Danning Yu, 305087992
CS M152A, Lab 2
TA: Logan Kuo

Project 1 Report

Introduction

The purpose of this lab is to introduce us to the use of Verilog as a hardware description language, the use of the Xilinx ISE as a software suite for the simulation and synthesis of digital circuits, and the FPGA design process as a whole. A simple combinational module consisting of basic logic gates and a multiplexer selector will be implemented in Verilog, as well as a simple sequential 4-bit counter using both logical gates and Verilog arithmetic. Finally, a counter is used to create a 10 kHz to 1 Hz frequency divider. After coding the designs, test benches will be created to verify the functionality of all modules to run simulations in the Xilinx Isim software, schematics will be generated using the RTL schematic tool, and synthesis and mapping reports will be examined.

Design

Combinational Circuit

The combinational circuit consists of a 5-bit input called `sw` ("switch"), where the lower 2 bits are used as inputs to 8 unary or binary input logic gates, while the upper 3 bits of the input are used as the selection bits for an 8-to-1 multiplexer that takes in the outputs of the 8 logical gates and forwards one of them depending on the value of the selection bits. The gates that are implemented are NOT, non-inverting buffer, XNOR, XOR, OR, NOR, AND, and NAND. The NOT and non-inverting buffer gates take in the `sw[0]`, lowest bit of the module input, while the other 6 gates also take in the second lowest bit of the module input, `sw[1]`. The inputs and outputs are summarized in the table below.

Input/Output		Description
`sw[4:0]`	`sw[0]`	Input: an input for all 8 logical gates
	`sw[1]`	Input: a second input for the 6 logical gates that take 2 inputs
	`sw[4:2]`	Input: selection bits for the multiplexer, determines which logic gate signal to forward
`result`		Output: the result of a particular logical gate selected by the selection bits

Table 1: A summary of the input and outputs of the combination module and their functions. If we had access to an FPGA board, the inputs would be connected to a physical switch and the output connected to something that could display its value.

At the high level, the circuit can be thought of as containing a logic gate module containing all of the logic gates, and it outputs to a multiplexer, as shown in the figure below. The figures after it show gate level schematics of the circuit provided by the project specification and generated by Xilinx ISE after synthesis.

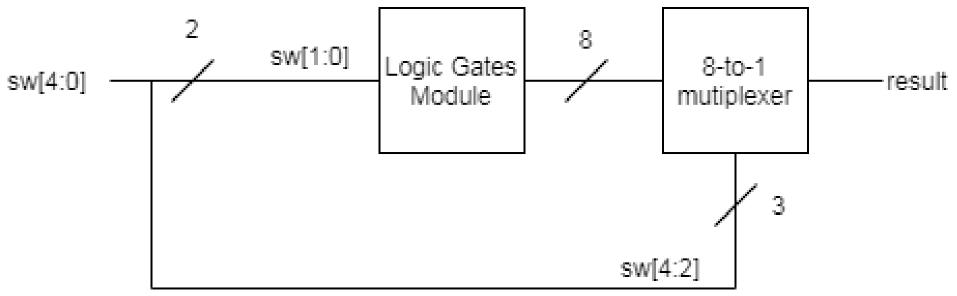

Figure 1: High level overview of the combinational module. The 5 bit input `sw` is split into a 2 bit input for the logic gates module, which has an 8 bit output, where each bit corresponds to a particular logic gate. The 8 bit output feeds into an 8-to-1 multiplexer, and the selected input is determined by the other 3 bits of the input and emerges as the 1 bit output `result`.

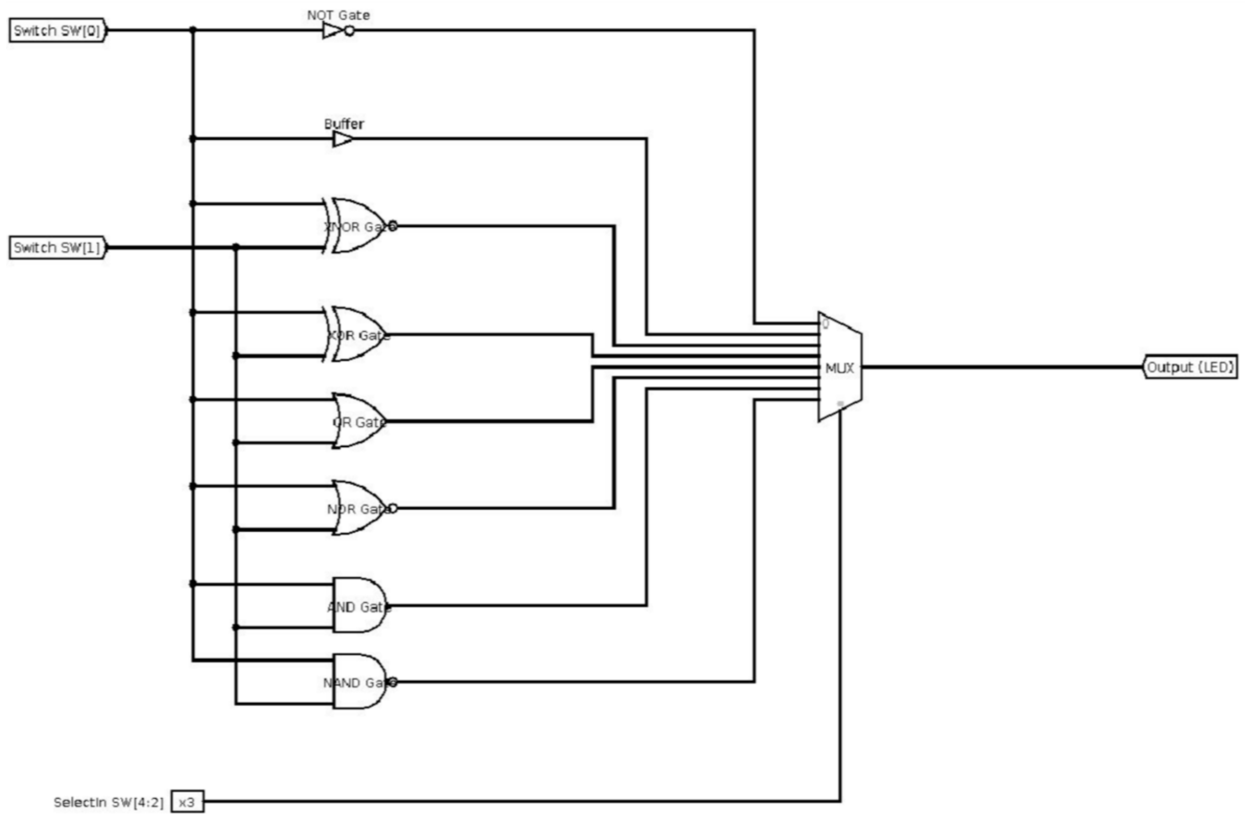

Figure 2: The combinational circuit to be implemented, at the gate level. From top to bottom, the gates are NOT, non-inverting buffer, XNOR, XOR, OR, NOR, AND, and NAND, and they all feed into an 8-to-1 multiplexer. Picture courtesy of the project 1 assignment document.

