Design a Pattern Recognizer

Argyrios Kokkinis
a.kokkinis@student.utwente.nl
s2252406

Design of Digital Systems University of Twente

February 2019

Contents

1 Introduction **3**

2 Behavioral Description **4**

3 Structural Description **5**

4 Automated Testing **7**

5 Synthesis **9**

6 Post Simulation Analysis **16**

1 Introduction

The aim of the assignment Pattern Recognizer was to design a pattern recognizer using VHDL , performing simulations on ModelSim , synthesizing it on Quartus Prime and then programming it on a Cyclone FPGA.

The design of the pattern recognizer had the following specifications:

- Synchronous system with asynchronous reset.
- The active edge of the clock is only the rising edge.
- The pattern is 11100 (from left to right).
- It counts the number of occurrences of this pattern.
- It uses two 7 segment displays.
- After more than 99 occurrences the displays show '- - '.
- After reset the counter is zero again if a part of the pattern was already recognized it is ignored.
- Before the first reset the circuit has undefined behavior.
- The counter is updated after the last bit is detected and before the next rising edge of the clock.

2 Behavioral Description

The VHDL files used for that part are :

1. `pattern_recogniser.vhd`[1]
2. `pattern_recogniser_tb.vhd`[2]

At first the specifications of the above pattern recognizer were translated into one VHDL file creating a testable informal description[1] of the design. Then a test bench file was created[2] in order to validate the pattern recognizers functionality.

For simulation purposes the test bench generated a clock with a period of 20 ns and 110 data pulses each one having a period of 160 ns (80 ns high and 80 ns low).

After that it performed a reset and then it generated other 50 data pulses with a period of 190 ns (110 ns high and 80 ns low).

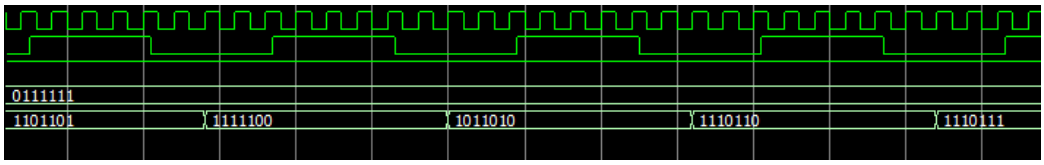

Figure 1: Testing the behavioral

1. 1st waveform : clock
2. 2nd waveform : data pulses
3. 3rd : reset signal
4. 4th : display no1
5. 5th : display no2

3 Structural Description

The VHDL files used for that part are :

1. `pattern_recogniser_str.vhd`[1]
2. `pattern_recogniser_str_tb.vhd`[2]
3. `list_det.vhd`[3]
4. `counter.vhd`[4]
5. `dsp_driver1.vhd`[5]
6. `dsp_driver2.vhd`[6]

The pattern recognition consists of the following entities :

`list_det`[3]: identifies the pattern and activates the output signal match to the counter.

A shift register Q was used to store the input. When the pattern 1110 is stored in the register then the signal match is activated. The value of this signal is changed concurrently in respect to the data.

`counter`[4]: receives an input signal match from the `list_det`[3] and increments the value of the bcd code. A lookup table has been used in order to translate the counter values into bcd.

`dsp_driver1.vhd`[5]:receives an input bcd signal from the `counter`[4] unit which corresponds to the 7 segment value for the first display.

`dsp_driver2.vhd`[6]:receives an input bcd signal from the `counter`[4] unit which corresponds to the 7 segment value for the second display.

```
architecture behavior of list_det is

signal matchValue : std_ulogic := '0';
begin
 process(clk,reset)

 variable Q : std_logic_vector(4 downto 0);

 begin
-- reset initialization
  if reset='0' then
     matchValue <= '0';
  elsif (rising_edge(clk) and (reset='1')) then
     Q(4 downto 1) := Q(3 downto 0);
     Q(0) := data;
  if(Q(3 downto 0) = "1110") then
      matchValue <= '1';
  else
      matchValue <='0';
  end if ;
end if;
end process;

process(data,matchValue)
 begin
 match <= matchValue and (not data);
 end process;
end behavior;
```

Figure 2: list det

For both the display drivers a function has been used in order to convert the bcd value into decimal and then a lookup table corresponds the decimal value into a 7-segment code.

