CAD Assignment 1

CNG331 – Computer Organization / Architecture 1

Fatma Erem AKSOY

2315075

INTRODUCTION

What is the goal of this CAD assignment? Why is it important to learn about synthesis, functional verification, timing analysis, cost estimation and power dissipation analysis on FPGAs?

- The goal of this assignment is to be able to analyze a particular circuit and understand how everything works and design the most appropriate circuit considering the needs of a particular situation. With the power dissipation analysis, we can know the overall power consumption of a particular circuit and design different circuits or techniques to reduce this amount. This way, we can also reduce the amount we need to pay so cost estimation is also very important because then we can work on reducing the amount by different analysis and techniques. Timing analysis is also important in terms of reducing the power consumption and cost. By knowing how much time a particular circuit spends, we can think of some other components to replace them to gain more time. So, it helps to pick the most efficient components regarding our purpose in a circuit.

QUESTION 1: What is a Tribonacci Sequence?

- A Tribonacci Sequence is a sequence of numbers that each term is the sum of the previous three numbers starting from the fourth term, and the first few tribonacci numbers can be given as: 0, 0, 1, 1, 2, 4, 7, …

QUESTION 2: Open up and study the declaration spaces, data types, structures, procedural blocks and statements in the design file tribonacci_calculator.sv. Read up on their description using your reference book (SystemVerilog for Design). After each line in the code, add a descriptive comment about the purpose of the line. Save the file with your comments and make sure it compiles correctly in the following steps.

module tribonacci_calculator (input logic clk, reset_n, //module implements a function with certain values, which are given as input and output variables.

input logic [4:0] input_s,

input logic begin_tribo,

output logic [15:0] tribo_out,

output logic done);

enum logic [1:0] {IDLE=2'b00, COMPUTE=2'b01, DONE=2'b10} state; //enum is used to define a new data type, our own data type that we create, which is logic here. The variables are 2-bit here.

logic [4:0] count; //declares a 5-bit logic type variable

logic [15:0] R0, R1, R2; //declares a 16-bit logic type variable

always_ff @(posedge clk, negedge reset_n) // this command line is used to model sequential flip-flop logic(because of _ff part and the fact that there is a clock), and it only

//operates when the clock is on the positive edge and the reset is on the negative edge.

begin //goes inside the loop if any of the two conditions above is true

if (!reset_n) begin //operates when reset_n value = 0 (so that it'll be 1 inside the if condition, which is true)

```
    state <= IDLE;                  //assigns state variable as 0, IDLE, when reset_n variable equals to 0
    done <= 0;              //assigns done variable to 0
   end else                 //operates when reset_n = 1
    case (state)            //this block of code will execute based on the given value of state variables in the design.
      IDLE:         //if the current state is IDLE, which is 0, executes the next given operations
       if (begin_tribo) begin               //executes when begin_tribo = 1
        count <= input_s;                   //assigns the count value to the given input value at the beginning
        R0 <= 1;            //assigns R0 value as 1
        R1 <= 1;            //assigns R1 value as 1
                                    R2 <= 0;                    //assigns R2 value as 0
        state <= COMPUTE;               //makes the state COMPUTE 2'b01 now
       end
      COMPUTE:                  //if the state is at COMPUTE, executethe next lines
       if (count > 2) begin                 //executes if count value equals to 2
        count <= count - 1;                 //decreases the count value by 1
        R0 <= R0 + R1 + R2;                 //assigns R0 value as the sum of the three values, which are R0, R1, and R2
                                    R1 <= R0;                   //saves R0 value to R1
        R2 <= R1;                   //saves R1 value to R2. So after the last 3 lines of code, the R2 holds the sum of R0, R1, and R2 values
        $display("state = %s, count = %3d, R0 = %4d, R1 = %4d, R2 = %4d", state, count, R0, R1, R2);                   //displays the arguments in the given order.
       end else begin               //executes if count value <= 2
        state <= DONE;              //makes state value DONE now, which is 2'b10
        done <= 1;                  //assigns done value as 1
        tribo_out <= R0;                    //saves the value of R0 to tribo_out which is an output
       end
      DONE:                 //is the state varible is at the DONE, execute the upcoming lines
```

```
        state <= IDLE;              //makes state value IDLE, which is 2'b00
    endcase                 //ends the conditional blocks of code
 end
endmodule
```

