

Project 2: Clock Design Methodology

Haley Kim
405111152

Introduction and Requirement

Clocks play a major role in synchronous data transmission, and are a basis of timer systems used in embedded systems. Clocks allow high testability since engineers can run through waveform simulations to check for expected behavior.

This project has four major design requirements that must be met:

1. submodule dividing given clock by power of 2
2. submodule dividing given clock by even numbers
3. submodule dividing given clock by odd numbers
4. submodule using pulse/strobee/flag

Within the first submodule, we will divide the given clock by 2, 4, 8, and 16, and assign each to its own wire output, `clk_div_2`, `clk_div_4`, `clk_div_8`, and `clk_div_16`.

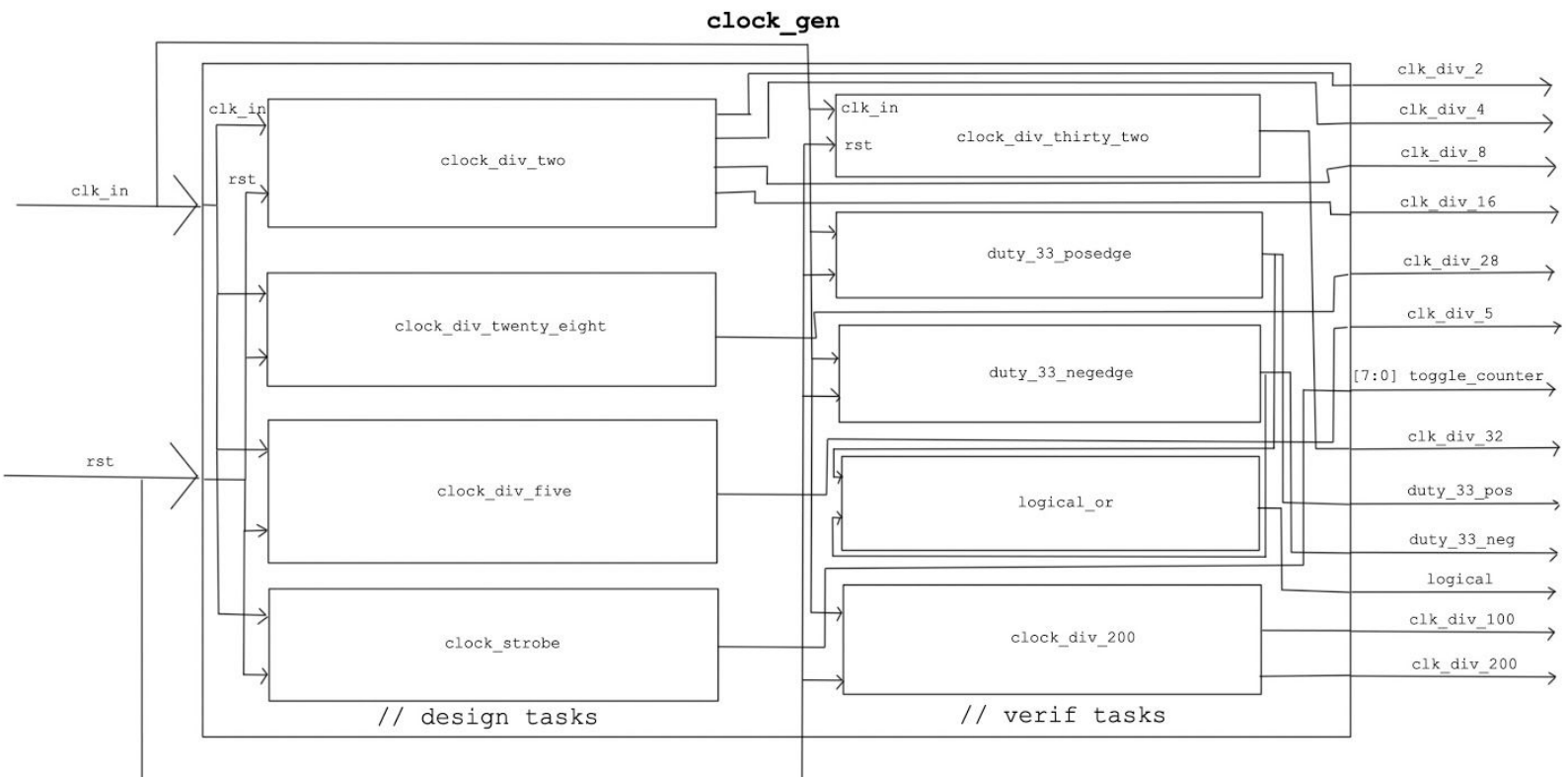
Within the second submodule, we will divide the given clock by 28, and assign it to a wire output, `clk_div_28`.

