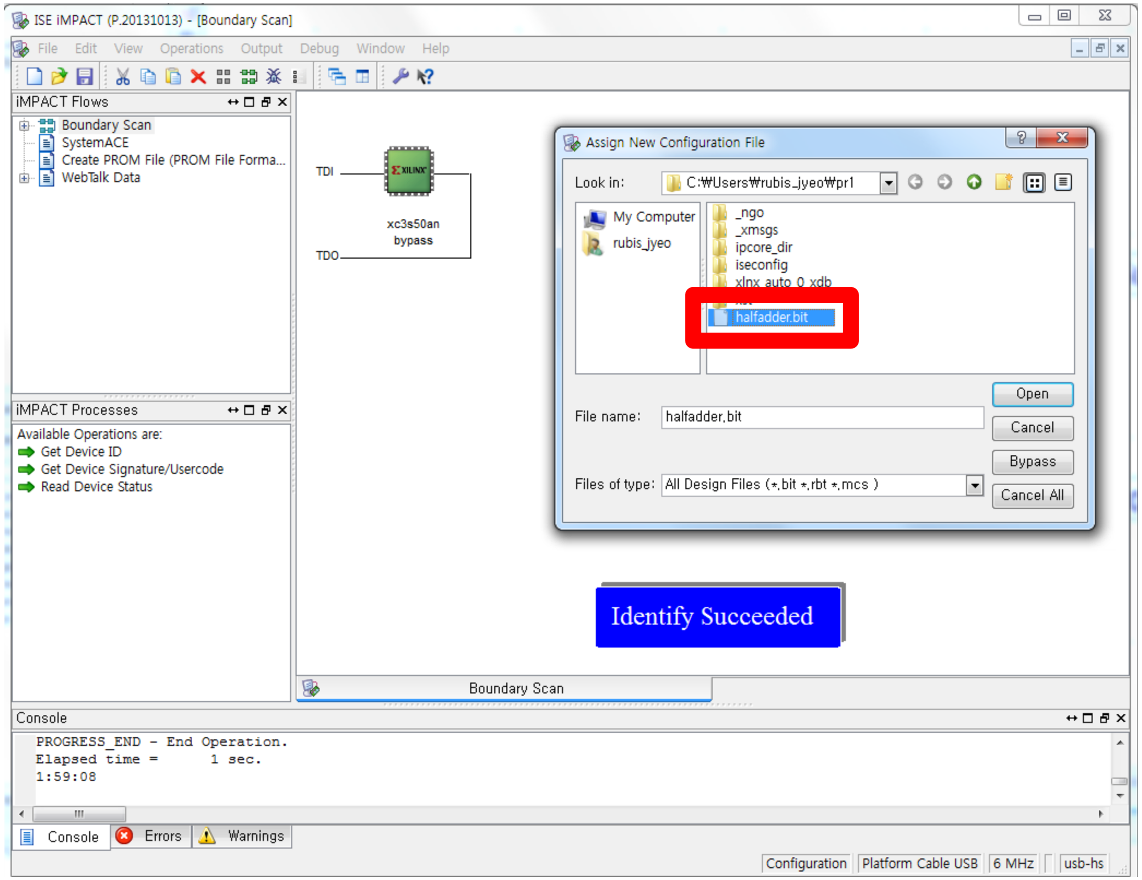


Available Operations are:

Right click to Add

SNU Logic Design Board

- Choose the schematic source you've written.



