HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF TELECOMMUNICATIONS ENGINEERING

Lab 1
Warmup

INSTRUCTORS : PhD. Trần Hoàng Linh
: Nguyễn Tuấn Hùng
STUDENT : Trần Minh Trí
ID STUDENT : 2051022

Ho Chi Minh City – 2024

Table of Contents

I. Introduction

This project will focus on an overview of System Verilog techniques, setting up an FPGA programming environment, and more. Specifically, the task of this project will be to design and test the top module that has the function of calculating cumulative sums and communicating with peripheral devices.

II. Objectives

In this project, I am provided with the following modules to calculate the cumulative sum of a given number and then capture the value of the calculated sum to display on a 7-segment LED.

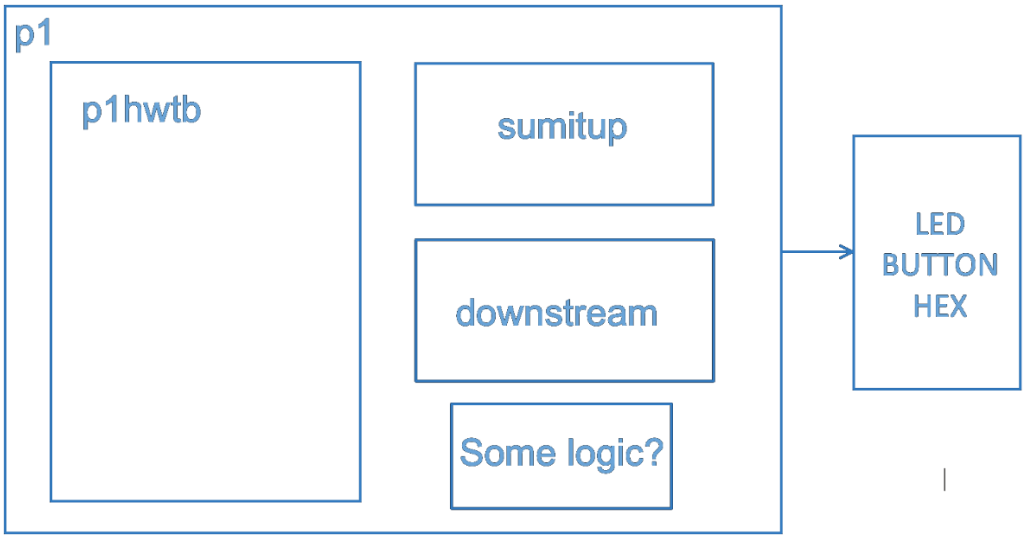


Figure 1: Basic organization

Module	Description
p1hwtb	Generate random values and sends them to your 'sumitup' module. Display the 8-bit sum of the input series in HEX3 and HEX2 (leftmost digits), with the calculated sum shown in HEX1 and HEX0 Activate LEDR0 to illuminate the LED when matching between expected sum and calculated sum

sumitup	Accumulate a series of input numbers
downstream **(main task in project)**	Capture the calculated sum and holds it for display while the next sum is being calculated. Wait for the done signal and then loads the calculated sum into its own register. Display the captured value on the two hex displays.
p1 **(main task in project)**	Top module of design, connecting everything and synthesize on Quartus

Table1: Description of modules

III. Design architecture & Requested program

Based on the above goals, below will be the detailed design structure of this project, with 5 main blocks: p1(top-level), p1hwtb, sumitup, downstream, bcdtohex (supporting to display on 7 - segment displays).