In order to validate the functionality of the pattern recognizer a testbench file `pattern_recogniser_str_tb`[2] was created . The testbench file generates a clock signal with a period of 20 ns and 110 data pulses each one with a period of 100 ns (60 ns high and 40 ns low).Then it performs a reset and after that it generates 50 data pulses each having a period of 200 ns (110 ns high and 90 ns low).

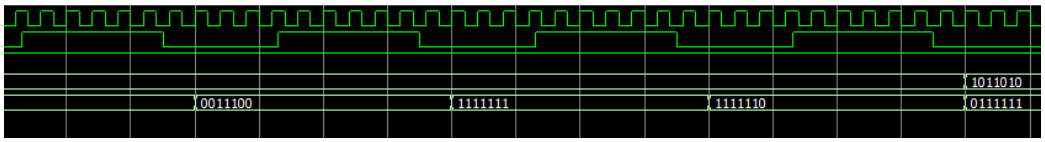

Figure 3: Testing the above description

4 Automated Testing

The VHDL files used for that part are :

1. `pattern_recogniser_str.vhd`[1]
2. `testing`[2]
3. `pattern_recogniser.vhd`[3]
4. `env_golden_unit.vhd`[4]
5. `compare.vhd`[5]

In order to compare the output of the structural pattern recognizer and the behavioral one a golden unit environment was created. The entity `env_golden_unit`[4] has no inputs or outputs and is composed of the following components :

1. `Testing`[2]: generates the clock, reset and data.
2. `pattern_recogniser`[3]:the behavioral description of the pattern recognizer.
3. `pattern_recogniser_str`[1] : the structural description of the pattern recognizer.
4. `compare`[5] : compares the output between the structural and the behavioral pattern recognizer.

In case that one or both of the outputs are different then a warning message DIFFERENCE is printed to the console.

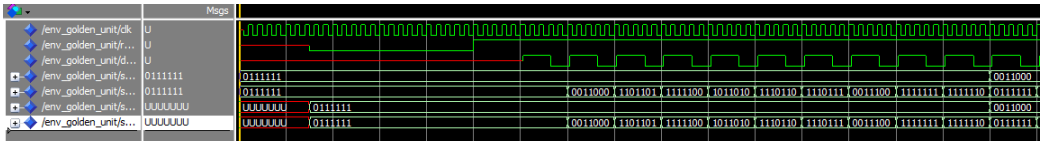

Figure 4: Comparison between the two pattern recognizers

1. 1st clock signal.
2. reset signal.
3. data signal.
4. segment 1 output from the second pattern recognizer (structural).
5. segment 2 output from the second pattern recognizer (structural).
6. segment 1 output from the first pattern recognizer (behavioral).
7. segment 2 output from the first pattern recognizer (behavioral).

For the above simulation the testing file testing.vhd[2] generated a clock of 20 ns period . After 150 ns it performed a reset and then it generated 110 data pulses of 100 ns period (60 ns high and 40 ns low). After that it performed another reset and then again it generated 50 data pulses of 190 ns period (110 ns high and 80 ns low).

5 Synthesis

Having validated the correctness of the design the next step was to synthesize it on Quartus.

