Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

BEng(H) Software & Electronic Engineering

System on Chip Design & Verification
Year 3

FPGA Stopwatch Project Assignment

Dan Koskiranta

Atlantic Technological University Galway

2023 - 2024

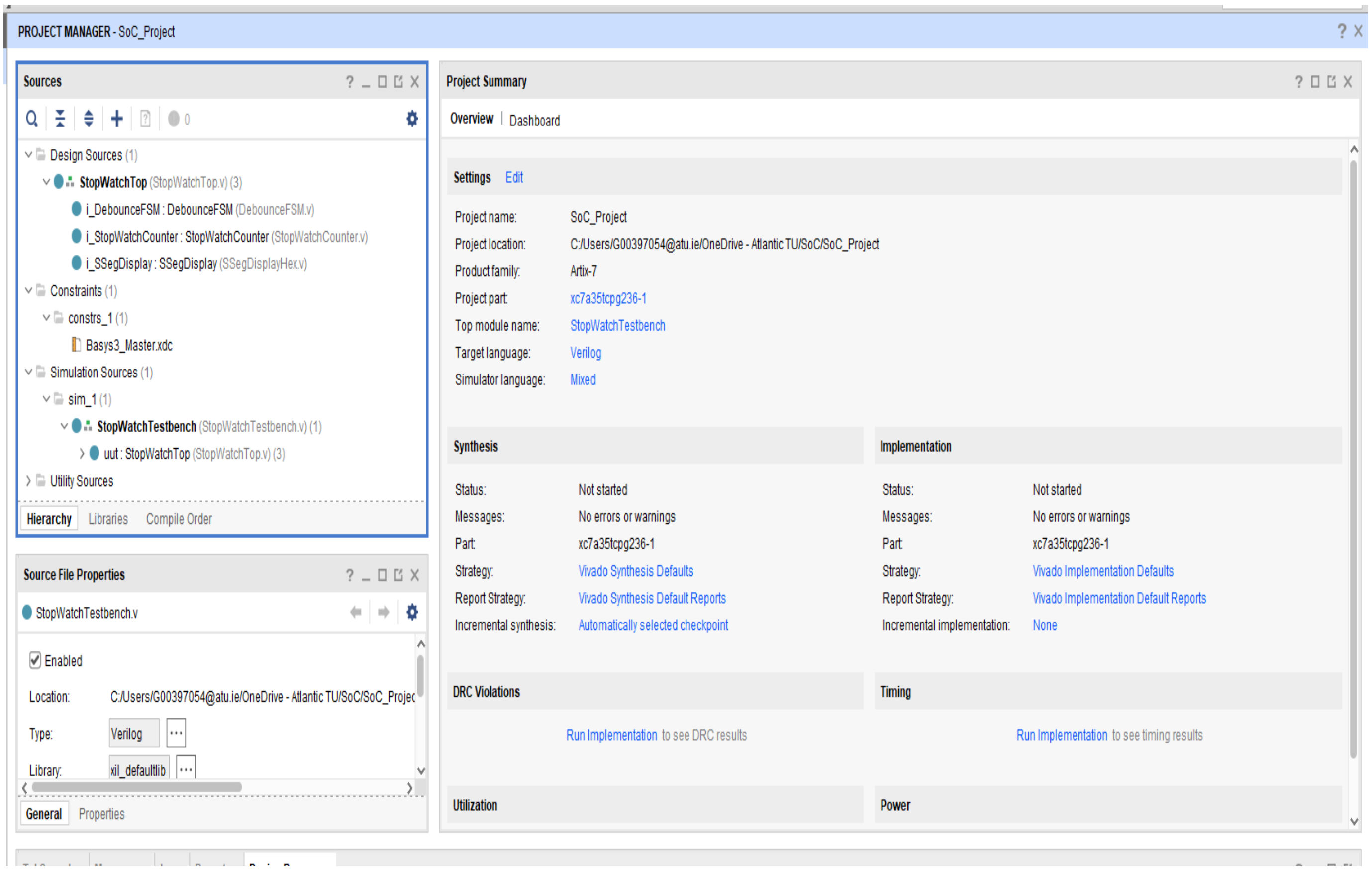
Table of Contents

1. Introduction

The purpose of this project is to create a digital stopwatch on a reconfigurable System-on-Chip (SoC) platform. The digital stopwatch is created by integrating the hardware description FPGA technology and the Xilinx Vivado toolchain. The stopwatch will initially display three digits on the Basys3 FPGA Development Board. At the end of the project, code is modified to display four digits.

The project involves the testing and simulation of Verilog code to make sure the code works as expected. This is followed by synthesis to transform the code into a netlist to represent the logical structure of the design. Final part is the implementation to map the netlist on the target FPGA architecture and generate bitstream to configure the FPGA with the desired functionality.