Within the third submodule, we will divide the given clock by 5, and make sure it has a 50% duty cycle. We will then assign it to a wire output, `clk_div_5`.

Within the fourth submodule, we will divide the given clock by 4 to create a strobe, then use the strobe to subtract 5 from the toggle counter on every strobe, and increment by 2 on every positive edge of the given clock (to resemble $0 \Rightarrow 2 \Rightarrow 4 \Rightarrow 6 \Rightarrow 1 \Rightarrow 3 \Rightarrow 5 \Rightarrow 7 \Rightarrow 2 \dots$).

Along the project, there will also be five verification tasks to complete that will aid with building the main four submodules. The first is to create a divide-by-32 clock, the second is to create a 33% duty cycle clock on the positive edge of the given clock, the third is to create the same 33% duty cycle clock but on the negative edge of the given clock, the fourth is to apply a logical OR to the two 33% duty cycle clocks, and the fifth is to create a divide-by-100 1% duty cycle clock and a divide by 200 50% duty cycle clock.

Design Description



The main overarching module is **clock_gen**, with inputs:

clk_in,

rst,

and outputs:

clk_div_2, **clk_div_4**, **clk_div_8**, **clk_div_16**, **clk_div_28**, **clk_div_5**, **[7:0]**

toggle_counter (for design tasks), and

clk_div_32, **duty_33_pos**, **duty_33_neg**, **logical**, **clk_div_100**, **clk_div_200** (for verification tasks)

Main module **clock_gen** calls submodules:

clock_div_two, **clock_div_twenty_eight**, **clock_div_five**, and **clock_strobe** for the four design tasks.

Main module **clock_gen** calls submodules:

clock_div_thirty_two, **duty_33_posedge**, **duty_33_negedge**, **logical_or**, and

clock_div_200 for the five verification tasks.

clock_gen's logic:

has no real logic, uses submodules to create desired outputs

clock_div_two's logic:

takes in "clk_in" and "rst" as input

creates 4-bit counter "a"

assigns LSB to "clk_div_2_sub", assigns "a[1]" to "clk_div_4_sub", assigns

"a[2]" to "clk_div_8_sub", assigns MSB to "clk_div_16_sub"

outputs "clk_div_2_sub", "clk_div_4_sub", "clk_div_8_sub", "clk_div_16_sub"

clock_div_twenty_eight's logic:

takes in "clk_in" and "rst" as input

(100MHz divided by 28 is 3.57142857MHz)

(3.57142857MHz requires a 280ns period)

initially start 4-bit counter "a" at 0, start clk_div_28_sub at 0

flip clk_div_28_sub when "a" is 13, change "a" to 0

(the period of clk_div_28_sub becomes 28 counts long, which is 280ns)

outputs "clk_div_28_sub"

clock_div_five's logic:

takes in "clk_in" and "rst" as input

(100MHz divided by 5 is 20MHz)

(20MHz requires a 50ns period)

(50% duty cycle requires 25ns output 1 followed by 25ns output 0)

(impossible to do 5ns chunks because clk_in has period of 10ns)

always posedge loop creates "clk_div_pos_sub"

(creates 20ns high 30ns low signal)

initialize "clk_div_pos_sub" at 0, counter "a" at 0

flip "clk_div_pos_sub" when "a" is 1

flip "clk_div_pos_sub" when "a" is 4, set "a" back to 0

always negedge loop creates "clk_div_neg_sub"

(creates 20ns high 30ns low signal that is 5ns behind clk_div_pos_sub)

same logic as "clk_div_pos_sub" except on negedge of "clk_in"

perform logical OR on "clk_div_pos_sub" and "clk_div_neg_sub" to get

"clk_div_five_sub"

(creates 25ns high 25ns low signal)

outputs "clk_div_5_sub"

clock_strobe's logic:

- takes in "clk_in" and "rst" as input
 - implement divide-by-4 strobe
 - (100MHz divided by 4 is 25MHz, 25MHz requires a 40ns period)
 - initialize 4-bit counter "a" as 0, "strobe_sub" as 0
 - when "a" is 2, flip "strobe_sub"
 - when "a" is 3, flip "strobe_sub" and change "a" back to 0
 - (this creates a clock with 30ns low and 10ns high)
 - if "strobe_sub" is 1, subtract 5 from "toggle_counter_sub"
 - if "strobe_sub" is 0, add 2 to "toggle_counter_sub"
 - (this should happen on every posedge clk_in)
- outputs 7-bit "toggle_counter_sub"

clock_div_thirty_two's logic:

- takes in "clk_in" and "rst" as input
 - (100MHz divided by 32 is 3.125MHz, 3.125MHz requires a 320ns period)
 - initialize 4-bit counter "a" as 0, "clk_div_32_sub" as 0
 - if "a" is 0, flip "clk_div_32_sub"
 - (this creates a clock with 16 counter high and 16 counter low)
 - (equivalent to 160ns high and 160ns low)
- outputs "clk_div_32_sub"

duty_33_posedge's logic:

- takes in "clk_in" and "rst" as input
 - initialize 4-bit counter "a" as 0, "duty_33_posedge_sub" as 0
 - if "a" is 5, flip "duty_33_posedge_sub"
 - if "a" is 8, flip "duty_33_posedge_sub" and change "a" back to 0
 - (this creates a clock with 6 counts low and 3 counts high)
- outputs "duty_33_posedge_sub"

duty_33_negedge's logic:

- same logic as duty_33_negedge, but always loop triggered on negedge instead of posedge

logical_or's logic:

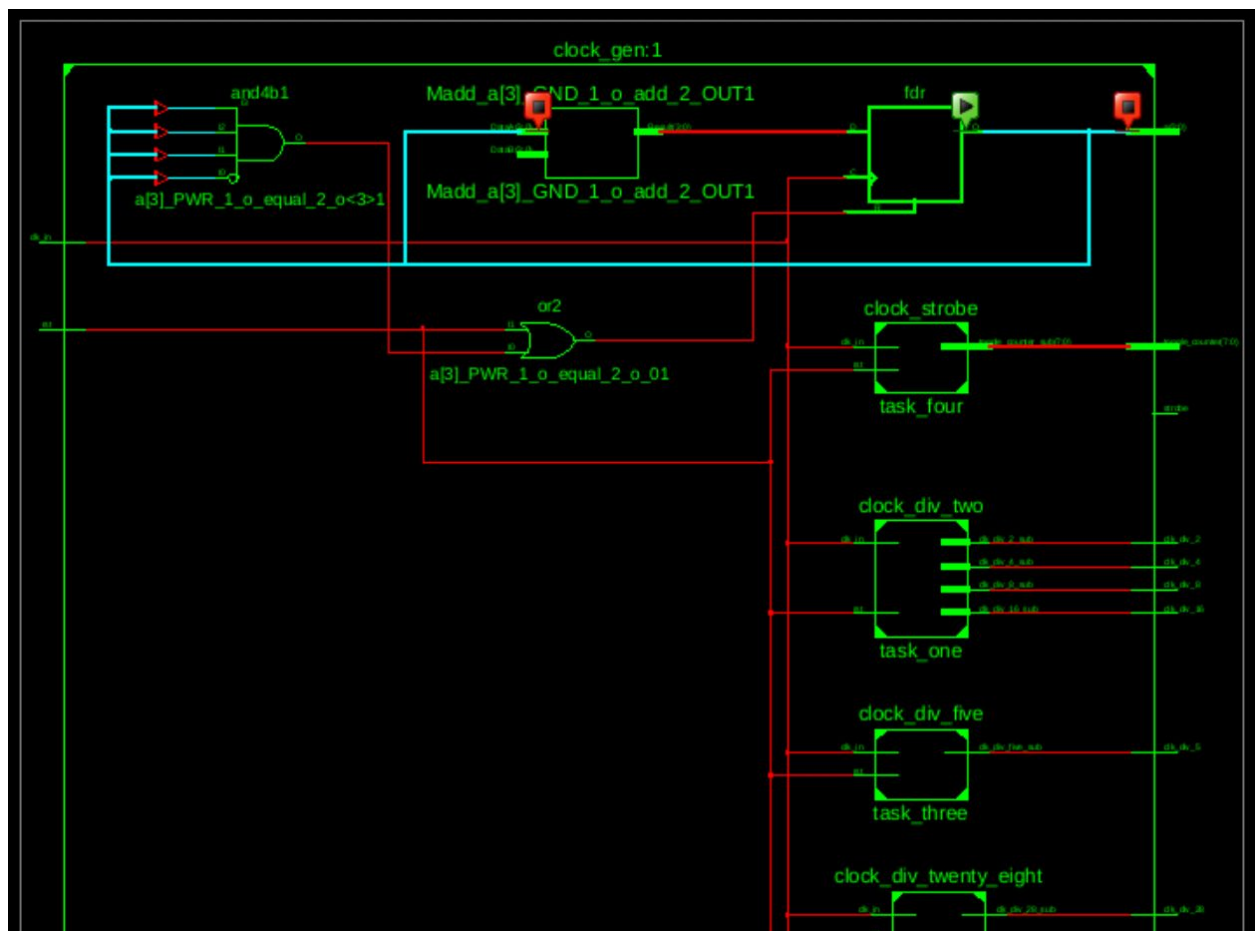
- takes in "duty_33_pos_in" and "duty_33_neg_in" as input
 - performs a logical OR on the two inputs and assigns it to "logical_or_sub"
- outputs "logical_or_sub"