Look in: C:₩Users₩rubis_jyeo₩pr1

File name: halfadder.bit

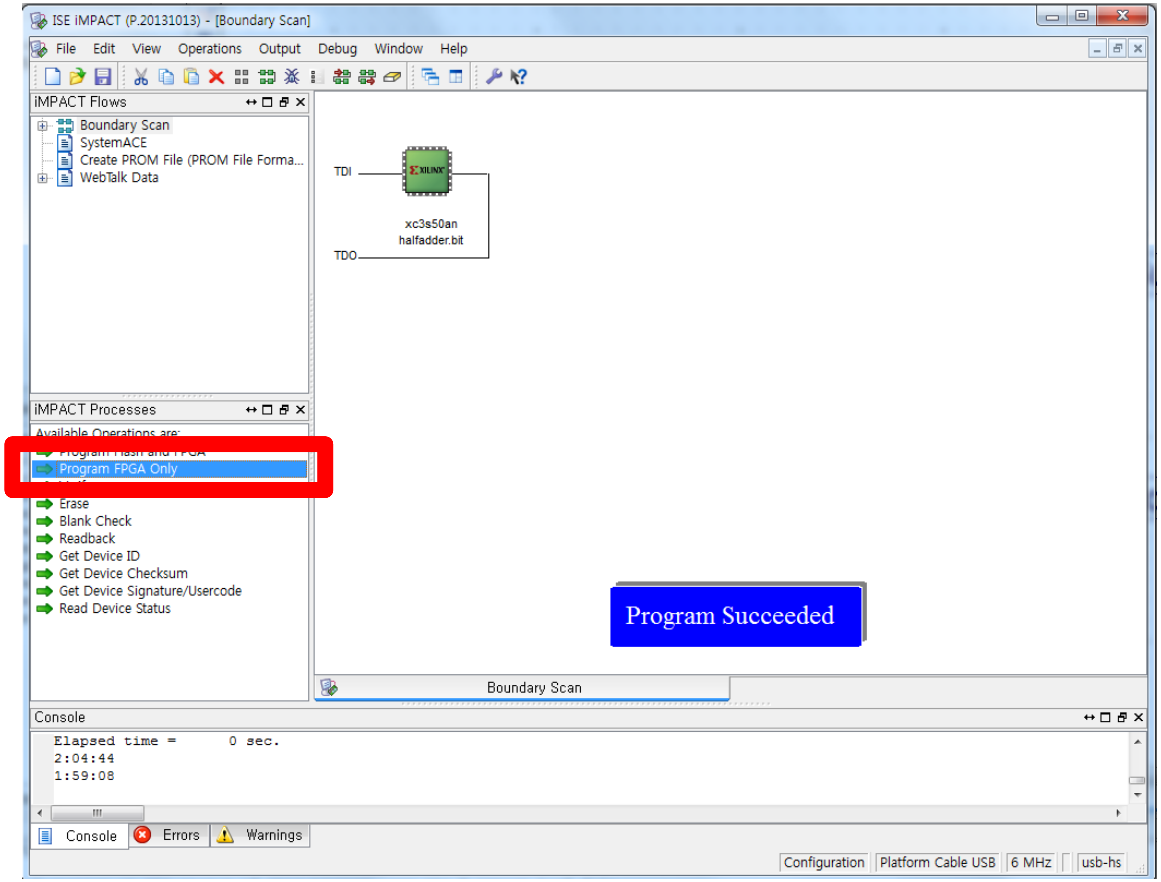
Files of type: All Design Files (*.bit *.rbt *.mcs)

Identify Succeeded

```
PROGRESS_END - End Operation.
Elapsed time =      1 sec.
1:59:08
```