2. Project Setup and Configuration



PROJECT MANAGER - SoC_Project

Sources

- Design Sources (1)
  - StopWatchTop (StopWatchTop.v) (3)
    - i_DebounceFSM : DebounceFSM (DebounceFSM.v)
    - i_StopWatchCounter : StopWatchCounter (StopWatchCounter.v)
    - i_SSegDisplay : SSegDisplay (SSegDisplayHex.v)
- Constraints (1)
  - constrs_1 (1)
    - Basys3_Master.xdc
- Simulation Sources (1)
  - sim_1 (1)
    - StopWatchTestbench (StopWatchTestbench.v) (1)
      - uut : StopWatchTop (StopWatchTop.v) (3)
- Utility Sources

Hierarchy | Libraries | Compile Order

Source File Properties

StopWatchTestbench.v

Enabled

Location:	C:/Users/G00397054@atu.ie/OneDrive - Atlantic TU/SoC/SoC_Projec
Type:	Verilog
Library:	xil_defaultlib

General | Properties

Project Summary

Overview | Dashboard

Settings Edit

Project name:	SoC_Project
Project location:	C:/Users/G00397054@atu.ie/OneDrive - Atlantic TU/SoC/SoC_Project
Product family:	Artix-7
Project part:	xc7a35tcpg236-1
Top module name:	StopWatchTestbench
Target language:	Verilog
Simulator language:	Mixed

Synthesis

Status:	Not started
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	Automatically selected checkpoint

Implementation

Status:	Not started
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

DRC Violations

Run Implementation to see DRC results

Timing

Run Implementation to see timing results

Utilization

Power

Figure 1 Vivado Project Window

Figure 1 shows the Sources Panel with project hierarchy.

3. Simulation

It's very important to test your design to make sure your hardware works as expected. Untested design and dysfunctional hardware might lead to big financial losses.

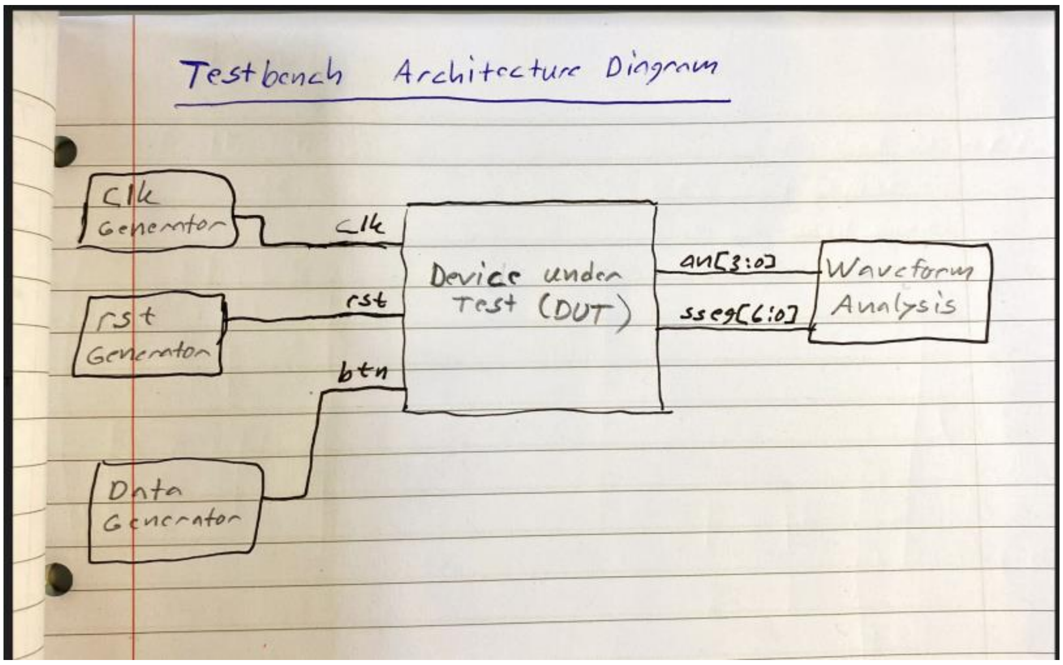
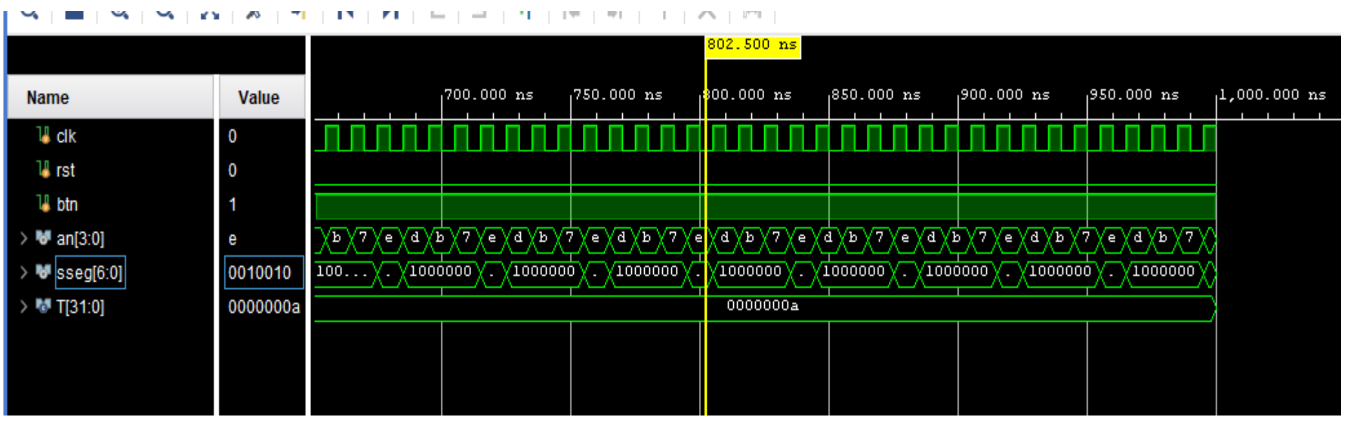


Figure 2 Testbench Architecture Diagram

Testbench is the simulation environment used to verify the functionality of your digital design before synthesizing it and implementing it on hardware.



