Experiment 8: Multiple String Detector

Debasish Panda 21D070021

October 11, 2022

1 Overview of the experiment

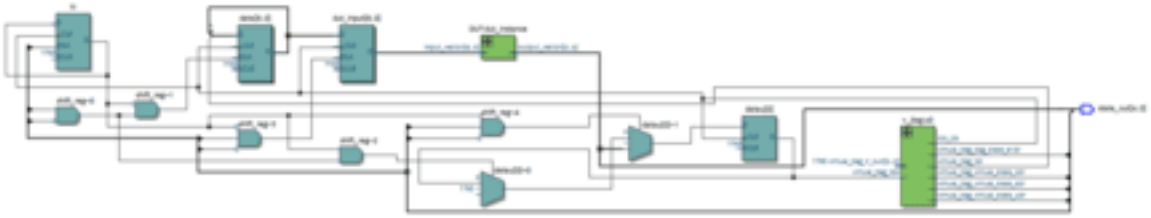
The purpose of the experiment is to design a string detector using a Mealy type FSM (Finite State Machine) which has been described in VHDL using Behavioral-Dataflow modelling. The FSM outputs '1' when input sequence of letters given thus far contains the sub-sequences: '**bomb**', '**gun**' and '**roam**'. We need to first determine the number of bits needed to distinguish the states individually, following which we have to draw the state transition diagram for the FSM. Following a successful RTL simulation, we have to demonstrate the same using Scanchain on Xenon board.

The appropriate code for Behavioral-Dataflow modelling of the system is written using VHDL on Quartus Prime software. Next, we have to demonstrate the Multiple String Detector practically by using the Xen10 FPGA board and Scanchain. In order to run the VHDL code on the board, we also need to determine the input and output pins correctly as per the specifications of the circuit board.

In this report first we discuss the setup required for the experiment and the approach used by me to solve the assignment. Attached alongwith are the RTL schematic of the FSM and picture showing the simulation output of the design.

2 Experimental setup

Note that we need to describe the Mealy FSM using behavioral modelling in order to execute the required operations on the input, so we do not need to design any hardware design as such.



RTL Schematic of the String Detector

Now that we have the required description of the inputs and expected outputs, we need to describe the same in VHDL. Observe that we will need to describe three different FSMs for the three different sub-sequences. The required code for the same is discussed in the next section.

2.1 Design Documentation and VHDL Code

The VHDL code for designing the entity-architecture pairs associated with the digital circuit is as follows:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity word_detection is
port(inp:in std_logic_vector(4 downto 0);
     reset,clock:in std_logic;
     outp: out std_logic);
end word_detection;

architecture bhv of word_detection is
```

```
---------------Define state type here-----------------------------
type state1 is (rst1,s1,s2,s3);
type state2 is (rst2,s4,s5);
type state3 is (rst3,s6,s7,s8);
---------------Define signals of state type-----------------------
signal y_present1,y_next1: state1:=rst1;
signal y_present2,y_next2: state2:=rst2;
signal y_present3,y_next3: state3:=rst3;
signal outp1, outp2, outp3: std_logic;

begin
clock_proc:process(clock,reset)
begin
    if(clock='1' and clock' event) then
          if(reset='1') then
                y_present1<=rst1;
  y_present2<=rst2;
  y_present3<=rst3;
          else
                y_present1<=y_next1;
  y_present2<=y_next2;
  y_present3<=y_next3;
          end if;
       end if;
end process;

state_transition_proc1:process(inp,y_present1)
begin
     case y_present1 is
           when rst1=>
                  if(unsigned(inp)=2) then --b has been detected
                         y_next1<=s1;
                  else
         y_next1<=y_present1;
 end if;
 when s1=>
             if(unsigned(inp)=15) then
             y_next1<=s2;
```

```
 else
          y_next1<=y_present1;
 end if;
 when s2=>
             if(unsigned(inp)=13) then
             y_next1<=s3;
 else
          y_next1<=y_present1;
 end if;
 when s3=>
             if(unsigned(inp)=2) then
             y_next1<=rst1;
 else
          y_next1<=y_present1;
 end if;
end case;
end process;


state_transition_proc2:process(inp,y_present2)
begin
     case y_present2 is
           when rst2=>
                   if(unsigned(inp)=7) then --g has been detected
                           y_next2<=s4;
                   else
         y_next2<=y_present2;
 end if;
 when s4=>
             if(unsigned(inp)=21) then
             y_next2<=s5;
 else
          y_next2<=y_present2;
 end if;
 when s5=>
             if(unsigned(inp)=14) then
             y_next2<=rst2;
 else
```

```
          y_next2<=y_present2;
 end if;
end case;
end process;


state_transition_proc3:process(inp,y_present3)
begin
     case y_present3 is
           when rst3=>
                   if(unsigned(inp)=18) then --r has been detected
                           y_next3<=s6;
                   else
         y_next3<=y_present3;
 end if;
 when s6=>
             if(unsigned(inp)=15) then
             y_next3<=s7;
 else
          y_next3<=y_present3;
 end if;
 when s7=>
             if(unsigned(inp)=1) then
             y_next3<=s8;
              else
          y_next3<=y_present3;
 end if;
 when s8=>
             if(unsigned(inp)=13) then
             y_next3<=rst3;
 else
          y_next3<=y_present3;
 end if;
end case;
end process;


output_proc1:process(y_present1, inp)
```

```
begin
    case y_present1 is
              when rst1=>
                     outp1<='0';
  when s1=>
               outp1<='0';
     when s2=>
               outp1<='0';
     when s3=>
      if(unsigned(inp)=2) then
                outp1<='1';
else
    outp1<='0';
end if;
end case;
end process;

output_proc2:process(y_present2, inp)
begin
    case y_present2 is
              when rst2=>
                     outp2<='0';
  when s4=>
               outp2<='0';
     when s5=>
      if(unsigned(inp)=14) then
                outp2<='1';
else
    outp2<='0';
end if;
end case;
end process;

output_proc3:process(y_present3, inp)
begin
    case y_present3 is
              when rst3=>
                     outp3<='0';
```

```
  when s6=>
                outp3<='0';
  when s7=>
                outp3<='0';
     when s8=>
      if(unsigned(inp)=13) then
                 outp3<='1';
else
    outp3<='0';
   end if;
end case;
end process;

outp<=outp1 or outp2 or outp3;
end bhv;
```

3 Observations

The experiment was conducted successfully, and the expected output was obtained through the RTL Simulation. The Scanchain also gave the desired result, with all test cases passing successfully. The expected output of the Universal Shifter is shown in the form of the final RTL Simulation result of the design.

RTL Simulation of FSM

Following this, we have to do the pin planning correctly and convert our VHDL file to a .svf file, which is executable by the Xen10 board and is also needed for running the Scanchain program.

4 References

[1] EE214 Github page

[2] Overleaf LaTeX tutorial for beginners

[3] Introduction to Xen10 Board