

# Experiment 6: String Recognizer

Hardik Panchal  
Roll Number 200070054  
EE-214, WEL, IIT Bombay  
October 5, 2021

## Overview of the experiment:

In this experiment, we will design a string detector using a Mealy-type FSM to detect the occurrence of the covid word in a string of letters. The design accepts a sequence of letters coded in binary and outputs a '1' if the required word is detected. The letters of covid can be present anywhere in the string but should be in sequence.  
e.g., letters a = "00001", b = "00010" and so on.  
For instance, "lclolmldl" is the input text then the output sequence would be "00000000010".

## Approach to the experiment:

We will make a state transition table from this given below figure of an FSM. In architecture, we have defined three different processes.

#	State	Transition	Table		
	Reset	Input	P.S.	N.S.	Output
1		X	xxx	RST	0
0		'c'	RST	1	0
0		'o'	1	2	0
0		'v'	2	3	0
0		'j'	3	4	0
0		'd'	4	RST	1

First is the clock process, second is the state transition process, and third is the output process.

Each process will run concurrently.

The output will be '1' if we reach state four and encounter 'd' as an input. In all other cases, the output is '0'.

Design document and VHDL code if relevant:

Design:

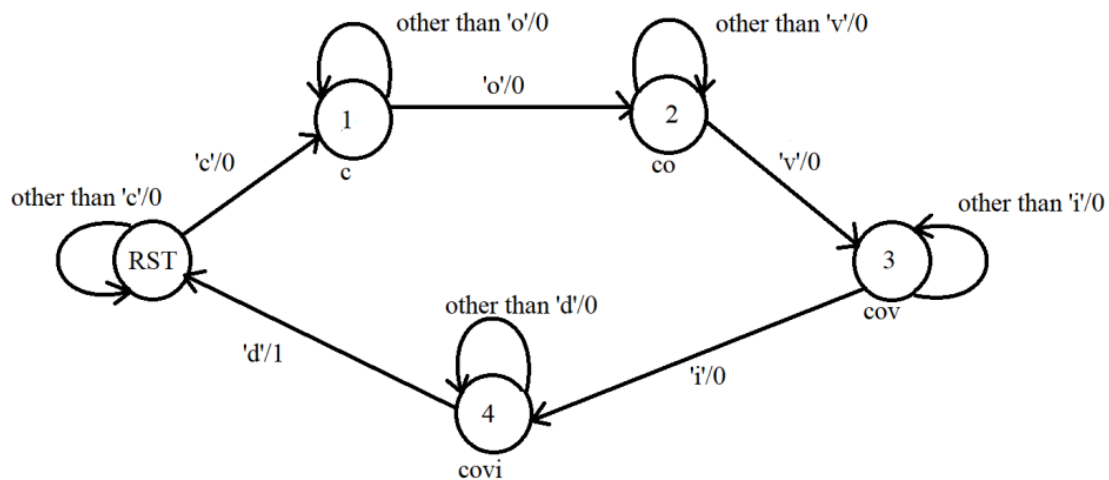


Figure 1: State diagram for detecting word "covid" in the given string of alphabets.

# State Transition Table					
Reset	Input	P.S.	N.S.	Output	
1	X	XXX	RST	0	
0	'c'	RST	1	0	
0	'o'	1	2	0	
0	'v'	2	3	0	
0	'i'	3	4	0	
0	'd'	4	RST	1	

The architecture of main logic:

```

architecture rch of cov_detect is
type state is (rst,s1,s2,s3,s4);
signal y_present,y_next: state:=rst;

begin
clock_proc:process(clock,reset)

```

```

begin
    if(clock='1' and clock' event) then
        if(reset='1') then
            y_present<=rst;
        else
            y_present<=y_next;
        end if;
    end if;
end process;

state_transition_proc:process(inp,y_present)
begin
    case y_present is

        when rst=>
            if(unsigned(inp)=3) then    --c
                y_next<=s1;
            else
                y_next<=rst;
            end if;

        when s1=>
            if(unsigned(inp)=15) then    --o
                y_next<=s2;
            else
                y_next<=s1;
            end if;

        when s2=>
            if(unsigned(inp)=22) then    --v
                y_next<=s3;
            else
                y_next<=s2;
            end if;

        when s3=>
            if(unsigned(inp)=9) then    --i
                y_next<=s4;
            else
                y_next<=s3;
            end if;

        when s4=>
            if(unsigned(inp)=4) then    --d