Name	Value
clk	0
rst	0
btn	1
an[3:0]	e
sseg[6:0]	0010010
T[31:0]	0000000a

Figure 3 Simulation Waveform

The clock clk input makes the output change each time the clock input goes from LOW to HIGH. The reset rst returns the counter to zero.

The btn signal generates the data. The 7-segment display won't start counting before btn goes HIGH.

an[3:0] represents the internal signals. There are 4 bits on the an signal and the an signal drives the 7-segment display.

The line sseg[6:0] represents the 7-segment displays. Number 1 on the 7-bit value indicates which segment (LED) is off on the 7-segment display.

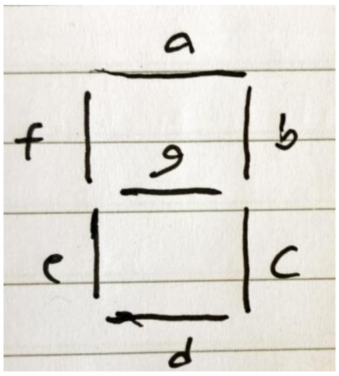


Figure 4: 7-segment Display

g f e d c b a

0 = 1 0 0 0 0 0 0

Figure 4 represents a seven-segment display.

4. RTL Analysis

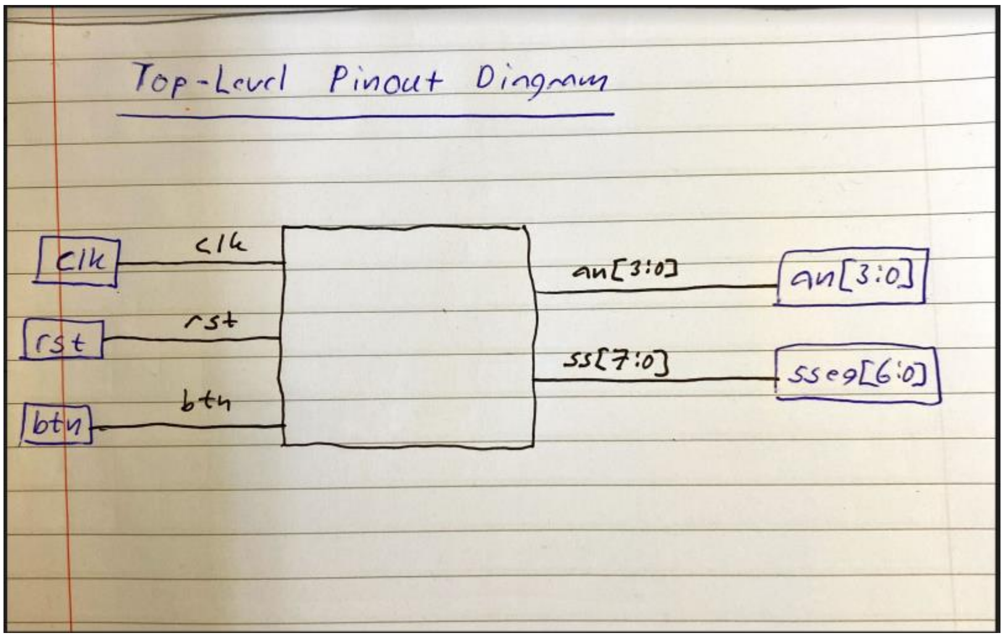


Figure 5 Top-Level Pinout Diagram for StopWatch Module

The StopWatch top module in **figure 5** contains all the other modules. It coordinates the operation of a stopwatch. The functionality is to create a stopwatch system that can be started, stopped, and reset.