SNU Logic Design Board

- **Program FPGA Only > Program Succeeded**

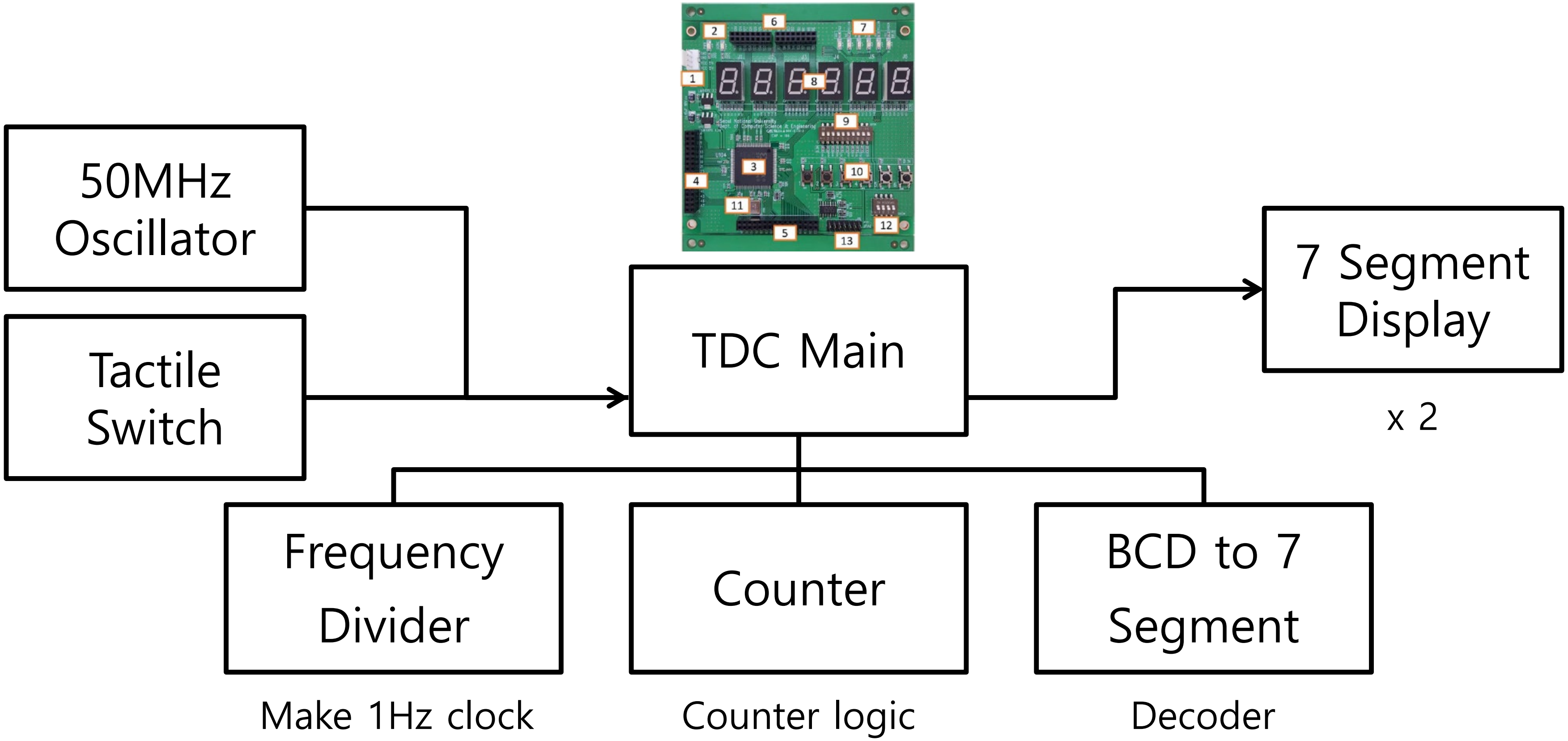


Program Succeeded

```
Elapsed time =      0 sec.
2:04:44
1:59:08
```

Two Digit Counter

Two Digit Counter Structure



50MHz Oscillator

Tactile Switch

TDC Main

7 Segment Display

x 2

Frequency Divider

Make 1Hz clock

Counter

Counter logic

BCD to 7 Segment

Decoder

Homework

1. Implement the **Counter** Module in Verilog. It should start from 0, and increment 1 every **positive edge of clock**. The next value of 99 should be 0. Additionally, your implementation should handle external reset signal, it should set the counter back to 0 when the signal is HIGH.

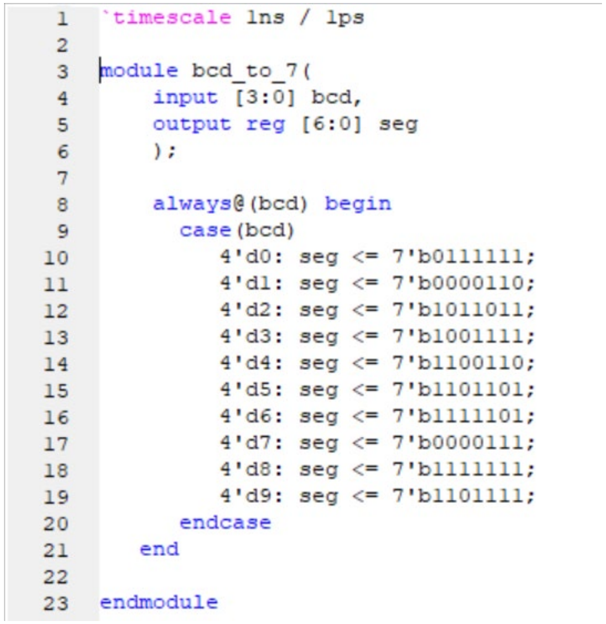
Homework

2. Implement the **Two Digit Counter** in Verilog and upload your implementation to the Logic Design Board. It should start from 0, and increment 1 **every second**. The next value of 99 should be 0. Additionally, your implementation should handle external reset signal, it should set the counter back to 0 when the signal is HIGH. Use 50MHz oscillator (pin 57) as clock input, and a tactile switch (pin 47) as reset button. Also, use two 7-segment displays as your two-digit output.