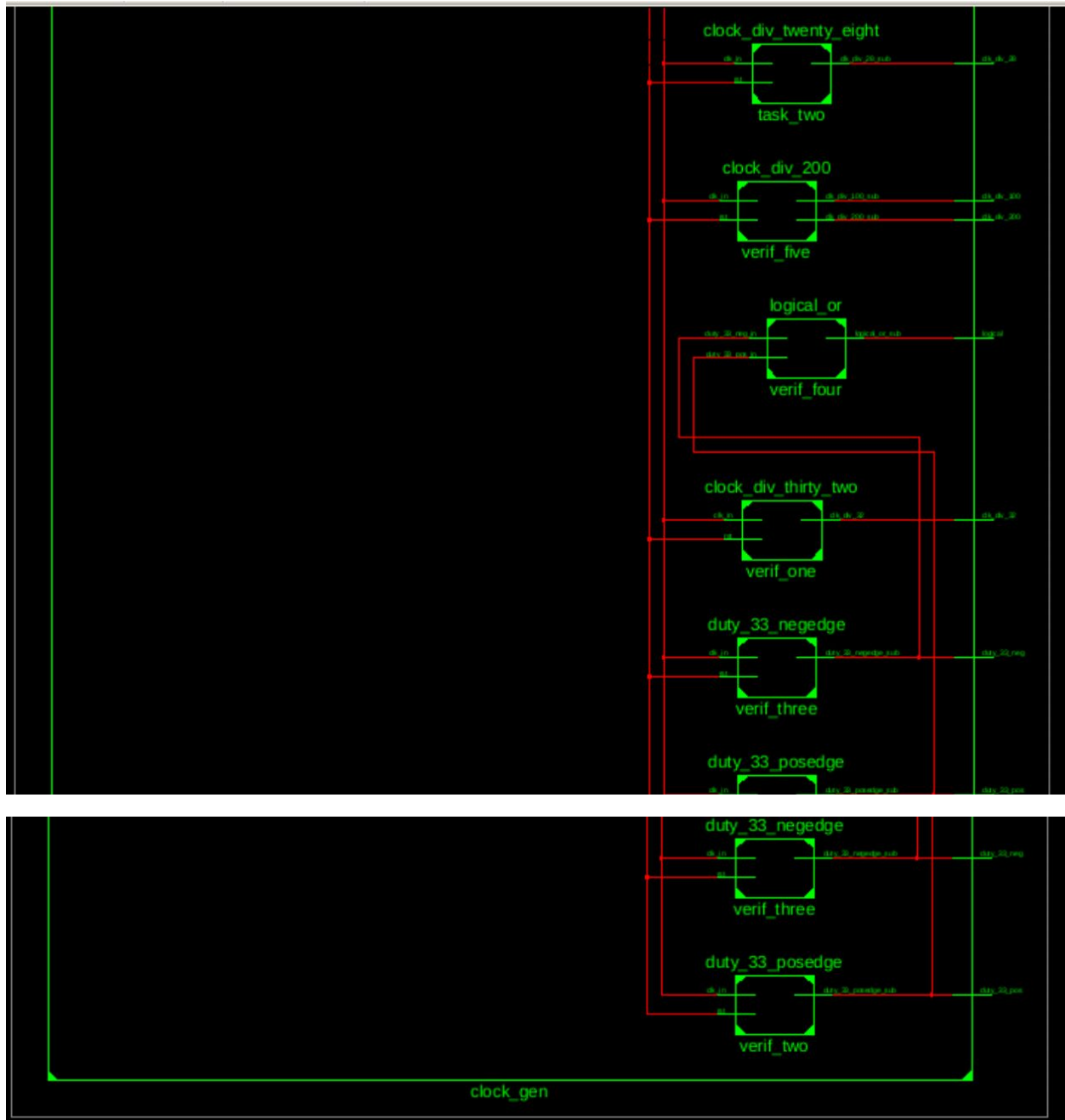
clock_div_200's logic:

- takes in "clk_in" and "rst" as input

initializes 7-bit counter "a" as 0, "clk_div_100_sub" as 0
 if "a" is 98, flip "clk_div_100_sub"
 if "a" is 99, flip "clk_div_100_sub" and change "a" back to 0
 (this creates a clock with 99 counter low and 1 counter high)
 another always posedge loop, initialize "clk_div_200_sub" as 0
 if "clk_div_100_sub is 1, flip "clk_div_200_sub"
 (this creates a clock with 100 counter low and 100 counter high, a divide
 by 200 clock with 50% duty cycle)
 outputs "clk_div_100_sub" and "clk_div_200_sub"

Simulation Documentation





===== HDL Synthesis Report

Macro Statistics

# Adders/Subtractors	: 11
4-bit adder	: 9
7-bit adder	: 1
8-bit addsub	: 1
# Registers	: 21
1-bit register	: 9
4-bit register	: 10
7-bit register	: 1
8-bit register	: 1
# Multiplexers	: 3
4-bit 2-to-1 multiplexer	: 2
8-bit 2-to-1 multiplexer	: 1


HDL Synthesis Report:


This report tells us which parts were utilized for the synthesis of this project. A large number of registers are used, a total of 21, which is significantly larger than the number of registers used for project 1.

clock_gen Project Status (11/07/2020 - 03:43:50)			
Project File:	Proj2.xise	Parser Errors:	No Errors
Module Name:	clock_gen	Implementation State:	Placed and Routed
Target Device:	xc6slx4-3tqg144	• Errors:	No Errors
Product Version:	ISE 14.7	• Warnings:	5 Warnings (5 new)
Design Goal:	Balanced	• Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	All Constraints Met
Environment:	System Settings	• Final Timing Score:	0 (Timing Report)

Device Utilization Summary					[1]
Slice Logic Utilization	Used	Available	Utilization	Note(s)	
Number of Slice Registers	47	4,800	1%		
Number used as Flip Flops	47				
Number used as Latches	0				
Number used as Latch-thrus	0				
Number used as AND/OR logics	0				
Number of Slice LUTs	35	2,400	1%		
Number used as logic	34	2,400	1%		
Number using O6 output only	19				
Number using O5 output only	0				
Number using O5 and O6	15				
Number used as ROM	0				
Number used as Memory	0	1,200	0%		
Number used exclusively as route-thrus	1				
Number with same-slice register load	1				
Number with same-slice carry load	0				
Number with other load	0				
Number of occupied Slices	21	600	3%		
Number of MUXCYs used	0	1,200	0%		
Number of LUT Flip Flop pairs used	37				
Number with an unused Flip Flop	4	37	10%		
Number with an unused LUT	2	37	5%		
Number of fully used LUT-FF pairs	31	37	83%		
Number of unique control sets	5				
Number of slice register sites lost to control set restrictions	17	4,800	1%		
Number of bonded IOBs	26	102	25%		
Number of RAMB16BWERS	0	12	0%		
Number of RAMB8BWERS	0	24	0%		