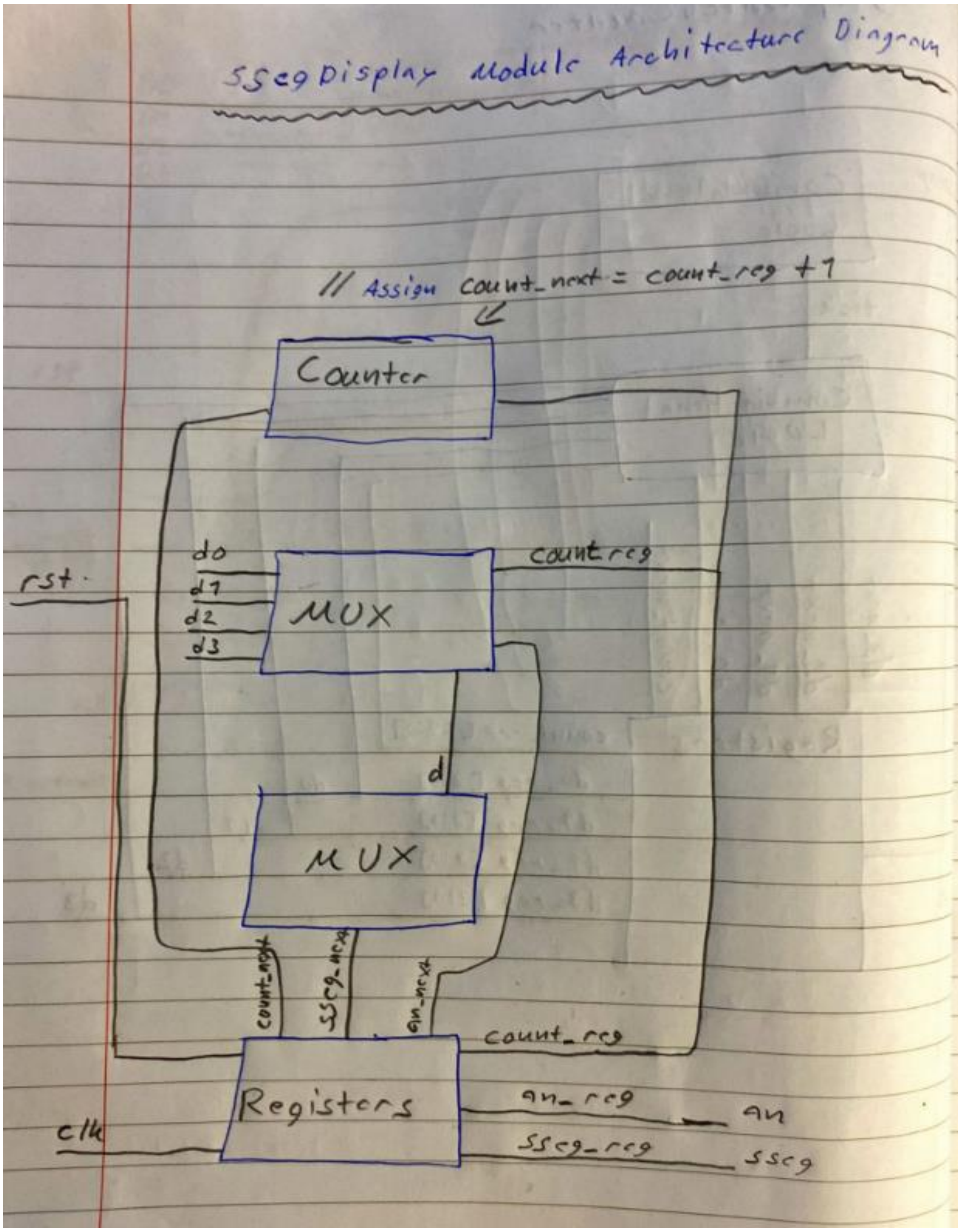


Figure 6 SSegDisplay Module Architecture Diagram

The SSegDisplay Module in **figure 6** is designed for multiplexing a seven-segment display, which displays numbers between 0 and 9 and hexadecimal numbers A to F.

