Laboratory Report #7

Name: **German E Felisarta III** **Group Number: 3**

Laboratory Exercise Title: **Modularized Design of Digital Systems** **Date Completed:** **11/25/2020**

Target Course Outcomes:

CO1: Create descriptions of digital hardware components such as in combinational and sequential circuits using synthesizable Verilog HDL constructs.
CO2: Verify the functionality of HDL-based components through design verification tools.

Exercise 7A:

In this exercise, a disk spinner is to be simulated. Two sub modules are made, the counter module and the 7-segment decoder module. In the main module, the SSeg output is given a wire so that it can be assigned the value from the SSegDecoder module. The same goes for the C0, or count_out variable which can be found in the counterModule module. The counterModule module is the one responsible for checking the counter value and looping the counter. The SSegDecoder module is responsible for assigning the disk state to its corresponding counter value as shown in the lab guide.

Fig 1a. Design Entry of Disk Spinner

```
//GERMAN E FELISARTA III 16101002  CpE3101L Grp 3

module DiskSpinAnimation_Top (Clk, nReset, Start, SSeg);

	input wire Clk, nReset, Start;
	output reg [7:0]SSeg;
	wire [2:0]C0;
	wire [7:0]sSSeg;

	counterModule CM0(Clk, Start, nReset, C0);
	SSegDecoder  SSD0(C0, sSSeg);

	always @ (*) begin

		SSeg <= sSSeg;

	end

endmodule

module counterModule (cClk, cStart, cReset, cCounter_Out);

	input cClk, cStart, cReset;
	output reg [2:0]cCounter_Out;

	always @ (posedge cClk) begin
```

```
                if(cStart) begin

                        cCounter_Out <= 3'b000;

                        if(!cReset) begin

                                cCounter_Out <= (cCounter_Out + 1'b1);

                                        if (cCounter_Out == 3'b111)
                                                cCounter_Out <= 3'b000;

                        end
                        else cCounter_Out <= 3'b000;

                end
                else cCounter_Out <= 3'b000;

        end

endmodule


module SSegDecoder(sCounter_Out, SSeg);

        input wire [2:0]sCounter_Out;
        output reg [7:0]SSeg;

        always @ (*) begin

                case (sCounter_Out)

                        3'b000 : SSeg <= 8'b00000000;
                        3'b001 : SSeg <= 8'b01100010; // BGF
                        3'b010 : SSeg <= 8'b01100001; // GFA
                        3'b011 : SSeg <= 8'b00100011; // FAB
                        3'b100 : SSeg <= 8'b01000011; // ABG
                        3'b101 : SSeg <= 8'b01100011; // ABGF
                        3'b111 : SSeg <= 8'b01100011; // ABGF
                        default : SSeg <= 8'b00000000;

                endcase

        end

endmodule
```

