Connor Cousineau

Ece 3700

Lab 4

In this lab the objective was to design a stopwatch. The over all design needed four components, an incrementor, a DFF, a BCD to 7 seg and a clock divider. At first the task was a little over whelming. To lessen the burden the total design was broken down into two phases. The first phase was completed in week one. During this week we designed everything but the clock divider. The incrementor was fairly simple. Its an always block that checks if the value contained is 9 otherwise add one and output said value.

The next block was the DFF. This item required an always block to check the clock and only transfer on the posedge of the clock. Next was to check if the reset button was active. If neither of these are true then copy and store the data. If none of the conditions are met then just keep the value and change nothing. During this week we are not using the FPGA clock input and are only using push buttons. The input of these buttons is inverted and needed to be inverted in the checks. Without the inversion it didn't work properly.

The last item for this week was the seven seg converter. This code was provided to us already. We added compatibility up to 15 which turned out to be pointless in the end. The last step for this week was to combine all three. The data from the incrementor is fed into the DFF which then feeds the incrementor and seven seg. As long as reset is not high the circuit will count ever time the clock is high and will update the seven seg. If reset is high it will only display zero. Once tested and compiled It was put on the board and works. See the video.

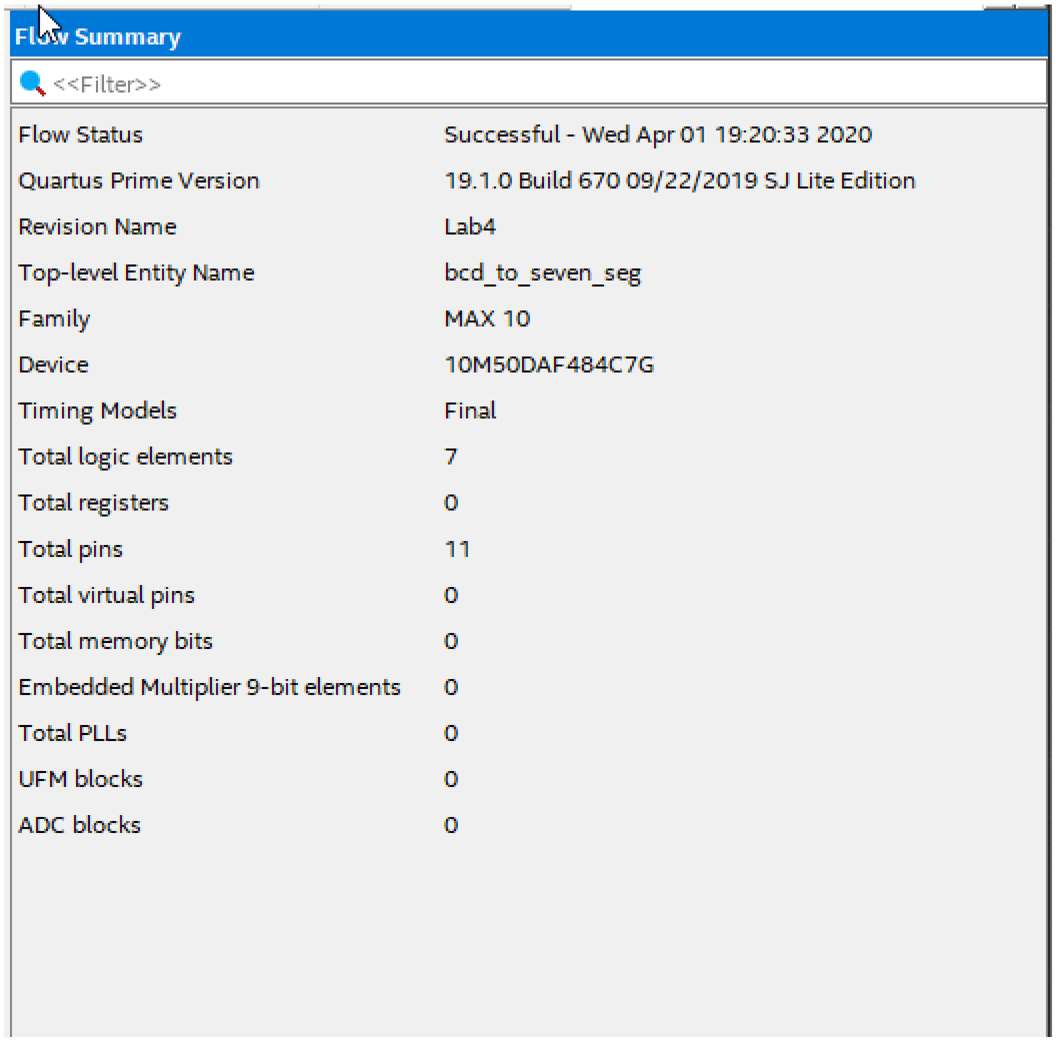
Next came phase two. In this week the objective was to create the clock divider. The purpose of this item is to take the 50 MHz clock and tone it down to 1 Hz to count in seconds. To do this we divided to clocks 50M/1 = 50M ticks / 2 = 25 M ticks. Twenty-five million ticks are the rate at which our counter will count. The block has an internal counter that counts until twenty-five million and then resets. This action produces the 1 Hz clock that we are looking for. This is all done in an always block checking for the posedge of the clock and the posedge of the reset signal. This signal is fed into the DFF for part 2.

In this phase we also needed to modify the DFF to accept start, stop and resume signals. This introduced the complication that the DFF needs to know these values at all times to function. To solve this we introduced registers for all of them. Each signal will set the other values respectively. This time around start and stop are buttons and reset is a switch. If the switch is high the value will remain at zero. Otherwise when started it will loops zero to nine until either reset is set to high or stop is pushed. If at any point stop is set to high it will stop counting. Testing was done by putting it on the board and using the previous DFF test bench modified for the new buttons. Once all the components were connected. It was compiled and put on the board and ran successfully.

