EE214 Digital Circuits Laboratory

Wadhwani Electronics Laboratory
Electrical Engineering IIT Bombay

Problem set: 6 Date: September 13, 2023

Instructions:

1. Use **Dataflow** modeling for writing VHDL description
2. Perform RTL simulation using the provided testbench and tracefile.
3. Demonstrate the simulations to your TA
4. Perform **Scanchain** on the Xenon board and verify with your TA.
5. Submit the entire project files in .zip format in moodle.

Problem Statement

- Write VHDL description in **Structural-Dataflow** modeling to generate the sequence **1 0 1 1 0 0**.
- Use **structural-dataflow** modeling only.
- Reset is asynchronous in nature i.e. reset effects the output sequence irrespective of the input clock arrival.
- On Reset, sequence should start from the first '1'.
- Unused states should be mapped to one of the known state which is reset state.

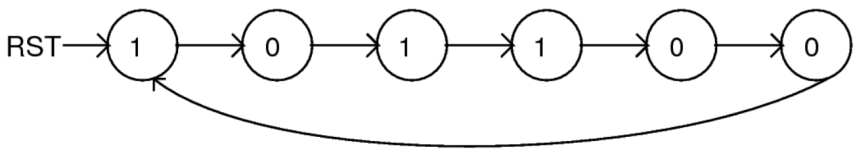

Figure 1: Sequence Generator

- Determine number of bits required to distinguish the states individually.
- Draw a state diagram to generate the states so that LSB of the states will generate the required sequence.
- Draw a state table consisting of Present State, Nest State and D FlipFlop inputs.

Present State(Qn..Q1 Q0)	Next State(N_Qn..N_Q1 N_Q0)	Dn...D1 D0
one state	next state	DFF inputs
–	–	–

- From the state table with the help of K-Maps generate equations for DFF inputs in terms of present state and reset. Take LSB of the states as your output.
- Tracefile format: ($< reset\ clock >$ $< output >$ $< Maskbit >$)
  Tracefile
- Perform Scanchain on the Xenon Board.

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


library ieee;
use ieee.std_logic_1164.all;
-- write the Flipflops packege declaration
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(...);
signal Q:std_logic_vector(...);
begin
-- write the equations in dataflow e.g z=a+bc written as z <= a or (b and c)
-- for DFF inputs which was derived and also for y.
-- Instantiate components dff_reset
-- and dff_set appropriately using port map statements.
end struct;
```

- Demo code snippet is given. Change the code accordingly.