Note: Because we do not have access to a physical FPGA board this quarter, the output is not connected a physical LED, and the switches are not actually connected to physical switches.

Using Figure 2, a Verilog module was written. In the code below, excerpted from the module, where `sw[4:0]` is an input and `result` is an output register, one can see a case statement was used, where based on the value of `sw[4:2]`, specific logical operations were carried out with `sw[0]` and `sw[1]` whenever the value of `sw[4:0]` changed.

```
----------------------Combinational Module Code----------------------
always @(sw)
    begin
      case (sw[4:2])
        3'b000: result <= ~sw[0];
        3'b001: result <= sw[0];
        3'b010: result <= ~(sw[0] ^ sw[1]);
        3'b011: result <= sw[0] ^ sw[1];
        3'b100: result <= sw[0] | sw[1];
        3'b101: result <= ~(sw[0] | sw[1]);
        3'b110: result <= sw[0] & sw[1];
        3'b111: result <= ~(sw[0] & sw[1]);
      endcase
    end
---------------------------------------------------------------------
```

After this module underwent the synthesis process, a RTL schematic was created, which is shown on the next page.

(project report continues on next page)

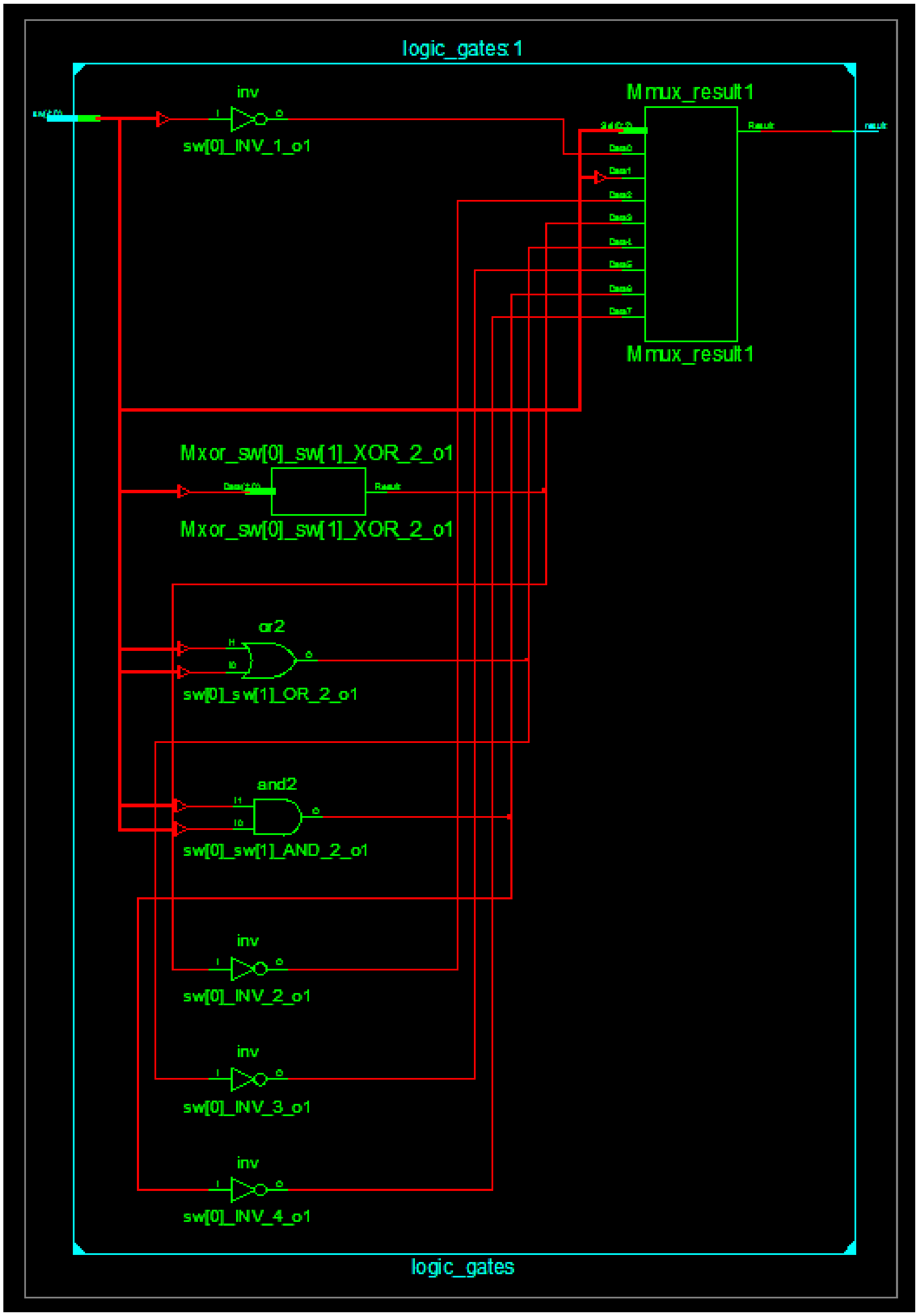

Figure 3: RTL synthesis diagram generated by the Xilinx ISE after synthesis. It is extremely similar to the provided circuit, except that it only uses NOT, AND, OR, and XOR gates, with the output signal being split to provide the non-inverted and inverted forms (for example, the output of one AND gate is used to create the AND and NAND outputs).

Sequential Circuit: 4 Bit Counter

The sequential circuit is a simple 4 bit counter, with a clock input that drives the counter and a synchronous reset for resetting it to 4’b0000. The reset is also used to initialize the counter to 4’b0000. It was implemented in 2 ways: first, using gates and D flip-flops, and then using the Verilog HDL.

The 4 bit counter implemented using logic gates consists of 4 D flip-flops, for storing the 4 bits of the counter. The clock and reset inputs (written as `clk` and `rst` in the code) feed into these flip-flops, and their outputs undergo combinational logic based on the state diagram below to provide the appropriate value to be loaded into the flip-flop at the next clock cycle. The resulting circuit is shown in Figure 5. To copy the design in Figure 5 over using Verilog, only the ~, &, and ^ operators were used, along with 4 1 bit registers representing the 4 D flip-flops.