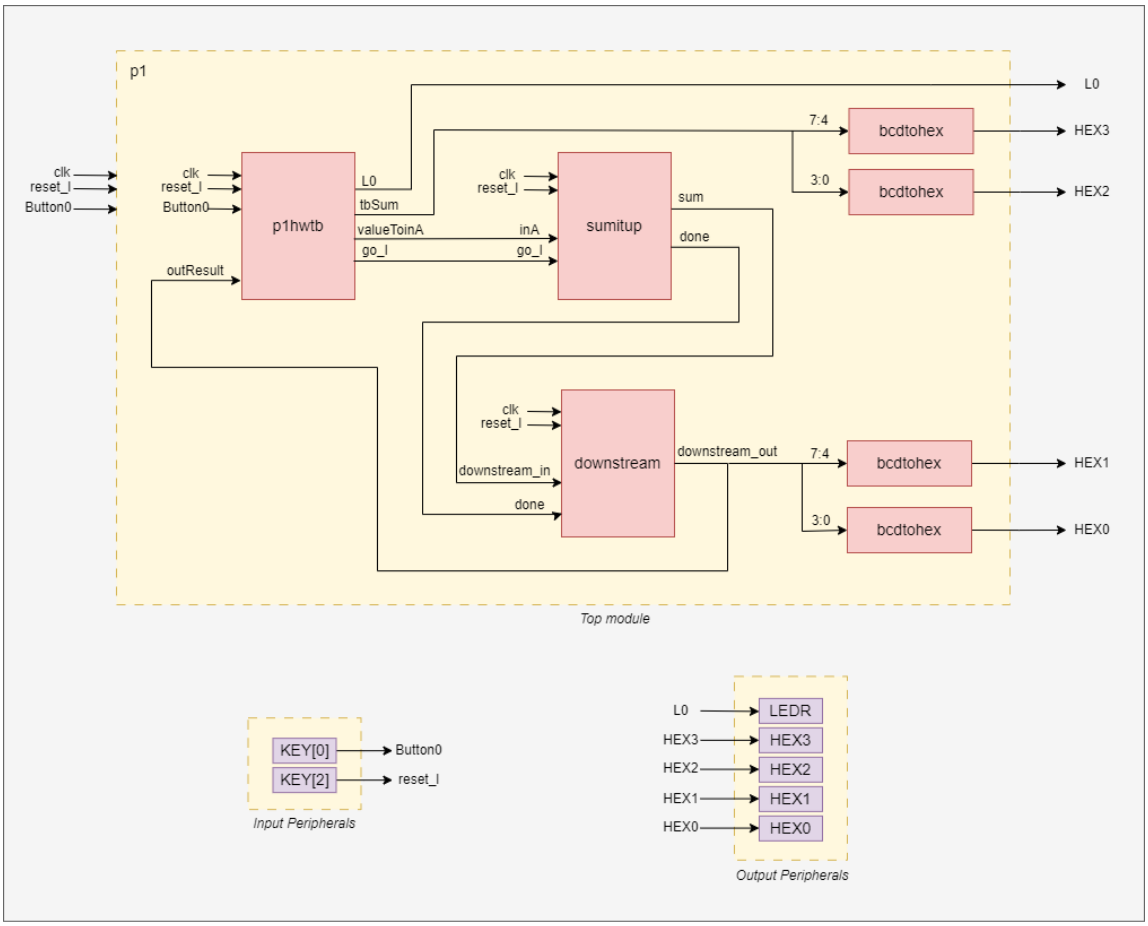


Figure 2: Main architecture design

```
1   module downstream // For Altera DE10s board
2   (input   logic          clk, reset_l,
3    input   logic done,
4    input logic [7:0] downstream_in,
5    output logic [7:0] downstream_out
6    );
7   logic [7:0] reg_sum;
8   ////////////////
9   //write your code here
10  always_ff @(posedge clk) begin
11     if (!reset_l) begin
12        downstream_out <= 8'b0;
13     end
14
15     else begin
16        if (!done) begin
17           reg_sum <= downstream_in;
18        end
19
20        else begin
21           downstream_out <= downstream_in;
22        end
23     end
24  end
25  ////////////////
26
27  endmodule
```

Figure 3: downstream.sv file

IV. Testcase

Testbench Operation

The testbench's operation is controlled by two buttons on the board. As a reset, KEY2 resets the Finite State Machine (FSM), turns off LEDR0, and makes the display show zeros. When the board is reset, all of the displays ought to show zeros. Pressing and holding KEY0 starts the process, which causes the hardware testbench to send a sequence of numbers to your sumitup thread at a 50 MHz rate. The total determined by the testbench is displayed on the higher hex displays (HEX3 and HEX2), and the value obtained by the downstream module—the outcome of my code—is displayed on the lower hex displays (HEX1 and HEX0).

LEDR0 illuminates if these two values are the same. The testbench's hex displays are reset to zero upon releasing KEY0, but my code keeps the calculated amount displayed in the lower digits. For the purpose of starting a fresh sequence of number transmission and display, the system waits for the next touch of KEY 0. When KEY2 is pressed, a

reset is completed and all screens display zeros. Interestingly, when a button is pressed, a logic 0 is produced because the buttons function on an active-low basis.