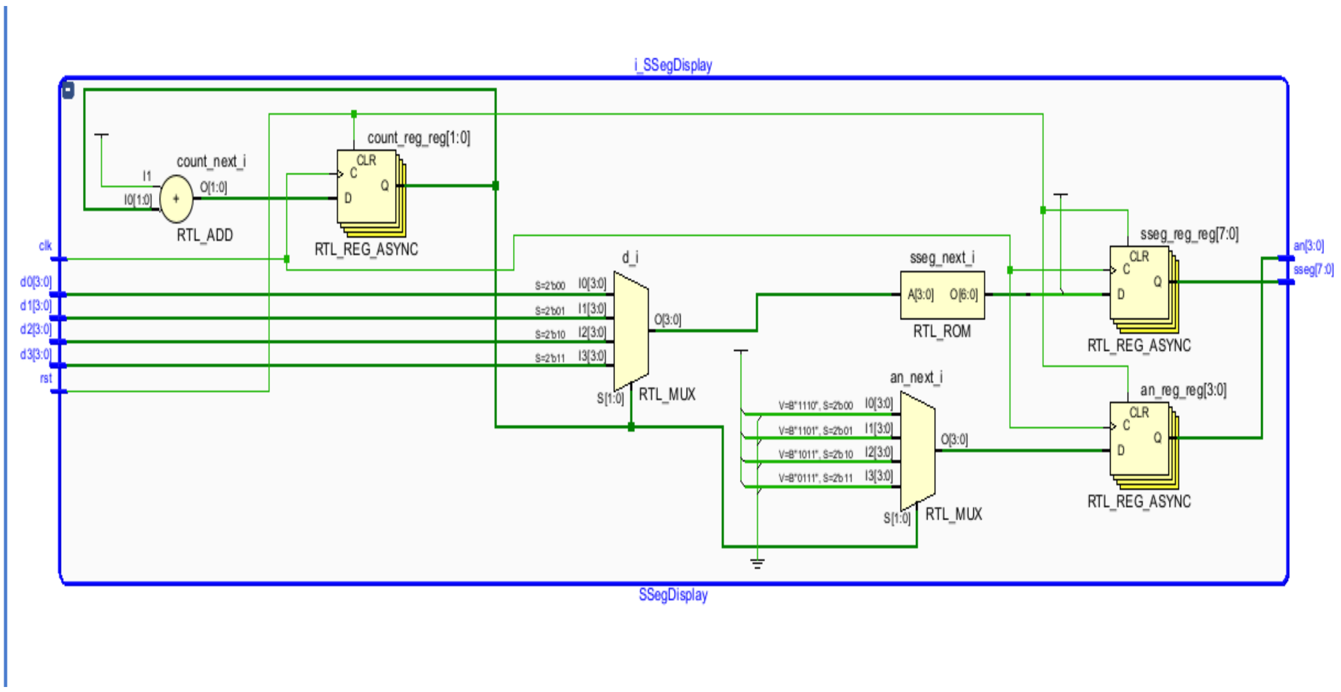


Figure 2 Elaborated Design Schematic for SSegDisplay Module

FPGA hardware elements used to implement the Verilog code. Match each hardware element to a piece of the Verilog code for this module:

Code: input wire [3:0] d3, d2, d1, d0, is associated with RTL_MUX

count_reg = counter register.

an_reg = 4 bit register for controlling the common anode.

sseg_reg = 8-bit register for driving the segments.

count_next = wire used to calculate the next value of the counter.

----

sseg_next = wire to represent the next value for driving the segments of a 7-segment display. Code below is for sseg_next:

```
4'h0: sseg_next[6:0] = 7'b1000000; // all leds on, bar g, for digit 0
    4'h1: sseg_next[6:0] = 7'b1111001; // leds b, c on for digit 1
    4'h2: sseg_next[6:0] = 7'b0100100; // leds c, f off for digit 2
    4'h3: sseg_next[6:0] = 7'b0110000; // leds e, f off for digit 3
    4'h4: sseg_next[6:0] = 7'b0011001;

    4'h5: sseg_next[6:0] = 7'b0010010;

    4'h6: sseg_next[6:0] = 7'b0000010;

    4'h7: sseg_next[6:0] = 7'b1111000;

    4'h8: sseg_next[6:0] = 7'b0000000; // all leds on for digit 8

    4'h9: sseg_next[6:0] = 7'b0010000;

    4'ha: sseg_next[6:0] = 7'b0001000;

    4'hb: sseg_next[6:0] = 7'b0000011;

    4'hc: sseg_next[6:0] = 7'b1000110;

    4'hd: sseg_next[6:0] = 7'b0100001;

    4'he: sseg_next[6:0] = 7'b0000110;

    4'hf: sseg_next[6:0] = 7'b0001110; // leds b, c, d off for hex digit F

    default: sseg_next[6:0] = 7'b1111111; // all off
```

----

an_next = 4-bit wire for representing the next value for controlling the common anode of a 4-digit 7-segment display.

d = 4-bit wire used to multiplex the digit input.

Code below is representing the **an_next** and **d:**

```
always @* begin

  case(count_reg[N-1:N-2])

    2'b00: begin  // Rightmost digit display on

      an_next = 4'b1110;

      d      = d0;

    end

    2'b01: begin  // Second display from right on

      an_next = 4'b1101;

      d      = d1;

    end
```

```
      2'b10: begin  // Third display from right on
        an_next = 4'b1011;
        d       = d2;
      end
      2'b11: begin  // Leftmost display on
        an_next = 4'b0111;
        d       = d3;
      end
      default: begin // Default, no display on
        an_next = 4'b1111;
        d       = 4'b0000;
      end
    endcase
end
```

5. Synthesis & Implementation

1: Bigger NUM_CLK_CYCLES value allows the synthesis tool to gather more accurate timing information about the design. Smaller value in simulation helps with faster verification and debugging.

2: 10,000,000 with a 100MHz clock sets d0 tick at 0.1s
Calculate the value of NUM_CLK_CYCLES required so that the rightmost digit, d0, will count in hundredths of seconds.
NUM_CLK_CYCLES = 0.01 s x (100 x 10^6)
NUM_CLK_CYCLES = 1000 000

3: After the code has been tested, next part is synthesis and implementation. The synthesis converts the code into a netlist which is a list of components that fit the target circuit. Next, the implementation tool will take the netlist as input and does the placing and routing for the design. Implementation is where the physical locations of the logic elements are determined. Finally, a bitstream is generated for programming the FPGA.

4. Resource utilization is crucial for optimizing a design to meet performance and resource Constraints. Resource utilization provides insights into the logical-to-physical mapping of the design. This will help to optimize the code for a specific FPGA design.