Fig 1b. Flow Summary of Disk Spinner

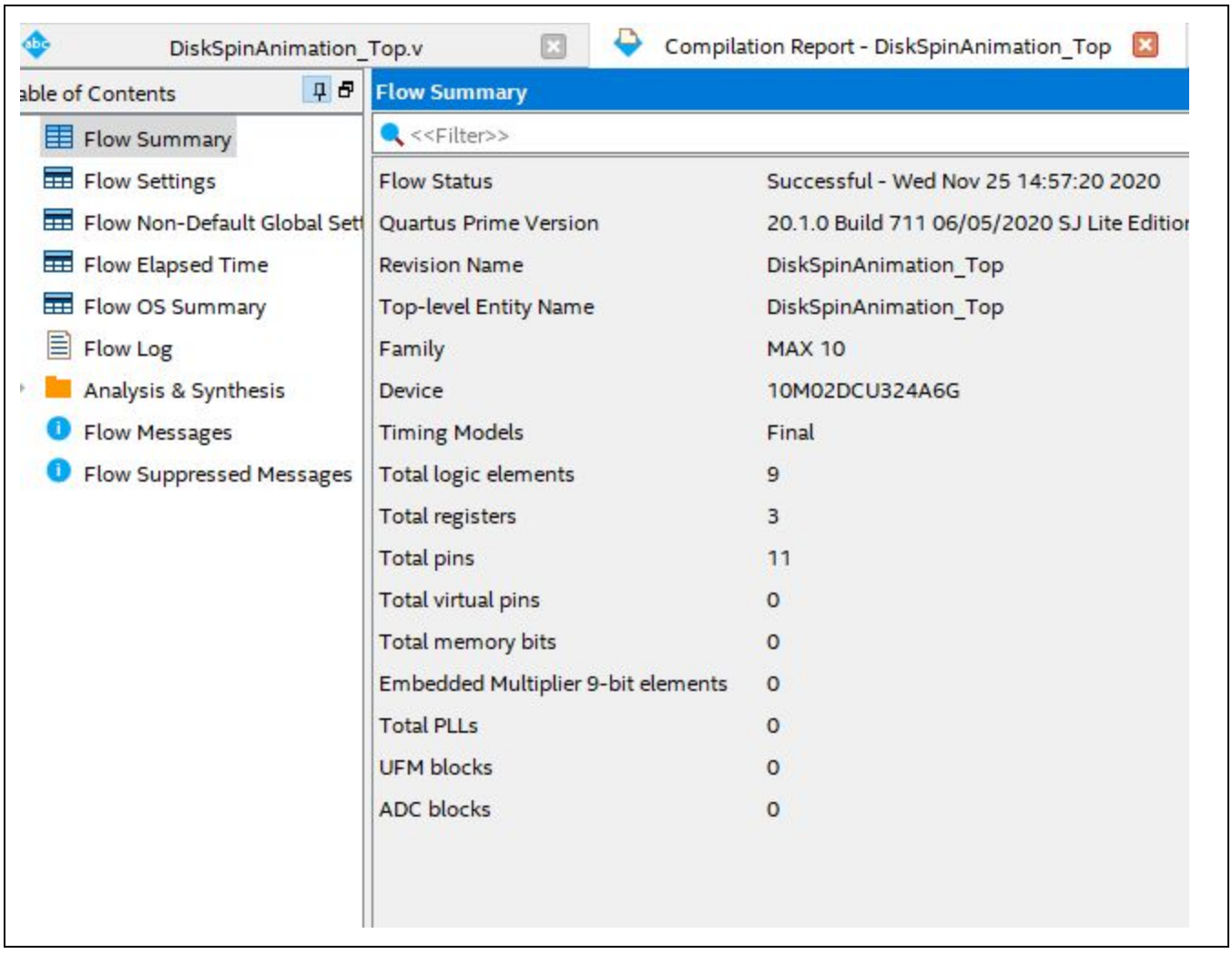
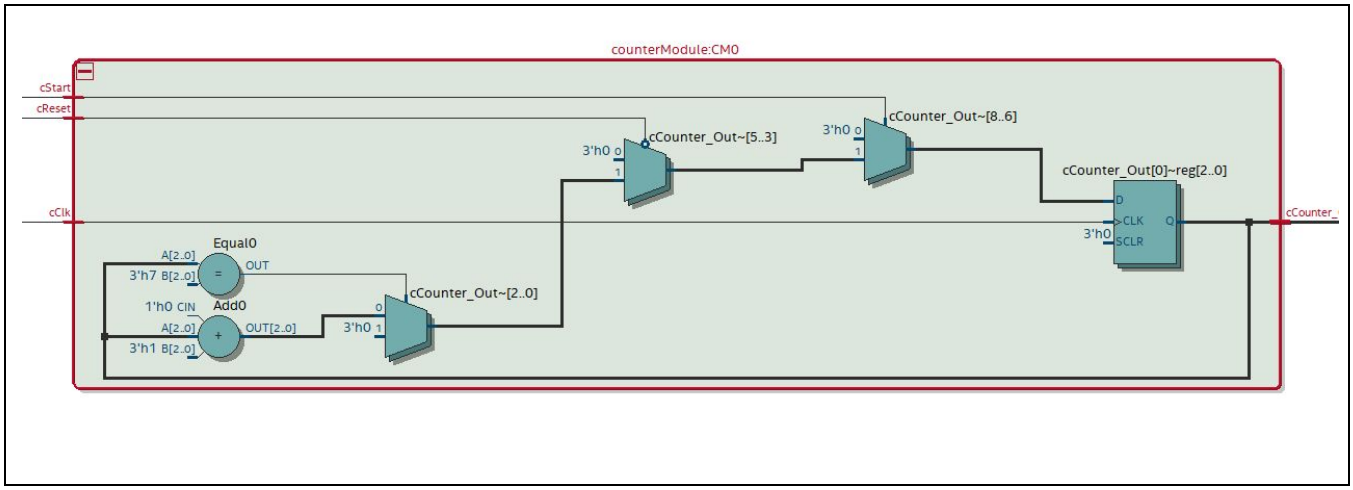