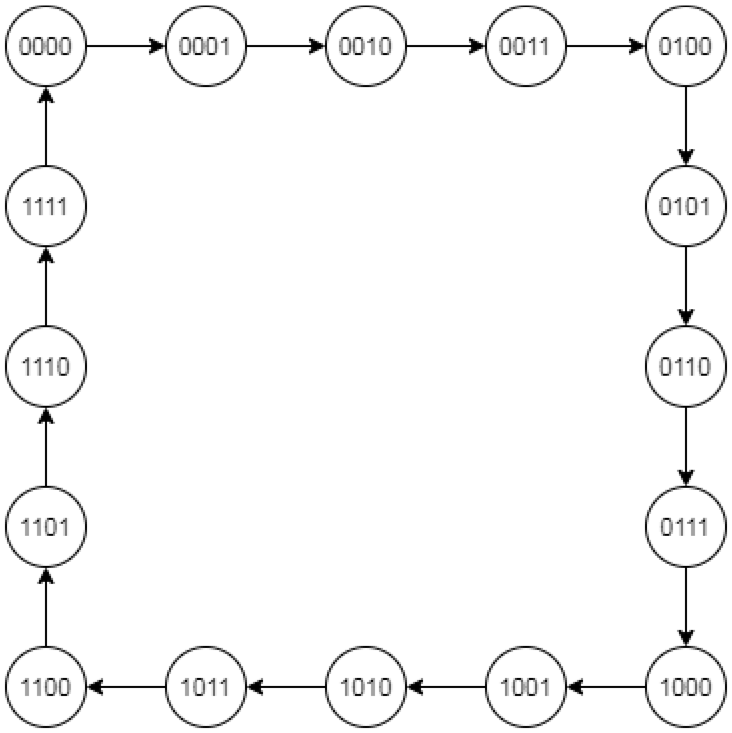

Figure 4: State diagram for a 4 bit counter. Each transition causes the value to increase by 1, except for when the counter reaches 4’b1111, at the next clock cycle, it resets to 4’b0000.

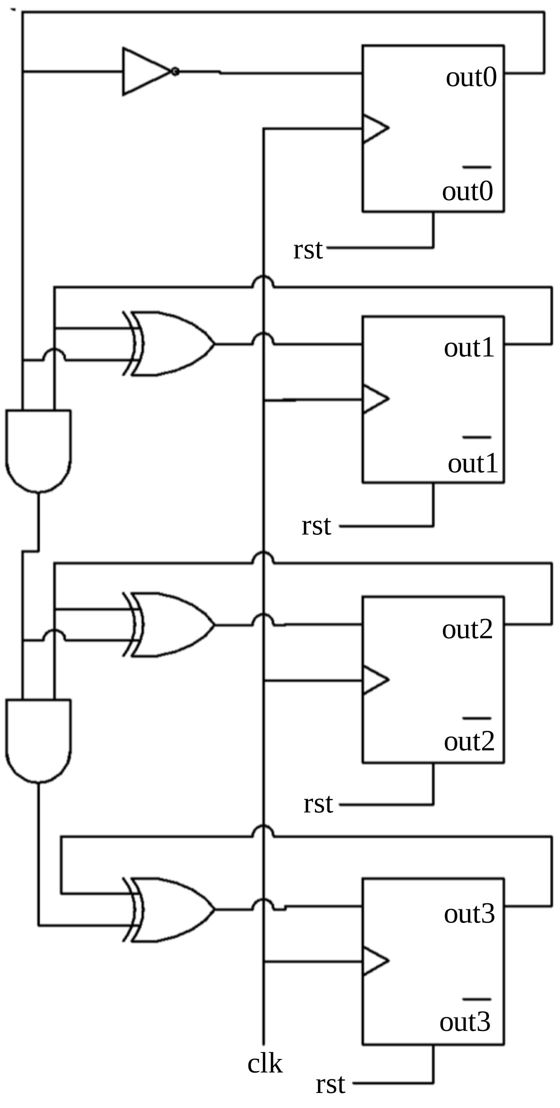

Figure 5: Implementation of a 4 bit counter using logic gates and D flip-flops, with clock and reset inputs. The value `out0` represents the least significant bit of the 4 bit counter, while `out3` represents the most significant bit. The `clk` input drives the circuit, and the `rst` signals zeroes out all the D flip flops.

Using the implementation shown in Figure 5, the following Verilog module on the next page was written. The values `clk` and `rst` are input wires, while `out0`, `out1`, `out2`, and `out3` are output registers. At every positive edge of the clock, a reset occurs, or the values in the output registers are updated according to Figure 5.

(project report continues on next page)

```
--------------------4 Bit Counter Using Logic Gates-------------------
always @(posedge clk)
    begin
      if(rst)
        begin
        out0 <= 1'b0;
        out1 <= 1'b0;
        out2 <= 1'b0;
        out3 <= 1'b0;
        end
      else
        begin
          out0 <= ~out0;
          out1 <= out0 ^ out1;
          out2 <= (out0 & out1)^out2;
          out3 <= ((out0 & out1)& out2)^ out3;
        end
    end
----------------------------------------------------------------------
```

The resulting RTL schematic can be found on the next page.

(project report continues on next page)

