Laboratory Report #8

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

Laboratory Exercise Title: Finite State Machines (FSMs) **Date Completed: 12/6/2020**

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 8A:

In this exercise, a Moore FSM is made for a Binary/Gray Counter. The state diagram was first accomplished. After finishing the state diagram, the design entry was made based on it. Since it is a Moore FSM, it is relatively easy to create. The timing diagram outputs exactly what was expected or displayed on the State Diagram.

Fig 1a. State Diagram for Binary/Gray Counter

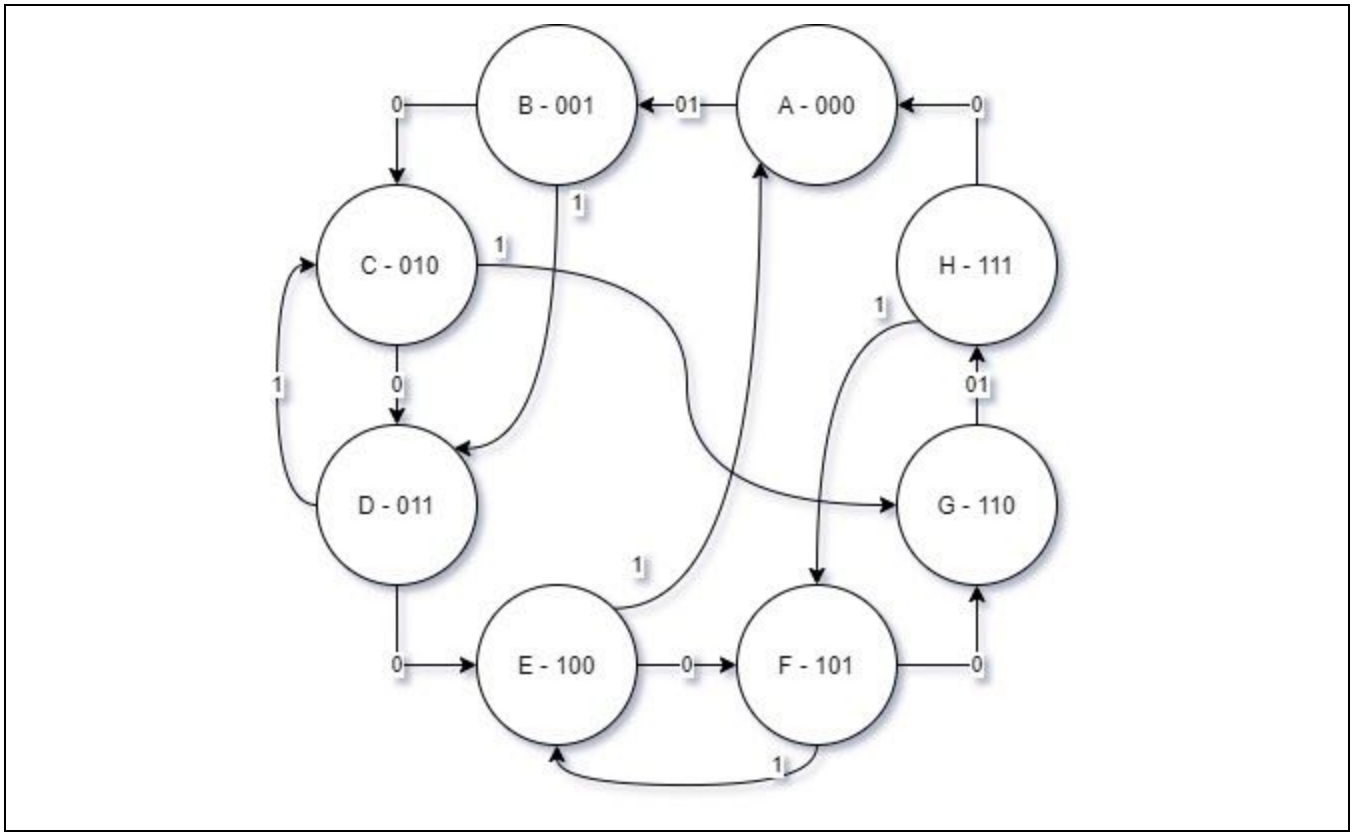


Fig 1b. Design Entry for Complex Counter

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

module ComplexCounter(CLOCK, nReset, M, COUNT);

	input wire CLOCK, nReset, M;
	output reg [2:0]COUNT;

	parameter A = 3'b000;
	parameter B = 3'b001;
	parameter C = 3'b010;
	parameter D = 3'b011;
	parameter E = 3'b100;
	parameter F = 3'b101;
	parameter G = 3'b110;
	parameter H = 3'b111;
	reg [2:0]cstate, nstate;

	always @ (posedge CLOCK or negedge nReset)begin

		if (!nReset)
			cstate <= A;
		else
			cstate <= nstate;

	end

	always @ (*) begin

		case(cstate)