Test cases

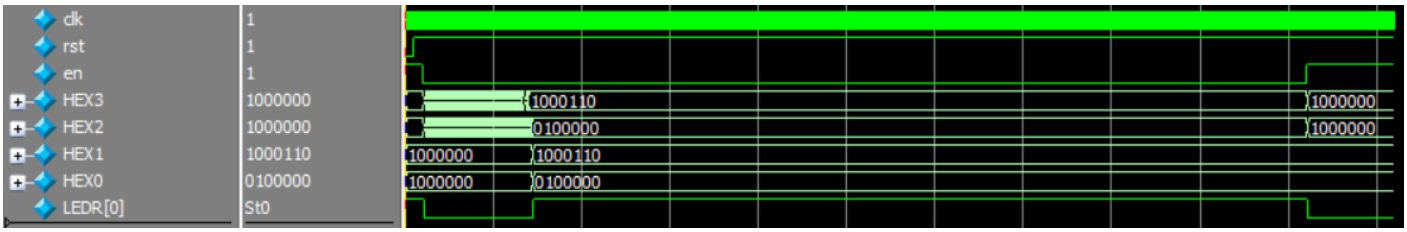
Besides executing the project with the given number (SEED = 8'd123) in file p1hwtb, the cumulative sum of this number in HEX format will consist of 4 digits (more than the number of HEX can display), So we also work with other numbers to be able to definitely evaluate the summation function of this module with SEED = 8'd11 and SEED = 8'd20 to be able to display all results on 7-segment LED.

Given number	Result	On HEX displays
SEED = 8'd123	1DCAh	CAh
SEED = 8'd11	42h	42h
SEED = 8'd20	D2h	D2h

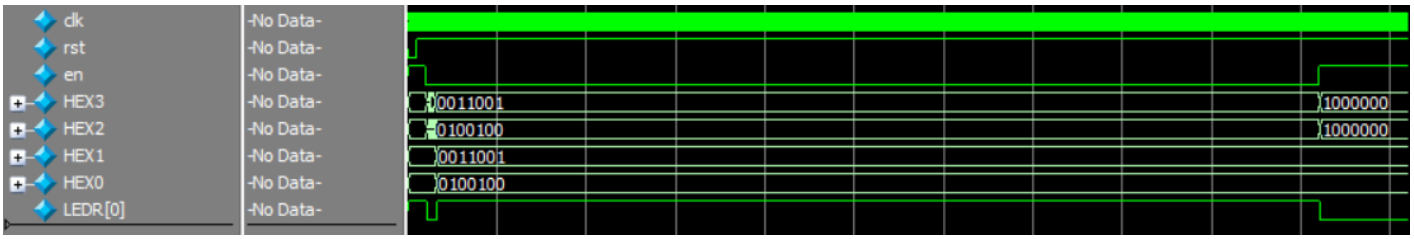
Table2: Some testcases

V. Simulation

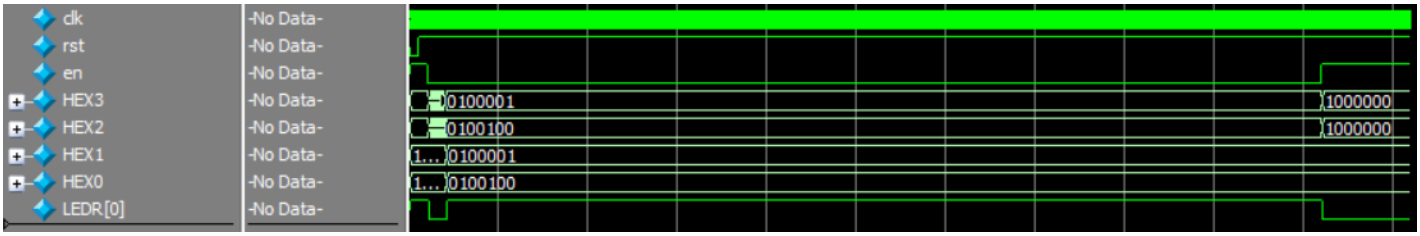
SEED = 8'd123



SEED = 8'd11



SEED = 8'd20



Report sentence in three cases

```
# Start testbench
# LED0 = x
# LED0 = 1
# LED0 = 0
# LED0 = 1
# Testbench finished, Check the output on Waveform
# If it is good, you should see at least two times that LED0 = 1
# 1 is when the testbench begins, 2 is when the testbench ends
# LED0 = 0
# ** Note: $finish    : D:/ktsnc/lab1/Lab1/plswtb.sv(44)
#    Time: 11200 ps  Iteration: 0  Instance: /plswtb
# 1
```