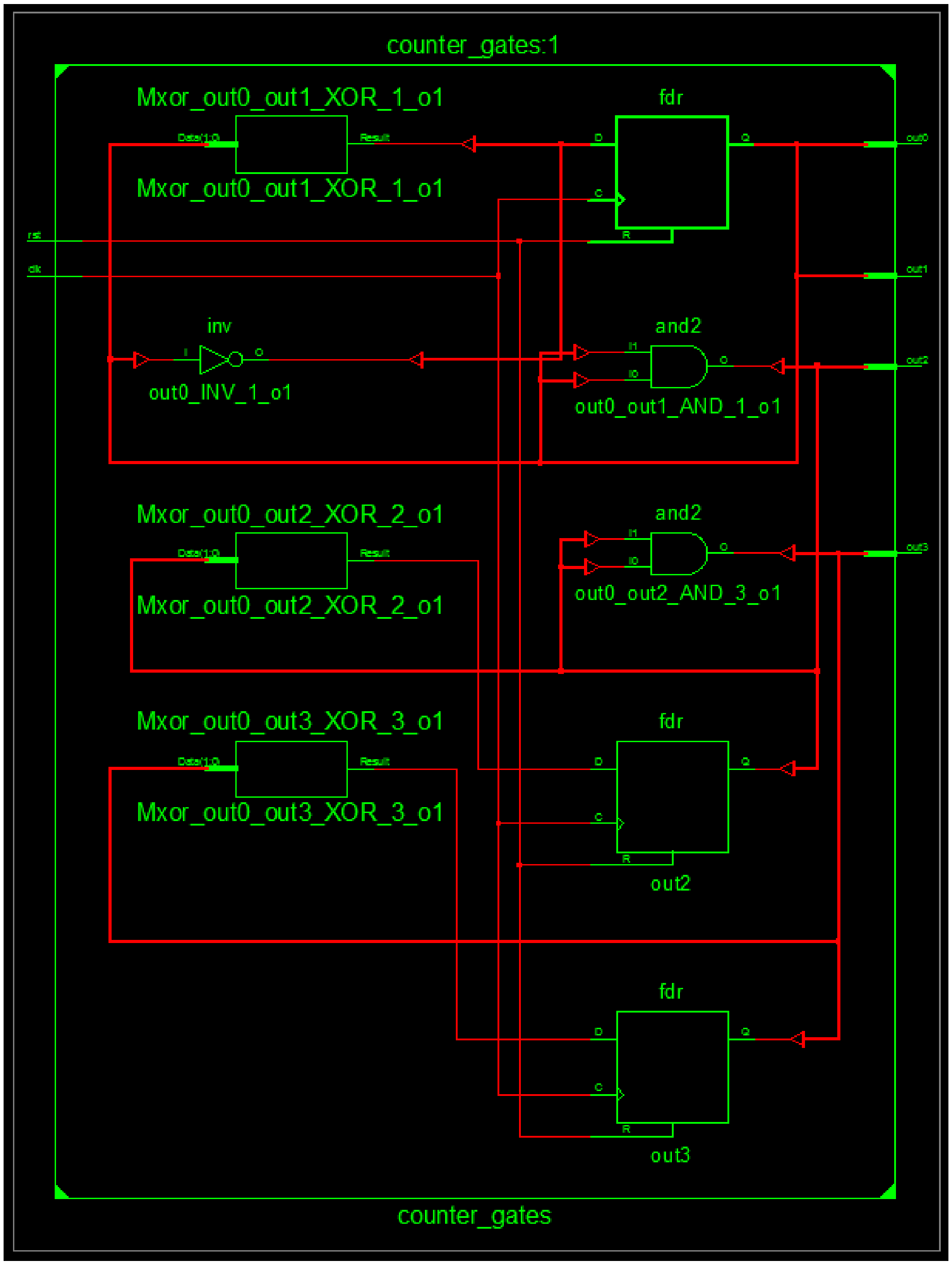

Figure 6: RTL schematic of 4 bit counter from Figure 5 after synthesis. The design is very similar to what was used to create the schematic, but there have been some optimizations, such as only using 3 D flip-flops to store the values. All the original logic gates are present.

The 4 bit counter was then implemented using the Verilog language, which is shown below. The code below provides the same behavior and functionality as the 4 bit counter implemented using logic gates, but in a more compact and simpler manner in terms of how much code is needed. It takes two inputs, a clock `clk` and reset `rst`, and has a 4 bit output register `out[3:0]`. At every positive edge of the clock, if reset is enabled, the values in the register are zeroed; otherwise, the values are incremented by 4'b1. I named it "counter_arith" to distinguish it from the previous counter, named "counter_gates," as this one is implemented using Verilog's arithmetic operations, while the previous counter was implemented using gates. The RTL schematic is shown in Figure 7.

```
------------------------Verilog 4 Bit Counter-----------------------
  always @(posedge clk)
    begin
      if(rst)
        out <= 4'b0;
      else
        out <= out + 4'b1;
    end
--------------------------------------------------------------------
```

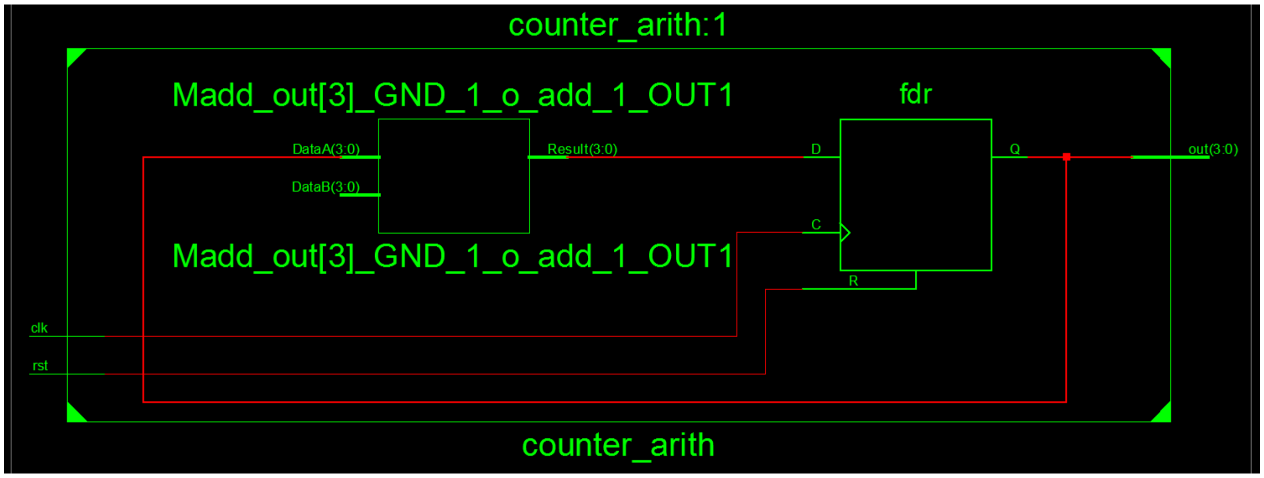

Figure 7: RTL schematic of 4 bit counter from the code above after synthesis. Because gates were not used, the schematic also abstracts away the gates, presenting it only as an adder combined with a 4 bit register.

Frequency Divider

The frequency divider takes in a 10 kHz input clock signal and divides it so that it outputs a 1 Hz signal, where the output signal is on for 0.5 seconds and off for 0.5 seconds (50% duty cycle). This is done by connecting the input clock signal to a counter that counts from 0 to 9999. When the counter reaches 9999, it goes back to 0 and sets the output signal to 1, and then when it hits 4999, it sets the output signal to 0. The result is an output signal that stays high for 5000 cycles and then low for 5000 cycles, which for a 10 kHz input clock corresponds to 0.5 seconds on and off, as desired. A synchronous reset signal is also provided to initialize the counter and output to 0. The design is shown in the Verilog code below, which has a clock `clk` and reset `rst` inputs, as well as a one bit output `out`. Inside the module, a 14 bit register `value` is declared to store the value of the counter. At the positive edge of the clock, if `rst` is enabled, `out` and `value` are zeroed out. If `value` is 9999, then it is be set to 0, as (9999+1)mod10000 = 0;

otherwise, `value` is incremented by 1. At the same time, if `value` is 9999, `out` is set to 1, and if `value` is 4999, then `out` is set to 0.

```
------------------------Frequency Divider Code------------------------
reg [13:0] value; //14 bits b/c 9999 = 14'b10011100001111

always @(posedge clk)
  begin
    if(rst)
      value <= 14'd0;
    else if(value == 14'd9999)
      value <= 14'd0;
    else
      value <= value + 14'd1;
  end

always @(posedge clk)
  begin
    if(rst)
      out <= 1'b0;
    else if(value == 14'd9999)
      out <= 1'b1;
    else if(value == 14'd4999)
      out <= 1'b0;
  end
--------------------------------------------------------------------
```

When synthesized, the resulting schematic is given in the following figure.