			A : nstate <= (M) ? B : B;
			B : nstate <= (M) ? D : C;
			C : nstate <= (M) ? G : D;
			D : nstate <= (M) ? C : E;
			E : nstate <= (M) ? A : F;
			F : nstate <= (M) ? E : G;
			G : nstate <= (M) ? H : H;
			default : nstate <= (M) ? F : A;

		endcase

	end

	always @ (*) begin

		COUNT <= cstate;

	end
endmodule
```

Fig 1c. Flow Summary for Complex Counter

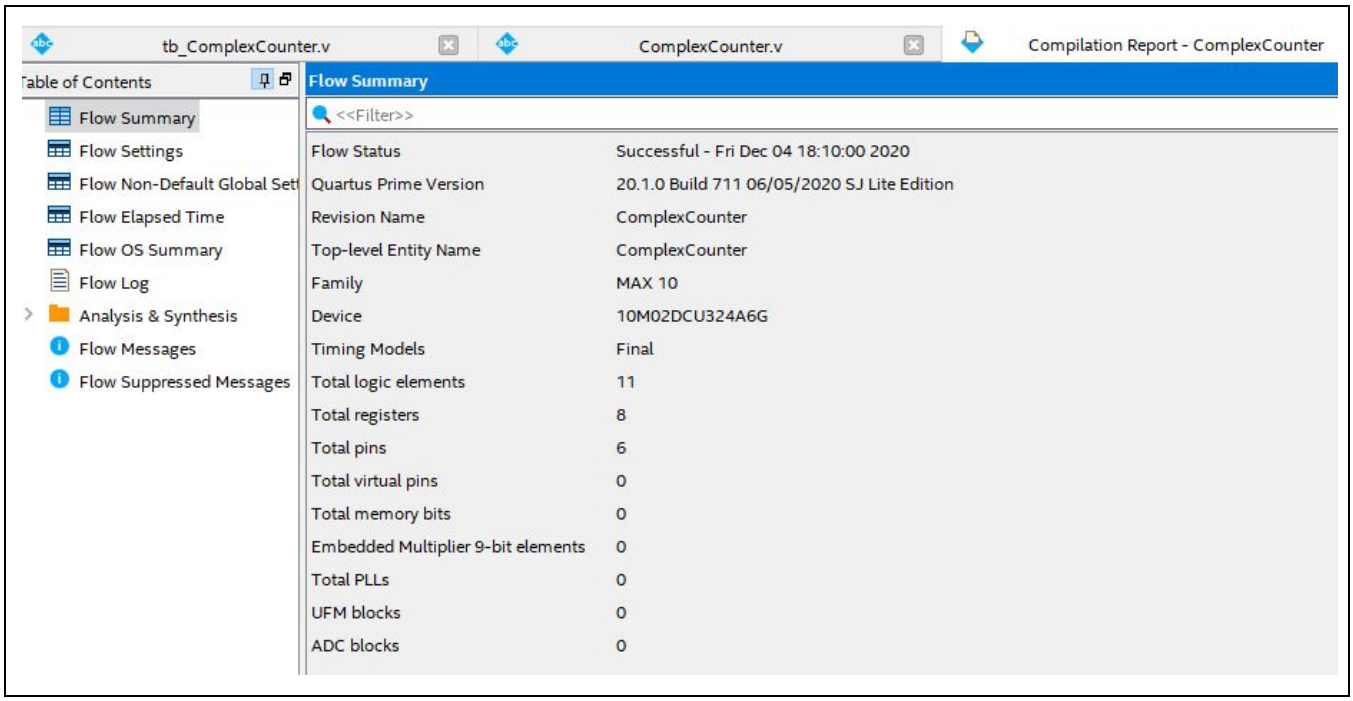


Fig 1d. Design Entry for Test Bench Complex Counter

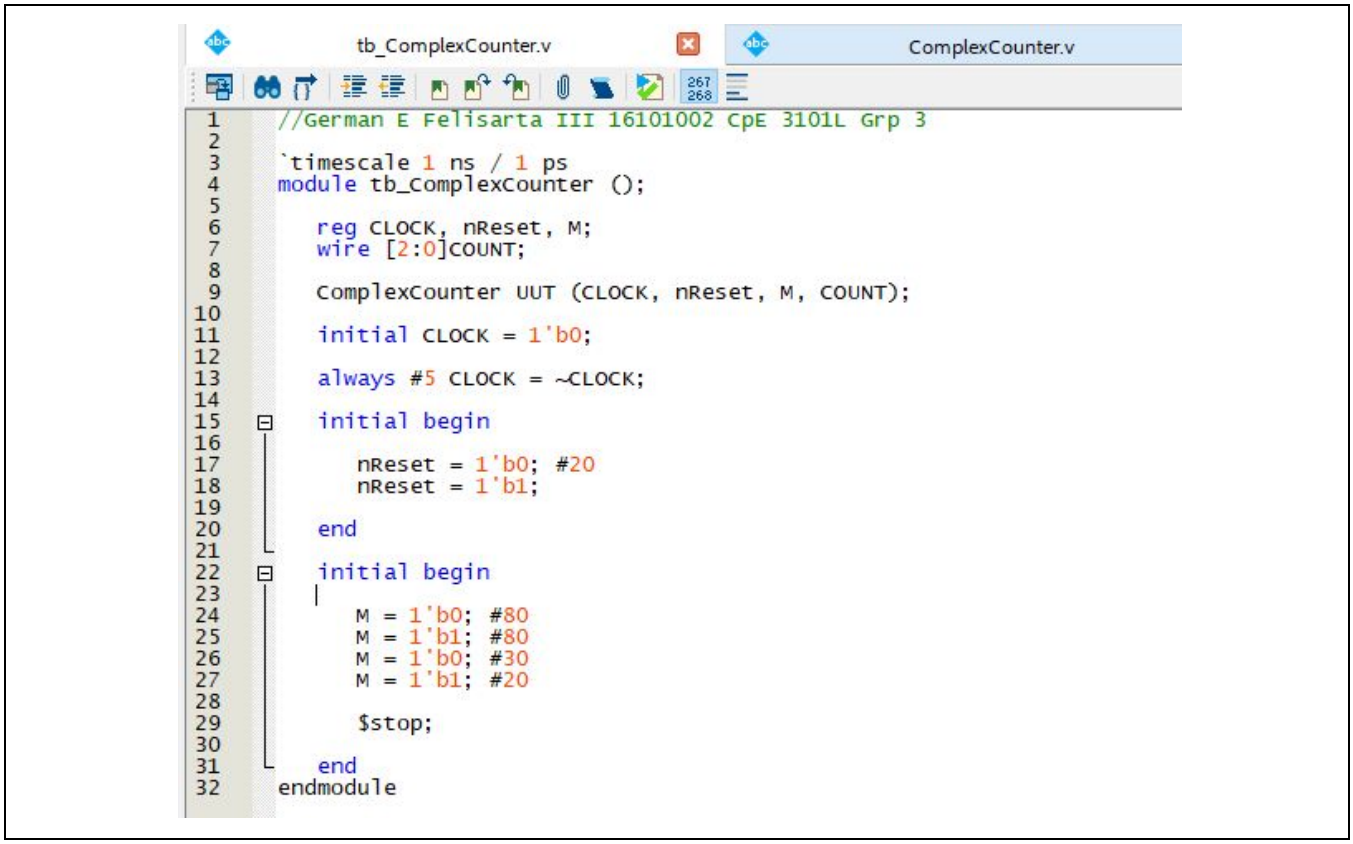
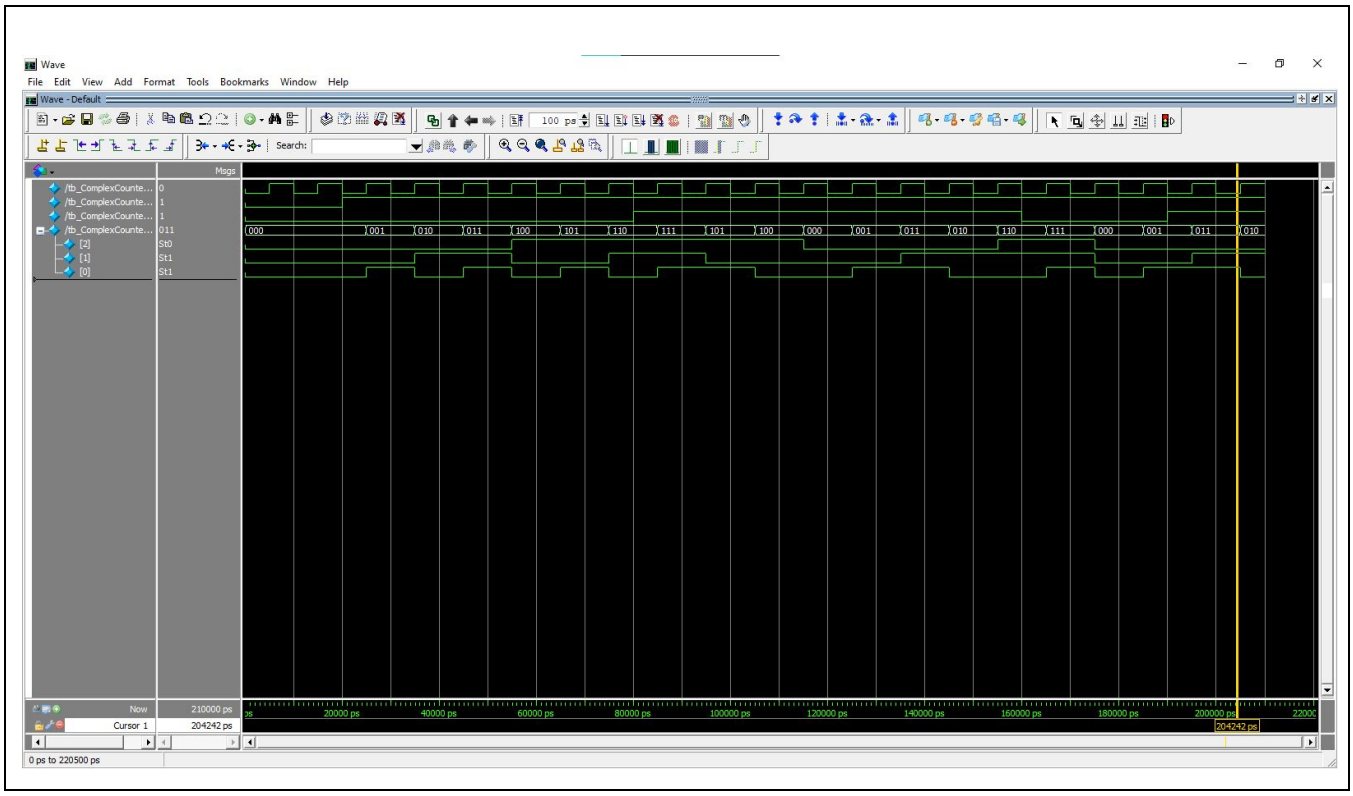


Fig 1e. RTL Simulation for Complex Counter



Exercise 8B:

In this exercise, a Race Lights Controller is to be made, a Mealy FSM. First, the state diagram is made and the basis for input is the START variable. The output is the concatenation of RED, YELLOW, and GREEN variable outputs. The design entry was modeled after the State diagram. The timing diagram shows that it follows the expected output and displays the outputs projected by the state diagram.