Number of BUFIO2/BUFIO2_2CLKs	0	32	0%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%
Number of BUFG/BUFGMUXs	1	16	6%
Number used as BUFGs	1		
Number used as BUFGMUX	0		
Number of DCM/DCM_CLKGENs	0	4	0%
Number of ILOGIC2/ISERDES2s	0	200	0%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	200	0%
Number of OLOGIC2/OSERDES2s	0	200	0%
Number of BSCANs	0	4	0%
Number of BUFHs	0	128	0%
Number of BUFPLLs	0	8	0%
Number of BUFPLL_MCBs	0	4	0%
Number of DSP48A1s	0	8	0%
Number of ICAPs	0	1	0%
Number of PCIOLOGICSEs	0	2	0%
Number of PLL_ADVs	0	2	0%
Number of PMVs	0	1	0%
Number of STARTUPs	0	1	0%
Number of SUSPEND_SYNCs	0	1	0%
Average Fanout of Non-Clock Nets	3.84		

Performance Summary				
Final Timing Score:	0 (Setup: 0, Hold: 0)	Pinout Data:	Pinout Report	
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report	
Timing Constraints:	All Constraints Met			

Detailed Reports						
Report Name	Status	Generated	Errors	Warnings	Infos	
Synthesis Report	Current	Sat Nov 7 01:56:12 2020	0	5 Warnings (5 new)	8 Infos (8 new)	
Translation Report	Current	Sat Nov 7 03:43:24 2020	0	0	0	
Map Report	Current	Sat Nov 7 03:43:37 2020	0	0	7 Infos (7 new)	
Place and Route Report	Current	Sat Nov 7 03:43:44 2020	0	0	3 Infos (3 new)	
Power Report						
Post-PAR Static Timing Report	Current	Sat Nov 7 03:43:48 2020	0	0	4 Infos (4 new)	
Bitgen Report						

Design Summary Report:

The design was synthesized with no errors. Other than this crucial information, the design summary report gives us how many of each part was utilized for the synthesis of the design.

Design Summary

Number of errors: 0

Number of warnings: 0

Slice Logic Utilization:

Number of Slice Registers:	47 out of	4,800	1%
Number used as Flip Flops:	47		
Number used as Latches:	0		
Number used as Latch-thrus:	0		
Number used as AND/OR logics:	0		
Number of Slice LUTs:	35 out of	2,400	1%
Number used as logic:	34 out of	2,400	1%
Number using 06 output only:	19		
Number using 05 output only:	0		
Number using 05 and 06:	15		
Number used as ROM:	0		
Number used as Memory:	0 out of	1,200	0%
Number used exclusively as route-thrus:	1		
Number with same-slice register load:	1		
Number with same-slice carry load:	0		
Number with other load:	0		

Slice Logic Distribution:

Number of occupied Slices:	21 out of	600	3%
Number of MUXCYs used:	0 out of	1,200	0%
Number of LUT Flip Flop pairs used:	37		
Number with an unused Flip Flop:	4 out of	37	10%
Number with an unused LUT:	2 out of	37	5%
Number of fully used LUT-FF pairs:	31 out of	37	83%
Number of unique control sets:	5		
Number of slice register sites lost to control set restrictions:	17 out of	4,800	1%

A LUT Flip Flop pair for this architecture represents one LUT paired with one Flip Flop within a slice. A control set is a unique combination of clock, reset, set, and enable signals for a registered element.

The Slice Logic Distribution report is not meaningful if the design is over-mapped for a non-slice resource or if Placement fails.

Average Fanout of Non-Clock Nets: 3.84

Peak Memory Usage: 759 MB

Total REAL time to MAP completion: 9 secs

Total CPU time to MAP completion: 8 secs

Map Report:

While other information is similar to that given by the HDL Synthesis Report and Design Summary Report, the Map Report uniquely identifies the peak memory usage and the total REAL time to MAP completion. The total REAL time to completion is greater by 80% compared to that of project 1.

```
Starting initial Timing Analysis.  REAL time: 2 secs
Finished initial Timing Analysis.  REAL time: 2 secs
```

```
Starting Router
```

```
Phase 1 : 235 unrouted;      REAL time: 2 secs
```

```
Phase 2 : 199 unrouted;      REAL time: 3 secs
```

```
Phase 3 : 55 unrouted;       REAL time: 3 secs
```

```
Phase 4 : 55 unrouted; (Par is working to improve performance)  REAL time: 3 secs
```

```
Updating file: clock_gen.ncd with current fully routed design.
```