QUESTION 3: Open up and study the declaration spaces, data types, structures, procedural blocks and statements in the verification (testbench) file tb_tribonacci_calculator.sv. Read up on their description using your reference book (SystemVerilog for Verification). After each line in the code, add a descriptive comment about the purpose of the line. Save the file with your comments and make sure it compiles correctly in the following steps.

```
module tb_tribonacci_calculator;
 logic clk, reset_n;  //logic introduces a new 4-state data type called clk and reset_n
 logic [4:0] input_s;
 logic begin_tribo;
 logic [15:0] tribo_out;
 logic done;
 // instantiate your design
tribonacci_calculator uut(clk, reset_n, input_s, begin_tribo, tribo_out, done);
 // Clock Generator
 always
 begin
        #1250     //waits for 1250, a delay of 1250 ns we can say
        clk = 1'b1;    //assigns clock to 1
        #1250      //delay of 1250 ns again
        clk = 1'b0;    //assigns clock to 0
 end
 initial      //creates an initial block
 begin    //beginning of the intial block
```

```
        /* ------------- Input of 5 ------------- */
      // Reset
        @(posedge clk) reset_n = 0;     //when the clock is on the positive edge, assign reset_n value to 0
        for (int k = 0; k < 2; k++) @(posedge clk);     //executes when the condition, k<2, is true at the positive edge of the clock
        reset_n = 1;       //makes reset_n value 1
        begin_tribo = 1'b0;     //makes begin_tribo value 0
        for (int k = 0; k < 2; k++) @(posedge clk);     //executes when the condition is held
        // Inputs into module/ Assert begin_tribo
        input_s = 5;       //assigns the input_s value to 5
        begin_tribo = 1'b1;     //makes begin_tribo value 1
        for (int k = 0; k < 2; k++) @(posedge clk);
        begin_tribo = 1'b0;     //makes the begin_tribo value 0


        wait (done == 1);       // Wait until calculation is done

        // Idle cycles before next input
        for (int k = 0; k < 2; k++) @(posedge clk);

        // Display
        $display("\n----------------------\n");      //displays the given text
        $display("Input: 5\n");     //displays the given text

        if (tribo_out === 7)
           $display("CORRECT RESULT: %d, GOOD JOB!\n", tribo_out);
        else
           $display("INCORRECT RESULT: %d, SHOULD BE: 7\n", tribo_out);
```

```
    $display("----------------------\n");



    /* ------------- Input of 9 ------------- */
// Reset
    @(posedge clk) reset_n = 0;
    for (int k = 0; k < 2; k++) @(posedge clk);
    reset_n = 1;
    begin_tribo = 1'b0;
    for (int k = 0; k < 2; k++) @(posedge clk);

    // Inputs into module/ Assert begin_tribo
    input_s = 9;
    begin_tribo = 1'b1;
    for (int k = 0; k < 2; k++) @(posedge clk);
    begin_tribo = 1'b0;

    // Wait until calculation is done
    wait (done == 1);

    // Idle cycles before next input
    for (int k = 0; k < 2; k++) @(posedge clk);

    // Display
    $display("\n----------------------\n");
    $display("Input: 9\n");

    if (tribo_out === 81)
       $display("CORRECT RESULT: %d, GOOD JOB!\n", tribo_out);
```