Fig 2a. State Diagram for Race Lights Controller

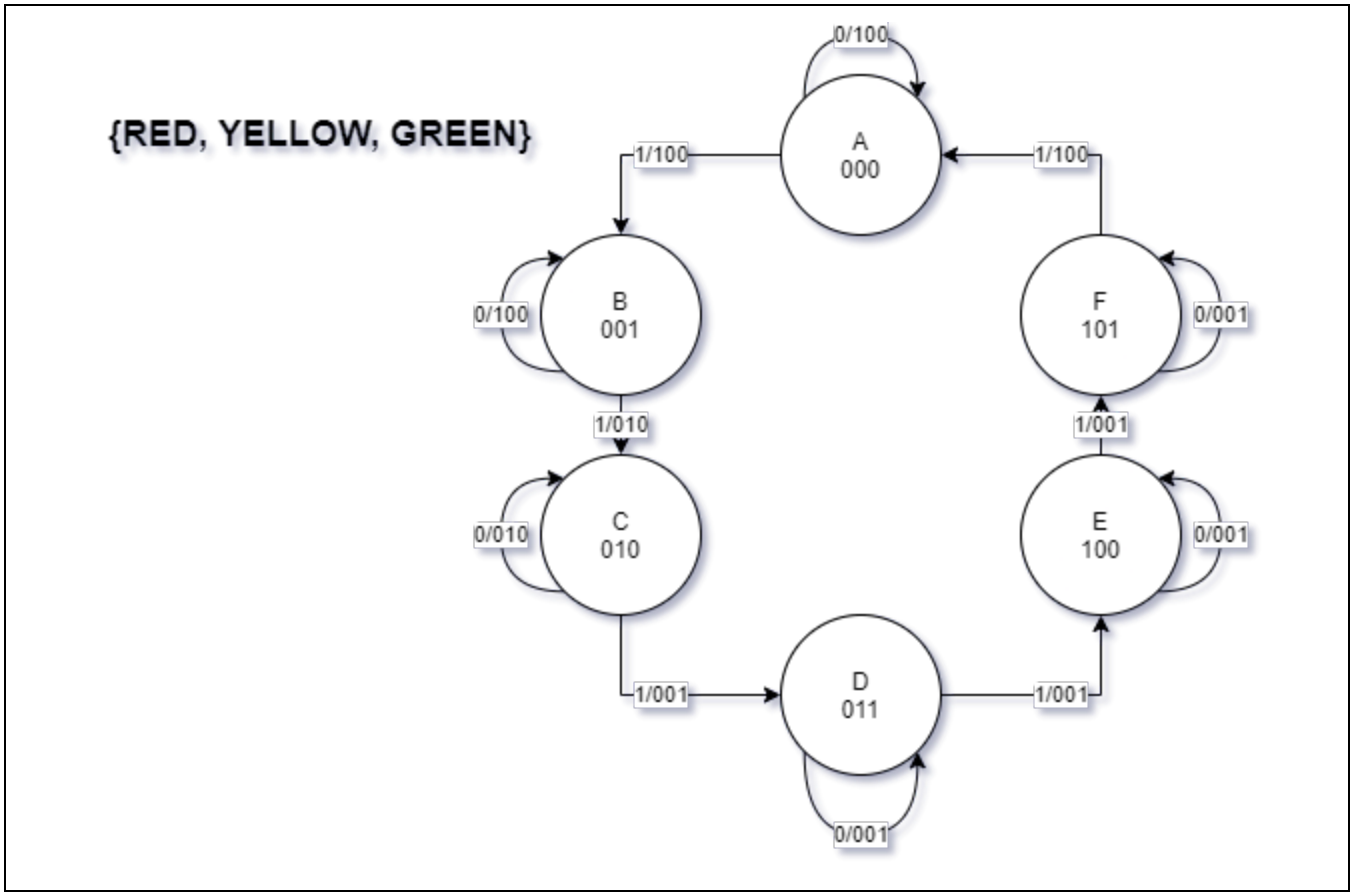


Fig 2b. Design Entry for Race Lights Controller

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

module RaceLightsController(CLOCK, nReset, START, RED, YELLOW, GREEN);

        input wire CLOCK, nReset, START;
        output reg RED, YELLOW, GREEN;

        parameter R = 3'b100; //RED
        parameter Y = 3'b010; //YELLOW
        parameter G = 3'b001; //GREEN

        parameter A = 3'b000;
        parameter B = 3'b001;
        parameter C = 3'b010;
        parameter D = 3'b011;
        parameter E = 3'b100;
        parameter F = 3'b101;

        reg [2:0] cstate, nstate;
        reg [1:0] counter;//00, 01, 10

        always @ (posedge CLOCK or negedge nReset) begin

                        if(!nReset)
                                cstate <= A;
                        else
                                cstate <= nstate;

        end

        always @ (START, cstate) begin
                if (START) begin
                        case(cstate)
                                3'b000 : nstate <= B;
                                3'b001 : nstate <= C;
                                3'b010 : nstate <= D;
                                3'b011 : nstate <= E;
                                3'b100 : nstate <= F;
                                default : nstate <= A;
                        endcase
                end
        end


        always @ (*) begin
                case(cstate)
                        3'b000 : {RED,YELLOW,GREEN} <= R;
                        3'b001 : {RED,YELLOW,GREEN} <= Y;
                        3'b010 : {RED,YELLOW,GREEN} <= G;
                        3'b011 : {RED,YELLOW,GREEN} <= G;
                        3'b100 : {RED,YELLOW,GREEN} <= G;
                        3'b101 : {RED,YELLOW,GREEN} <= G;
```

```
                    default: {RED,YELLOW,GREEN} <= R;
                endcase
        end
endmodule
```