6. Basys3 FPGA Development Board Demonstration

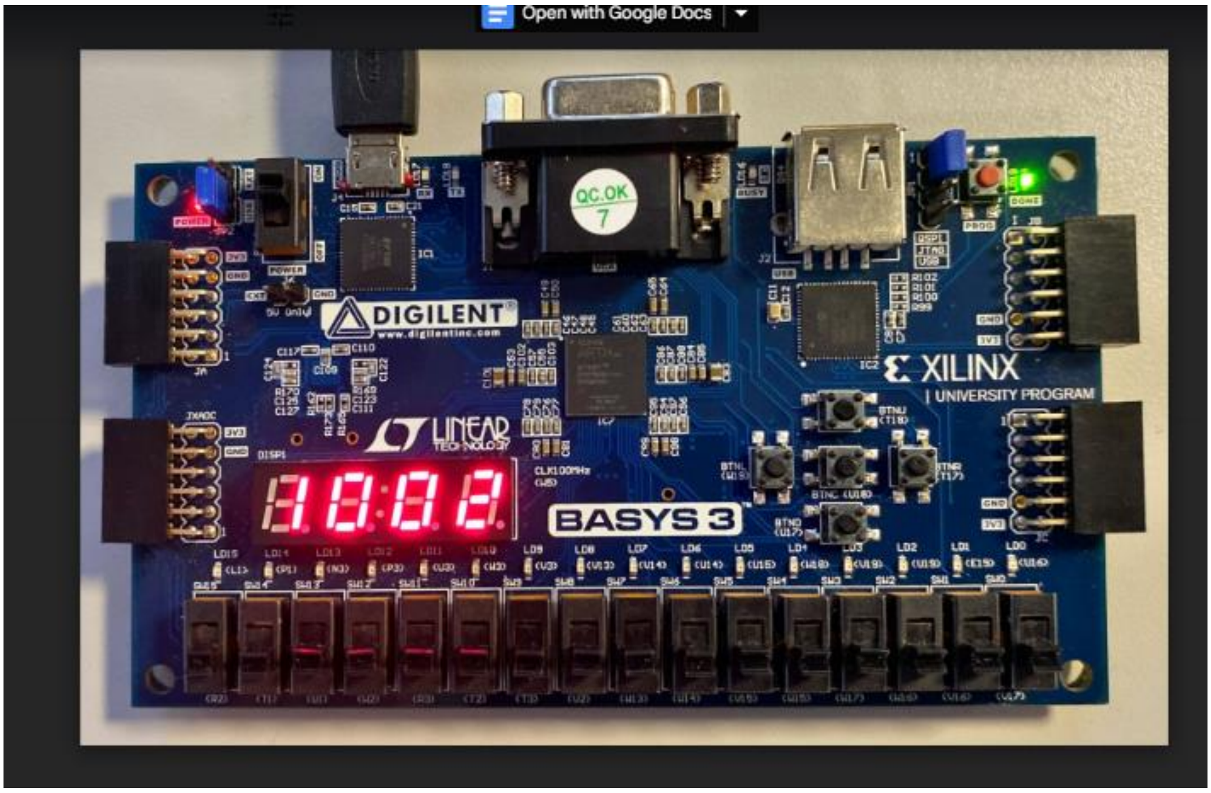


Figure 8: Stopwatch working on the Basys3 FPGA Development Board.

7. Code Analysis

Code for the StopWatchCounter module uses the ternary operator in assigning values to two signals: count_next and tick. Re-write this code in both cases, as pseudo-code, to use if-else statements instead of the ternary operator.

```
assign count_next = (rst || (count_reg == NUM_CLK_CYCLES && go)) ? 4'b0 : (go) ?
count_reg + 1 : count_reg;
assign tick = (count_reg == NUM_CLK_CYCLES) ? 1'b1 : 1'b0;
```

Rewritten code using if-else statements (Pseudo code)

```
if(rst || (count_reg == NUM_CLK_CYCLES && 90))
        count_next = 4’b0;
else if(90)
        count_next + 1;
else
        count_next = count_reg;
```

```
if(count_reg == NUM_CLK_CYCLES)
        tick = 1’b1;
else
        tick = 1’b0;
```

8. Design Adaptation

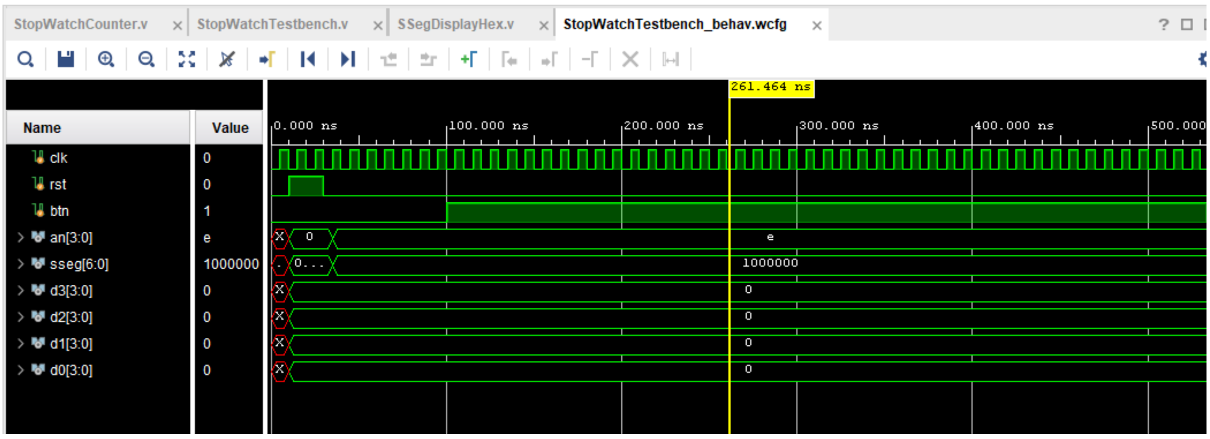


Figure 9: Simulation output of the enhanced design implementing the four digits in Vivado.