```

```

        y_next<=rst;
    else
        y_next<=s4;
    end if;

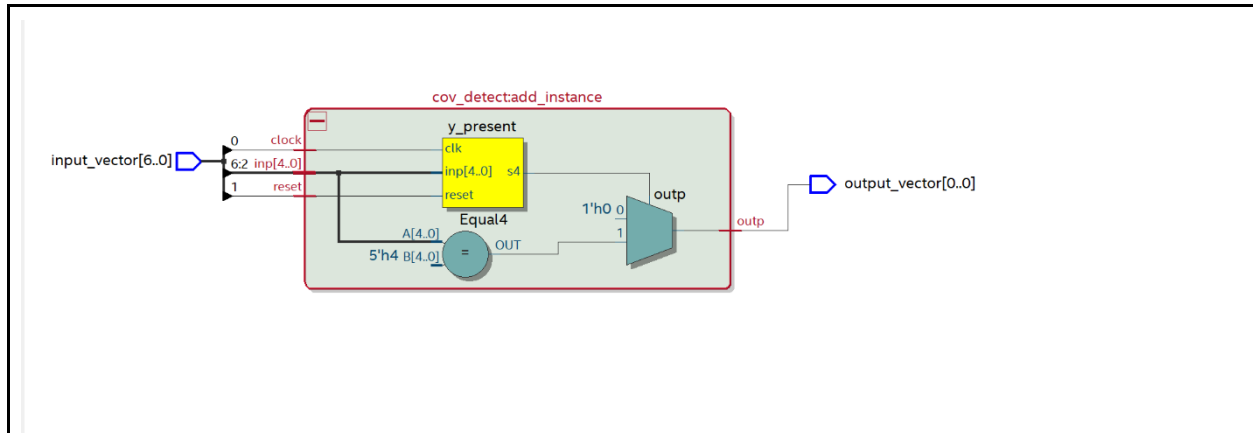
    end case;
end process;

output_proc:process(inp,y_present)
begin
    case y_present is
        when rst=>
            if(unsigned(inp)=3) then    --c
                outp<='0';
            else
                outp<='0';
            end if;
        when s1=>
            if(unsigned(inp)=15) then    --o
                outp<='0';
            else
                outp<='0';
            end if;
        when s2=>
            if(unsigned(inp)=22) then    --v
                outp<='0';
            else
                outp<='0';
            end if;
        when s3=>
            if(unsigned(inp)=9) then    --i
                outp<='0';
            else
                outp<='0';
            end if;
        when s4=>
            if(unsigned(inp)=4) then    --d
                outp<='1';
            else
                outp<='0';
            end if;
        end case;
    end process;

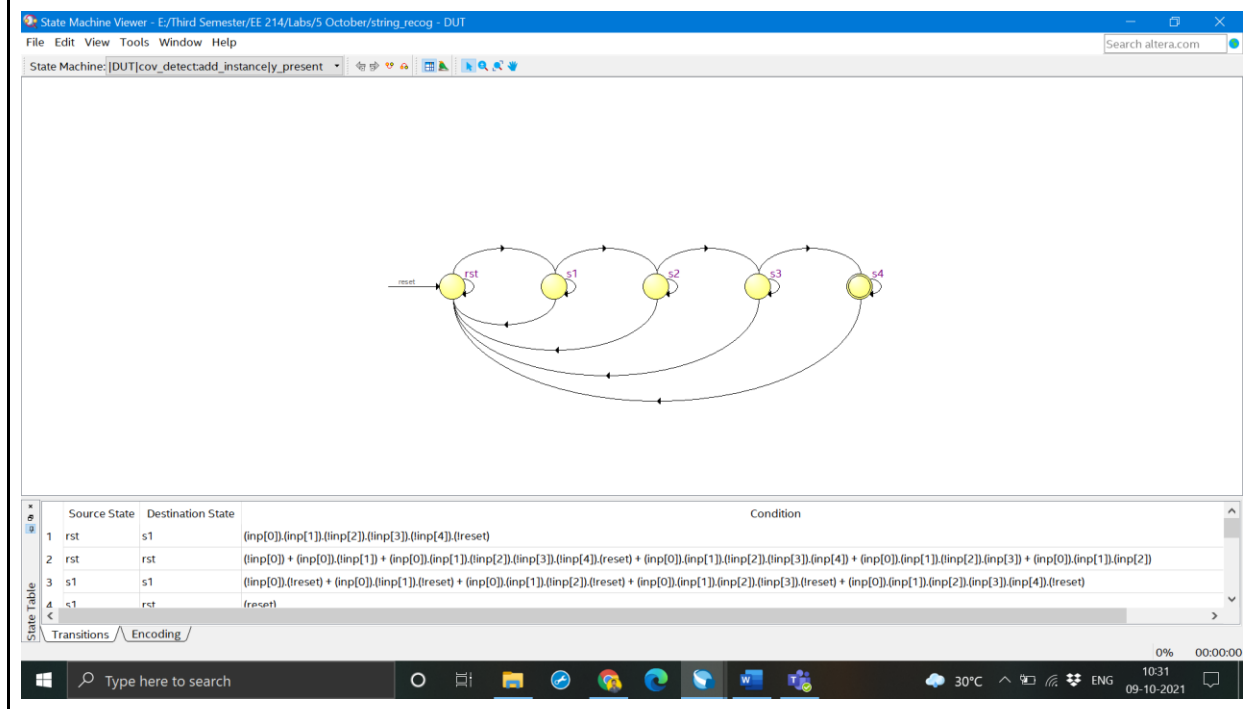
```

```
end rch;
```