Fig 2c. Flow Summary for Race Lights Controller

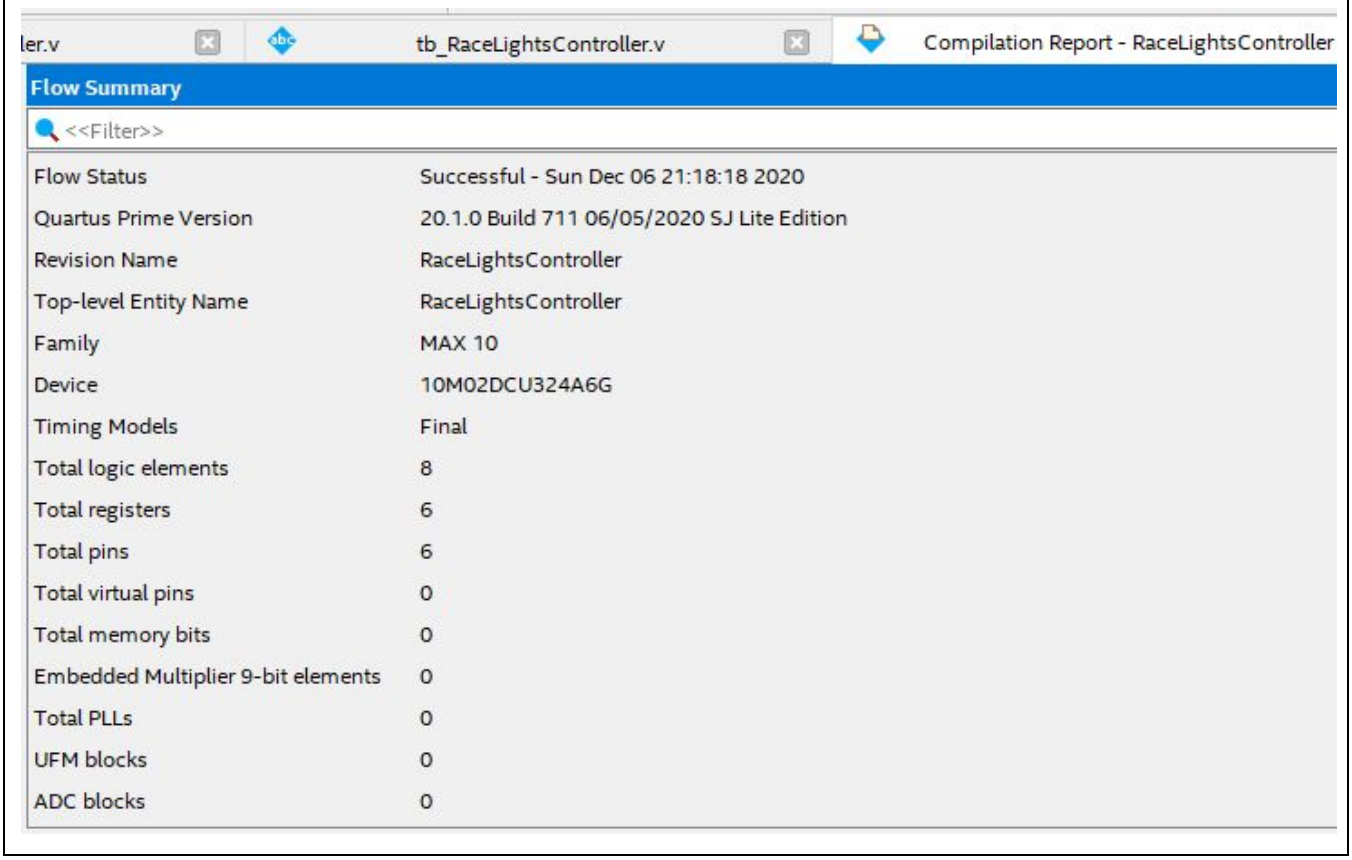
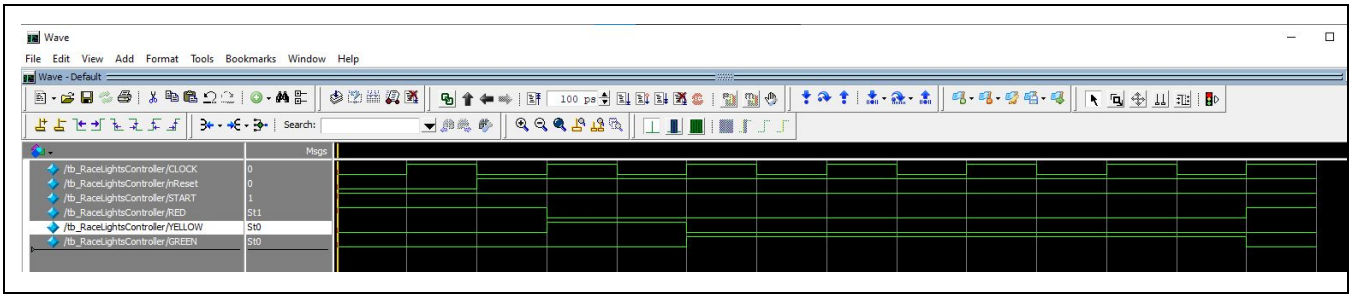


