Ain Shams University

Fuculty of Engineering - CHEP
CESS Program

Fall 2019

Electronic Design Automation (CSE215)

STRING RECOGNIZER PROJECT

Name: Ahmed Abd Elnasser

ID: 17P8113

Email: aahmed.aabdelnasser@gmail.com

Program: CESS

INTRODUCTION:

A finite state recognizer has one input (In) and two outputs (Out) and (Termination), in addition to the (Reset) and (Clock) inputs (+VDD and VSS).

The output is asserted whenever the input sequence 010 (detection string) has been observed, as long as the sequence 100 (termination string) has never been seen.

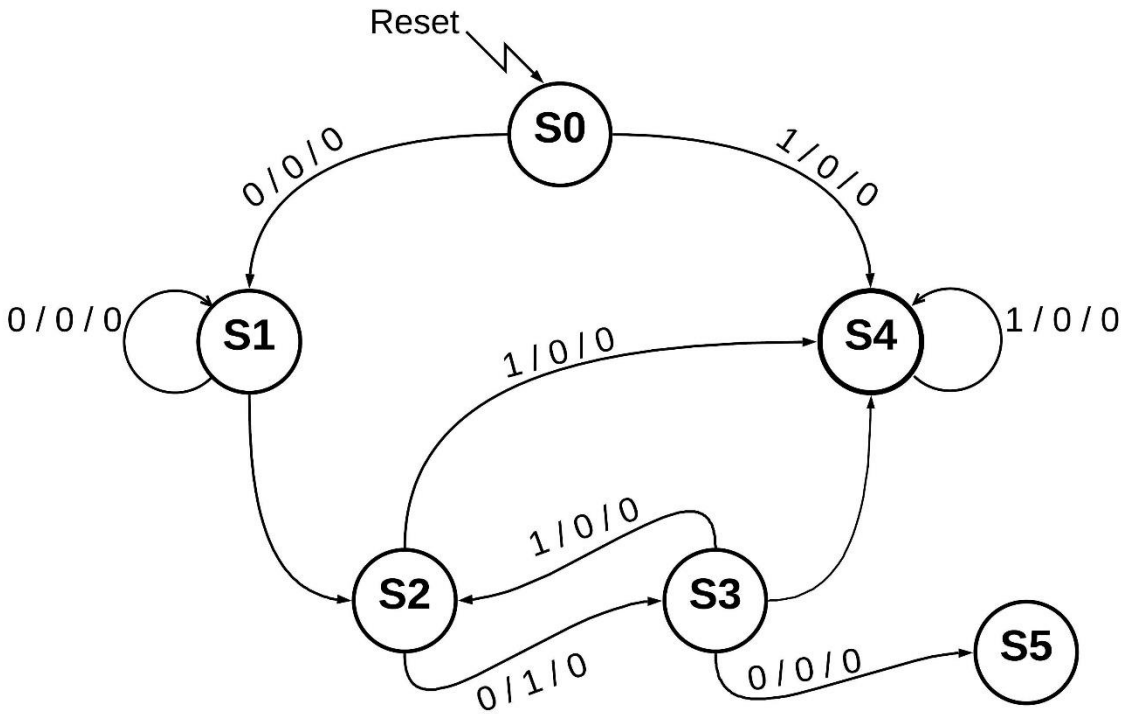
Once 100 appears, the (Termination) output is asserted and the output remains 0 till the (Reset) is asserted.

1. PART 1: FSM:

This Finite State Machine is designed using a **MEALY** output and a **SYNCHRONOUS** reset signal.

1.1. State Diagrams:

1.1.1 Transition diagram:



1.1.2. Transition table:

Current State				In	Next State				Outputs	
ID	**CS2**	**CS1**	**CS0**	**i**	**ID**	**NS2**	**NS1**	**NS0**	**out**	**term**
S0	0	0	0	0	S1	0	0	1	0	0
S0	0	0	0	1	S4	1	1	0	0	0
S1	0	0	1	0	S1	0	0	1	0	0
S1	0	0	1	1	S2	0	1	1	0	0
S2	0	1	1	0	S3	0	1	0	1	0
S2	0	1	1	1	S4	1	1	0	0	0
S3	0	1	0	0	S5	1	1	1	0	1
S3	0	1	0	1	S2	0	1	1	0	0
S4	1	1	0	0	S3	0	1	0	0	0
S4	1	1	0	1	S4	1	1	0	0	0
S5	1	1	1	0	S5	1	1	1	0	0
S5	1	1	1	1	S4	1	1	1	0	0

1.2. TestBench Strategy:

The strategy for testing will depend on finding an input sequence that satisfies all state transitions and gets through all possible outputs, then put the FSM under testing this sequence and assert the expected outputs.

Also testing the reset signal by starting with the circuit on then switching the reset to 1 after a certain amount of delay time.

Feature	Proposed Test		Expected Outputs	
	Transition	Input	out	term
State Transition	S0 ~ S1	0	0	0
	S1 ~ S2	0	0	0
	S1 ~ S2	1	0	0
Output	S2 ~ S3	0	1	0
State Transition	S3 ~ S2	1	0	0
	S2 ~ S4	1	0	0
	S4 ~ S4	1	0	0
	S4 ~ S3	0	0	0
	S3 ~ S2	1	0	0
Output	S2 ~ S3	0	1	0
Terminate & Remain	S3 ~ S5	0	0	1
	S5 ~ S5	1	0	1
	S5 ~ S5	0	0	1
Proposed Test Sequence			**001010111010010**	