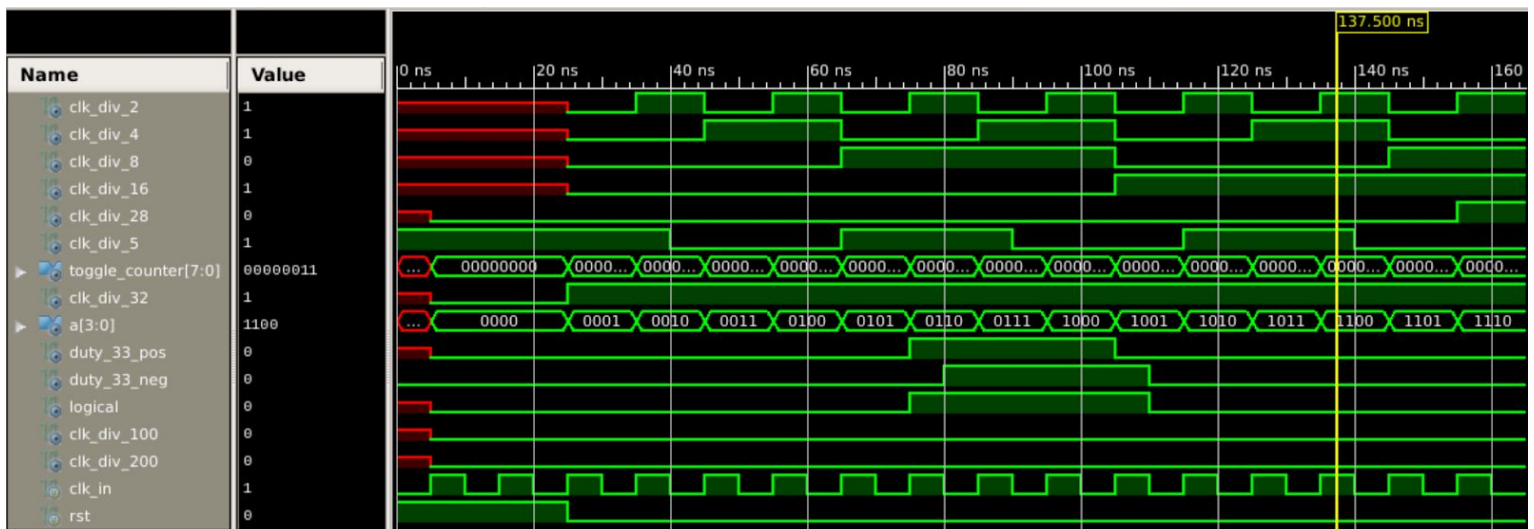
```
Phase 5 : 0 unrouted; (Par is working to improve performance)  REAL time: 3 secs
```