(project report continues on next page)

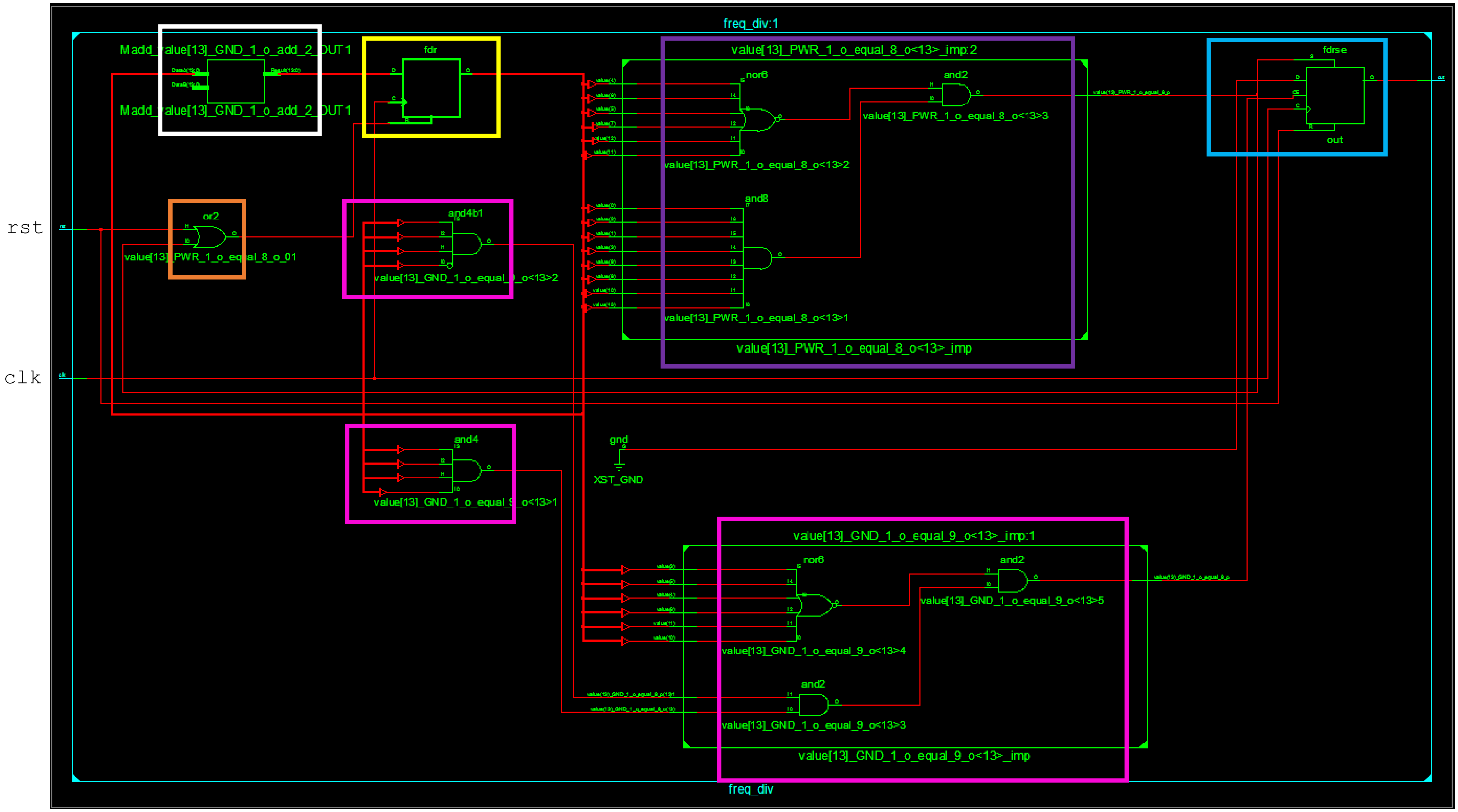
freq_div:1
Madd_value[13]_GND_1_o_add_2_OUT1
fdr
value[13]_PWR_1_o_equal_8_o<13>_imp:2
nor6
and2
value[13]_PWR_1_o_equal_8_o<13>3
value[13]_PWR_1_o_equal_8_o<13>2
fdrse
out
rst
or2
value[13]_PWR_1_o_equal_8_o_01
and4b1
value[13]_GND_1_o_equal_9_o<13>2
and8
value[13]_PWR_1_o_equal_8_o<13>1
value[13]_PWR_1_o_equal_8_o<13>_imp
clk
and4
value[13]_GND_1_o_equal_9_o<13>1
gnd
XST_GND
value[13]_GND_1_o_equal_9_o<13>_imp:1
nor6
and2
value[13]_GND_1_o_equal_9_o<13>5
value[13]_GND_1_o_equal_9_o<13>4
and2
value[13]_GND_1_o_equal_9_o<13>3
value[13]_GND_1_o_equal_9_o<13>_imp
freq_div

Figure 8: RTL schematic of the frequency divider. There is a 14 bit register to store the 14-bit result of the counter (yellow box, `value` in the code), as well as a 14-bit adder to increment `value` (white box) and a register to store the current value of the output (blue box, `out` in the code). The violet box is for testing if `value` is 9999, and the orange boxes test if `value` is 4999. The line remaining OR gate in orange is for zeroing out the 14 bit register, either when the `rst` signal is active or `value` is 9999.

Simulation Documentation

Combinational Circuit

The requirement for the combinational circuit is that it outputs the correct value given any particular 5 bit input. Thus, I tested all possible 5 bit inputs, from 5'b00000 to 5'b11111 to verify that the output was correct. This was done in a test bench (code given below) that initialized `sw` to 5'b00000, and every 10 ns, it was incremented by 1 in an always block so that it would eventually pass through every value from 5'b00000 to b'11111. The always block eliminates the need to manually type in all 32 possible values for `sw[4:0]`. The truth table is given in Table 2, followed by a code extract from the test bench and simulation results in Figures 9 and 10, which match the truth table and thus verify the implementation.

```
--------------Test Bench Code for Combinational Circuit---------------
initial begin
  // Initialize sw to 0
  sw[4:0] = 5'b0;
  #100;
end

always
  begin
    #10;
    sw[4:0] = sw[4:0]+5'b1; //increment by 1 in 10 ns increments
  end
----------------------------------------------------------------------
```

sw[4:2]	sw[1]	sw[0]	Gate	result	sw[4:2]	sw[1]	sw[0]	Gate	result
000	0	0	NOT	1	100	0	0	OR	0
000	0	1		0	100	0	1		1
000	1	0		1	100	1	0		1
000	1	1		0	100	1	1		1
001	0	0	Buffer	0	101	0	0	NOR	1
001	0	1		1	101	0	1		0
001	1	0		0	101	1	0		0
001	1	1		1	101	1	1		0
010	0	0	XNOR	1	110	0	0	AND	0
010	0	1		0	110	0	1		0
010	1	0		0	110	1	0		0
010	1	1		1	110	1	1		1
011	0	0	XOR	0	111	0	0	NAND	1
011	0	1		1	111	0	1		1
011	1	0		1	111	1	0		1
011	1	1		0	111	1	1		0

Table 2: Truth table for a 5-bit input combinational module. The 5 bit input is divided into the least significant bit, second lowest bit, and three highest bits for easier readability, along with the gate that particular set of inputs corresponds to.