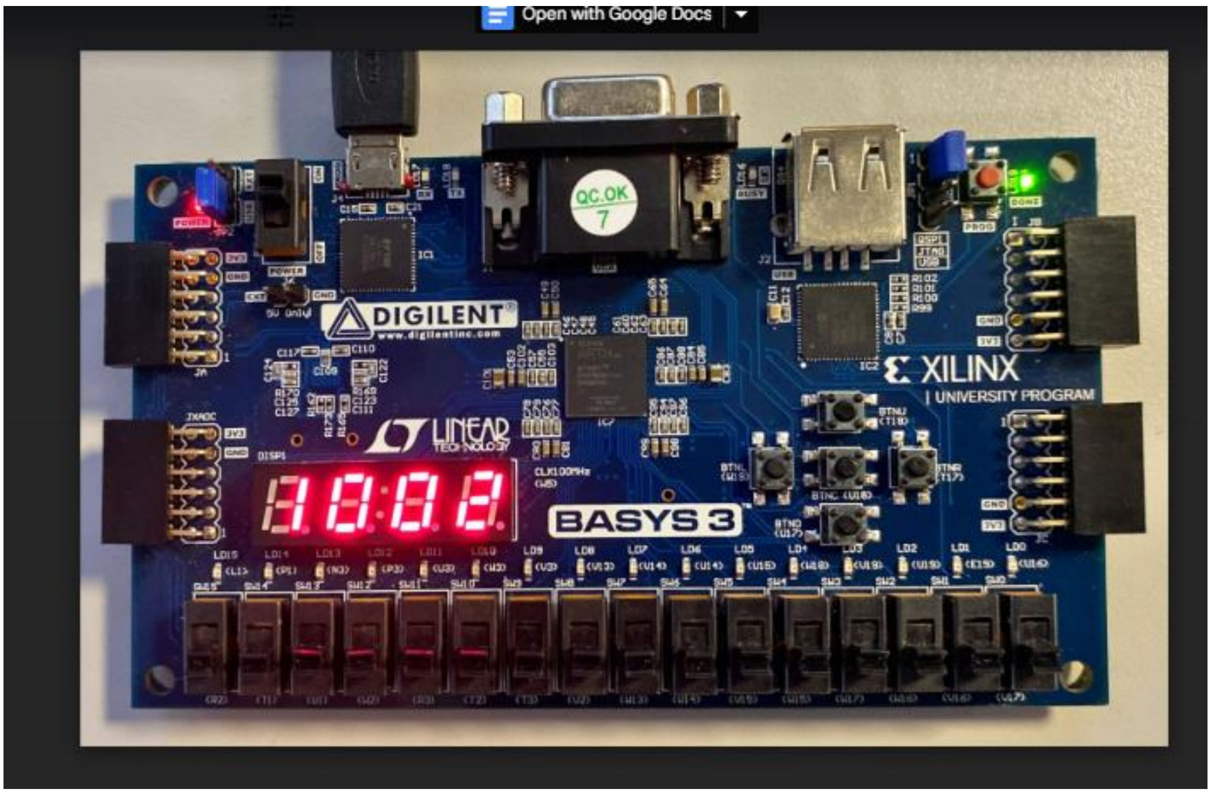
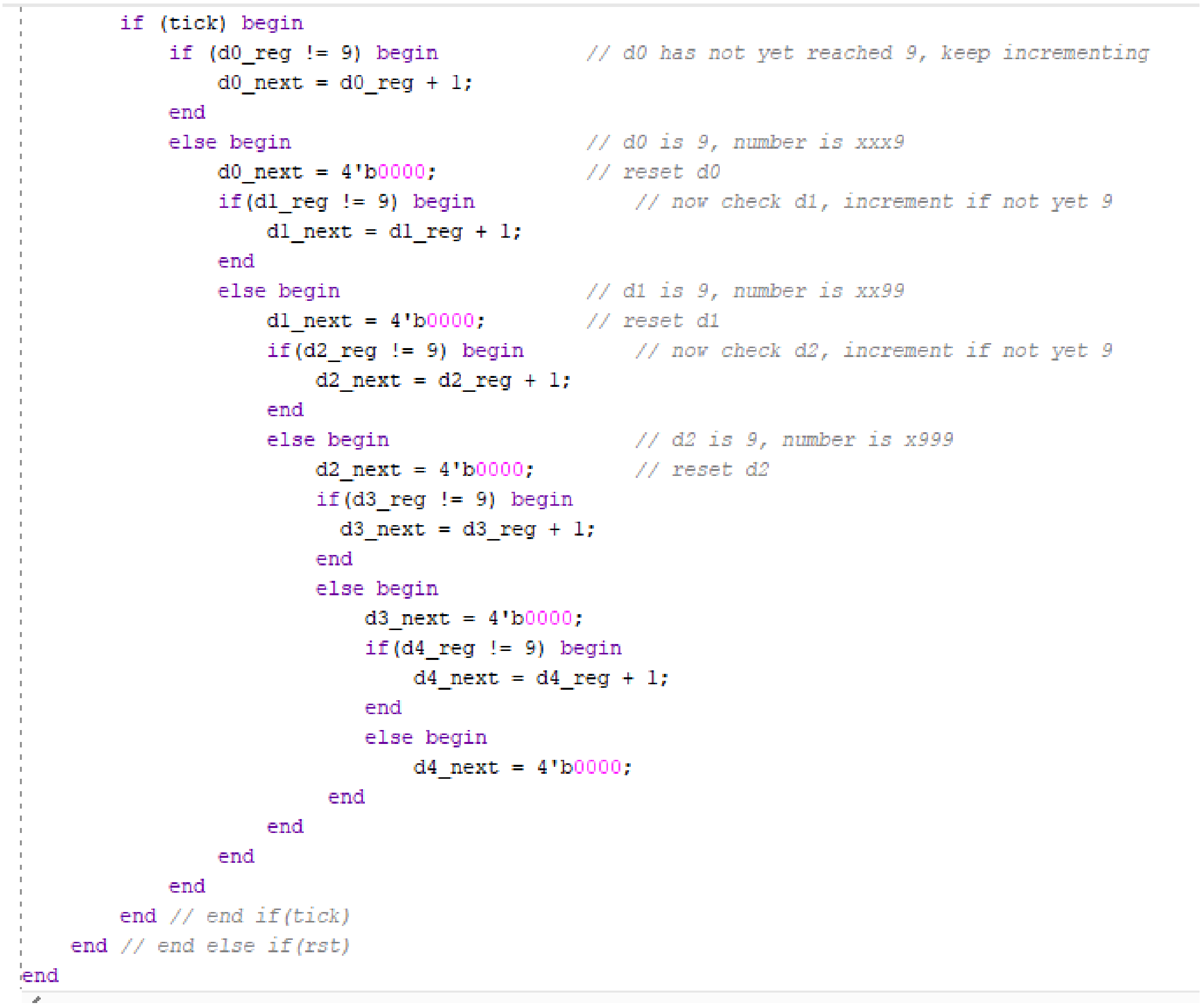