Fig 1c. RTL View of Disk Spinner



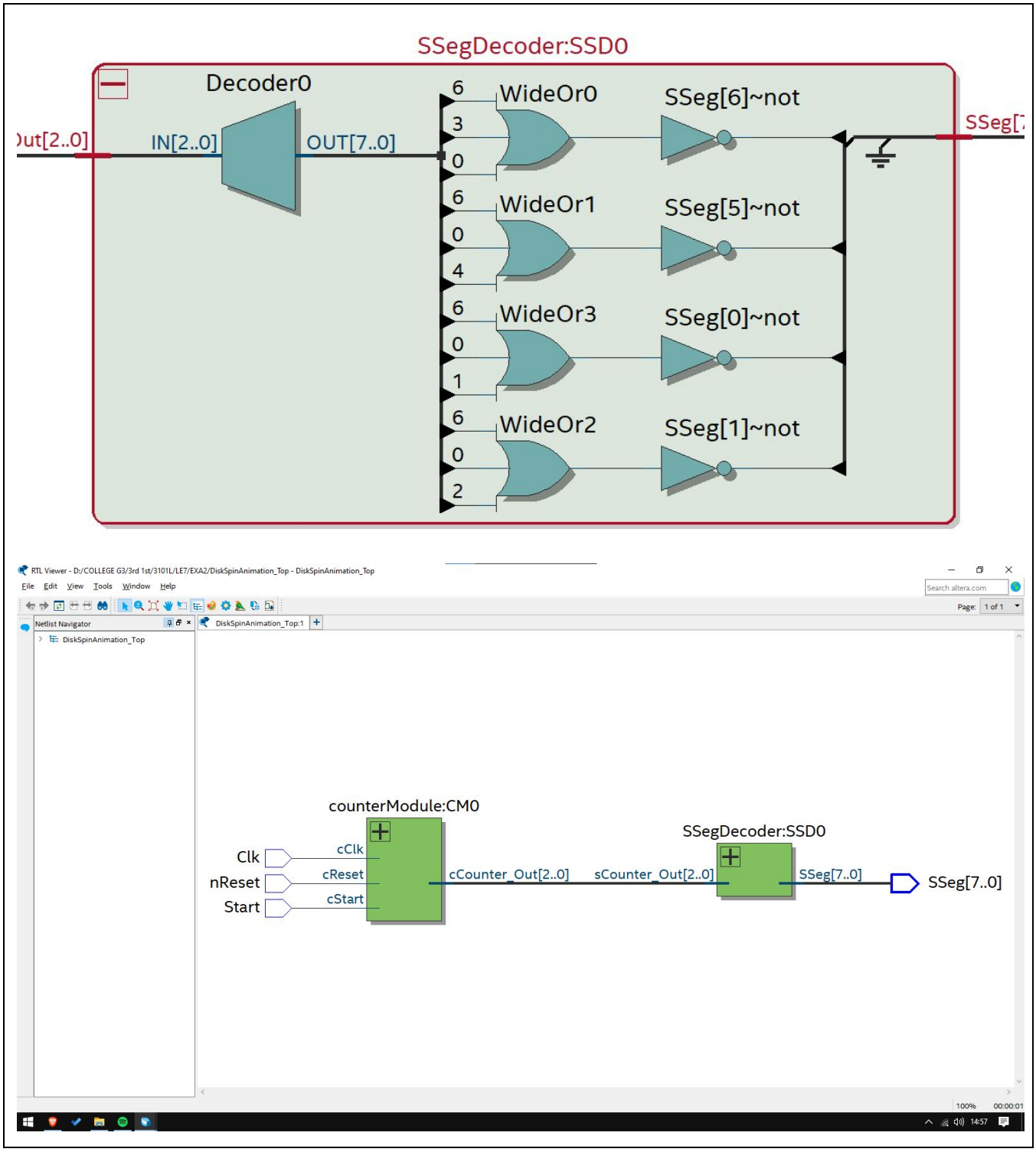
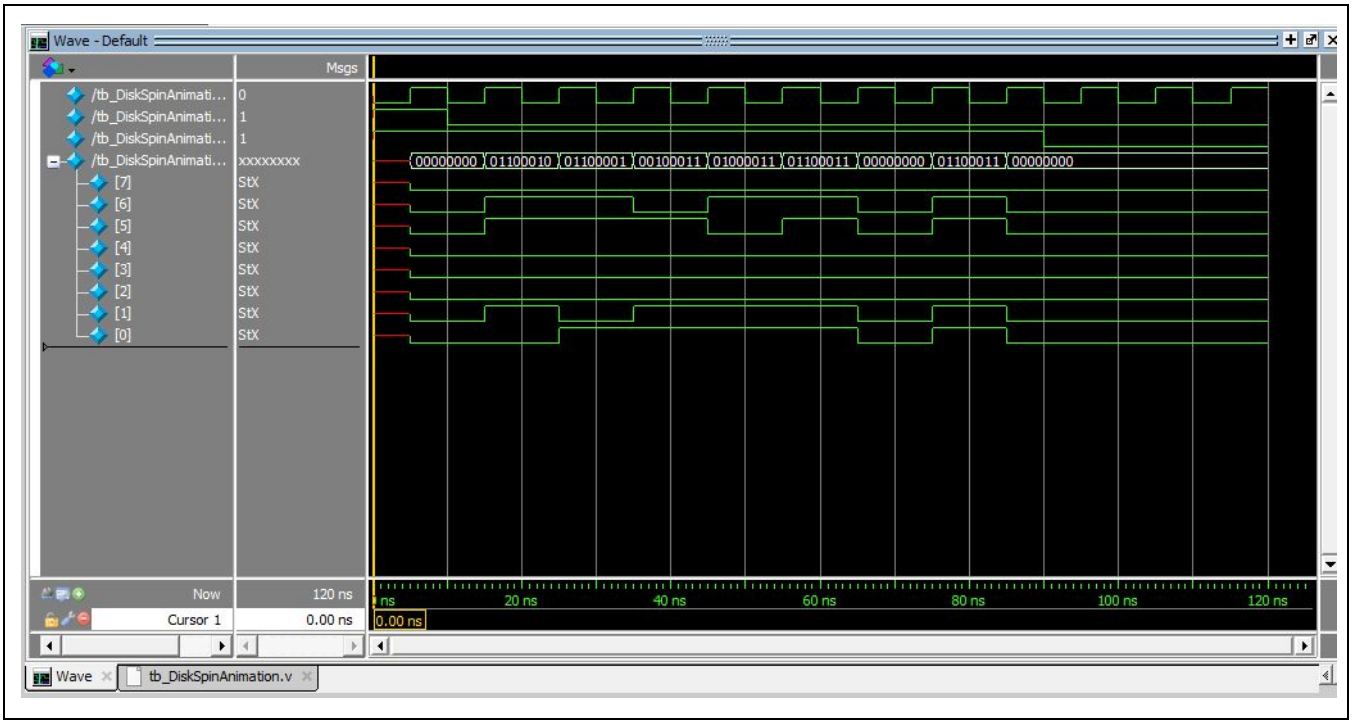

