**Sequential Logic Design with Counters in Verilog Design Report**

Benjamin Betancourt

Lab Dates: February 10th – February 18th, 2025

Date of Submission: February 17th, 2025

Table of Contents

**Introduction3**

**Implementation3-5**

**Results5-10**

**Conclusion11**

***Introduction:***

This experiment seeks to act as an introduction to sequential logic design and its challenges in Verilog. The main difference between sequential logic and combinational logic is the use of time as a factor. This means that previous circuit states are a factor in the current state. This introduces additional design considerations such as clock domains, old states, and gate delay. These complicate the design process therefore making the overall design harder to implement than combinational designs are. Sequential design was introduced thriygh the requirement of implementing a counter with functions. This counter is four bits wide meaning that it can take on 16 states. This means that hexadecimal was used to demonstrate the current state. Additional features including a switch direction, reset, set with four bit input, and clock switch were also implemented into the design adding design complexity. Overall, this design project sets a good foundation for future sequential logic designs.

***Implementation:***

To start, a clock divider needed to be created. The max10 FPGA operates at a frequency of 50Mhz which is much too fast for this project, so a lower frequency is needed. The project expects there to be three frequencies offered with the ability to switch between them based on a two bit input. This added the requirement for a multiplexer to the clock divider circuit. My clock divider circuit can be seen in figure 1.

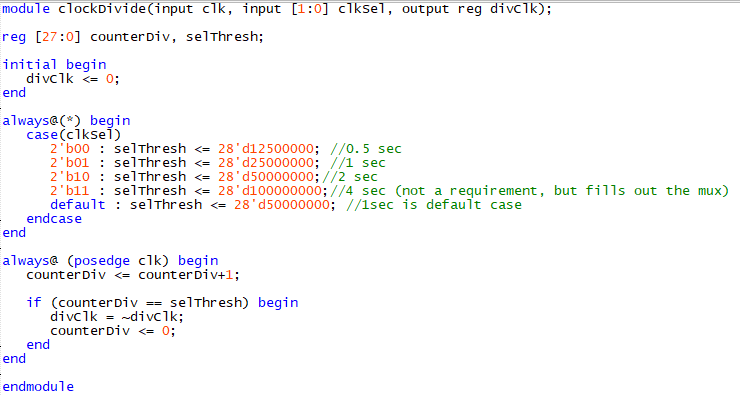


Figure : Clock divider with clock selection

This clock divider circuit makes use of a 28 bit wide bus as its counter. This counter is used to count clock cycles. When a threshold is met, the output clock is then toggled, therefore generating a reduced output clock. The output threshold is decided with a multiplexer with selection pins connected to the two bit selection input.

The other part of the final design is the counter. This takes a clock, reset, load with load value, up down, and a clock selector input. It outputs a counter shown on both a seven segment display and four LEDs. The reset, up down, and load inputs were concatenated together into a single bus. This bus was then used with a case statement to implement the counter. Inline if statements were also employed to handle the overflow scenarios based on whether the counter is incrementing or decrementing. This kept the code clean and manageable as the case statement improved readability over the nested if else statement. Lastly, the seven segment to hex decoder used in the last project was reused for this project to display the result on a seven segment display for ease of reading. The final design can be seen in figure 2.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure : Top level design module making use of concatenation and a case statement

A diagram of a computer program

AI-generated content may be incorrect.

Figure : RTL diagram of final design

***Results:***

The simulation results are as expected. All components of the final counter design work as intended given a constant clock. The only issue with the simulations was the different clock domains and the clock divider. The clock divider was very hard and resource intensive to simulate due to the massive discrepancy with the input and output clocks. This was solved by splitting the code into two which allowed the simulation shown in figure 4 to generate. The last line in figure 4 is the counter and its dependent on the various input control lines. The design works fully on the FPGA. The test bench for the final design can be seen in figure 5 and the test bench used to debug the clock divider alone is in figure 6. The testbench for the clock had the primary timescale set to 1s/1ns. This allowed for both domains to be represented using seconds for the inputs and nanoseconds for the clock.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Waveforms for the top level design

A close up of a text

AI-generated content may be incorrect.

Figure : Top level testbench

A screenshot of a computer code

AI-generated content may be incorrect.

Figure : testbench for clock divider used in debugging

***Observations and Conclusion:***

In conclusion, this activity introduced sequential logic design in the context of Verilog. This project required the implementation of a counter circuit with various functions. This clock circuit makes use of several clock domains in its design which allows for clock domains to be introduced. Sequential testbench methodologies were also introduced. This lab allowed me to learn about clock domains through the clock divider. It also allowed me to experiment with different clock switching circuits making use of things such as flip flops, latches, and busses. Several iterations of the clock circuit were created before the final version. Lastly, on the creation of a testbench, the learning of timescales was accomplished. I had trouble representing both hertz and megahertz in the same test bench, this was resolved through the splitting of the time scale. The second scale was used as the offsets operate in seconds. The nanosecond precision was also used to allow for a 50Mhz clock to be generated for the testing of the divider. Overall, this lab acted as a good introduction to the process and challenges of sequential logic development. This will act as a good foundation for future projects making use of both combinational and sequential logic combined.