Fig 2d. Design Entry for Test Bench Race Lights Controller

RaceLightsController.v | tb_RaceLightsController.v

```
1   //GERMAN E FELISARTA III 16101002 CpE 3101L Grp 3
2
3   `timescale 1 ns / 1 ps
4   module tb_RaceLightsController();
5
6      reg CLOCK, nReset, START;
7      wire RED, YELLOW, GREEN;
8
9      RaceLightsController UUT (CLOCK, nReset, START, RED, YELLOW, GREEN);
10
11     initial CLOCK = 1'b0;
12
13     always #5 CLOCK = ~CLOCK;
14
15     initial begin
16        nReset = 1'b0; #10;
17        nReset = 1'b1;
18     end
19
20     initial begin
21
22        START = 1'b1; #70
23
24        $stop;
25     end
26
27  endmodule
```

Fig 2e. RTL Simulation for Race Lights Controller



Exercise 8C:

In this exercise, a Washing Machine Controller is to be made, a Moore FSM was used. First, the state diagram is made and the basis for input are MEDIUM, LARGE, DIRTY, WET, T1Done, and T2Done. The output is the concatenation of MEDIUM, LARGE, WASH, RINSE, DRY, T1Start, and T2Start variable outputs. The design entry was modeled after the State diagram. The timing diagram shows that it follows the expected output but only up to STATE C, the design cannot reach other states. The experiment is unsuccessful.