SSegDecoder:SSD0
Decoder0
)ut[2..0]
IN[2..0]
OUT[7..0]
6
3
0
WideOr0
SSeg[6]~not
SSeg[7
6
0
4
WideOr1
SSeg[5]~not
6
0
1
WideOr3
SSeg[0]~not
6
0
2
WideOr2
SSeg[1]~not
RTL Viewer - D:/COLLEGE G3/3rd 1st/3101L/LE7/EXA2/DiskSpinAnimation_Top - DiskSpinAnimation_Top
File Edit View Tools Window Help
Netlist Navigator
DiskSpinAnimation_Top
DiskSpinAnimation_Top:1
counterModule:CM0
Clk
nReset
Start
cClk
cReset
cStart
cCounter_Out[2..0]
sCounter_Out[2..0]
SSegDecoder:SSD0
SSeg[7..0]
SSeg[7..0]

Fig 1d. Test Bench Design Entry of Disk Spinner

DiskSpinAnimation_Top.v | Compilation Report - DiskSpinAnimation_Top | tb_DiskSpinAnimation.v

```
1  //GERMAN E FELISARTA III 16101002   CpE 3101L Grp 3
2
3  `timescale 1 ns / 1 ps
4  module tb_DiskSpinAnimation();
5
6     reg Clk, nReset, Start;
7     wire [7:0]SSeg;
8
9     DiskSpinAnimation_Top  UUT (Clk, nReset, Start, SSeg);
10
11    initial
12       Clk = 1'b0;
13
14    always
15       #5 Clk = ~Clk;
16
17    initial begin
18       nReset = 1'b1; #10
19       nReset = 1'b0;
20    end
21
22    initial begin
23       $display("Starting Simulation at %0d ns...", $time);
24
25       Start = 1'b1; #90
26       Start = 1'b0; #30
27
28       $display("Finished simulation at %0d ns...", $time);
29       $stop;
30
31    end
32 endmodule
33
34
```

Fig 1e. RTL Simulation of Disk Spinner



Exercise 7B:

In this exercise, a 3-digit stopwatch is to be made. After all the tries made, this exercise was unsuccessfully delivered. The output shows that it does not count up the values and instead just skips to another value.

Fig 2a. Design Entry of StopWatch

```
//German E Felisarta III 16101002          CpE 3101L Grp 3

module StopWatch(Clk, nReset, Start, S0, S1, S2);

        input wire Clk, nReset, Start;
        output reg [7:0]S0;
        output reg [7:0]S1;
        output reg [7:0]S2;

        wire [3:0]wHex0;
        wire [3:0]wHex1;
        wire [3:0]wHex2;

        wire [7:0]wS0;
        wire [7:0]wS1;
        wire [7:0]wS2;

        reg EnableDec, EnableOnes, EnableTens;

        BCDCounter BC0(Clk, nReset, EnableDec, wHex0);
        BCDCounter BC1(Clk, nReset, EnableOnes, wHex1);
        BCDCounter BC2(Clk, nReset, EnableTens, wHex2);

        hexaDigit  HD0(wHex0, 1'b0, wS0);
        hexaDigit  HD1(wHex1, 1'b0, wS1);
        hexaDigit  HD2(wHex2, 1'b0, wS2);



        always @ (negedge Clk) begin

                S0 <= wS0;
                S1 <= wS1;
                S2 <= wS2;

                if(Start) begin
                        EnableDec <= 1'b1;
                        if(wS0 == 4'b1001) EnableOnes <= 1'b1;
                        if(wS1 == 4'b1001) EnableTens <= 1'b1;

                end
                else begin

                        EnableDec <= 1'b0;
                        EnableOnes <= 1'b0;
```

```
                        EnableTens <= 1'b0;

                end
        end
endmodule



module BCDCounter(xClk, xnReset, Enable, Count_out);

        input xClk, xnReset, Enable;
        output reg [3:0]Count_out;

        always @ (negedge xClk) begin


                if(!xnReset) Count_out <= 4'b0000;
                else begin

                        if(Enable) begin
                                if (Count_out == 4'b1010)
                                        Count_out <= 4'b0000;
                                else
                                        Count_out <= (Count_out + 1'b1);

                        end
                        else Count_out <= Count_out;

                end

        end

endmodule


//GERMAN E FELISARTA III 16101002  CpE 3101L EX5B

module hexaDigit(Hex, DP, SSeg);

        input wire [3:0]Hex;
        input wire DP;
        output reg [7:0]SSeg;

        always @ (*) begin

                                case ({DP, Hex})
                                        5'b00000 : SSeg = 8'b0111111;
                                        5'b00001 : SSeg = 8'b0000110;
                                        5'b00010 : SSeg = 8'b1011011;
                                        5'b00011 : SSeg = 8'b1001111;
                                        5'b00100 : SSeg = 8'b1100110;
                                        5'b00101 : SSeg = 8'b1101101;
                                        5'b00110 : SSeg = 8'b1111101;
```

```
                                    5'b00111 : SSeg = 8'b0000111;
                                    5'b01000 : SSeg = 8'b1111111;
                                    5'b01001 : SSeg = 8'b1101111;
                                    5'b01010 : SSeg = 8'b1110111;
                                    5'b01011 : SSeg = 8'b1111100;
                                    5'b01100 : SSeg = 8'b0111001;
                                    5'b01101 : SSeg = 8'b1011110;
                                    5'b01110 : SSeg = 8'b1111001;
                                    5'b01111 : SSeg = 8'b1110001;
                                    default  : SSeg = 8'b0000000;

                            endcase
                end

endmodule
```