Sample Code (Behavioral Description)

```
`timescale 1ns / 1ps

module freq_divider(
    input clr,
    input clk,
    output reg clkout
    );
         reg[31:0] cnt;
         always@ (posedge clk) begin
                if(clr) begin
                        cnt <=32'd0;
                        clkout <= 1'b0;
                end
                else if (cnt == 32'd25000000) begin
                        cnt <= 32'd0;
                        clkout <= ~clkout;
                end
                else begin
                        cnt <= cnt + 1;
                end
        end

endmodule
```

Frequency Divider

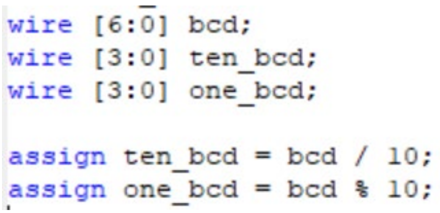


```
`timescale 1ns / 1ps

module bcd_to_7(
    input [3:0] bcd,
    output reg [6:0] seg
    );

    always@(bcd) begin
      case(bcd)
         4'd0: seg <= 7'b0111111;
         4'd1: seg <= 7'b0000110;
         4'd2: seg <= 7'b1011011;
         4'd3: seg <= 7'b1001111;
         4'd4: seg <= 7'b1100110;
         4'd5: seg <= 7'b1101101;
         4'd6: seg <= 7'b1111101;
         4'd7: seg <= 7'b0000111;
         4'd8: seg <= 7'b1111111;
         4'd9: seg <= 7'b1101111;
      endcase
   end

endmodule
```

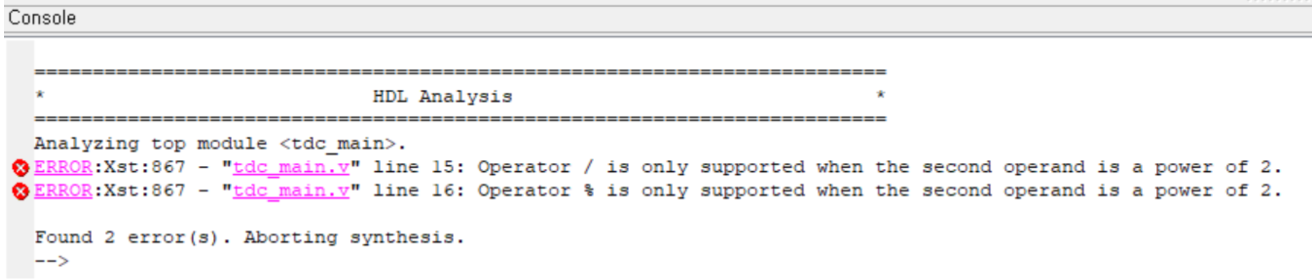
BCD to 7-segment Decoder

Homework

- **Hint : There is modular operation in Verilog. But..**
  - Our board does not support division by 10.
  - We recommend using 4 bit for each digit in counter logic



```
wire [6:0] bcd;
wire [3:0] ten_bcd;
wire [3:0] one_bcd;

assign ten_bcd = bcd / 10;
assign one_bcd = bcd % 10;
```