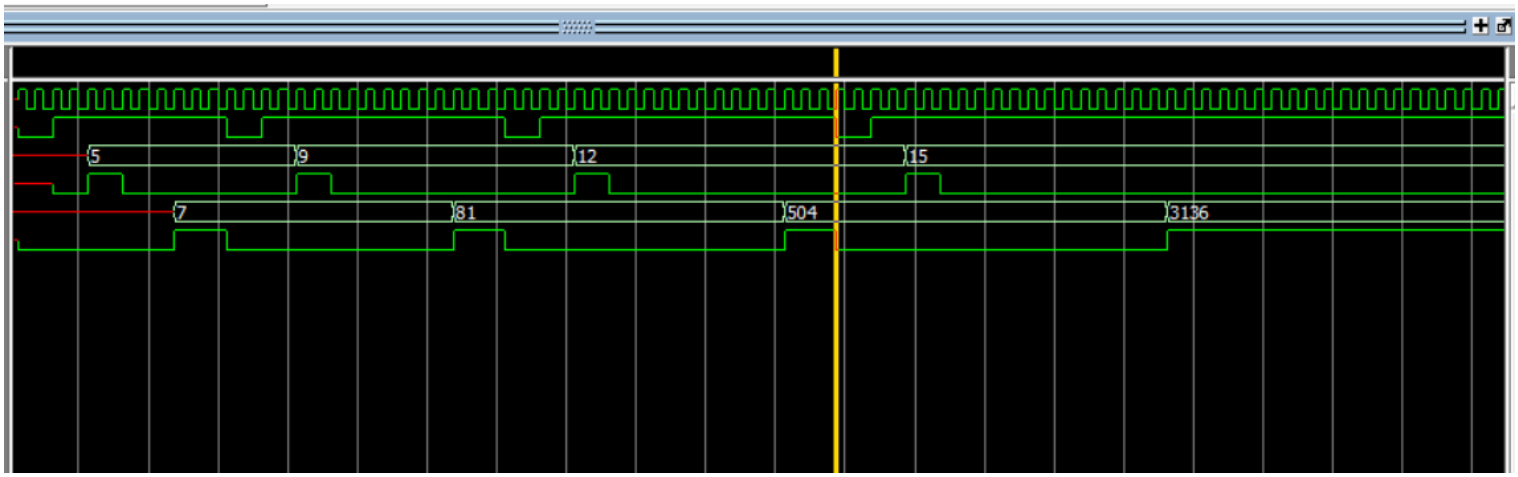
```
    else
        $display("INCORRECT RESULT: %d, SHOULD BE: 81\n", tribo_out);

    $display("----------------------\n");



    /* ------------- Input of 12 ------------- */
// Reset
    @(posedge clk) reset_n = 0;
    for (int k = 0; k < 2; k++) @(posedge clk);
    reset_n = 1;
    begin_tribo = 1'b0;
    for (int k = 0; k < 2; k++) @(posedge clk);

    // Inputs into module/ Assert begin_tribo
    input_s = 12;
    begin_tribo = 1'b1;
    for (int k = 0; k < 2; k++) @(posedge clk);
    begin_tribo = 1'b0;

    // Wait until calculation is done
    wait (done == 1);

    // Idle cycles before next input
    for (int k = 0; k < 2; k++) @(posedge clk);

    // Display
    $display("\n----------------------\n");
```

```
        $display("Input: 12\n");

        if (tribo_out === 504)
           $display("CORRECT RESULT: %d, GOOD JOB!\n", tribo_out);
        else
           $display("INCORRECT RESULT: %d, SHOULD BE: 504\n", tribo_out);

        $display("----------------------\n");



        //MODIFIED PART OF THE CODE FOR INPUT 15 (THE TRIBONACCI OF FIRST 15 NUMBERS)

        /* ------------- Input of 15 ------------- */
      // Reset
        @(posedge clk) reset_n = 0;
        for (int k = 0; k < 2; k++) @(posedge clk);
        reset_n = 1;
        begin_tribo = 1'b0;
        for (int k = 0; k < 2; k++) @(posedge clk);

        // Inputs into module/ Assert begin_tribo
        input_s = 15;
        begin_tribo = 1'b1;
        for (int k = 0; k < 2; k++) @(posedge clk);
        begin_tribo = 1'b0;

        // Wait until calculation is done
        wait (done == 1);
```

```
        // Idle cycles before next input
        for (int k = 0; k < 2; k++) @(posedge clk);

        // Display
        $display("\n----------------------\n");
        $display("Input: 15\n");

        if (tribo_out === 3136)
           $display("CORRECT RESULT: %d, GOOD JOB!\n", tribo_out);
        else
           $display("INCORRECT RESULT: %d, SHOULD BE: 3136\n", tribo_out);

        $display("----------------------\n");


        $stop;     //stops the execution
  end
endmodule
```

QUESTION 4: **Modify the test bench in order to calculate the tribonacci of first 15 numbers at the end of the existing test. Recompile and run your test. Submit both the transcript text and waveforms captured from the end of the existing test to demonstrate functionality.**

- The waveform:



- And the transcript text:

Transcript

```
run
# state = COMPUTE, count =   5, R0 =    1, R1 =    1, R2 =    0
# state = COMPUTE, count =   4, R0 =    2, R1 =    1, R2 =    1
run
# state = COMPUTE, count =   3, R0 =    4, R1 =    2, R2 =    1
#
# ---------------------
#
# Input: 5
#
# CORRECT RESULT:     7, GOOD JOB!
#
# ---------------------
#
run
run
# state = COMPUTE, count =   9, R0 =    1, R1 =    1, R2 =    0
# state = COMPUTE, count =   8, R0 =    2, R1 =    1, R2 =    1
run
# state = COMPUTE, count =   7, R0 =    4, R1 =    2, R2 =    1
# state = COMPUTE, count =   6, R0 =    7, R1 =    4, R2 =    2
# state = COMPUTE, count =   5, R0 =   13, R1 =    7, R2 =    4
# state = COMPUTE, count =   4, R0 =   24, R1 =   13, R2 =    7
run
# state = COMPUTE, count =   3, R0 =   44, R1 =   24, R2 =   13
#
# ---------------------
#
# Input: 9
#
# CORRECT RESULT:    81, GOOD JOB!
#
# ---------------------
#
```

```
run
# state = COMPUTE, count =  12, R0 =    1, R1 =    1, R2 =    0
# state = COMPUTE, count =  11, R0 =    2, R1 =    1, R2 =    1
run
# state = COMPUTE, count =  10, R0 =    4, R1 =    2, R2 =    1
# state = COMPUTE, count =   9, R0 =    7, R1 =    4, R2 =    2
# state = COMPUTE, count =   8, R0 =   13, R1 =    7, R2 =    4
# state = COMPUTE, count =   7, R0 =   24, R1 =   13, R2 =    7
run
# state = COMPUTE, count =   6, R0 =   44, R1 =   24, R2 =   13
# state = COMPUTE, count =   5, R0 =   81, R1 =   44, R2 =   24
# state = COMPUTE, count =   4, R0 =  149, R1 =   81, R2 =   44
# state = COMPUTE, count =   3, R0 =  274, R1 =  149, R2 =   81
run
#
# ---------------------
#
# Input: 12
#
# CORRECT RESULT:   504, GOOD JOB!
run
# state = COMPUTE, count =  15, R0 =    1, R1 =    1, R2 =    0
# state = COMPUTE, count =  14, R0 =    2, R1 =    1, R2 =    1
# state = COMPUTE, count =  13, R0 =    4, R1 =    2, R2 =    1
run
# state = COMPUTE, count =  12, R0 =    7, R1 =    4, R2 =    2
# state = COMPUTE, count =  11, R0 =   13, R1 =    7, R2 =    4
# state = COMPUTE, count =  10, R0 =   24, R1 =   13, R2 =    7
# state = COMPUTE, count =   9, R0 =   44, R1 =   24, R2 =   13
run
# state = COMPUTE, count =   8, R0 =   81, R1 =   44, R2 =   24
# state = COMPUTE, count =   7, R0 =  149, R1 =   81, R2 =   44
# state = COMPUTE, count =   6, R0 =  274, R1 =  149, R2 =   81
# state = COMPUTE, count =   5, R0 =  504, R1 =  274, R2 =  149
run
# state = COMPUTE, count =   4, R0 =  927, R1 =  504, R2 =  274
# state = COMPUTE, count =   3, R0 = 1705, R1 =  927, R2 =  504
run
#
# ---------------------
#
# Input: 15
#
# CORRECT RESULT:  3136, GOOD JOB!
#
# ---------------------
#
# Break in Module tb_tribonacci_calculator at D:/Fifth Semester/CNG 331/PROJ
```

QUESTION 5:

	Dynamic Thermal Power	Core Static Thermal Power	Core Junction Temperature	Cydn
100 MHz	**0.00 mW**	**154.96 mW**	**25.0 degrees Celcius**	**0**
200 MHz	**55.12 mW**	**155.68 mW**	**26.7 degrees Celcius**	**0.191 mW.ns/V^2**

- As the frequency increases from 100 MHz to 200 MHz, there is a huge difference between core dynamic thermal power values while the core static thermal power and core junction temperature values are very close to each other. But we can say that, obviously, all the values mentioned are higher when the clock frequency is 200 MHz. As the core dynamic thermal power is 0 for clock frequency 100 MHz, the Cydn value is also 0. When we look at the dominant power dissipation mode for this particular design, as the change is negligible for static case, we can say that the dominant mode is the dynamic one.

CONCLUSION

My estimated success for this CAD assignment is 80% because I think the first 4 questions are the ones I did without struggling, but for the last question, as it needs more analysis and calculations, there might be some calculation mistakes or mismatches with the design values taken depending on the time we simulate them. I simulated all the required files without any issue so I can say that the simulation of this design was very successful for me. I only struggled to look up for some values asked but after searching, I also learnt how to do it. I did so much research to understand some core basics about the topic to be able to complete those questions and I believe I've learnt so much by this. I learnt how to simulate power analysis, the differences between static and dynamic power dissipation, and how to calculate Cydn value for a specific clock frequency. I also observed the changes in power dissipation values with the different clock frequencies.