Some design considerations were that the buttons were inverted and placement of the start_reg etc allowed the code to be more readable hopefully. Other than that the lab was done to the specifications of the lab worksheet.

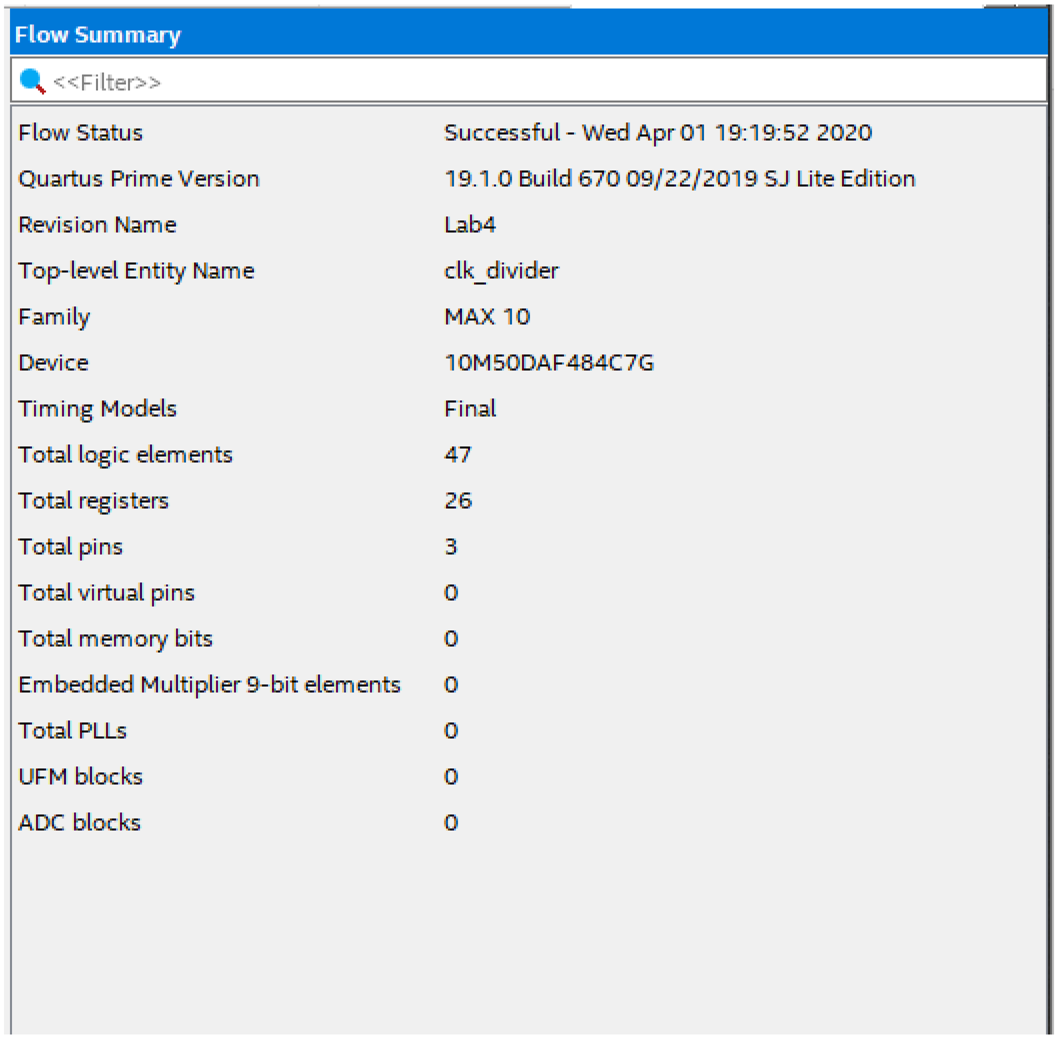
All the rest of the Labs pictures, the RTL Netlist, Test benches, waveforms, Verilog code, synthesis reports can be found below. Unfortunately its not in any particular order, except that the code is at the bottom.

7 seg rpt



Flow Status	Successful - Wed Apr 01 19:20:33 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	bcd_to_seven_seg
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	7
Total registers	0
Total pins	11
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Clk Divider rpt