## RTL View:



## State Machine Viewer:



## DUT Input/Output Format:

**Input:** 5-bit input reset clock **LSB** = clock **MSB** = inp(4)

**Output:** outp **LSB** = **MSB** = outp

### Some Test Cases from TRACEFILE.txt

**Format:** 5-bit input reset clock outp mask-bit

```
0011000 0 1
0011001 0 1
1001100 0 1
1001101 0 1
0011000 0 1
0011001 0 1
0111100 0 1
0111101 0 1
0101000 0 1
0101001 0 1
0101000 0 1
0101001 0 1
0101100 0 1
0101101 0 1
0101100 0 1
0101101 0 1
0101100 0 1
0101101 0 1
0101100 0 1
0110000 0 1
```

## RTL Simulation:

### Transcript

```
ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help

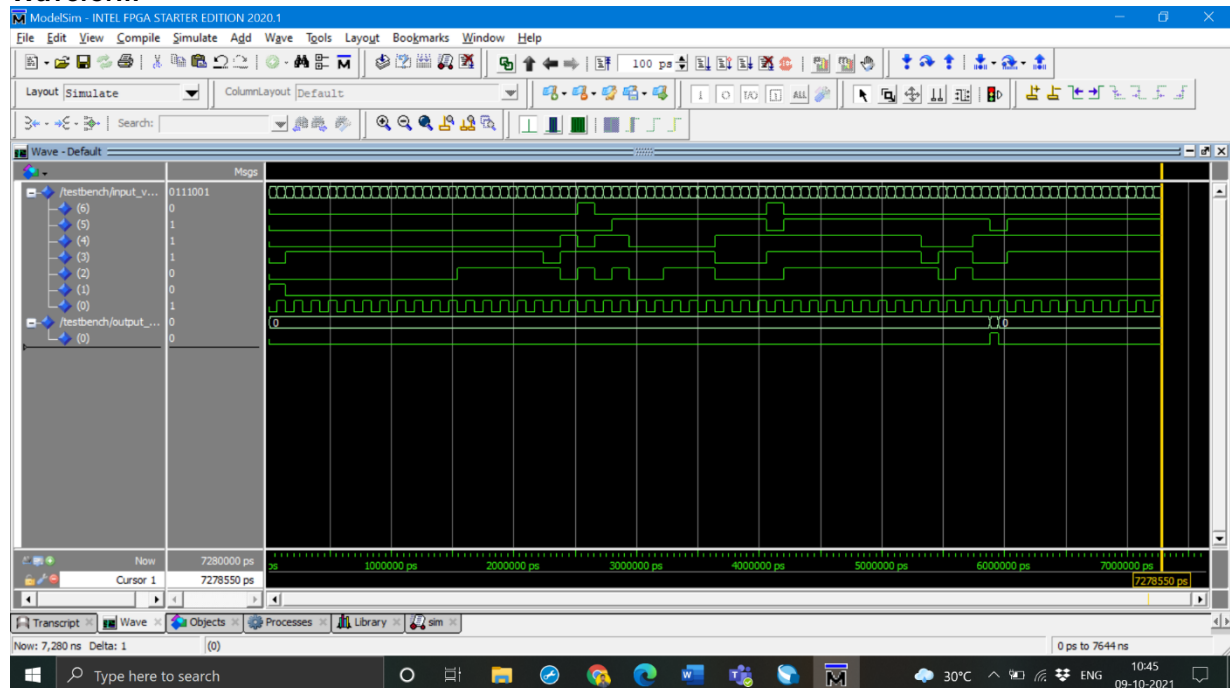
Layout Simulate ColumnLayout Default

Search:

Transcript
# -- Loading package std_logic_1164
# -- Compiling entity Testbench
# -- Compiling architecture Behave of Testbench
# End time: 10:33:35 on Oct 09, 2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t