Fig 2b. Flow Summary of StopWatch

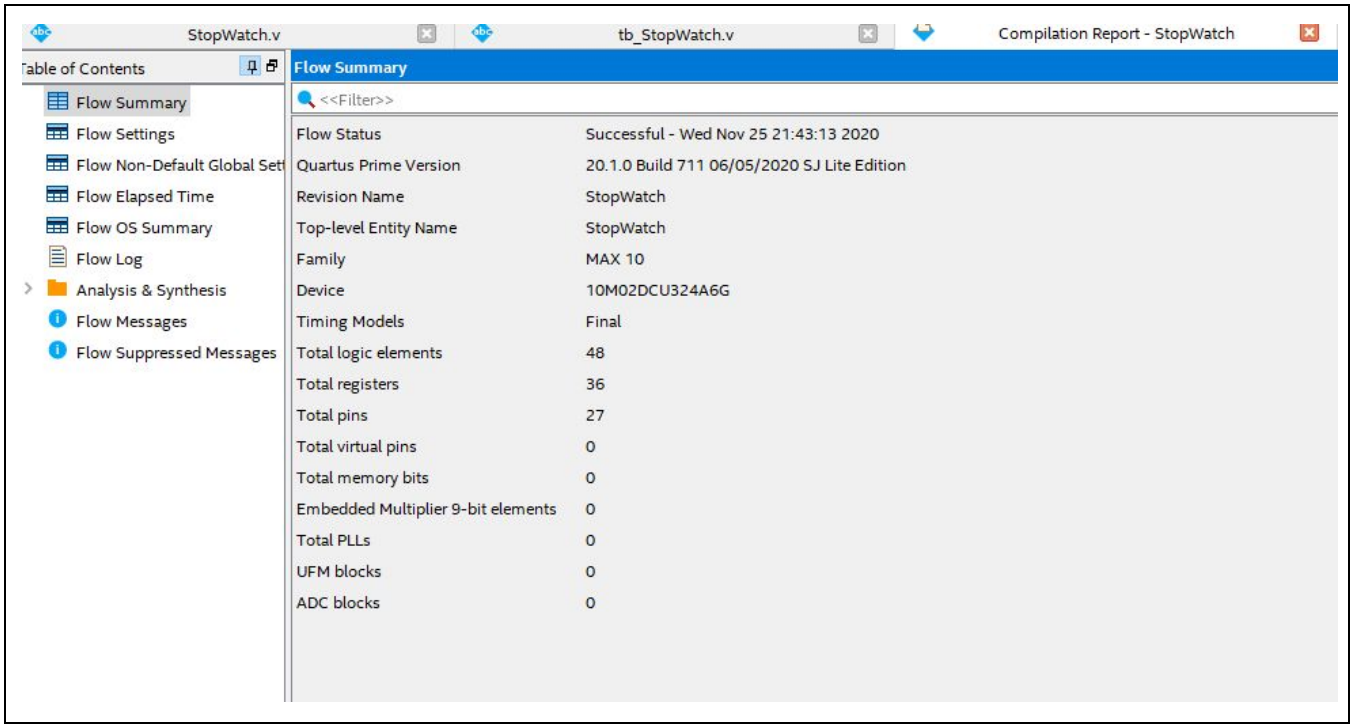


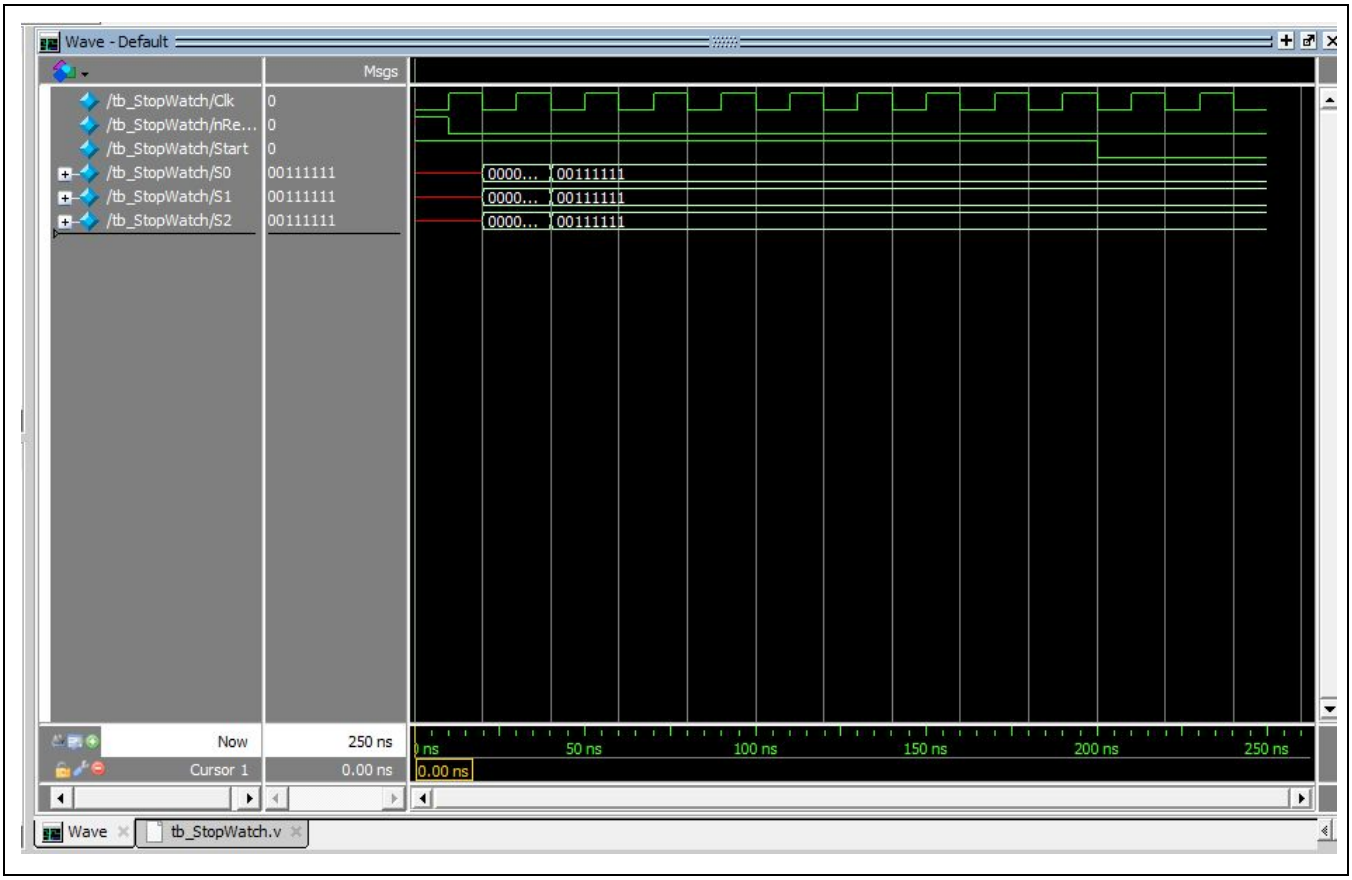
Fig 2c. RTL View of StopWatch

Fig 2d. Test Bench Design Entry of StopWatch

```
//German E Felisarta III 16101002 CpE 3101L Grp 3

`timescale 1 ns / 1 ps
module tb_StopWatch();

	reg Clk, nReset, Start;
	wire [7:0]S0;
	wire [7:0]S1;
	wire [7:0]S2;

	StopWatch UUT (Clk, nReset, Start, S0, S1, S2);

	initial
		Clk = 1'b0;

	always
		#10 Clk = ~Clk;

	initial begin
		nReset = 1'b1; #10
		nReset = 1'b0;
	end
```

```
        initial begin
                $display("Starting simulation");

                Start = 1'b1; #200
                Start = 1'b0; #50

                $display("Finished Simulation");
                $stop;

        end

endmodule
```

Fig 2e. RTL Simulation of StopWatch



Exercise 7C:

In this Exercise, the n-Bit ALU from Exercise 5 was improved. Instead of just basic arithmetic instructions, the Shift Instructions were added. The original ALU module is directly copied from the verilog file from Exercise 5B. The shift instructions were carried out for the other module named CombShiftMod. In the main module ALUnBit, the logic for picking the module is written. If the MODE input is greater then 1'b0111, then it would use the result from CombShiftMod, otherwise, from the OrigALU module.

Fig 3a. Design Entry of ALU

```
//German E Felisarta III  16101002          CpE 3101L Grp 3

module ALUnBit(A, B, CB_in, Mode, Result, CB_out);

        localparam n = 4;
        input wire [(n-1):0]A;
        input wire [(n-1):0]B;
        input wire CB_in;
        input wire [3:0]Mode;

        output reg CB_out;
        output reg [(n-1):0]Result;

        wire [(n-1):0]oResult;
        wire [(n-1):0]cResult;
