SFU SIMON FRASER UNIVERSITY

Department of Engineering Science
Faculty of Applied Science

Course Code:	ENSC 252
Course Title:	Fundamentals of Digital Logic
Semester Code:	1204

Instructor:	Anita Tino

Asst/Lab No. :	LA05
Asst/Lab Title:	FSM

Submission Date:	10th july 2020

Student Name:	Piyush Khurana
SFU ID:	301401571
Signature*:	Piyush

*By signing above, I certify that the contribution made to this lab/assignment is solely mine and confirm that all work completed is my own. Any suspicion of plagiarism and/or copying will result in an investigation in the ENSC department. A mark of zero or F may be assigned, and depending on the level of severity, an FD grade on my transcript and/or expulsion from the school, according SFU's Code of Academic Integrity found online at http://www.sfu.ca/policies/gazette/student/s10-01.html

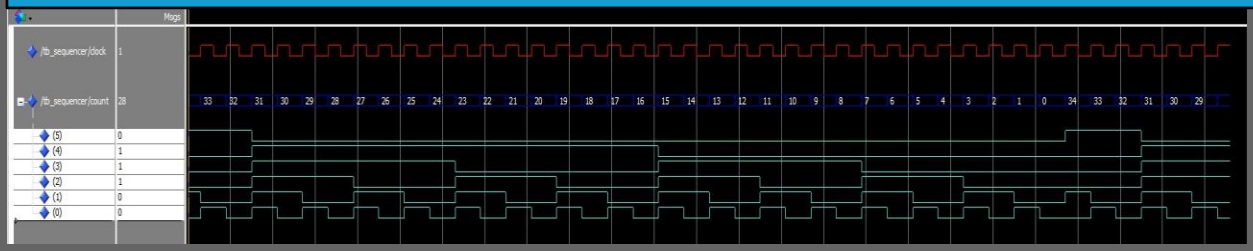
Code for Sequencer

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY sequencer IS
PORT( clk   : IN STD_LOGIC;
      count : OUT UNSIGNED(5 DOWNTO 0) );
END sequencer;

Architecture behaviour of sequencer IS
signal Q : UNSIGNED(5 downto 0) := "100010";
begin
   process(clk)
   begin
      if(rising_edge(clk)) then
         if(Q = 0) then
            Q <= "100010";
         else
            Q <= Q-1;
         end if;
      end if;
   end process;
count<=Q;
End Behaviour;
```

Code for tb_sequencer

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity tb_sequencer is
End tb_sequencer;

Architecture checking of tb_sequencer is

Component sequencer is
PORT( clk   : IN STD_LOGIC;
      count : OUT UNSIGNED(5 DOWNTO 0) );
End Component;

signal clock : STD_LOGIC;
signal count : UNSIGNED(5 DOWNTO 0);

Begin

DUT : sequencer
PORT MAP(clk=>clock, count=>count);
   process
      begin
      for ii in 0 to 40 loop
         clock<='0';
         wait for 2.5 ns;
         clock<='1';
         wait for 2.5 ns;            --    (ii)  number of rising edges
      end loop;

      wait;
   end process;
End checking;
```

Functional Simuation of Sequencer



In, this waveform, the red input is the clock whose period is 5 ns (for functional simulation). The count (output) from 33 to 0, and then again from 33 to 0 is shown in blue color, with its individual bit in cyan.

Code for async_output_decoder aka decoder1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity async_output_decoder is
PORT( seq     :  IN  UNSIGNED(5 downto 0);
      waveOut :  OUT STD_LOGIC   );
End async_output_decoder;

Architecture Behaviour of async_output_decoder is
begin
    waveOut<= '1' when seq = 33 else
              '1' when seq = 31 else
              '1' when seq = 29 else
              --S and space
              '1' when seq = 25 else
              '1' when seq = 24 else
              '1' when seq = 23 else
              --space
              '1' when seq = 21 else
              '1' when seq = 20 else
              '1' when seq = 19 else
              --space
              '1' when seq = 17 else
              '1' when seq = 16 else
              '1' when seq = 15 else
              -- O and space
              '1' when seq = 11 else
              --space
              '1' when seq = 09 else
              --space
              '1' when seq = 07 else
              --S and
              --i am dead
              '0';
End Behaviour;
```

Code for tb_decoder

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity tb_decoder is
End tb_decoder;

Architecture checking of tb_decoder is

Component async_output_decoder is
PORT( seq     :  IN  UNSIGNED(5 downto 0);
      waveOut :  OUT STD_LOGIC   );
End Component;

signal sequencer_in : UNSIGNED(5 downto 0);
signal wave_out     : STD_LOGIC;
signal temp         : UNSIGNED(5 downto 0) := "100010";

Begin
    DUT : async_output_decoder
    Port map( seq=>sequencer_in, waveOut=>wave_out);

    process
        begin
            for ii in 0 to 34 loop
                temp <= temp-1;
                sequencer_in <= temp;
                wait for 5 ns;
            end loop;
        wait;
    end process;
End checking;
```