lps -L altera -L lpm -L sgate -L altera_mf -L altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# vsim -t lps -L altera -L lpm -L sgate -L altera_mf -L altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# Start time: 10:33:36 on Oct 09, 2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(bhavior)
# Loading work.dut(dutwrap)
# Loading ieee.numeric_std(body)
# Loading work.cov_detect(rch)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Warning: NUMERIC_STD.*: metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /testbench/dut_instance/add_instance
# ** Warning: NUMERIC_STD.*: metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /testbench/dut_instance/add_instance
# ** Note: SUCCESS, all tests passed.
# Time: 7280 ns Iteration: 0 Instance: /testbench
V$IM 2>

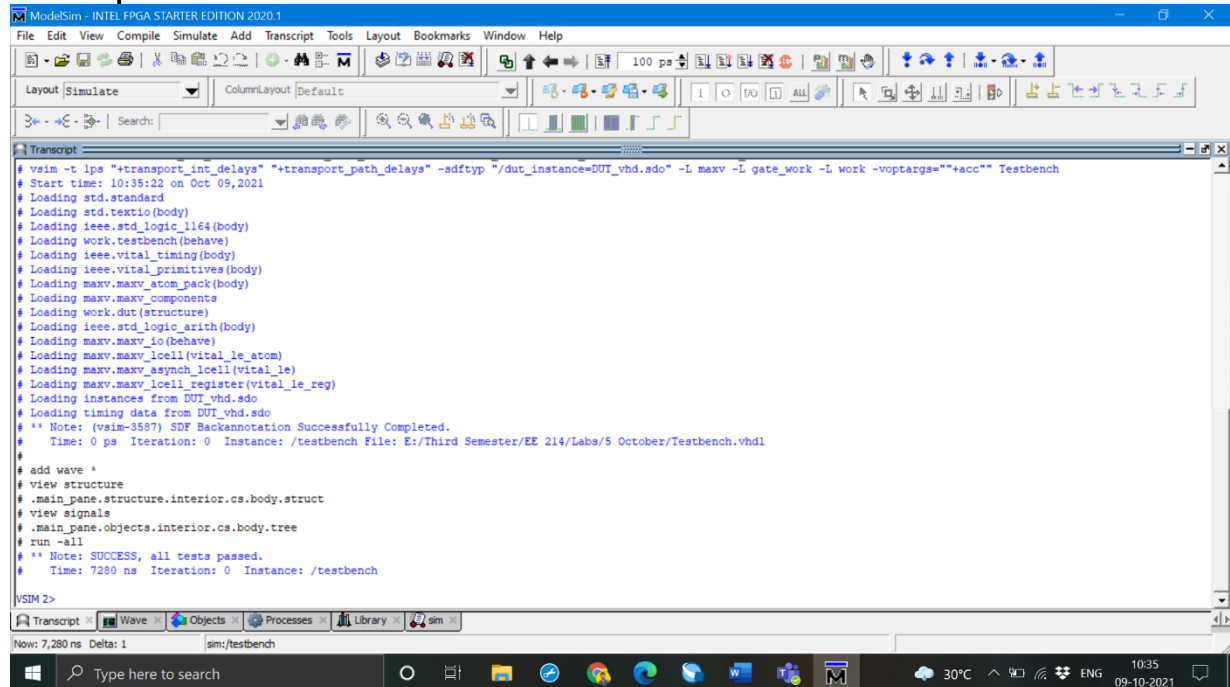
Now: 7,280 ns Delta: 1 sim:/testbench
```