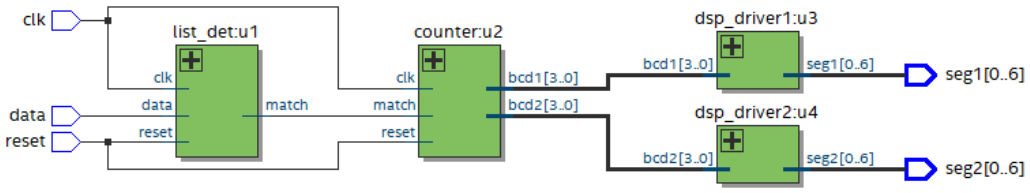

Figure 5: RTL view

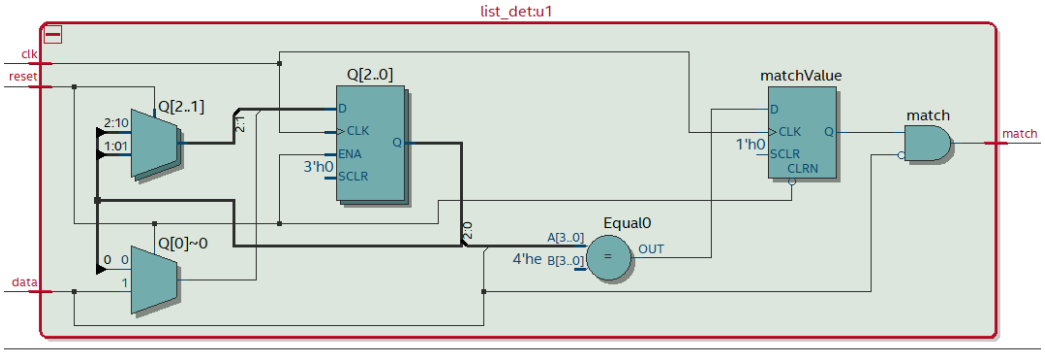

Figure 6: List det unit

The `list_det` is composed of 4 D flip flops . 3 of them compose the Q shift register that is used to store the data values . The 4th D flip-flop matchValue stores the value of the match signal which becomes 1 once the pattern 1110 has been met. In any other case it remains 0 . The signal match which is used to trigger the counter comes from an AND gate between the matchValue and the inverted data (inverted because the last bit of the pattern has to be 0 11100).

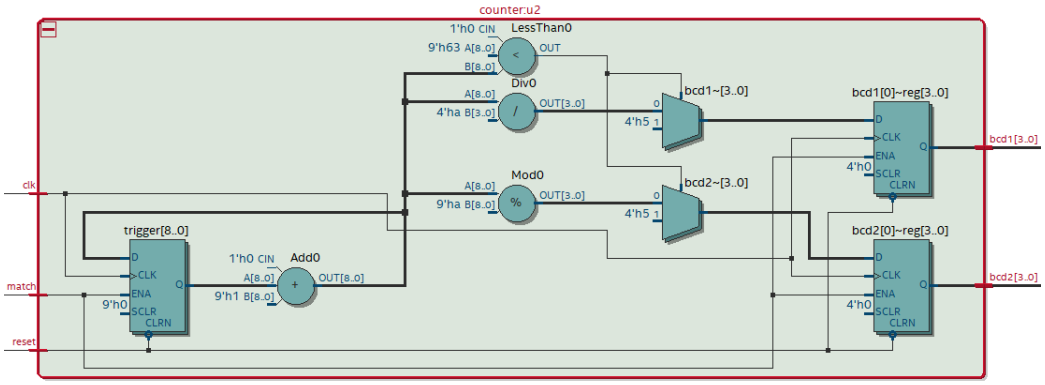

Figure 7: Counter unit

The counter is composed of 9 D flip flops trigger that compose a register on which the number of the pattern occurrences is stored . The other 8 D flip flops are registers that store the bcd values for the driver1 and the driver2.

The dsp driver1 receives bcd1 value as an input from the counter and through a lookup table the bcd is converted into the 7 segment output.