Functional Simuation of decoder1

In, this waveform, the first series of numbers is our counter that runs from 34 down to 0 without any interruption (input from sequencer). The output in cyan is our wave of Morse code of “SOS”.

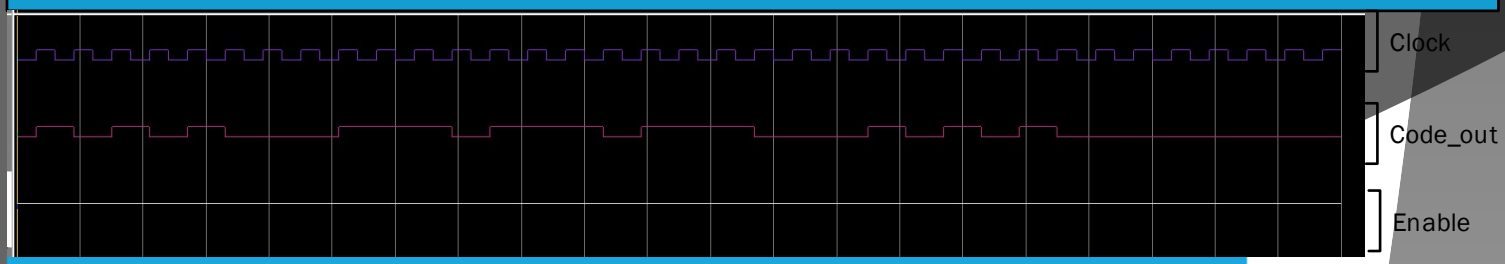
Code for box1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity box1 is
Port( clock, enable : IN STD_LOGIC;
       code_out      : OUT STD_LOGIC  );
End box1;

Architecture struct of box1 is

Component sequencer is
PORT( clk   : IN STD_LOGIC;
      count : OUT UNSIGNED(5 DOWNTO 0) );
End Component;

Component async_output_decoder is
PORT( seq     :  IN  UNSIGNED(5 downto 0);
      waveOut :  OUT STD_LOGIC   );
End Component;

signal ordinary_wire : UNSIGNED(5 downto 0);
signal output_hold   : STD_LOGIC;

begin

    sequence_producer : sequencer port map(clk=>clock, count=>ordinary_wire );
    decoder           : async_output_decoder port map( seq=>ordinary_wire,
waveOut=>output_hold);

    code_out <= output_hold when Enable = '1' else
                '0';
End struct;
```

Code for tb_box1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity tb_box1 is
End tb_box1;

Architecture checking of tb_box1 is

Component box1 is
Port( clock, enable : IN STD_LOGIC;
      code_out      : OUT STD_LOGIC  );
End Component;

signal clock      : STD_LOGIC;
signal enable     : STD_LOGIC;
signal code_out   : STD_LOGIC;

Begin
   DUT : box1
   Port map(clock, enable, code_out);

   enable<='1';
   process
      begin
      for ii in 0 to 34 loop
          clock<='0';
          wait for 2.5 ns;
          clock<='1';
          wait for 2.5 ns;            --   (ii)  number of rising edges
      end loop;

      wait;
   end process;
End checking;
```

Functional Simuation of tb_box1



In, this waveform, the sequencer and decoder from previous designs are integrated from previous designs are integrated such that the Code_out. Whenever the Enable is high, which is high for the entire time, the code_out is making a morse output else it produces an active low.