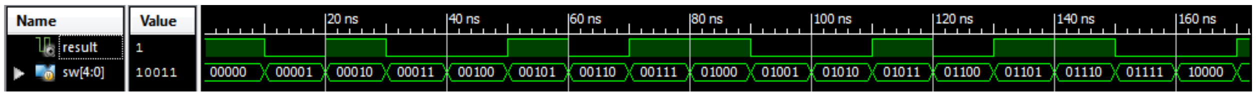

Figure 9: Simulation waveform results for inputs from 5'b00000 to 5'b01111. This is a combinational circuit, so there is no need for a clock. All inputs matched what was expected, thus verifying the design for inputs from 5'b00000 to 5'b01111.

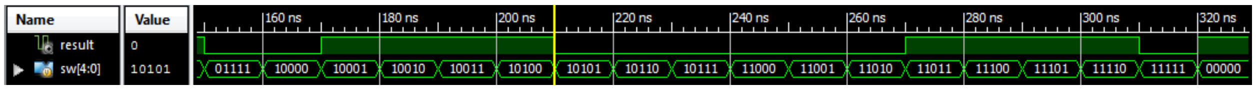

Figure 10: Simulation waveform results for inputs from 5'b10000 to 5'b11111. This is a combinational circuit, so there is no need for a clock. All inputs matched what was expected, thus verifying the design for inputs from 5'b10000 to 5'b11111.

After synthesis and implementation, the reports indicated no registers and 1 lookup table (LUT) was used. 1 of 9112 slice LUTs were used after synthesis, all for logic. Overall, the LUT and slice register usage is very low, indicating this design is feasible to implement on the actual board. Detailed extracts from the synthesis and implementation reports are attached to this report in Appendix A.

Sequential Circuit: 4 Bit Counter

The requirements for both the counter implemented using gates/flip-flops and the counter implemented using Verilog language is that they follow the state machine in Figure 4, transitioning from one state to the other at every positive edge of the clock and synchronously going to 4'b0000 when `rst` is 1. I combined the two counters into one test bench, as they should provide the same output. The code for this test bench was very simple, where reset was initialized to 1, and then set to 0 100 ns later. A perpetually running 50 MHz clock was used to provide stimulus to the counter. Examining the results in Figures 11 and 12 show that they match the behavior of the state machine in Figure 4, thus verifying the design.

```
-------------------Test Bench Code for 4 Bit Counter------------------
initial begin
  // Initialize Inputs
  clk = 0;
  rst = 1;
  #100;
  rst = 0;
end

always
  begin
    clk = ~clk;
    #10;
  end
----------------------------------------------------------------------
```

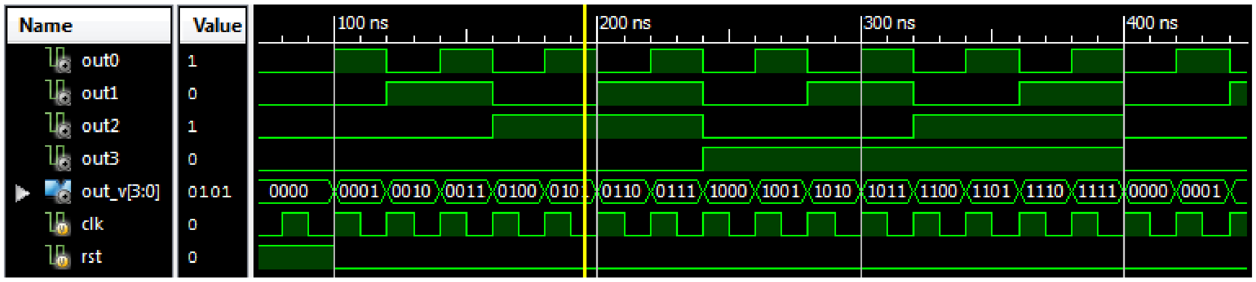

Figure 11: Waveform simulation results for the 4 bit counter implemented using logic gates (`out0`, `out1`, `out2`, and `out3`) and Verilog (`out_v[3:0]`). The inputs are `clk` and `rst`. The clock has a frequency of 50 MHz, and the simulation ran for 550 ns, allowing for the counter to visit all possible states and transitions. The waveforms and state transitions match what is expected, thus verifying the design and Verilog code.

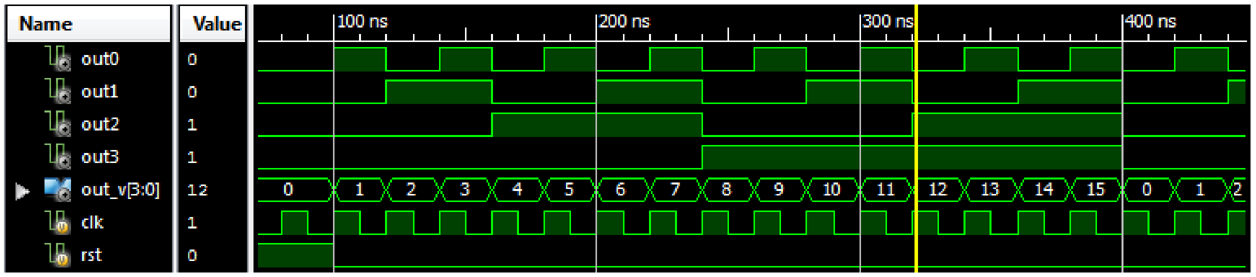

Figure 12: The same simulation as depicted in Figure 9, but with the output of the 4-bit counter implemented in Verilog depicted in unsigned decimal, confirming that it counts from 0 to 15, and when it reaches 15, its next value is 0.

For both counter implementations, the register and LUT usage was the same. After synthesis and implementation, the reports indicated 4 registers and 4 LUTs were used. Specifically, 4 out of 18224 slice registers were used (all for flip flops) to form a 4 bit register. 4 of 9112 slice LUTs were used after synthesis, all for logic. Initially, after synthesis, 8 LUT flip flop pairs were used, where 4 had an unused flip flop and 4 had an unused LUT, but after mapping, it was consolidated down into 4 LUT flip flop pairs that had both their LUT and flip flop used. A 4 bit counter was also used. Overall, the LUT and register usage is very low, indicating this design is feasible to implement on the actual board. Detailed extracts from the synthesis and implementation reports are attached to this report in Appendix B (for the implementation using gates and 1 bit registers) and Appendix C (for the implementation using higher level Verilog abstractions).