        wire wCB_out;

        OrigALU OA0 (A, B, CB_in, Mode, oResult, wCB_out);
        CombShiftMod CSM0(A, B, CB_in, Mode, cResult);

        always @ (*) begin

                CB_out <= wCB_out;

                if (Mode <= 4'b0111) Result <= oResult;
               else Result <= cResult


        end


endmodule

//From EX5C

module OrigALU (A, B, CB_in, Mode, Result, CB_out);

        localparam n = 4;
        input [(n-1):0]A;
        input [(n-1):0]B;
        input CB_in;
        input [3:0]Mode;
```

```
	output reg CB_out;
	output reg [(n-1):0]Result;

	always @(*)
		begin
			case (Mode)

				4'b0000 :
					begin Result = (A + B); CB_out = 1; end
				4'b0001 :
					begin Result = (A - B); CB_out = 1; end
				4'b0010 :
					Result = (A & B);
				4'b0011 :
					Result = (A | B);
				4'b0100 :
					Result = (A ^ B);
				4'b0101 :
					Result = (~A);
				4'b0110 :
					Result = (A + 1'b1);
				4'b0111 :
					Result = (A - 1'b1);

			endcase
		end
endmodule

module CombShiftMod(A, B, CB_in, Mode, Result);

	localparam n = 4;
	input [(n-1):0]A;
	input [(n-1):0]B;
	input CB_in;
	input [3:0]Mode;

	output reg [(n-1):0]Result;

	always @ (*)
		begin
			case (Mode)

				4'b1000 :
					Result = A << 1;
				4'b1001 :
					Result = ~(A << 1);
				4'b1010 :
					Result = A >> 1;
				4'b1011 :
					Result = ~(A >> 1);
				4'b1100 :
					Result = A <<< 1;
				4'b1101 :
```

```
                                    Result = A >>> 1;
                                4'b1110 :
                                    Result = {A[n-1], A[(n-2):0]};
                                4'b1111 :
                                    Result = {A[0], A[(n-1):1]};


                        endcase
                end

endmodule
```