Fig 3a. State Diagram for Washing Machine

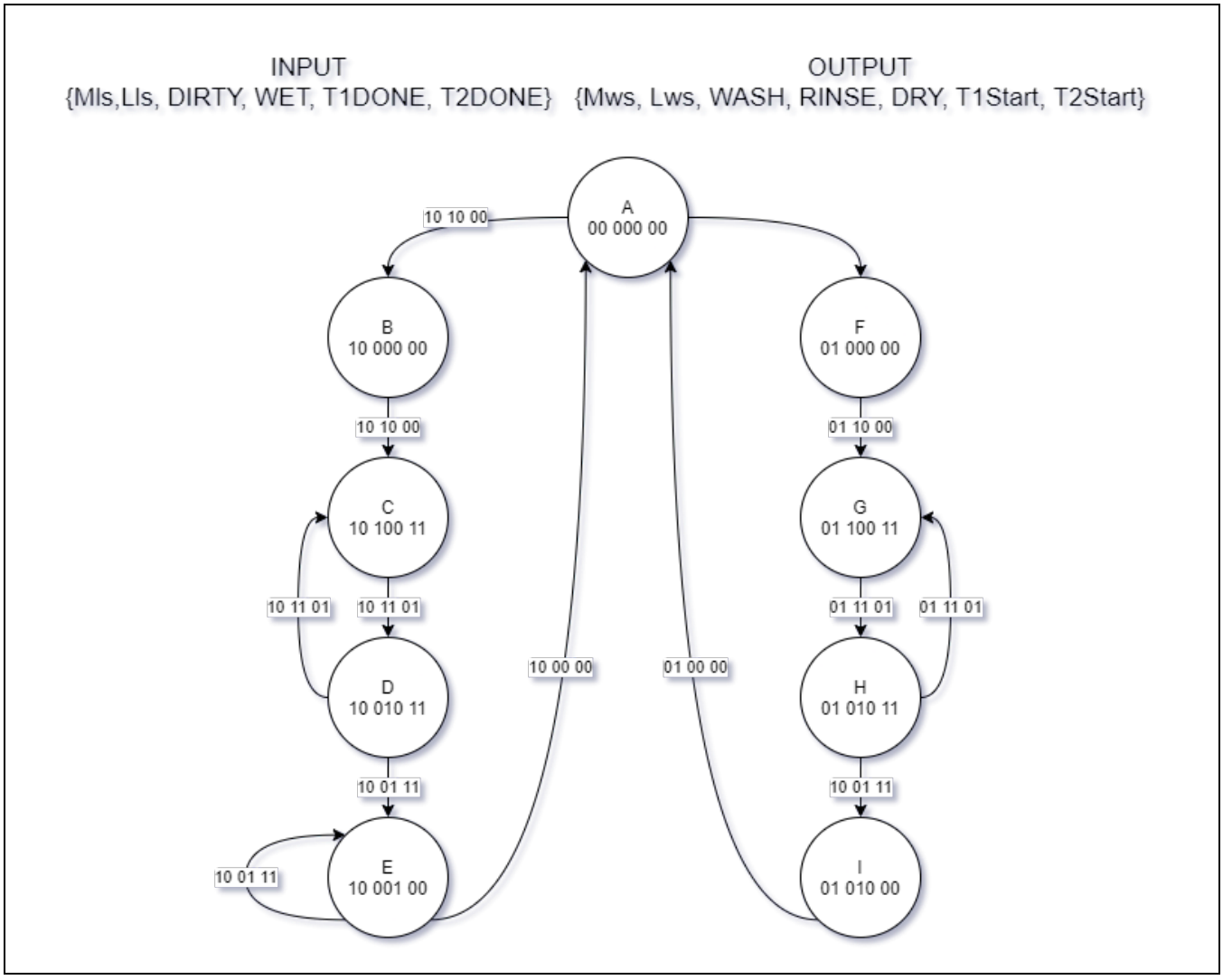


Fig 3b. Design Entry for Washing Machine

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

module WashingMachineController(CLOCK, nReset, START, Mls, Lls, DIRTY, WET, T1Done, T2Done, Mws,
Lws, WASH, RINSE, DRY, T1Start, T2Start);

        input wire CLOCK, nReset, START, Mls, Lls, DIRTY, WET, T1Done, T2Done;
        output reg Mws, Lws, WASH, RINSE, DRY, T1Start, T2Start;

        //INPUTS
        //CLOCK, nReset
        //START
        //{Mls,Lls, DIRTY, WET, T1DONE, T2DONE}

        //OUTPUTS
        //{Mws, Lws, WASH, RINSE, DRY, T1Start, T2Start}

        reg [6:0]cstate, nstate;
        reg [2:0]counter;

        parameter A = 7'b0000000;
        parameter B = 7'b1000000;
        parameter C = 7'b1010011;
        parameter D = 7'b1001011;
        parameter E = 7'b1000100;
        parameter F = 7'b0100000;
        parameter G = 7'b0110011;
        parameter H = 7'b0101011;
        parameter I = 7'b0101000;

        always @ (posedge CLOCK or negedge nReset)begin

                if(!nReset)
                        cstate <= A;
                else
                        cstate <= nstate;


        end

        always @ (*) begin

        if (START) begin

        case(cstate)
                A : nstate <= ({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b101000) ? B : F;
                B : nstate <= ({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b101000) ? C : B;
                C : nstate <= ({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b101101) ? D : C;

                D : begin
                        if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b100111) nstate <= E;
                        else if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b101101) nstate <= C;
                   end
```

```
                E : begin
                        if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b100000) nstate <= A;
                        else if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b100111) nstate <= E;
                 end

                F : nstate <= ({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b011000) ? G : F;
                G : nstate <= ({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b011101) ? H : G;
                H : begin
                        if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b010111) nstate <= I;
                        else if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b011101) nstate <= G;
                end

                I : begin
                        if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b010111) nstate <= I;
                        else if({Mls,Lls, DIRTY, WET, T1Done, T2Done} == 6'b010000) nstate <= A;
                 end

        endcase
end
end

        always @ (*)
                {Mws, Lws, WASH, RINSE, DRY, T1Start, T2Start} <= cstate;