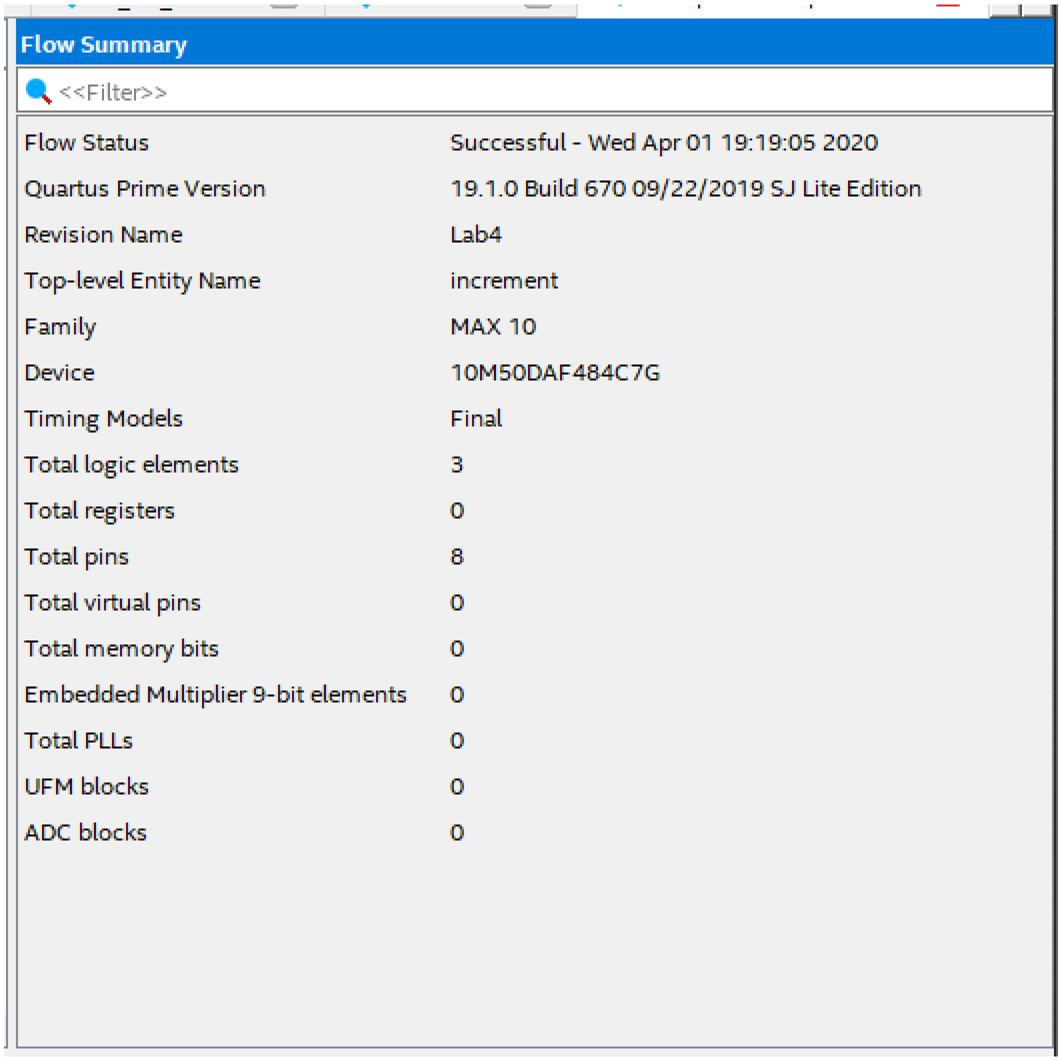


Flow Summary

<<Filter>>

Flow Status	Successful - Wed Apr 01 19:19:52 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	clk_divider
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	47
Total registers	26
Total pins	3
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Incrementer rpt

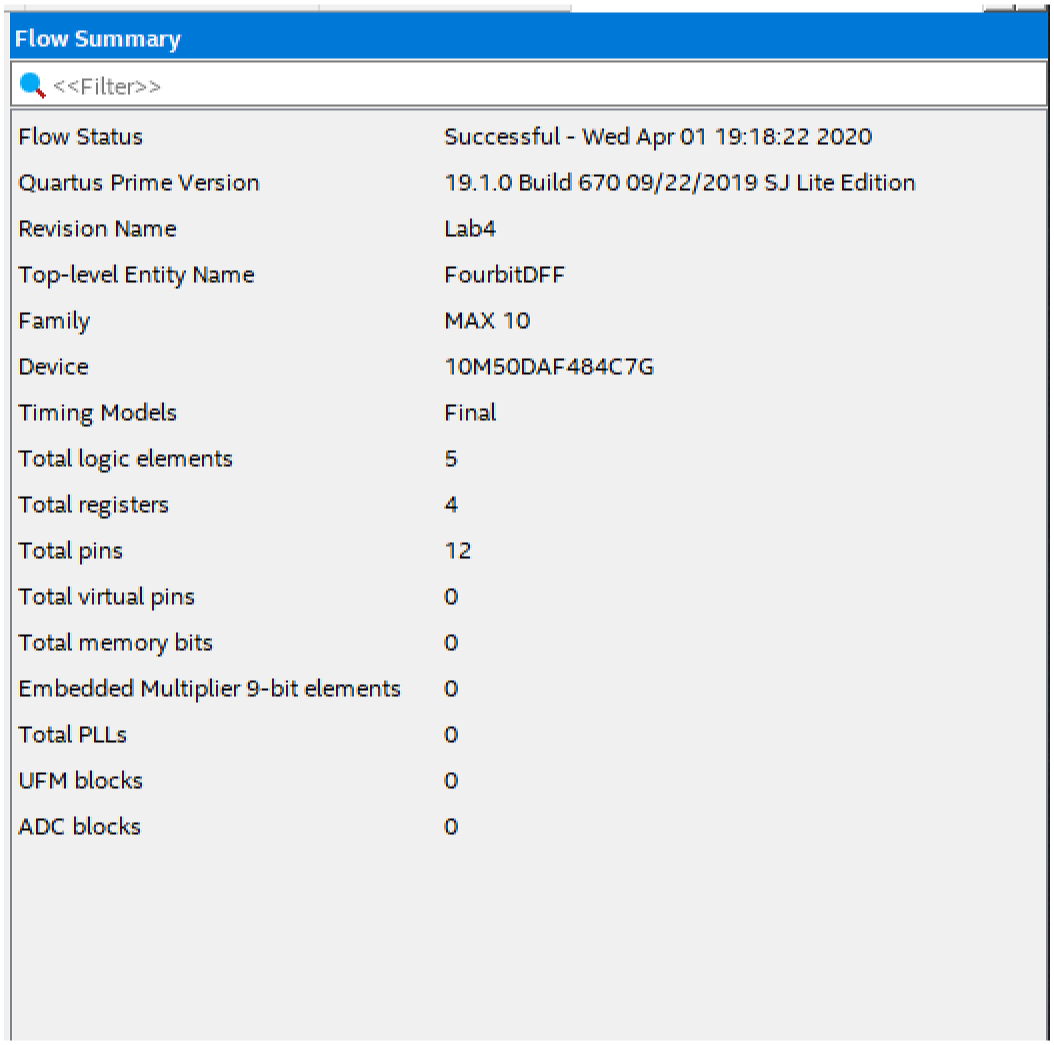


Flow Summary

<<Filter>>

Flow Status	Successful - Wed Apr 01 19:19:05 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	increment
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	3
Total registers	0
Total pins	8
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Final DFF rpt

