Experiment 7: Sequence Generator

Debasish Panda 21D070021

October 4, 2022

1 Overview of the experiment

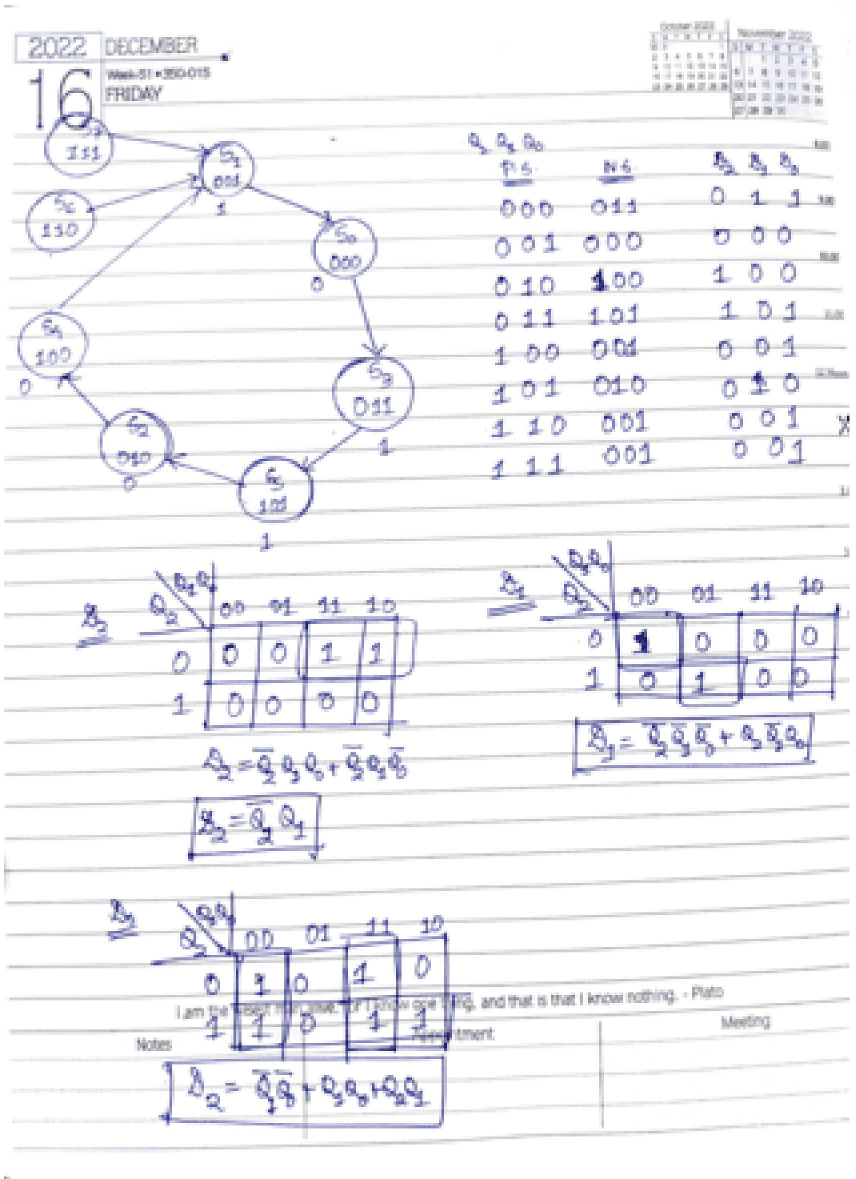
The purpose of the experiment is to generate the sequence 101100 using an FSM (Finite State Machine) which has been described in VHDL using Behavioral-Dataflow modelling. We need to first determine the number of bits needed to distinguish the states individually, following which we have to draw the state 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 Sequence Generator 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 pen-and-paper description of the FSM and picture showing the simulation output of the design.

2 Experimental setup

Note that we need to describe the 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.



$Q_2\,Q_1\,Q_0$

P.S	N.S	$D_2\ D_1\ D_0$
000	011	0 1 1
001	000	0 0 0
010	100	1 0 0
011	101	1 0 1
100	001	0 0 1
101	010	0 1 0
110	001	0 0 1 X
111	001	0 0 1

D_2

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	0	0	1	1
1	0	0	0	0

$D_2 = \bar{Q}_2 Q_1 Q_0 + \bar{Q}_2 Q_1 \bar{Q}_0$

$\boxed{D_2 = \bar{Q}_2 Q_1}$

D_1

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	1	0	0	0
1	0	1	0	0

$\boxed{D_1 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + Q_2 \bar{Q}_1 Q_0}$