Frequency Divider

The requirement of this module is that it divides a 10 kHz input clock signal into a 1 Hz output signal. The code for this test bench has reset `rst` being initialized to 1 to reset the counter, and then a clock `clk` that inverts the signal every 50,000 ns, resulting in a 10 kHz clock. Examining the simulation results in Figures 13 and 14 and measuring the period of the signals shows the input clock has a frequency of 10 kHz and the output signal has a frequency of 1 Hz, thus verifying the design.

```
-----------------Test Bench Code for Frequency Divider----------------
initial begin
  // Initialize Inputs
  clk = 1'b0;
  rst = 1'b1;
  #50100;
  rst = 1'b0;
  end

always
  begin
    #50000; //creates a 10 kHz clock
    clk = ~clk;
  end
----------------------------------------------------------------------
```

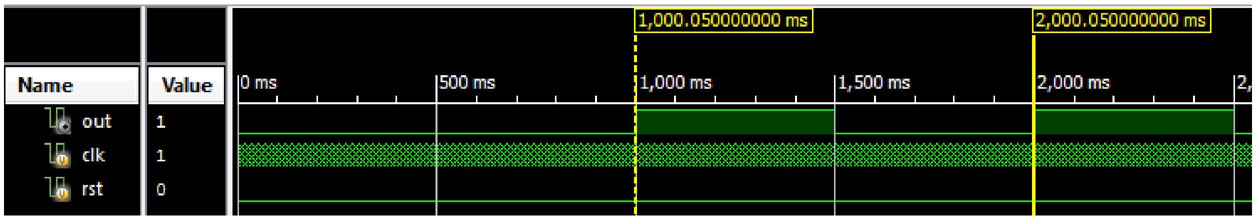

Figure 13: Waveform simulation result for the 10,000-to-1 frequency divider implemented using a 14 bit counter. The inputs are `clk` and `rst`, with the `clk` having a frequency of 10 kHz. The cursors, located at two successive rising edges of the output signal, have times of 1000.05 ms and 2000.05 ms, thus showing the output signal has a frequency of 1 Hz, as desired.

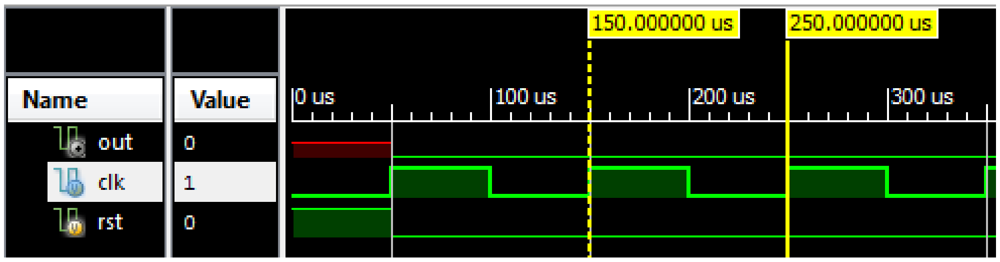

Figure 14: Waveform simulation results from the same test run as used to create Figure 11a, but zoomed in on the clock signal. The two cursors located at two successive edges of the `clk` signal are at 150 µs and 250 µs, confirming that the input clock has a frequency of 10 kHz.

After synthesis and implementation, the reports indicated 34 LUTs and 15 registers were used. Specifically, 15 out of 18224 slice registers were used (all for flip flops) along with 34 of 9112 slice LUTs. Of these 34 slice LUTs, 19 had an unused flip flop and 15 had a fully used LUT-FF pair. In addition, of the 34 slice LUTs, 33 were used for logic, while 1 was used for route-thrus only. Registers were grouped into a 1 bit register (for `out`) and a 14 bit register (for the internal counter named `value`). A 14 bit counter was also used. Overall, the LUT and slice register usage is very low, indicating this design is feasible to implement on the actual board. Detailed extracts from the synthesis and implementation reports are attached to this report in Appendix D.

Conclusion

In this lab, a combinational module consisting of 8 basic logic gates and an 8-to-1 multiplexer, 4 bit counter consisting of logic gates and D flip-flops, 4 bit counter using Verilog language, and 10 kHz to 1 Hz frequency divider were implemented and tested. In the process, I learned how to code modules and test benches in Verilog, run simulations in Isim, synthesize designs, and view RTL schematics. The simulations showed expected behavior, thus validating my designs.

The main difficulty I encountered with this project was figuring out how to set up the test benches, especially the 4 bit counter so that I could test both types of the 4 bit counter in one simulation, as I implemented them in different modules. However, the TA provided an example of this in class, so I went back and re-watched the lecture videos to see how it was done. Overall, I would say the lab was a good introduction to Verilog and the Xilinx ISE suite. The project guidelines, combined with the lab sessions and FPGA and Verilog resources provided on CCLE, were enough to complete this lab without too much effort. Thus, I don't have any suggestions for improving the lab.

Supplemental Question

What is an "`.ucf`" file? How would you use it for Nexys3 in an actual setting on a real board to connect the inputs of the combinational circuitry to the switches on the FPGA?

A `.ucf` file, which stands for user constraint file, is used to map inputs and outputs specified in a Verilog module to particular physical pins on the FPGA board. It is specific to each board, in this case a Nexys3 board for this class. Thus, for connecting the inputs of combinational circuity to the FPGA switches, you would add the Nexys3 `.ucf` file to your Xilinx workbench, uncomment the lines for the switches you are using in the `.ucf` file, and then use the names of those switches in the combinational circuit module written in Verilog. Then, you would synthesize the design and implement it (consisting of, but not limited to translation, mapping, placement, and routing), and Xilinx will automatically match the inputs to the physical pins in the programming file that it generates.

Appendix

Note: Only sections pertaining to slice registers, slice LUTs, and gates are included. Duplicate information within the same report may be omitted (for example, if “HDL Synthesis Report” and “Advanced HDL Synthesis Report” are the same).

Appendix A: Synthesis and Map Reports for Logic Gates Module

Synthesis Report

```
=========================================================================
*                           HDL Synthesis                               *
=========================================================================
HDL Synthesis Report
Macro Statistics
# Multiplexers                                        : 1
 1-bit 8-to-1 multiplexer                             : 1
# Xors                                                : 1
 1-bit xor2                                           : 1

=========================================================================
*                            Design Summary                             *
=========================================================================
Top Level Output File Name         : logic_gates.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 1
#      LUT5                        : 1
# IO Buffers                       : 6
#      IBUF                        : 5
#      OBUF                        : 1

Device utilization summary:
---------------------------
Selected Device : 6slx16csg324-3

Slice Logic Utilization:
 Number of Slice LUTs:                    1  out of   9112     0%
    Number used as Logic:                 1  out of   9112     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:      1
   Number with an unused Flip Flop:       1  out of      1   100%
   Number with an unused LUT:             0  out of      1     0%
   Number of fully used LUT-FF pairs:     0  out of      1     0%
   Number of unique control sets:         0
```

Map Report

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                     0 out of  18,224    0%
  Number of Slice LUTs:                          1 out of   9,112    1%
    Number used as logic:                        1 out of   9,112    1%
      Number using O6 output only:               1
      Number using O5 output only:               0
      Number using O5 and O6:                    0
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   2,176    0%

Slice Logic Distribution:
  Number of occupied Slices:                     1 out of   2,278    1%
  Number of MUXCYs used:                         0 out of   4,556    0%
  Number of LUT Flip Flop pairs used:            1
    Number with an unused Flip Flop:             1 out of       1  100%
    Number with an unused LUT:                   0 out of       1    0%
    Number of fully used LUT-FF pairs:           0 out of       1    0%
    Number of slice register sites lost
      to control set restrictions:               0 out of  18,224    0%
```

Appendix B: Synthesis and Map Reports for 4 Bit Counter (Gates & Registers Version)

Synthesis Report

```
======================================================================
*                           HDL Synthesis                            *
======================================================================
HDL Synthesis Report