Code for sequencer2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY sequencer2 IS
PORT( clk   : IN STD_LOGIC;
      count : OUT UNSIGNED(5 DOWNTO 0) );
END sequencer2;

Architecture behaviour of sequencer2 IS
signal Q : UNSIGNED(5 downto 0) := "111111";
begin
   process(clk)
   begin
      if(rising_edge(clk)) then
         if(Q = 0) then
            Q <= "111111";
         else
            Q <= Q-1;
         end if;
      end if;
   end process;
count<=Q;
End Behaviour;
```

Code for decoder2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity decoder2 is
PORT( seq     :  IN  UNSIGNED(5 downto 0);
      waveOut :  OUT STD_LOGIC   );
End decoder2;

Architecture Behaviour of decoder2 is
begin
    waveOut<= '1' when seq = 62 else
              --pause
              '1' when seq = 60 else
              '1' when seq = 59 else
              '1' when seq = 58 else
              --pause
              '1' when seq = 56 else
              '1' when seq = 55 else
              '1' when seq = 54 else
               --pause
              '1' when seq = 52 else
              --first alphabet P is  completed (skipping 3 for a space)
              '1' when seq = 48 else
              --pause
              '1' when seq = 46 else
              -- second Alphabet I is completed (skipping 3 for a space)
              '1' when seq = 42 else
              '1' when seq = 41 else
              '1' when seq = 40 else
              --pause
              '1' when seq = 38 else
              --pause
              '1' when seq = 36 else
              '1' when seq = 35 else
              '1' when seq = 34 else
              --pause
              '1' when seq = 32 else
              '1' when seq = 31 else
              '1' when seq = 30 else
              --- 3rd alphabet Y is completed (skipping 3 for a space)
              '1' when seq = 26 else
              --pause
              '1' when seq = 24 else
              --pause
              '1' when seq = 22 else
              '1' when seq = 21 else
              '1' when seq = 20 else
              -- 4th letter U is completed (skipping 3 for a space)
              '1' when seq = 16 else
              --pause
              '1' when seq = 14 else
              --pause
              '1' when seq = 12 else
              -- 5th letter S is completed (skipping 3 for a space)
              '1' when seq = 8 else
              --pause
              '1' when seq = 6 else
              --pause
              '1' when seq = 4 else
              --pause
              '1' when seq = 2 else
              --last alphabet of my name is H and its over and
              -- and i am dead (again)
              '0';
End Behaviour;
```

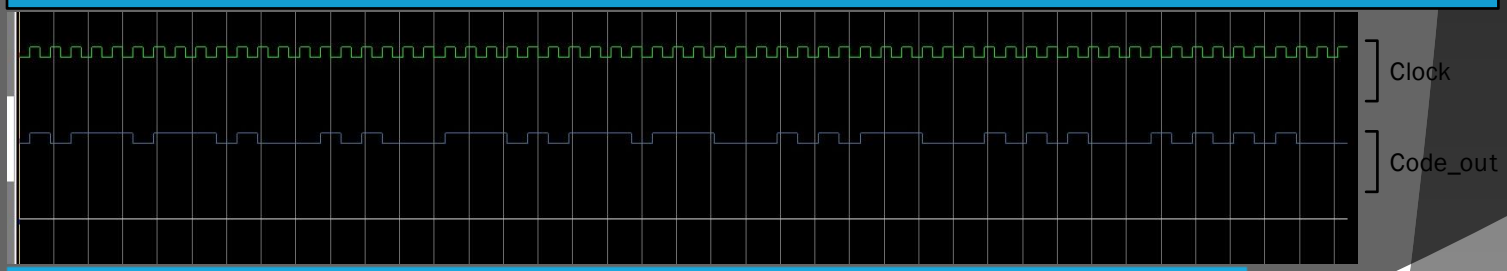
Code for box2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity box2 is
Port( clock, enable : IN STD_LOGIC;
      code_out      : OUT STD_LOGIC  );
End box2;

Architecture struct of box2 is

Component sequencer2 is
PORT( clk   : IN STD_LOGIC;
      count : OUT UNSIGNED(5 DOWNTO 0) );
End Component;

Component decoder2 is
PORT( seq     :  IN  UNSIGNED(5 downto 0);
      waveOut :  OUT STD_LOGIC   );
End Component;

signal ordinary_wire : UNSIGNED(5 downto 0);
signal output_hold   : STD_LOGIC;

begin

   sequence_producer : sequencer2 port map(clk=>clock, count=>ordinary_wire );
   decoder           : decoder2 port map( seq=>ordinary_wire, waveOut=>output_hold);

   code_out <= output_hold when Enable = '1' else
               '0';
End struct;
```

Code for tb_box2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity tb_box2 is
End tb_box2;

Architecture checking of tb_box2 is

Component box2 is
Port( clock, enable : IN STD_LOGIC;
      code_out      : OUT STD_LOGIC  );
End Component;

signal clock      : STD_LOGIC;
signal enable     : STD_LOGIC;
signal code_out   : STD_LOGIC;

Begin
   DUT : box2
   Port map(clock, enable, code_out);

   enable<='1';
   process
      begin
      for ii in 0 to 63 loop
         clock<='0';
         wait for 2.5 ns;
         clock<='1';
         wait for 2.5 ns;            --   (ii)  number of rising edges
      end loop;

      wait;
   end process;
End checking;
```