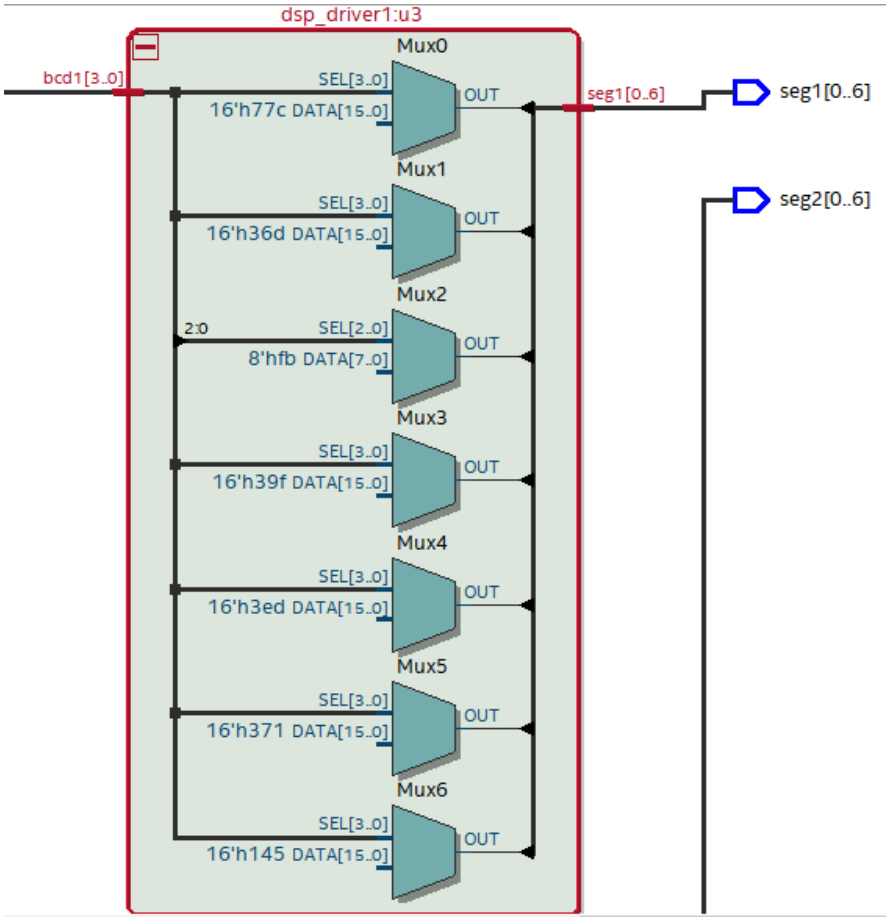

Figure 8: display driver 1 unit

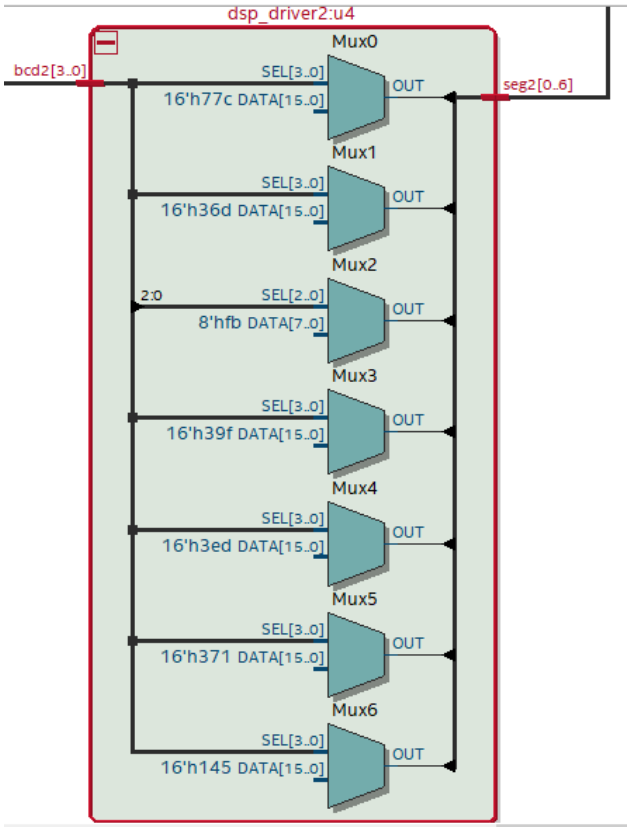

Figure 9: display driver 2 unit

The operating conditions for the Timing Analysis were The operating conditions were Fast at 1100mV at 85 degrees Celsius

