

## Zak Rowland

Lab 5 -2020

Paste images and respond to questions in this document.

### Part 1

1. Using Quartus Prime, write a 32 bit counter
  - a. Positive edge triggered
  - b. Counts up
  - c. Active high synchronous reset
  - d. Counts up to 32'hFFF\_FF00 then sets a flag 'GO' for one cycle and counter goes back to zero.
  - e. For your comparison, use less than or equal.

```
1 module counter_32bit(  
2     input clk,  
3     input reset,  
4     output reg [31:0] count,  
5     output reg go  
6 );  
7  
8  
9  
10 parameter compare_value = (32'hFFFFFFF); // For <=  
11 //parameter compare_value = (32'hFFFFFF00); // For !=  
12  
13 always @ (posedge clk) begin  
14     if(reset == 1'b1) begin  
15         count <= 32'd0;  
16         go <= 1'b0;  
17     end  
18     else begin  
19         if(count <= compare_value) begin  
20             count <= count + 32'd1;  
21             go <= 1'b0;  
22         end  
23         else begin  
24             count <= 32'd0;  
25             go <= 1'b1;  
26         end  
27     end  
28 end  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41 endmodule
```

2. Go into the Timing Analyzer. Determine the maximum speed of the 32-bit counter. Take a screenshot

Set Operating Conditions

☐ Slow 1100mV 85C Model  
☐ Slow 1100mV 0C Model  
☐ Fast 1100mV 85C Model  
☒ Fast 1100mV 0C Model

Report

Fast 1100mV 0C Model

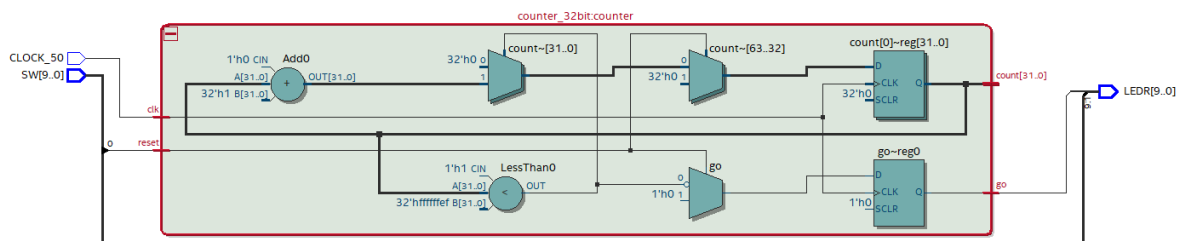
	Fmax	Restricted Fmax	Clock Name	Note
1	535.91 MHz	535.91 MHz	CLOCK_50	

## 3. Take a screenshot of the resources used



Fitter Resource Usage Summary			
<<Filter>>			
	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	23 / 32,070	< 1 %
2	> ALMs needed [=A-B+C]	23	
3			
4	Difficulty packing design	Low	
5			
6	> Total LABs: partially or completely used	4 / 3,207	< 1 %
7			
8	> Combinational ALUT usage for logic	40	
9	Combinational ALUT usage for route-throughs	0	
10			
11	> Dedicated logic registers	33	
12			
13	Virtual pins	0	
14	> I/O pins	21 / 457	5 %
15			
16	> Hard processor system peripheral utilization		
17			
18	M10K blocks	0 / 397	0 %
19	Total MLAB memory bits	0	
20	Total block memory bits	0 / 4,065,280	0 %
21	Total block memory implementation bits	0 / 4,065,280	0 %

Fitter Resource Usage Summary			
<<Filter>>			
	Resource	Usage	%
21	Total block memory implementation bits	0 / 4,065,280	0 %
22			
23	Total DSP Blocks	0 / 87	0 %
24			
25	Fractional PLLs	0 / 6	0 %
26	> Global signals	1	
27	SERDES Transmitters	0 / 100	0 %
28	SERDES Receivers	0 / 100	0 %
29	JTAGs	0 / 1	0 %
30	ASMI blocks	0 / 1	0 %
31	CRC blocks	0 / 1	0 %
32	Remote update blocks	0 / 1	0 %
33	Oscillator blocks	0 / 1	0 %
34	Impedance control blocks	0 / 4	0 %
35	Hard Memory Controllers	0 / 2	0 %
36	Average interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%	
37	Peak interconnect usage (total/H/V)	0.3% / 0.3% / 0.3%	
38	Maximum fan-out	33	
39	Highest non-global fan-out	32	
40	Total fan-out	233	
41	Average fan-out	2.01	

4. Take a screenshot of the RTL



5. Repeat 1 and use != instead of <= for your comparison value.

Report  

6. Take a screenshot of the resources used. Do you notice a difference between != and <=?

Fitter Resource Usage Summary			
	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	24 / 32,070	< 1 %
2	➤ ALMs needed [=A-B+C]	24	
3			
4	Difficulty packing design	Low	
5			
6	➤ Total LABs: partially or completely used	4 / 3,207	< 1 %
7			
8	➤ Combinational ALUT usage for logic	42	
9	Combinational ALUT usage for route-throughs	0	
10			
11	➤ Dedicated logic registers	33	
12			
13	Virtual pins	0	
14	➤ I/O pins	21 / 457	5 %
15			
16	➤ Hard processor system peripheral utilization		
17			
18	M10K blocks	0 / 397	0 %
19	Total MLAB memory bits	0	
20	Total block memory bits	0 / 4,065,280	0 %
21	Total block memory implementation bits	0 / 4,065,280	0 %