Fig 3b. Flow Summary of ALU

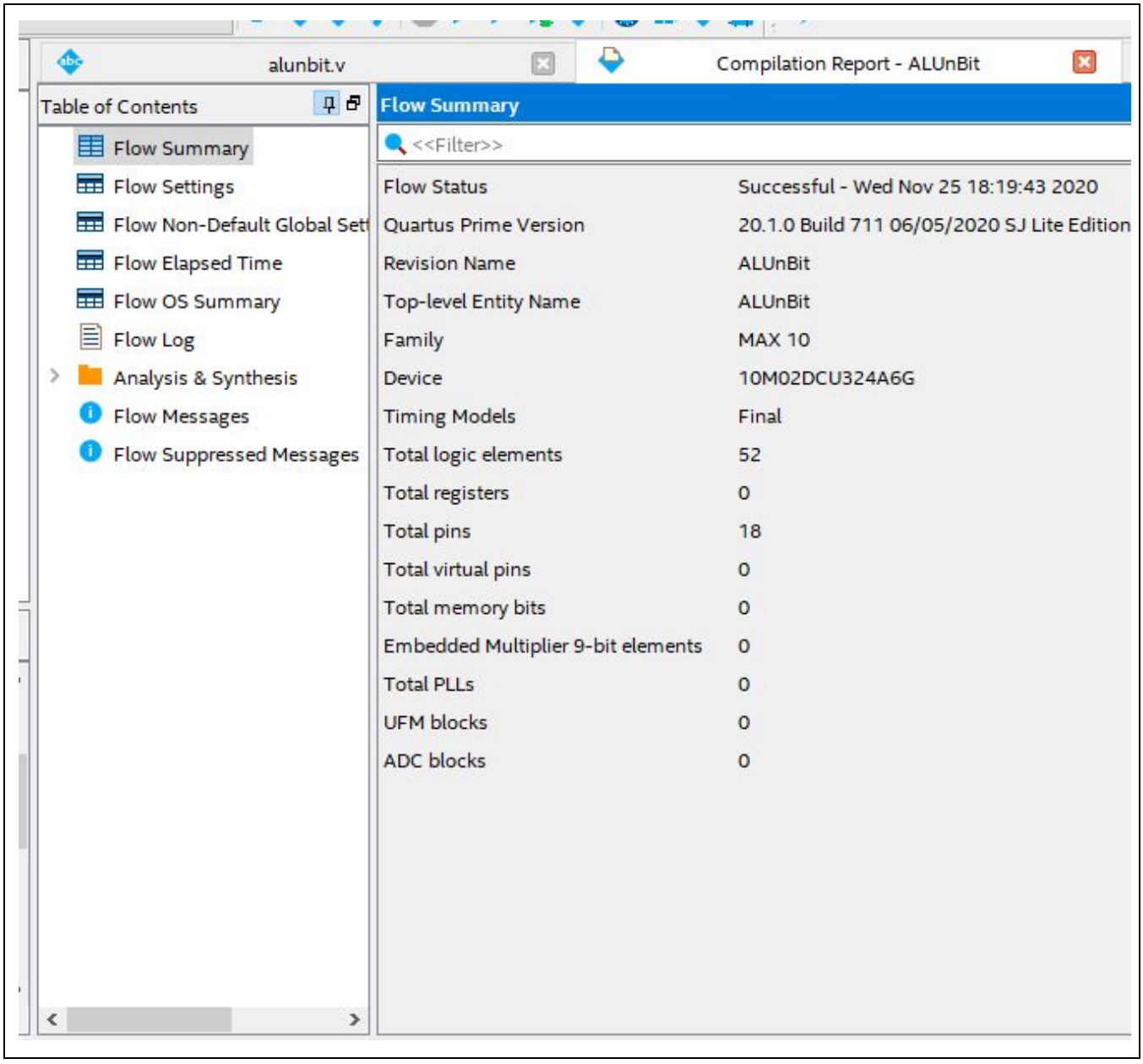


Fig 3c. RTL View of ALU

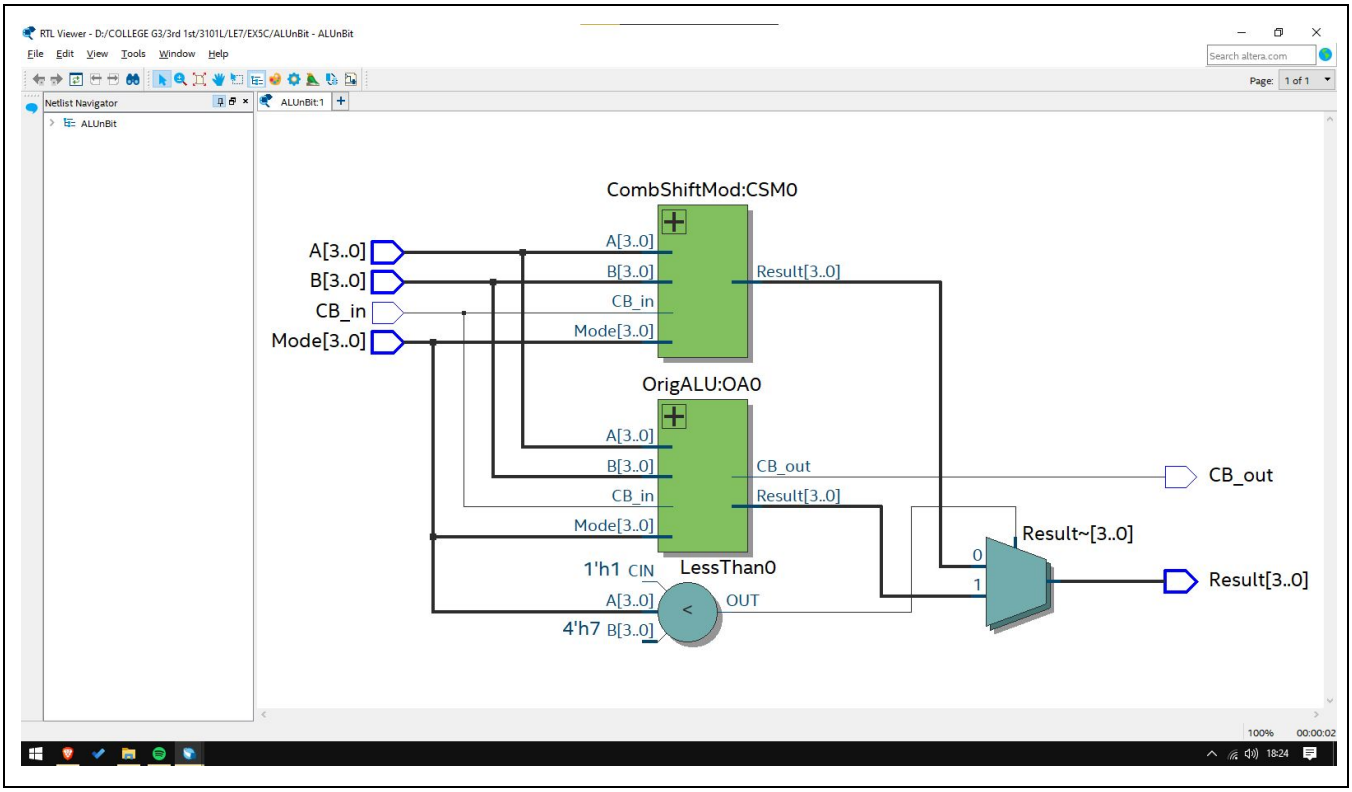


Fig 3d. Test Bench Design Entry of ALU

```
//German E Felisarta III  16101002        CpE 3101L Grp 3

`timescale 1 ns / 1 ps
module tb_ALUnBit();

        localparam n = 4;

        reg [(n-1):0]A;
        reg [(n-1):0]B;
        reg CB_in;
        reg [3:0]Mode;

        wire CB_out;
        wire [(n-1):0]Result;

        ALUnBit UUT (A, B, CB_in, Mode, Result, CB_out);

        initial begin
                $display("Starting simulation at %0d ns...", $time);

                //0000
                Mode = 4'b0000;
```

```
A = 4'b0001; B = 4'b0010; CB_in = 1'b1; #5

A = 4'b0011; B = 4'b0001; CB_in = 1'b1; #5

A = 4'b0101; B = 4'b0010; CB_in = 1'b0; #5

//0001
Mode = 4'b0001;

A = 4'b0011; B = 4'b0010; #5


A = 4'b0111; B = 4'b0011; #5

A = 4'b0001; B = 4'b0010; #5

//0010
Mode = 4'b0010;

A = 4'b0011; B = 4'b0010; #5

A = 4'b0111; B = 4'b0111; #5

A = 4'b0010; B = 4'b0010; #5

//0011
Mode = 4'b0011;

A = 4'b0011; B = 4'b0010; #5


A = 4'b0111; B = 4'b0111; #5

A = 4'b0010; B = 4'b0010; #5

//0100
Mode = 4'b0100;

A = 4'b0011; B = 4'b0010; #5

A = 4'b0111; B = 4'b0111; #5

A = 4'b0010; B = 4'b0010; #5

//0101
Mode = 4'b0101;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5
```

```
//0110
Mode = 4'b0110;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5

//0111
Mode = 4'b0111;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5

//1000
Mode = 4'b1000;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5

//1001
Mode = 4'b1001;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5

//1010
Mode = 4'b1010;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5

//1011
Mode = 4'b1011;

A = 4'b0011; #5

A = 4'b0111; #5

A = 4'b0010; #5
```

```
                //1100
                Mode = 4'b1100;

                A = 4'b0011; #5

                A = 4'b0111; #5

                A = 4'b0010; #5

                //1101
                Mode = 4'b1101;

                A = 4'b0011; #5

                A = 4'b0111; #5

                A = 4'b0010; #5

                //1110
                Mode = 4'b1110;

                A = 4'b0011; #5

                A = 4'b0111; #5

                A = 4'b0010; #5

                //1111
                Mode = 4'b1111;

                A = 4'b0011; #5

                A = 4'b0111; #5

                A = 4'b0010; #5


                $display("Finished simulation at %0d ns.", $time);
                $stop;
        end

endmodule
```

Fig 3e. RTL Simulation of ALU