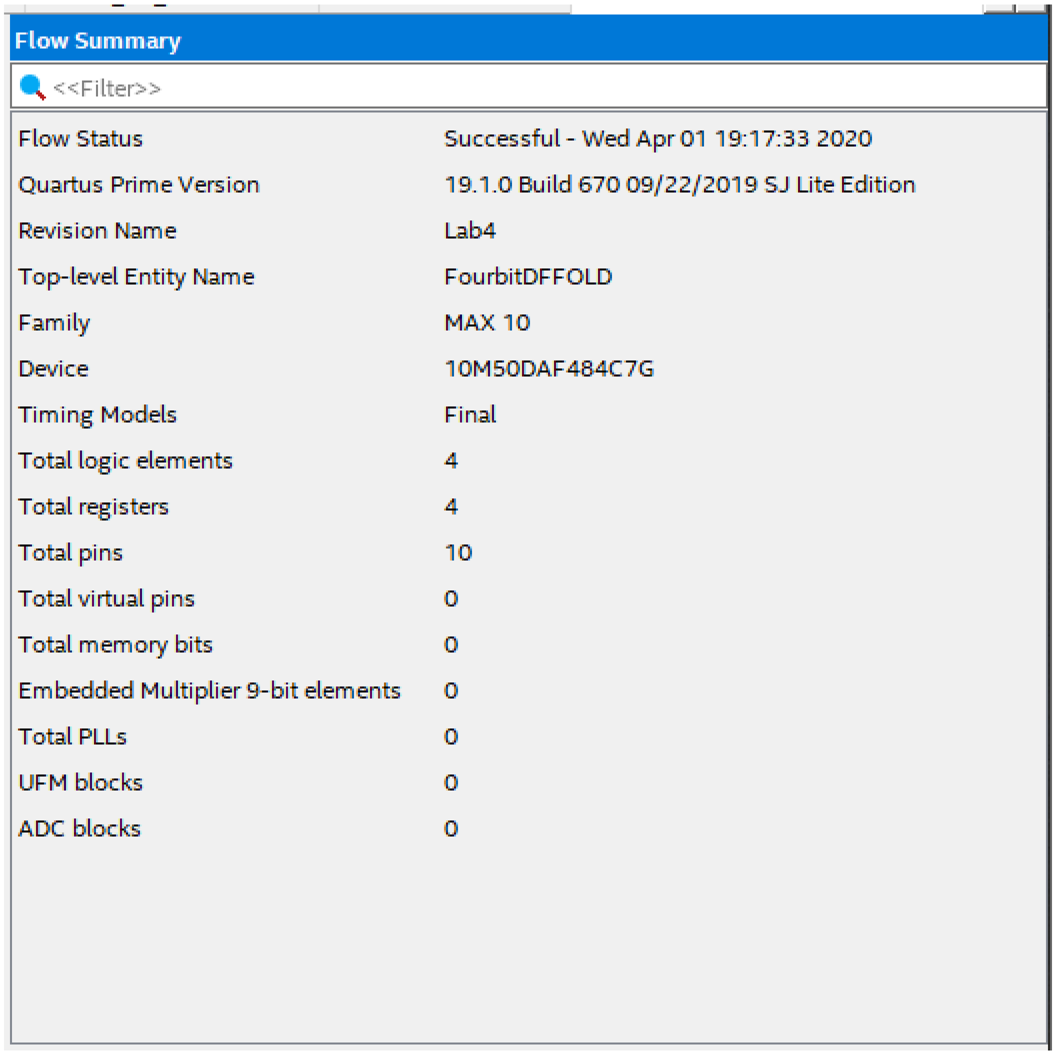


Flow Summary

<<Filter>>

Flow Status	Successful - Wed Apr 01 19:18:22 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	FourbitDFF
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	5
Total registers	4
Total pins	12
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