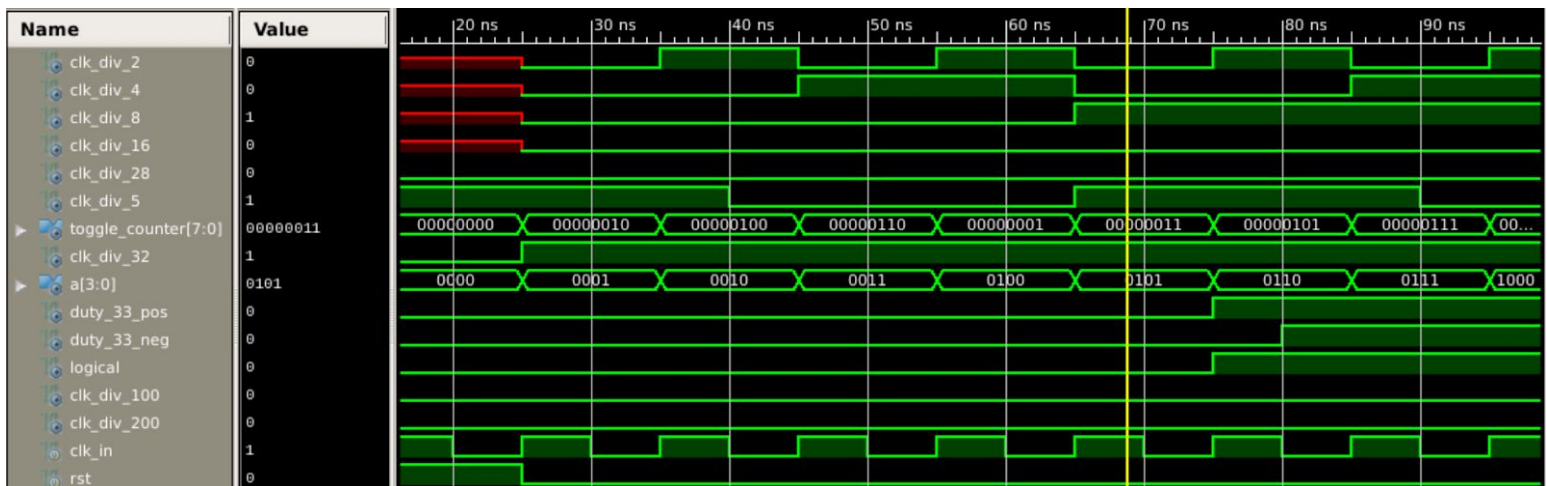
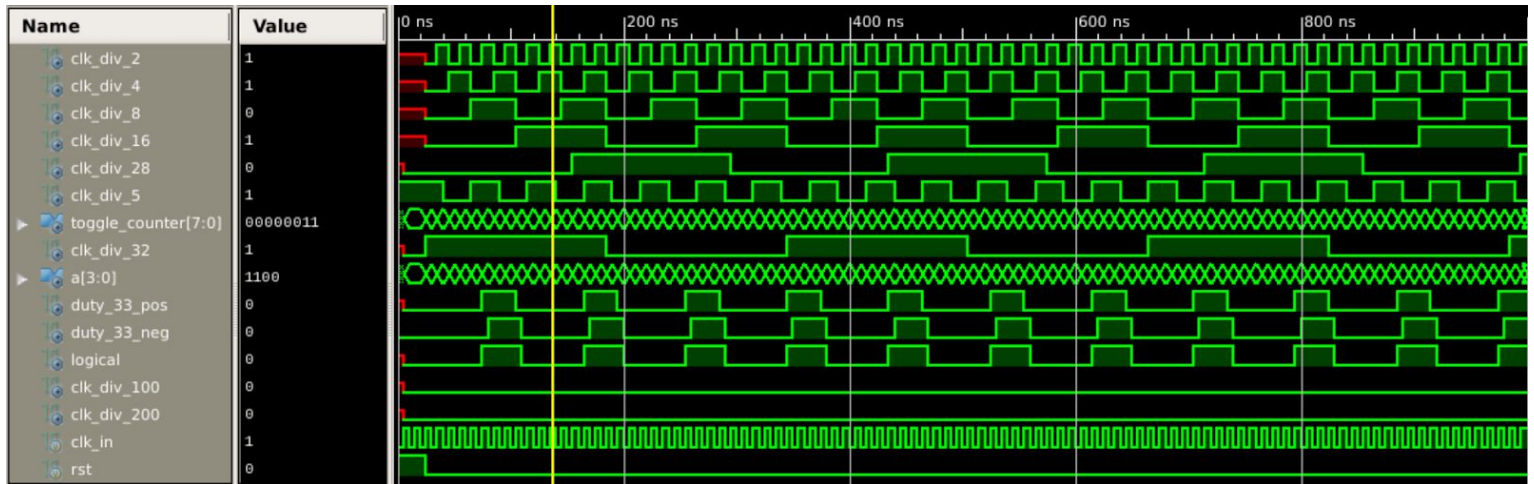
```
Phase 6 : 0 unrouted; (Par is working to improve performance)  REAL time: 3 secs
```

```
Phase 7 : 0 unrouted; (Par is working to improve performance)  REAL time: 3 secs
```

Place and Route Report:

While the other information given by the Place and Route Report overlaps with the HDL Synthesis Report, the Design Summary Report, and the Map Report, the Place and Route Report alone gives us information about the phases taken to route, and the REAL time it took. For this specific design, routing was completed in 4 phases, and 11 seconds REAL time.





Simulation Waveforms:

The three waveforms above include outputs from design tasks and verification tasks. I was able to test my design using the waveforms by visualizing each signal's period, and figuring out its frequency and duty cycle.

Conclusion

The design of clock_gen includes 4 submodules for design tasks and 5 submodules for verification tasks. The submodules have little communication with each other, since they do not work alongside each other to create a functional program — they are each doing simple tasks regarding the clk_in input.

The most difficult part of this project is working with possible delays regarding if statements using the 4-bit counter, and figuring out what the length of the period needed to be to satisfy the divide-by-x requirement.

I cannot think of ways to improve the overall lab experience. It was succinct and well-explained, and the TA's preemptive advice and pointers were highly useful when implementing the difficult features of the lab.