1.3. Simulation:

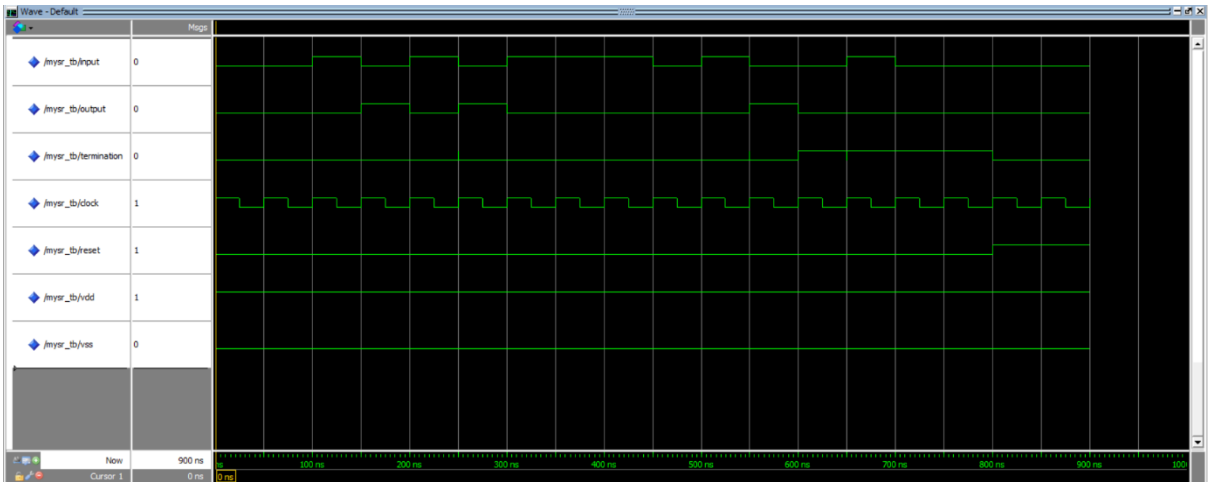
In this stage we shall apply the test strategy proposed earlier on a simulator (ModelSim) and assert the results.

1.3.1. Expected results:

Sequence	001010111010010
out	000101000001000
term	000000000000111

1.3.2 ModelSim simulation results:

After simulating the TestBench, the observation of the outputs waves confirms that the FSM sutisfies the requiremets.



APPENDIX

FSM VHDL Source Code:

```
--String Recognizer Finite State Machine (mysr) Simulation


entity mysr is

port(
i: in bit;
outp: out bit;
term: out bit;
clk: in bit;
rs: in bit;
vdd,vss: in bit
);

end entity mysr;


architecture mealy_arch of mysr is

--Enumeration Type for states
type state is (s0,s1,s2,s3,s4,s5);

--Declaring signals
signal cs: state;
signal ns: state;
```

```
begin

--Next_State transition process with SYNCHRONOUS reset
process(cs,i,rs)
begin

if(rs = '1') then ns <= s0; outp <= '0'; term <= '0';

else

case cs is

when s0 =>
if(i = '0') then ns <= s1; outp <= '0'; term <= '0';
else ns <= s4; outp <= '0'; term <= '0';
end if;

when s1 =>
if(i = '0') then ns <= s1; outp <= '0'; term <= '0';
else ns <= s2; outp <= '0'; term <= '0';
end if;

when s2 =>
if(i = '0') then ns <= s3; outp <= '1'; term <= '0';
else ns <= s4; outp <= '0'; term <= '0';
end if;

when s3 =>
if(i = '0') then ns <= s5; outp <= '0'; term <= '1';
else ns <= s2; outp <= '0'; term <= '0';
```

```
end if;

when s4 =>
if(i = '0') then ns <= s3; outp <= '0'; term <= '0';
else ns <= s4; outp <= '0'; term <= '0';
end if;

when s5 =>
ns <= s5; outp <= '0'; term <= '1';

when others => assert(false) report "illegal state" severity error;

end case;

end if;

end process;


--state update process
process(clk,rs)
begin

if(clk = '1' and not clk' stable) then
cs <= ns;
end if;

end process;

end mealy_arch;
```

TestBench:

```
--TestBench for the string recognizer FSM

entity mysr_tb is
end entity mysr_tb;



architecture test of mysr_tb is

--Component declaration for the DUT
component mysr is

port(
i : in bit;
outp : out bit;
term : out bit;
clk : in bit;
rs : in bit;
vdd,vss : in bit
);

end component mysr;

for mysr_dut: mysr use entity work.mysr(mealy_arch);



--Declaring input signals and initialize them
signal input: bit := '0';
signal output: bit := '0';
signal termination: bit := '0';
```

```
signal clock: bit := '1';
signal reset: bit := '0';
signal vdd: bit := '1';
signal vss: bit := '0';

constant clock_period: time := 50 ns;
constant seq: bit_vector := "001010111010010";



begin

--Instantiating the DUT
mysr_dut: mysr port map(input,output,termination,clock,reset,vdd,vss);



--Clock process with 50% duty cycle
process
begin

clock <= '1';
wait for clock_period/2;
clock <= '0';
wait for clock_period/2;

end process;



--Stimulus process
process
begin

reset <= '1' after (16*clock_period); --To test the reset signal after termination
```

```
for x in 0 to (seq'length-1) loop
input <= seq(x);
wait for clock_period;

if(x >= 2) then

if(seq(x) = '0') then

--The next 2 if conditions check for the output
if(seq(x-1) = '1') then

if(seq(x-2) = '0') then
report "output = 1";
assert output = '1' report "output should be 1 here" severity error;
end if;

--This else if condition checks for the termination
elsif(seq(x-1) = '0') then

if(seq(x-2) = '1') then
report "termination = 1";
assert termination = '1' report "termination should be 1 here" severity error;
end if;
end if;
end if;
end if;
end loop;
wait;
end process;
end test;
```