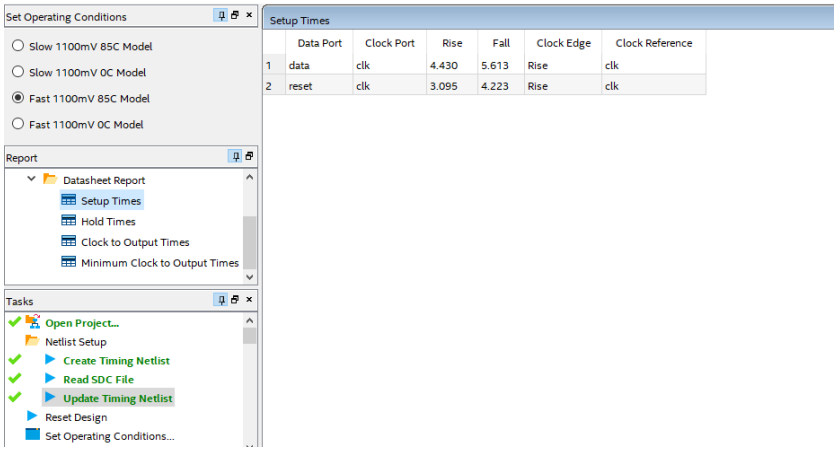

Figure 10: Timing Analysis

Setup Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	data	clk	4.430	5.613	Rise	clk
2	reset	clk	3.095	4.223	Rise	clk

Figure 11: Setup time

Hold Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	data	clk	-2.434	-3.566	Rise	clk
2	reset	clk	-2.392	-3.328	Rise	clk

Figure 12: Hold time

Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	seg1[*]	clk	5.467	5.275	Rise	clk
1	s...]	clk	5.004	5.275	Rise	clk
2	s...]	clk	5.353	5.051	Rise	clk
3	s...]	clk	5.391	5.163	Rise	clk
4	s...]	clk	5.457	5.129	Rise	clk
5	s...]	clk	5.356	5.072	Rise	clk
6	s...]	clk	5.393	5.099	Rise	clk
7	s...]	clk	5.467	5.136	Rise	clk
2	seg2[*]	clk	6.049	5.654	Rise	clk
1	s...]	clk	5.227	5.493	Rise	clk
2	s...]	clk	5.505	5.278	Rise	clk
3	s...]	clk	5.831	5.500	Rise	clk
4	s...]	clk	5.737	5.455	Rise	clk
5	s...]	clk	6.012	5.571	Rise	clk
6	s...]	clk	6.049	5.654	Rise	clk
7	s...]	clk	5.589	5.283	Rise	clk

Figure 13: Clock to ouput time

Minimum Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	seg1[*]	clk	4.469	4.531	Rise	clk
1	s...]	clk	4.469	4.684	Rise	clk
2	s...]	clk	4.785	4.531	Rise	clk
3	s...]	clk	4.630	4.598	Rise	clk
4	s...]	clk	4.875	4.597	Rise	clk
5	s...]	clk	4.759	4.547	Rise	clk
6	s...]	clk	4.809	4.577	Rise	clk
7	s...]	clk	4.856	4.604	Rise	clk
2	seg2[*]	clk	4.605	4.629	Rise	clk
1	s...]	clk	4.605	4.779	Rise	clk
2	s...]	clk	4.814	4.629	Rise	clk
3	s...]	clk	5.105	4.940	Rise	clk
4	s...]	clk	5.031	4.819	Rise	clk
5	s...]	clk	5.261	4.914	Rise	clk
6	s...]	clk	5.357	4.968	Rise	clk
7	s...]	clk	4.918	4.650	Rise	clk

Figure 14: Minimum clock to output time

6 Post Simulation Analysis

Having extracted the .vho file `pattern_recogniser_str` from Quartus a post simulation was run in the golden unit environment by comparing the outputs of the .vho file to the ones of the behavioral pattern recognizer.

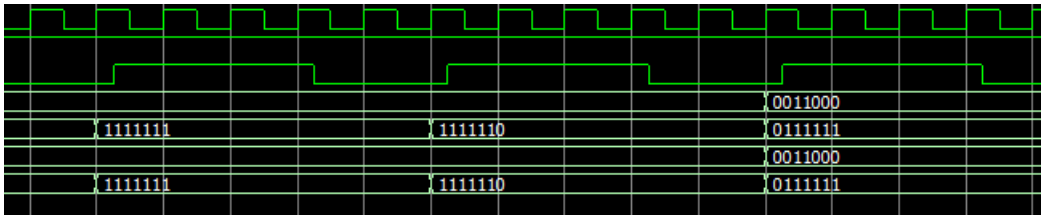

Figure 15: Post simulation

The results from the post simulation analysis match with the simulations that were applied to the previous system.