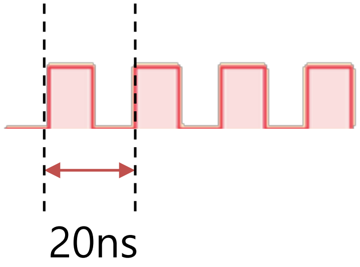


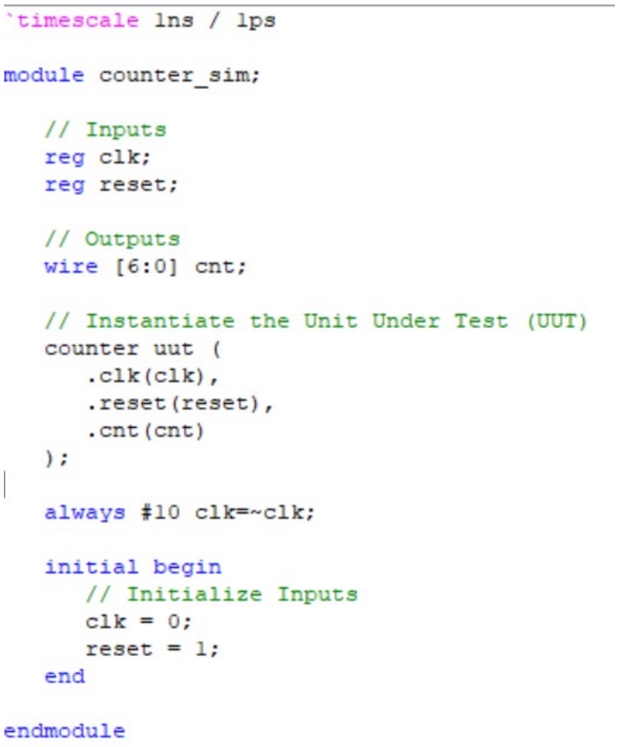
```
=========================================================================
*                          HDL Analysis                                 *
=========================================================================
Analyzing top module <tdc_main>.
ERROR:Xst:867 - "tdc_main.v" line 15: Operator / is only supported when the second operand is a power of 2.
ERROR:Xst:867 - "tdc_main.v" line 16: Operator % is only supported when the second operand is a power of 2.

Found 2 error(s). Aborting synthesis.
-->
```

Homework

- **Simulating 50MHz clock in test bench**
  - 50MHz : period is 20ns

    (So we should flip clock every 10ns)

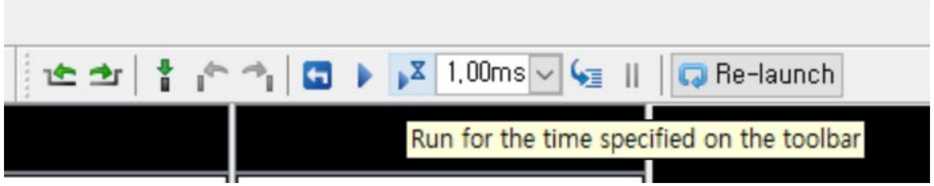




```
`timescale 1ns / 1ps

module counter_sim;

   // Inputs
   reg clk;
   reg reset;

   // Outputs
   wire [6:0] cnt;

   // Instantiate the Unit Under Test (UUT)
   counter uut (
      .clk(clk),
      .reset(reset),
      .cnt(cnt)
   );

   always #10 clk=~clk;

   initial begin
      // Initialize Inputs
      clk = 0;
      reset = 1;
   end

endmodule
```

Homework

- **Remind) Simulating more than 1ms**

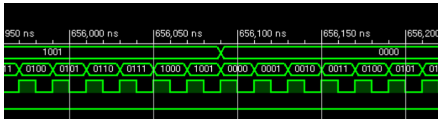


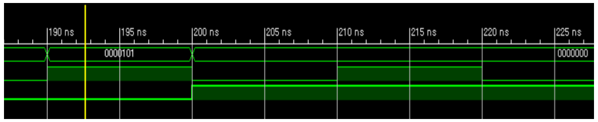
Run for the time specified on the toolbar

Report

- ## Counter Module (Verilog)

  1. Result of simulation (should attach waveform of three case)
     - should increase 1 in normal case
     - should set 0 when counter = 100
     - should be 0 when reset = 1
       (The bus value of the counter must be **clearly visible** in picture)





- ## Two Digit Counter

  1. All Verilog source code
     - should include code for module, submodule
       (i.e., TDC main, counter, freq divider, BCD-7 seg decoder)
     - only module code, **no test bench code**

  2. Result of implementation of three case
     - should increase 1 in normal case
     - should set 0 at t = 100s
     - should be 0 when pressing the reset button
       (at least 2 pictures should be attached for each case)

- ## Discussion