PRE FPGA clk DFF

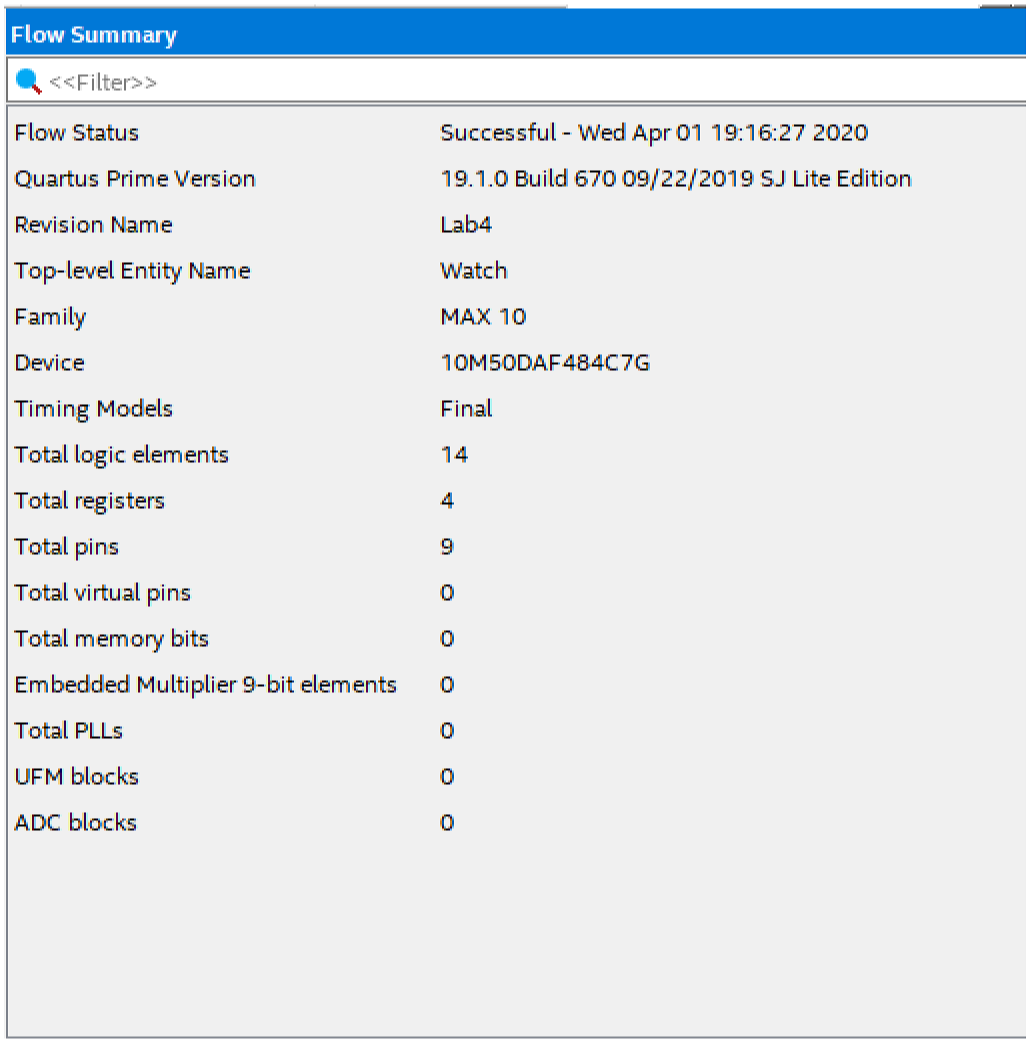


Flow Summary

<<Filter>>

Flow Status	Successful - Wed Apr 01 19:17:33 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	FourbitDFFOLD
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	4
Total registers	4
Total pins	10
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Pre FPGA clk watch

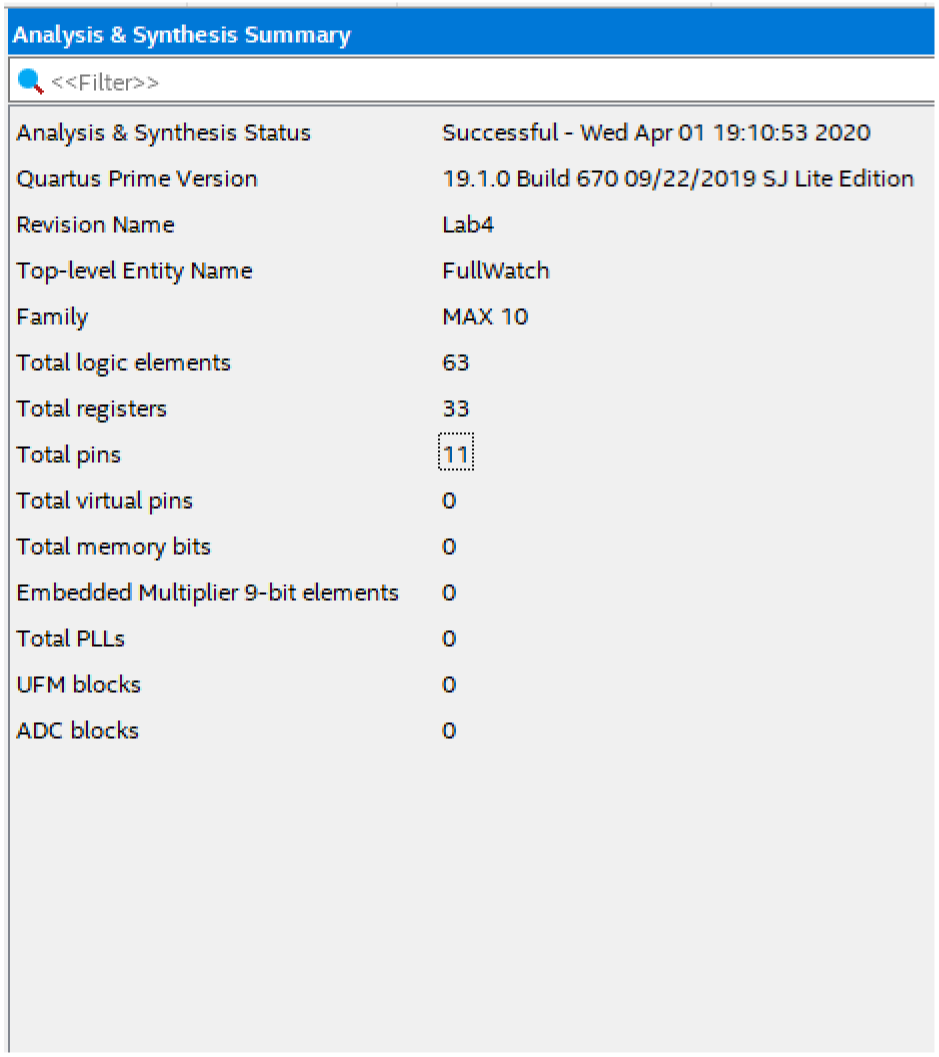


Flow Summary

<<Filter>>

Flow Status	Successful - Wed Apr 01 19:16:27 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	Watch
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	14
Total registers	4
Total pins	9
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Full stopwatch Sythesis rpt