VI. Real result pictures

SEED = 8'd123

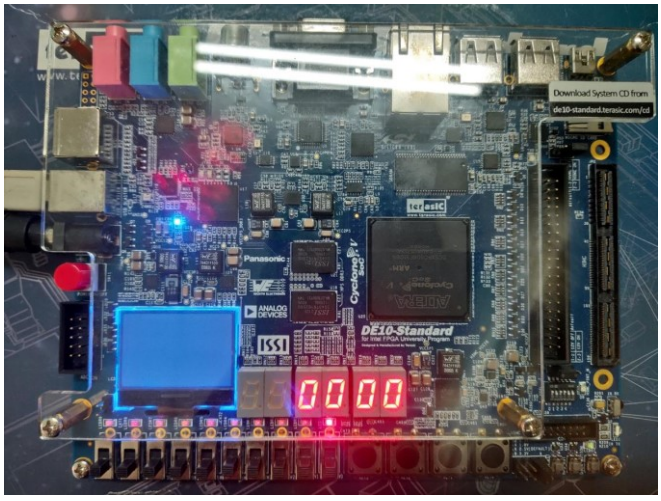


Figure 1. Initial Operation

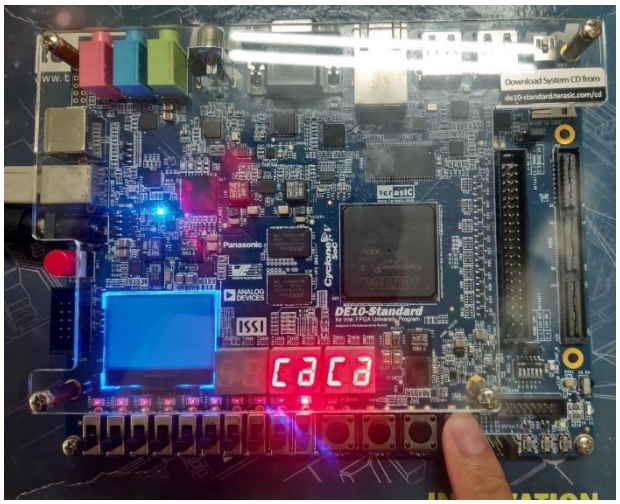


Figure 2. Start Operation

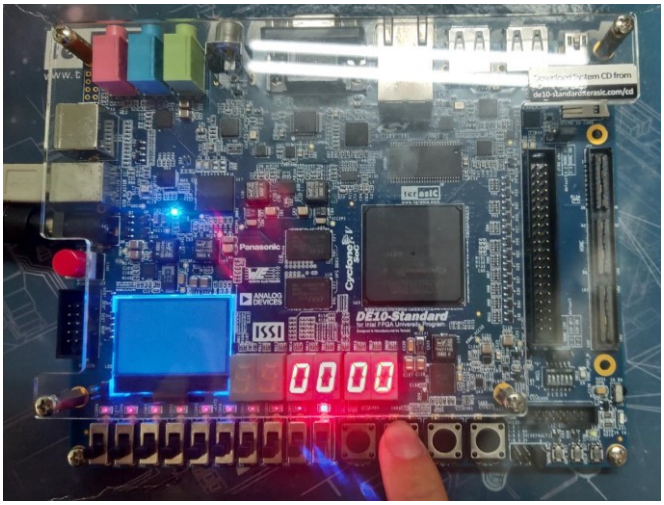


Figure 3. Wait Operation

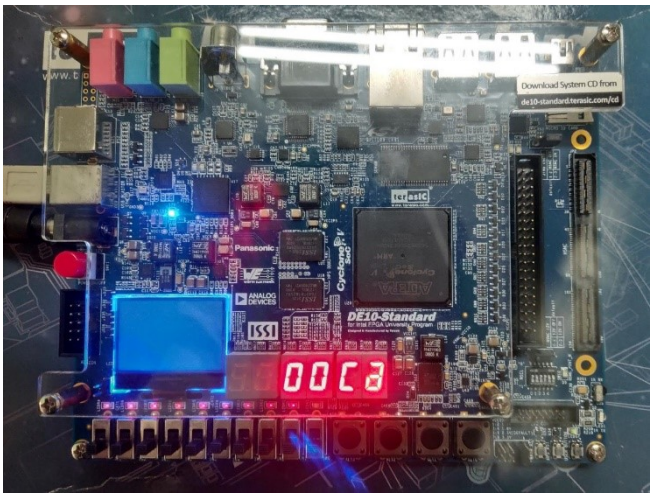


Figure 4. Reset Operation

SEED = 8'd11 (add later)

SEDD = 8'd20 (add later)

VII. Conclusion

In conclusion, this project focuses mostly on module connections and a small amount of coding expertise. Furthermore, during configuration, Quartus carefully mimics every signal. Additionally, use the Teacher's p1swtb.sv file to double-check on Modelsim. Once simulators being executed with 3 case of SEED have produced the same outcome (LEDR0 = 1) at least twice (once at the beginning of the testbench and once at its conclusion) and have a correct hexadecimal number result on HEX displays by mannual methods, compared to the expected above. As a result, my design functions properly.