Figure 10: Enhanced design to implement 4 digits on the Basys3 FPGA Development Board.

9. Problem Solving



```
        if (tick) begin
            if (d0_reg != 9) begin              // d0 has not yet reached 9, keep incrementing
                d0_next = d0_reg + 1;
            end
            else begin                          // d0 is 9, number is xxx9
                d0_next = 4'b0000;              // reset d0
                if(d1_reg != 9) begin               // now check d1, increment if not yet 9
                    d1_next = d1_reg + 1;
                end
                else begin                      // d1 is 9, number is xx99
                    d1_next = 4'b0000;          // reset d1
                    if(d2_reg != 9) begin           // now check d2, increment if not yet 9
                        d2_next = d2_reg + 1;
                    end
                    else begin                      // d2 is 9, number is x999
                        d2_next = 4'b0000;          // reset d2
                        if(d3_reg != 9) begin
                          d3_next = d3_reg + 1;
                        end
                        else begin
                            d3_next = 4'b0000;
                            if(d4_reg != 9) begin
                                d4_next = d4_reg + 1;
                            end
                            else begin
                                d4_next = 4'b0000;
                         end
                    end
                end
            end
        end // end if(tick)
    end // end else if(rst)
end
```

Figure 11: Problem Solving Example

The biggest problem I encountered during this project was the implementation of the fourth digit on the Basys3 FPGA Development Board. Initially, I was trying to run the code with the four outputs:
assign d0 = d0_reg;
assign d1 = d1_reg;
assign d2 = d2_reg;
assign d3 = d3_reg;

However, I was unable to get this work. So, I added a fifth output: assign d4 = d4_reg. After this, I ran into problems with synthesizing it. It turned out that I was missing an **end** keyword in the code block that increments the four-digit counter.

10.References

[1] Digilent, “Basys 3 Reference,” 2021. [Online]. Available: https://reference.digilentinc.com/basys3/refmanual.

[2] P. J. Ashenden, Digital Design (Verilog): An Embedded Systems Approach Using Verilog, Burlington: Morgan Kaufmann, 2007.

[3] M. Lynch, “Combinational and Sequential Logic with Verilog & Xilinx Vivado,” 2021. [Online]. Available: https://learnonline.gmit.ie.

[4] M. Lynch, “FPGA Stopwatch Project Source Code,” 2021. [Online]. Available: https://learnonline.gmit.ie.

[5] “chat.openai.com,” OpenAI, 30 11 2022. [Online]. Available: https://chat.openai.com/chat. [Accessed 5 12 2023].