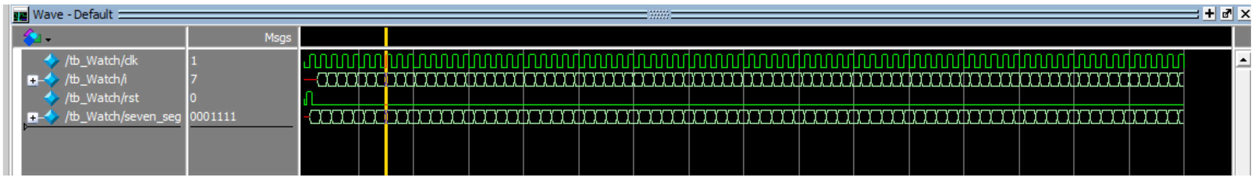


Analysis & Synthesis Summary

<<Filter>>

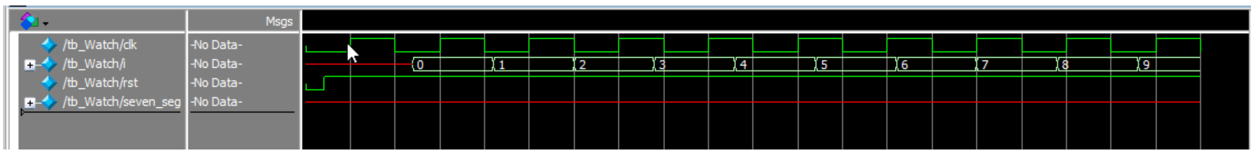
Analysis & Synthesis Status	Successful - Wed Apr 01 19:10:53 2020
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	Lab4
Top-level Entity Name	FullWatch
Family	MAX 10
Total logic elements	63
Total registers	33
Total pins	11
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

No clk divider watch reset low



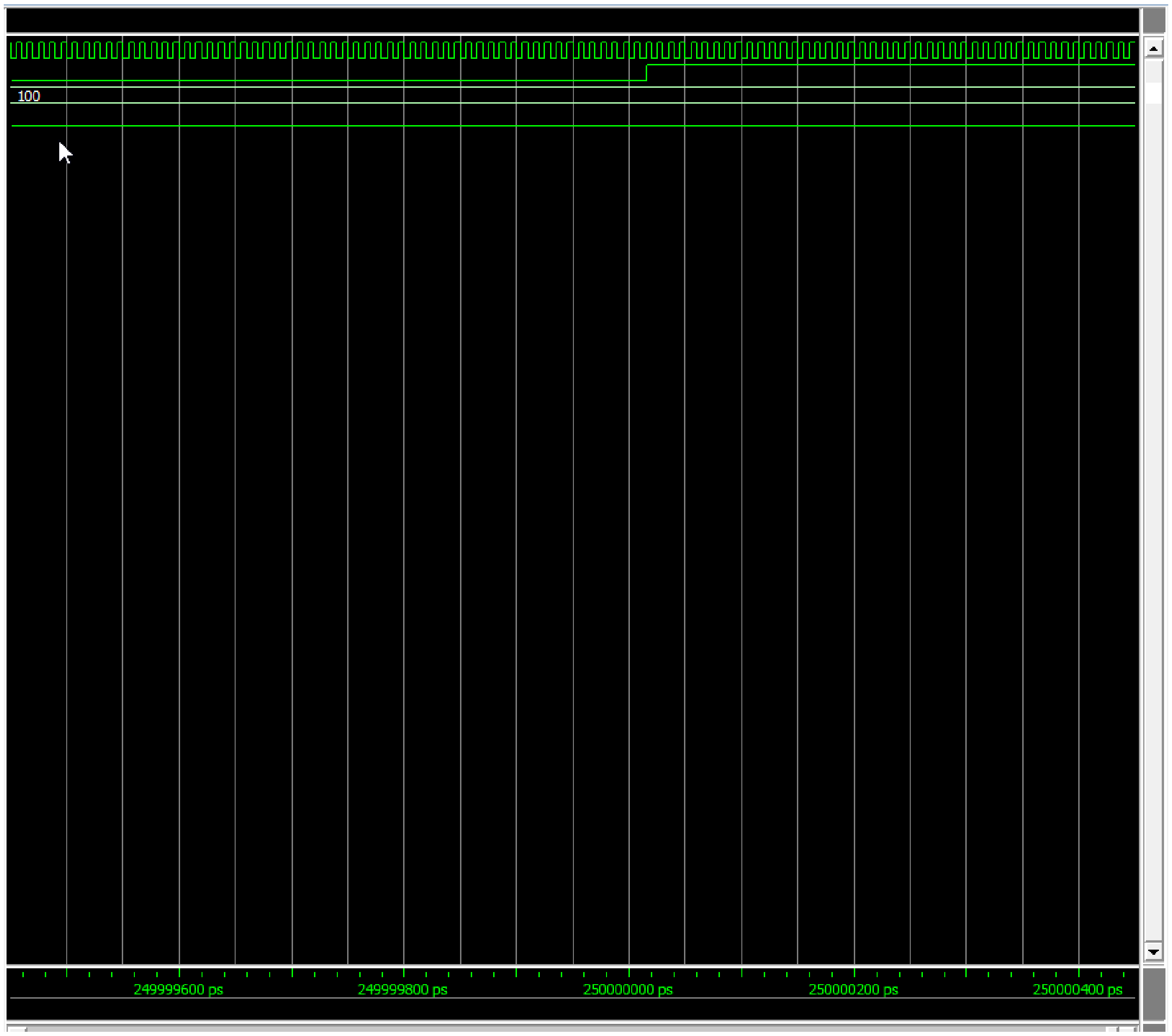
```
/tb_Watch/clk          1
/tb_Watch/i            7
/tb_Watch/rst          0
/tb_Watch/seven_seg    0001111
```