Functional Simuation of tb_box2



In, this waveform, the sequencer2 and decoder2 from previous designs are integrated from previous designs are integrated such that the Code_out. Whenever the Enable is high, which is high for the entire time, the code_out is making a morse output for “PIYUSH” else it produces an active low.

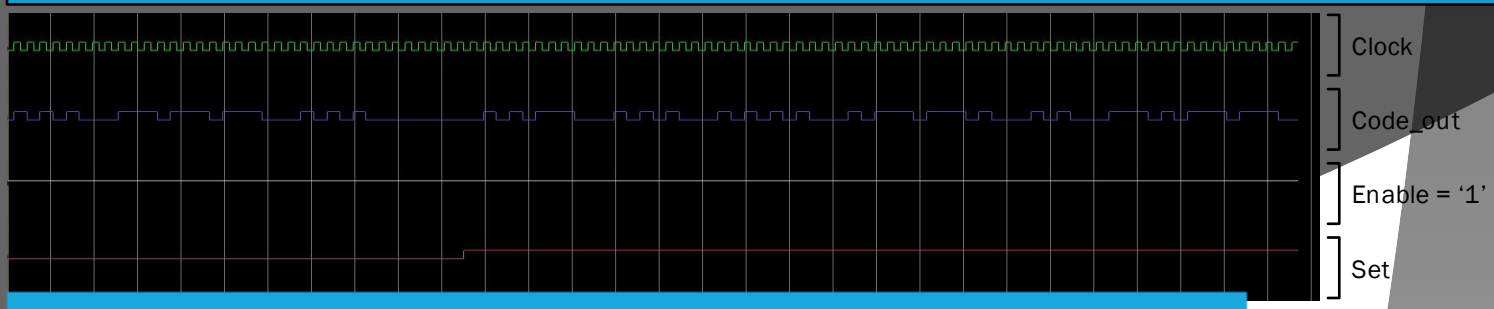
Code for distress_box

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity distress_box  is
PORT( clock, set, enable : IN STD_LOGIC;
      code_out           : OUT STD_LOGIC );
End distress_box;

Architecture checking of distress_box is

Component box1 is
Port( clock, enable : IN STD_LOGIC;
      code_out      : OUT STD_LOGIC  );
End Component;

Component box2 is
Port( clock, enable : IN STD_LOGIC;
      code_out      : OUT STD_LOGIC  );
End Component;

signal out_hold_box1   : STD_LOGIC;
signal out_hold_box2   : STD_LOGIC;

begin

   B1 :  box1 port map(clock, enable, out_hold_box1);
   B2 :  box2 port map(clock, enable, out_hold_box2);

   code_out <= out_hold_box1 when set<= '0' else
               out_hold_box2 when set<= '1';

End checking;
```

Code for distress_box

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

Entity tb_distress_box is
End tb_distress_box;

Architecture checking of tb_distress_box is

Component distress_box is
PORT( clock, set, enable : IN STD_LOGIC;
      code_out           : OUT STD_LOGIC );
End Component;

signal clock      : STD_LOGIC;
signal enable     : STD_LOGIC;
signal code_out   : STD_LOGIC;
signal set          : STD_LOGIC;

Begin
   DUT : distress_box
   Port map(clock, set, enable, code_out);
   enable<='1';
   process
      begin
      set<='0';
      for ii in 0 to 34 loop
         clock<='0';
         wait for 2.5 ns;
         clock<='1';
         wait for 2.5 ns;            --   (ii)  number of rising edges
      end loop;
      set<='1';
      for ii in 0 to 63 loop
         clock<='0';
         wait for 2.5 ns;
         clock<='1';
         wait for 2.5 ns;            --   (ii)  number of rising edges
      end loop;
      wait;
   end process;
End checking;
```

Functional Simuation of distress_box



In this waveform, the period of clock is 5 ns again, Enable is set to 1 all the time as we see here, we have given active low to Set for starting 34 clock cycles and then active high for rest of the 63 loops. When the set is low, we get “SOS” in the code_out(output), and when set is high, we get “PIYUSH” as our output.