D_0

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	1	0	1	0
1	1	0	1	1

$\boxed{D_0 = \bar{Q}_1 \bar{Q}_0 + Q_1 Q_0 + Q_2 Q_1}$

Description of FSM

Now that we have the required description of the inputs and expected outputs, we need to describe the same in VHDL. 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:

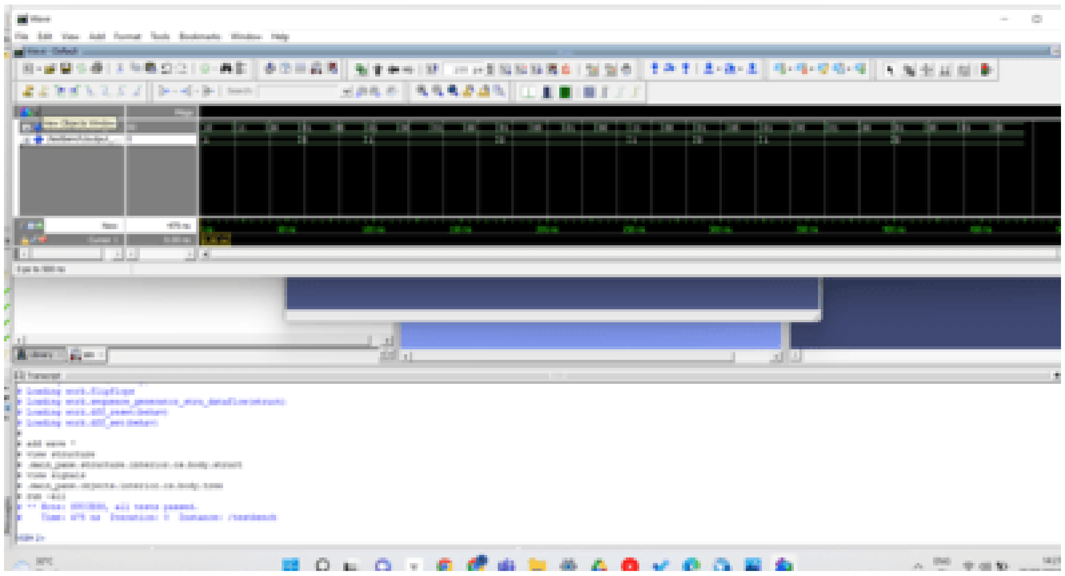
```
library ieee;
use ieee.std_logic_1164.all;

package Flipflops is
component dff_set is
port(D,clock,set:in std_logic;Q:out std_logic);
end component dff_set;

component dff_reset is
port(D,clock,reset:in std_logic;Q:out std_logic);
end component dff_reset;

end package Flipflops;

--D flip flop with set
library ieee;
use ieee.std_logic_1164.all;

entity dff_set is
port(D,clock,set:in std_logic;Q:out std_logic);
end entity dff_set;

architecture behav of dff_set is
begin
dff_set_proc: process (clock,set)
begin
if(set='1')then -- set implies flip flip output logic high
Q <= '1'; -- write the flip flop output when set
```

```
elsif (clock'event and (clock='1')) then
Q <= D; -- write flip flop output when not set
end if ;
end process dff_set_proc;
end behav;

--D flip flop with reset
library ieee;
use ieee.std_logic_1164.all;

entity dff_reset is
port(D,clock,reset:in std_logic;Q:out std_logic);
end entity dff_reset;

architecture behav of dff_reset is
begin
dff_reset_proc: process (clock,reset)
begin
if(reset='1')then -- reset implies flip flip output logic low
Q <= '0'; -- write the flip flop output when reset
elsif (clock'event and (clock='1')) then
Q <= D; -- write flip flop output when not reset
end if ;
end process dff_reset_proc;
end behav;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.Flipflops.all;

entity Sequence_generator_stru_dataflow is
port (reset,clock: in std_logic;
y:out std_logic);
end entity Sequence_generator_stru_dataflow;

architecture struct of Sequence_generator_stru_dataflow is
```

```
 signal D: std_logic_vector(2 downto 0);
 signal Q: std_logic_vector(2 downto 0);
begin

ff1:dff_reset port map(d(2), Clock,reset, q(2));
ff2:dff_reset port map(d(1),Clock,reset,q(1));
ff3:dff_set port map(d(0), Clock, reset, q(0));

D(2)<= (not Q(2) and Q(1));
D(1)<= ((not Q(2) and not Q(1) and not Q(0)) or (Q(2) and not Q(1) and Q(0)));
D(0)<= ((not Q(1) and not Q(0)) or (Q(1) and Q(0)) or (Q(2) and Q(1)));

y<=Q(0);

end struct;
```

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