Fitter Resource Usage Summary			
<<Filter>>			
	Resource	Usage	%
21	Total block memory implementation bits	0 / 4,065,280	0 %
22			
23	Total DSP Blocks	0 / 87	0 %
24			
25	Fractional PLLs	0 / 6	0 %
26	> Global signals	1	
27	SERDES Transmitters	0 / 100	0 %
28	SERDES Receivers	0 / 100	0 %
29	JTAGs	0 / 1	0 %
30	ASMI blocks	0 / 1	0 %
31	CRC blocks	0 / 1	0 %
32	Remote update blocks	0 / 1	0 %
33	Oscillator blocks	0 / 1	0 %
34	Impedance control blocks	0 / 4	0 %
35	Hard Memory Controllers	0 / 2	0 %
36	Average interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%	
37	Peak interconnect usage (total/H/V)	0.4% / 0.5% / 0.3%	
38	Maximum fan-out	33	
39	Highest non-global fan-out	32	
40	Total fan-out	235	
41	Average fan-out	1.99	

- The not equal comparison uses slightly more logic.

## Part 2

1. Now write a 32 bit counter but use pipelining. That is, break the 32 bit counter into 4 8-bit counters. Each 8-bit counter triggers the next 8-bit counter.
2. Go into timing analyzer and determine the maximum speed of the pipelined counter.

Timing Analyzer - J:/School/2019-2020/Spring 2020/CST351/Lab5/Lab5\_main/Lab5 - lab5\_top

File View Netlist Constraints Reports Script Tools Window Help

Set Operating Conditions

Fast 1100mV OC Model

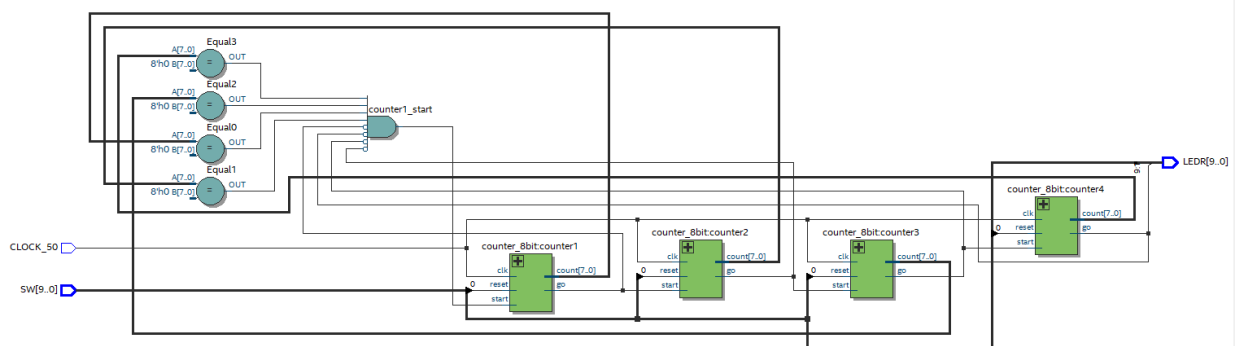
	Fmax	Restricted Fmax	Clock Name	Note
1	643.5 MHz	534.19 MHz	CLOCK_50	limit due to low minimum pulse width violation (tcl)

☐ Slow 1100mV 85C Model
 ☐ Slow 1100mV OC Model
 ☐ Fast 1100mV 85C Model
 ☒ Fast 1100mV OC Model

3. Take a screenshot of the resources used.

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	34
2		
3	> Combinational ALUT usage for logic	60
4		
5	Dedicated logic registers	40
6		
7	I/O pins	21
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	CLOCK_50~input
12	Maximum fan-out	40
13	Total fan-out	378
14	Average fan-out	2.66

4. Take a screenshot of the RTL



## Part 3

- What is the maximum speed improvement of Part 1 non-pipelined counter versus your Part 2 pipelined counter? What was the % increase in logic usage with pipelined versus non-pipelined.
  - The maximum speed improvement was 154.51MHz, and logic usage increased by 47.8%.
- Please explain why you saw a speed increase between the two implementations. (If you did not, you did something wrong.) Elaborate on this question.

- The speed increased because the processor can act upon the four counters in parallel every clock cycle.
3. What if we used 8 4-bit counters? (You don't actually have to try this.) Would you expect to be able to run the circuit faster or slower? Why?
- It would not be faster because the latency of 8 pipelined counters would be greater than the benefit of pipelining with registers that size.
4. What other ways could you improve the performance (space or speed) of the design? (Look up LFSR as one option)
- An LFSR is a Linear Feedback Shift Register which is a series of flip flops in the FPGA, and the output of the last flip flop feeds back to the input of the first flip flop. Other ways to improve the performance include removing unnecessary or duplicate logic or changing the speed grade of the device used.