Hardware Implementation

Soma Chandra lahari

May 2025

1. Introduction

In this report, I describe the hardware implementation of two basic Verilog modules using the Spartan-7 FPGA: a binary counter and a 7-segment display driver. Each module was written, simulated, synthesized, and tested independently. This allowed for better understanding of modular hardware design and real-time FPGA implementation.

2. Tools/Requirements

2.1 Xilinx Vivado

To simulate and synthesize the design, I used Xilinx Vivado, a comprehensive tool suite for FPGA development. Vivado supports Verilog and VHDL-based design and offers tools for simulation, synthesis, implementation, and bitstream generation.

2.2 Spartan-7 FPGA Board

The Spartan-7 FPGA is an affordable and efficient entry-level development board suitable for learning and prototyping. It is built on the Xilinx 7-series architecture and offers a robust set of I/O, making it well-suited for simple digital design applications like counters and display interfaces.

3. Project Overview

The implementation involved two separate Verilog modules:
1. A binary counter that increments with a clock pulse and can be observed via LEDs or internal signal monitoring.
2. A 7-segment display driver module that decodes binary inputs into corresponding 7-segment outputs to display hexadecimal digits.

4. Steps Followed

Step 1: Verilog Design

Wrote separate Verilog code for the binary counter and the 7-segment decoder. Each module was tested independently with its own testbench.

Step 2: Simulation

Performed behavioral simulation using Vivado. Verified functionality of both modules by comparing simulated outputs with expected values.

Step 3: RTL Analysis and Constraints

Generated the RTL schematic to understand design structure. Assigned FPGA pins using the .xdc constraint file. Ensured proper mapping of counter outputs to LEDs and 7-segment segments to the board's display pins.

Step 4: Synthesis and Implementation

Synthesized and implemented each module independently. Checked timing and logic utilization reports to ensure resource efficiency.

Step 5: Bitstream Generation

Generated bitstreams (.bit files) for both modules. Prepared the Spartan-7 board for hardware testing.

Step 6: Programming and Testing

Programmed the Spartan-7 board with each bitstream file. Used push-buttons for input and LEDs/7-segment display for observing output. Verified correct functionality in real-time.

5. Observations

- The binary counter correctly counted upward with each clock cycle and reflected output on LEDs.
- The 7-segment decoder accurately displayed digits 0 to F based on input binary values.
- Both modules worked independently as expected on the Spartan-7 board.

6. What I Learned

1. Writing modular Verilog code for reusable components.
2. Creating testbenches and running simulations in Vivado.
3. Mapping physical pins using .xdc constraints and I/O planning.
4. Synthesizing, implementing, and generating bitstreams for Spartan-7 FPGA.
5. Testing hardware modules independently using onboard peripherals.

7. Demonstration

This includes views of input switches, LEDs displaying counter values, and 7-segment output during active operation.

Binary counter

https://drive.google.com/file/d/17LiiU_9BcSSTLvMTrNEPu_ebCzjjF2LT/view?usp=sharing

7-segment decoder

https://drive.google.com/file/d/1rB4GO0461OQeo1QvmIs4Nf26TvZ0iR2m/view?usp=sharing

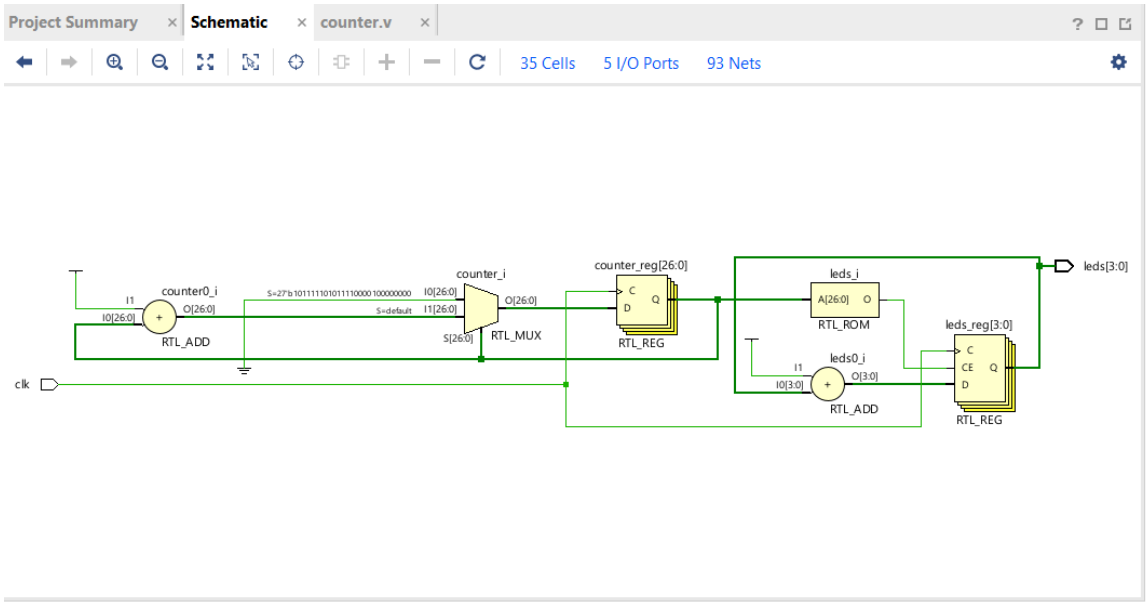
Binary counter code

Project Summary | counter.v

C:/Users/lahar/counter/counter.srcs/sources_1/new/counter.v

```
1  `timescale 1ns / 1ps
2
3
4
5  module counter(clk,leds);
6  input clk;
7  output reg [3:0]leds;
8  reg [26:0] counter;
9  always @(posedge clk)
10 begin
11 if (counter == 100000000)begin
12 leds<=leds+1;
13 counter<=0;
14 end
15 else begin
16 counter<=counter+1;
17 end
18 end
19
20
21 endmodule
22
```

RTL analysis schematic



7-segment decoder code

Project Summary | segment.v

C:/Users/lahar/7segment/7segment.srcs/sources_1/new/segment.v

```
1  `timescale 1ns / 1ps
2
3  module segment(w,x,y,z,a,b,c,d,e,f,g, dp,an);
4  input w,x,y,z;
5  output a,b,c,d,e,f,g;
6  output wire dp;
7  output wire [7:0]an;
8  assign an=8'b11111110;
9  assign dp=1;
10 assign a=~(y|w|(x&z)|(~x&~z));
11      assign b=~(~x|(~y&~z)|(y&z));
12      assign c=~(~y|z|x);
13      assign d=~(w|(y&~z|(~x&y)|(~z&~x)|(x&z&~y)));
14      assign e=~((~z&~x)|(y&~z));
15      assign f=~(w|(x&~y)|(~z&x)|(~y&~z));
16      assign g=~(w|(y&~z)|(x&~y)|(~x&y));
17
18 endmodule
19
```

RTL analysis schematic