No clk divider watch rst high

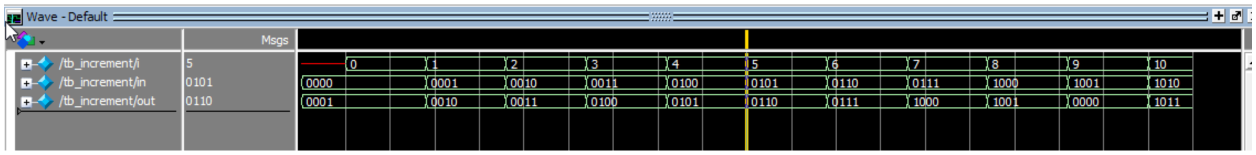


```
/tb_Watch/clk          -No Data-
/tb_Watch/i            -No Data-
/tb_Watch/rst          -No Data-
/tb_Watch/seven_seg    -No Data-
```

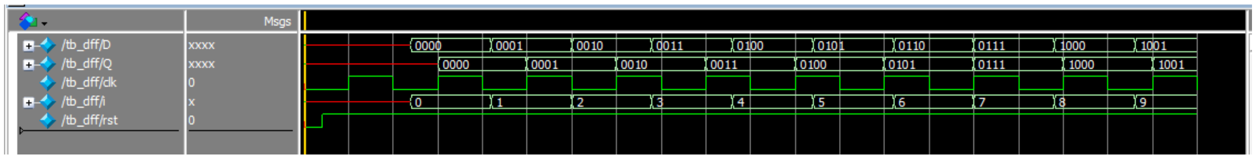
Clk Divider waveform



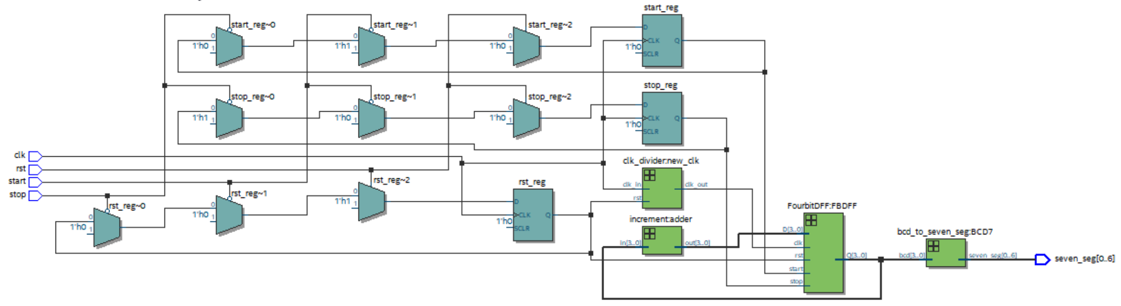
Incrementor wave



DFF PRE start stop etc



RTL NETLIST


start_reg~0
start_reg~1
start_reg~2
start_reg
stop_reg~0
stop_reg~1
stop_reg~2
stop_reg
clk
rst
start
stop
rst_reg~0
rst_reg~1
rst_reg~2
rst_reg
clk_divider:new_clk
increment:adder
FourbitDFF:FBDFF
bcd_to_seven_seg:BCD7
seven_seg[0..6]

FullWatch.V

```
module FullWatch(clk,start,stop,rst,seven_seg);

input clk,rst,start,stop;

output[0:6]seven_seg;

wire[3:0]Q,D;

wire clk_out;

reg rst_reg, start_reg, stop_reg;

clk_divider new_clk(clk,clk_out,rst_reg);

increment adder(Q,D);

FourbitDFF FBDFF(D,clk_out,rst_reg,stop_reg,start_reg,Q);

bcd_to_seven_seg BCD7(Q,seven_seg);

always@(posedge clk)begin

if(rst== 1)begin

rst_reg<=1;start_reg <= 0; stop_reg <= 0;

end

else if(start == 0)begin

rst_reg<= 0; start_reg <=1;stop_reg <= 0;

end

else if(stop == 0)begin

rst_reg<= 0; start_reg <=0;stop_reg <= 1;

end

else begin rst_reg <= rst_reg; start_reg <= start_reg; stop_reg <= stop_reg;

end

end

endmodule
```

tb_clk_divider.v

```
`timescale 1ps / 1ps

module tb_clk_divider;

wire clk_out;

reg clk_in,rst;

integer i;

clk_divider uut (

.clk_in(clk_in),

.clk_out(clk_out),

.rst(rst)

);

initial begin

//rst = 0; clk=0;#5;

//rst = 1; clk=1;#5;

//rst = 0; clk=0;#5;

clk_in = 0;

rst=0; #2;

rst=1; #5;

rst=0; #5;

for(i = 0; i < 100; i = i+1)begin

		#9;

		$display("%d, %d => %b",clk_in, rst, clk_out);

		end

end


always

begin

#5;

clk_in <= ~clk_in;
```

```
end


endmodule
```