Macro Statistics
# Registers                                                 : 3
 1-bit register                                             : 2
 2-bit register                                             : 1
# Xors                                                      : 3
 1-bit xor2                                                 : 3

======================================================================
*                       Advanced HDL Synthesis                       *
======================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Registers                                                 : 4
 Flip-Flops                                                 : 4
```

```
# Xors                                                  : 3
 1-bit xor2                                             : 3

======================================================================
*                         Low Level Synthesis                        *
======================================================================
Final Register Report

Macro Statistics
# Registers                                             : 4
 Flip-Flops                                             : 4

======================================================================
*                            Design Summary                          *
======================================================================
Top Level Output File Name         : counter_gates.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 4
#      INV                         : 1
#      LUT2                        : 1
#      LUT3                        : 1
#      LUT4                        : 1
# FlipFlops/Latches                : 4
#      FDR                         : 4
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 5
#      IBUF                        : 1
#      OBUF                        : 4

Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3

Slice Logic Utilization:
 Number of Slice Registers:                 4  out of  18224     0%
 Number of Slice LUTs:                      4  out of   9112     0%
    Number used as Logic:                   4  out of   9112     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:        8
   Number with an unused Flip Flop:         4  out of      8    50%
   Number with an unused LUT:               4  out of      8    50%
   Number of fully used LUT-FF pairs:       0  out of      8     0%
   Number of unique control sets:           1
```

Map Report

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                     4 out of  18,224    1%
    Number used as Flip Flops:                   4
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                          4 out of   9,112    1%
    Number used as logic:                        4 out of   9,112    1%
      Number using O6 output only:               4
      Number using O5 output only:               0
      Number using O5 and O6:                    0
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   2,176    0%

Slice Logic Distribution:
  Number of occupied Slices:                     1 out of   2,278    1%
  Number of MUXCYs used:                         0 out of   4,556    0%
  Number of LUT Flip Flop pairs used:            4
    Number with an unused Flip Flop:             0 out of       4    0%
    Number with an unused LUT:                   0 out of       4    0%
    Number of fully used LUT-FF pairs:           4 out of       4  100%
    Number of unique control sets:               1
    Number of slice register sites lost
      to control set restrictions:               4 out of  18,224    1%
```

Appendix C: Synthesis and Map Reports for 4 Bit Counter (Verilog HDL Version)

Synthesis Report

```
======================================================================
*                          HDL Synthesis                             *
======================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                    : 1
 4-bit adder                                            : 1
# Registers                                             : 1
 4-bit register                                         : 1

======================================================================
*                      Advanced HDL Synthesis                        *
======================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Counters                                          : 1
 4-bit up counter                                   : 1

=========================================================================
*                          Low Level Synthesis                          *
=========================================================================
Final Register Report

Macro Statistics
# Registers                                         : 4
 Flip-Flops                                         : 4

=========================================================================
*                             Design Summary                            *
=========================================================================

Top Level Output File Name         : counter_arith.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 4
#      INV                         : 1
#      LUT2                        : 1
#      LUT3                        : 1
#      LUT4                        : 1
# FlipFlops/Latches                : 4
#      FDR                         : 4
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 5
#      IBUF                        : 1
#      OBUF                        : 4

Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice Registers:               4  out of  18224     0%
 Number of Slice LUTs:                    4  out of   9112     0%
    Number used as Logic:                 4  out of   9112     0%
```

```
Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:      8
   Number with an unused Flip Flop:       4  out of      8    50%
   Number with an unused LUT:             4  out of      8    50%
   Number of fully used LUT-FF pairs:     0  out of      8     0%
   Number of unique control sets:         1
```

Map Report

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                     4 out of  18,224    1%
    Number used as Flip Flops:                   4
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                          4 out of   9,112    1%
    Number used as logic:                        4 out of   9,112    1%
      Number using O6 output only:               4
      Number using O5 output only:               0
      Number using O5 and O6:                    0
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   2,176    0%

Slice Logic Distribution:
  Number of occupied Slices:                     1 out of   2,278    1%
  Number of MUXCYs used:                         0 out of   4,556    0%
  Number of LUT Flip Flop pairs used:            4
    Number with an unused Flip Flop:             0 out of       4    0%
    Number with an unused LUT:                   0 out of       4    0%
    Number of fully used LUT-FF pairs:           4 out of       4  100%
    Number of unique control sets:               1
    Number of slice register sites lost
      to control set restrictions:               4 out of  18,224    1%
```

Appendix D: Synthesis and Map Reports for 10 kHz to 1 Hz Frequency Divider

Synthesis Report

```
======================================================================
*                          HDL Synthesis                             *
======================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 1
 14-bit adder                                          : 1
```

```
# Registers                                          : 2
 1-bit register                                      : 1
 14-bit register                                     : 1

======================================================================
*                      Advanced HDL Synthesis                        *
======================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Counters                                           : 1
 14-bit up counter                                   : 1
# Registers                                          : 1
 Flip-Flops                                          : 1

======================================================================
*                       Low Level Synthesis                          *
======================================================================
Final Register Report

Macro Statistics
# Registers                                          : 15
 Flip-Flops                                          : 15

======================================================================
*                          Design Summary                            *
======================================================================

Top Level Output File Name         : freq_div.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 63
#      GND                         : 1
#      INV                         : 1
#      LUT1                        : 13
#      LUT3                        : 2
#      LUT4                        : 1
#      LUT6                        : 17
#      MUXCY                       : 13
#      VCC                         : 1
#      XORCY                       : 14
# FlipFlops/Latches                : 15
#      FD                          : 14
#      FDR                         : 1
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 2
#      IBUF                        : 1
#      OBUF                        : 1

Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice Registers:             15  out of  18224     0%
 Number of Slice LUTs:                  34  out of   9112     0%
    Number used as Logic:               34  out of   9112     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     34
   Number with an unused Flip Flop:     19  out of     34    55%
   Number with an unused LUT:            0  out of     34     0%
   Number of fully used LUT-FF pairs:   15  out of     34    44%
   Number of unique control sets:        2
```

Map Report

```
Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:                    15 out of  18,224    1%
    Number used as Flip Flops:                  15
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                         34 out of   9,112    1%
    Number used as logic:                       33 out of   9,112    1%
      Number using O6 output only:              20
      Number using O5 output only:              12
      Number using O5 and O6:                    1
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   2,176    0%
    Number used exclusively as route-thrus:      1
      Number with same-slice register load:      0
      Number with same-slice carry load:         1
      Number with other load:                    0

Slice Logic Distribution:
  Number of occupied Slices:                     9 out of   2,278    1%
  Number of MUXCYs used:                        16 out of   4,556    1%
  Number of LUT Flip Flop pairs used:           34
    Number with an unused Flip Flop:          19 out of      34   55%
    Number with an unused LUT:                 0 out of      34    0%
    Number of fully used LUT-FF pairs:        15 out of      34   44%
    Number of unique control sets:             2
    Number of slice register sites lost
      to control set restrictions:             9 out of  18,224    1%
```