endmodule
```

Fig 3c. Flow Summary for Washing Machine

troller.v | tb_WashingMachineController.v | Compilation Report - Wa

Flow Summary

<<Filter>>

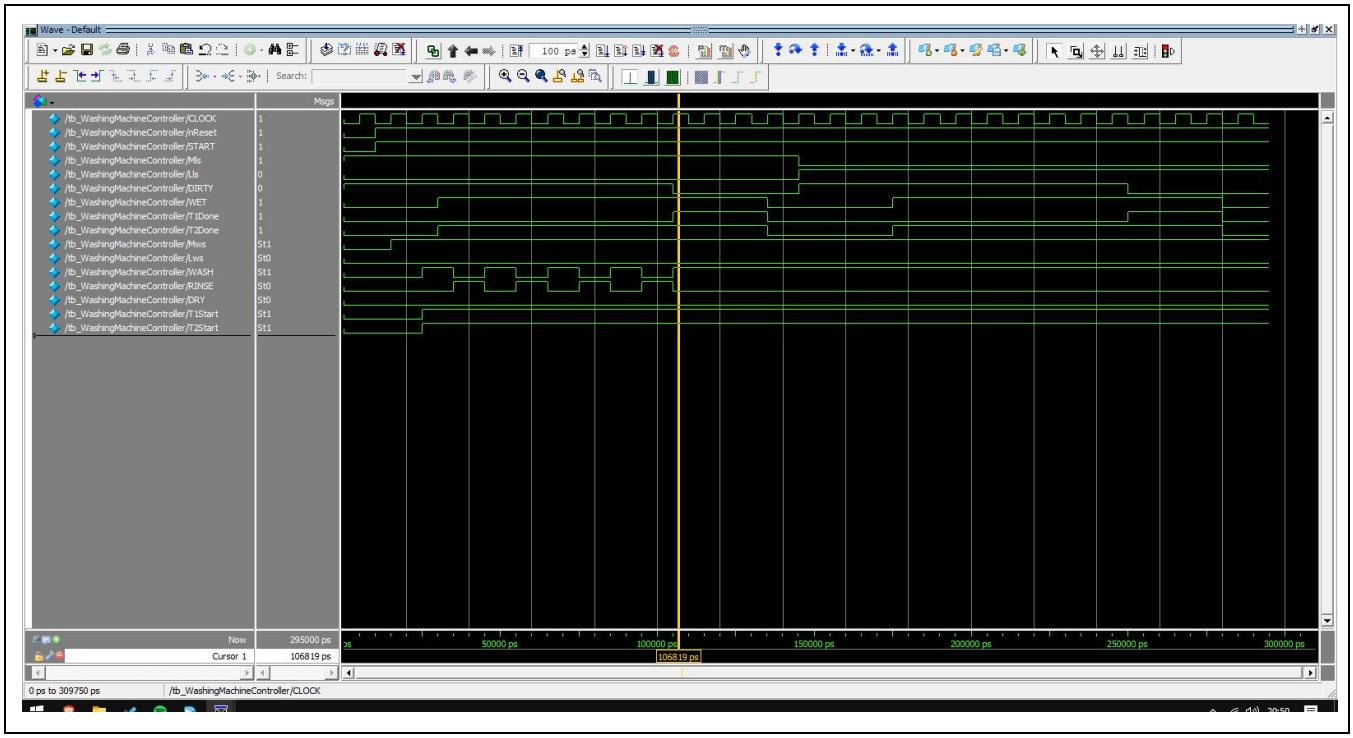
Flow Status	Successful - Sun Dec 06 20:47:10 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	WashingMachineController
Top-level Entity Name	WashingMachineController
Family	MAX 10
Device	10M02DCU324A6G
Timing Models	Final
Total logic elements	43
Total registers	9
Total pins	16
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Fig 3d. Design Entry for Test Bench Washing Machine

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

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

        reg CLOCK, nReset, START, Mls, Lls, DIRTY, WET, T1Done, T2Done;
        wire Mws, Lws, WASH, RINSE, DRY, T1Start, T2Start;

        WashingMachineController UUT (CLOCK, nReset, START, Mls, Lls, DIRTY, WET, T1Done, T2Done,
Mws, Lws, WASH, RINSE, DRY, T1Start, T2Start);

        initial CLOCK = 1'b0;

        always #5 CLOCK = ~CLOCK;

        initial begin
```

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

	initial begin
		START = 1'b0; #10
		START = 1'b1;
	end

	initial begin
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101000;#15 //@A going B
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101000;#15 //@B going C
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101101;#15	//@C going D	#1
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101101;#15	//@D going C
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101101;#15	//@C going D	#2
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101101;#15	//@D going C
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b101101;#15	//@C going D	#3
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b100111;#15	//@D going E
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b100111;#15	//@E first loop to 20Mins
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b100000;#10//@E going A

		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011000;#15 //@A going F
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011000;#15 //@F going G
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011101;#15	//@G going H	#1
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011101;#15	//@H going G
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011101;#15	//@G going H	#2
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011101;#15	//@H going G
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b011101;#15	//@G going H	#3
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b010111;#15	//@H going I
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b010111;#15	//@I first loop to 20Mins
		{Mls,Lls, DIRTY, WET, T1Done, T2Done} = 6'b010000;#15	//@I going A

		$stop;
	end

endmodule
```

Fig 3e. RTL Simulation for Washing Machine


Wave - Default
Msgs
/tb_WashingMachineController/CLOCK
/tb_WashingMachineController/nReset
/tb_WashingMachineController/START
/tb_WashingMachineController/Mis
/tb_WashingMachineController/Lis
/tb_WashingMachineController/DIRTY
/tb_WashingMachineController/WET
/tb_WashingMachineController/T1Done
/tb_WashingMachineController/T2Done
/tb_WashingMachineController/Mws
/tb_WashingMachineController/Lws
/tb_WashingMachineController/WASH
/tb_WashingMachineController/RINSE
/tb_WashingMachineController/DRY
/tb_WashingMachineController/T1Start
/tb_WashingMachineController/T2Start
Now
295000 ps
Cursor 1
106819 ps
50000 ps
100000 ps
150000 ps
200000 ps
250000 ps
300000 ps
0 ps to 309750 ps
/tb_WashingMachineController/CLOCK