Clk Divider.v

```
module clk_divider(clk_in,clk_out,rst);

input clk_in,rst;

output reg clk_out;

reg[24:0] count;

always@(posedge clk_in, posedge rst) begin

if(rst == 1)begin

        count <= 25'd0;

        clk_out<=0;

end

        else if(count == 25_000000)begin

                count <= 25'd0;

                clk_out <= ~clk_out;

                end

                else begin

                        count <= count + 1'b1;

                        clk_out <= clk_out;

                        end

                end

endmodule
```

Tb_increment.v

```
`timescale 1ps / 1ps
```

```
module tb_increment;
reg[3:0]in;
wire[3:0]out;
integer i;
increment uut (
.in(in),
.out(out)
);
initial begin
in = 0; #5;
for(i = 0; i < 100; i = i+1)begin
        in = i;
        #9;
        $display("%d => %d",in,out);
        end
end
endmodule
```

---

tb_wathc.v

```
`timescale 1ps / 1ps
module tb_Watch;
```

```
wire[0:6]seven_seg;

reg clk,rst;

integer i;

Watch uut (

.clk(clk),

.rst(rst),

.seven_seg(seven_seg)

);

initial begin

//rst = 0; clk=0;#5;

//rst = 1; clk=1;#5;

//rst = 0; clk=0;#5;

clk =0;

rst=0; #2;

rst=1; #5;

rst=1; #5;

for(i = 0; i < 100; i = i+1)begin

        #9;

        $display("%d, %d => %b",clk, rst, ~seven_seg);

        end

end

always

begin

#5;

clk <= ~clk;

end

endmodule
```

Watch.V

```
module Watch(clk,rst,seven_seg);
input clk,rst;
output[0:6]seven_seg;
wire[3:0]Q,D;
increment adder(Q,D);
FourbitDFF FBDFF(D,clk,rst,Q);
bcd_to_seven_seg BCD7(Q,seven_seg);
endmodule
```

tb_dff.v

```
`timescale 1ps / 1ps
module tb_dff;
reg [3:0]D;
reg clk,rst,stop,start;
wire [3:0]Q;
integer i;
FourbitDFF uut (
.D(D),
.clk(clk),
.rst(rst),
.stop(stop),
.start(start),
.Q(Q)
);
```

```
initial begin
//rst = 0; clk=0;#5;
//rst = 1; clk=1;#5;
//rst = 0; clk=0;#5;
clk =0;
stop = 1;
rst=0; #2;
rst=1; #5;
rst=1; #5;
for(i = 0; i < 16; i = i+1)begin
        D = i;
        #9;
        $display("%d => %d clk: %d rst: %d stop: %d", D, Q, clk, rst, stop);
        end
end
always
begin
#5;
clk <= ~clk;
end
endmodule
```

increment.v

```
module increment(in,out);

input [3:0]in;

output reg [3:0]out;

always @(in)

begin

            if(in == 9)

                        out = 0;

            else

                        out = in + 1;

end

endmodule
```

---

bcd_seven_seg.v

```
module bcd_to_seven_seg(bcd,seven_seg);

input [3:0] bcd;

output reg [0:6] seven_seg;

always @*

 begin

  case (bcd)

   0 : begin seven_seg  = ~7'b1111110; end

   1 : begin seven_seg  = ~7'b0110000; end

   2 : begin seven_seg  = ~7'b1101101; end

   3 : begin seven_seg  = ~7'b1111001; end

   4 : begin seven_seg  = ~7'b0110011; end

   5 : begin seven_seg  = ~7'b1011011; end

   6 : begin seven_seg  = ~7'b1011111; end

   7 : begin seven_seg  = ~7'b1110000; end

            8 : begin seven_seg  = ~7'b1111111; end
  9 : begin seven_seg  = ~7'b1110011; end
            10 : begin seven_seg = ~7'b1110111; end
            11: begin seven_seg  = ~7'b0011111; end
            12 : begin seven_seg = ~7'b1001110; end
            13 : begin seven_seg = ~7'b0111101; end
            14: begin seven_seg  = ~7'b1001111; end
            15 : begin seven_seg = ~7'b1000111; end
  default : begin seven_seg = ~7'b0000000; end
 endcase
 end
endmodule
```

FourBitDff.V

```
module bcd_to_seven_seg(bcd,seven_seg);

input [3:0] bcd;
output reg [0:6] seven_seg;
always @*
 begin
 case (bcd)
  0 : begin seven_seg  = ~7'b1111110; end
  1 : begin seven_seg  = ~7'b0110000; end
  2 : begin seven_seg  = ~7'b1101101; end
  3 : begin seven_seg  = ~7'b1111001; end
  4 : begin seven_seg  = ~7'b0110011; end
  5 : begin seven_seg  = ~7'b1011011; end
  6 : begin seven_seg  = ~7'b1011111; end
  7 : begin seven_seg  = ~7'b1110000; end
            8 : begin seven_seg  = ~7'b1111111; end
  9 : begin seven_seg  = ~7'b1110011; end
            10 : begin seven_seg = ~7'b1110111; end
            11: begin seven_seg  = ~7'b0011111; end
            12 : begin seven_seg = ~7'b1001110; end
            13 : begin seven_seg = ~7'b0111101; end
            14: begin seven_seg  = ~7'b1001111; end
            15 : begin seven_seg = ~7'b1000111; end
  default : begin seven_seg = ~7'b0000000; end
 endcase
end
endmodule
```

Pre FPGA clk DFF

```
module FourbitDFFOLD(D,clk,rst,Q);
input[3:0]D;
input clk,rst;
output reg [3:0]Q;

always@(posedge clk)
begin
```

```
        if(rst == 1)
        begin
        Q <= 4'b0000;
        end
        else
        begin
        Q <= D;
        end
end
endmodule
```