### Waveform



## Gate-level Simulation:

### Transcript



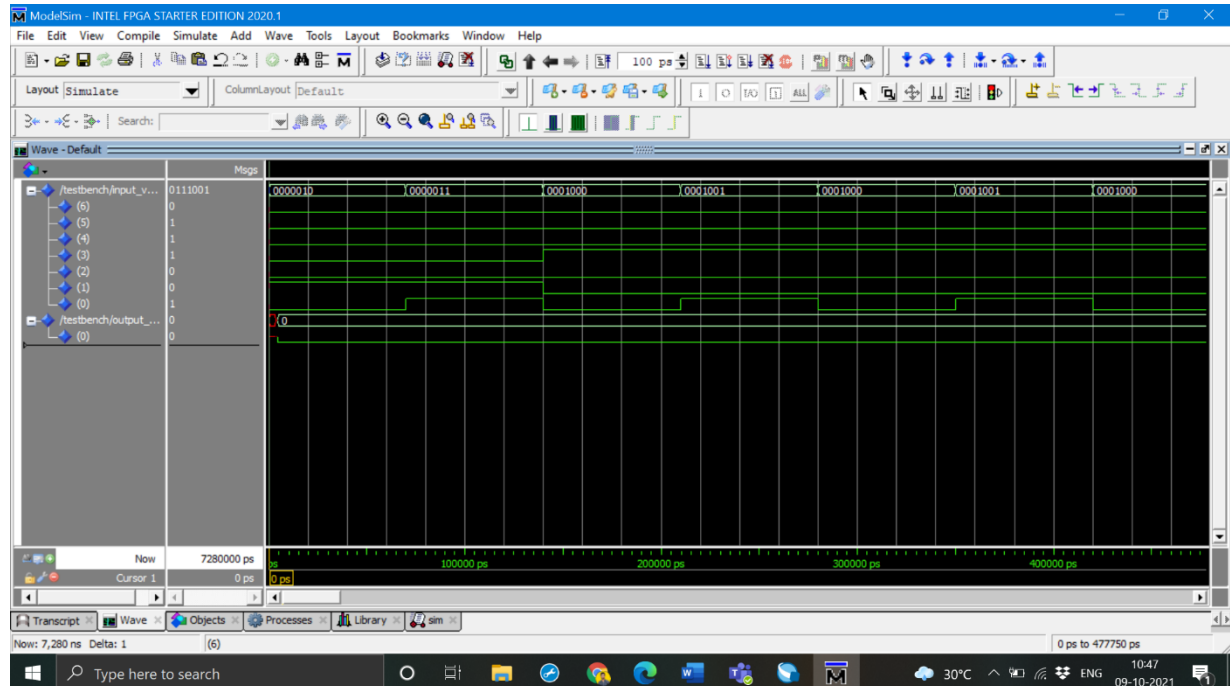
```
ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help

# vsim -t lps "+transport_int_delays" "+transport_path_delays" -sdftyp "/dut_instance=DUT_vhd.sdo" -L maxv -L gate_work -L work -voptargs="" +acc"" Testbench
# Start time: 10:35:22 on Oct 09, 2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behavior)
# Loading ieee.vital_timing(body)
# Loading ieee.vital_primitives(body)
# Loading maxv.maxv_atom_pack(body)
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behavior)
# Loading maxv.maxv_loell(vital_le_atom)
# Loading maxv.maxv_async_loell(vital_le)
# Loading maxv.maxv_loell_register(vital_le_reg)
# Loading instances from DUT_vhd.sdo
# Loading timing data from DUT_vhd.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#   Time: 0 ps Iteration: 0 Instance: /testbench File: E:/Third Semester/EE 214/Labs/5 October/Testbench.vhdl
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#   Time: 7280 ns Iteration: 0 Instance: /testbench

VSIM 2>
```

Now: 7,280 ns Delta: 1 sim:/testbench

### Waveform





## Krypton board:

We have used a scan chain for this experiment. So **out.txt** has an output which I got using scan chain.

```
E:\Third Semester\EE 214\UrJTAG\urjtag_windows\jtag.exe
UrJTAG 0.10 #1502
Copyright (c) 2002, 2003 ETC s.r.o.
Copyright (c) 2007, 2008, 2009 Kolja Waschk and the respective authors

UrJTAG is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
There is absolutely no warranty for UrJTAG.

WARNING: UrJTAG may damage your hardware!
Type "quit" to exit, "help" for help.

jtag> cable ft2232
Connected to libftd2xx driver.
jtag> detect
IR length: 10
Chain length: 1
Device Id: 00000010000010100011000011011101 (0x00000000020A30DD)
Manufacturer: Altera
Part(0): 5M1270
Stepping: 1
Filename: e:\third semester\ee 214\urjtag\urjtag_windows\data\altera\5m1270\5m1270
jtag> svf string_tue.svf progress
Warning svf: unimplemented mode 'ABSENT' for TRST
Parsing 40830/40835 ( 99%)
Scanned device output matched expected TDO values.
jtag>
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1266]
(c) Microsoft Corporation. All rights reserved.

E:\Third Semester\EE 214\Labs\5 October>scan_vjtag.exe TRACEFILE.txt out.txt
{'type': 6, 'id': 67330064, 'description': 'b'Dual RS232-HS A', 'serial': 'b'A'}

E:\Third Semester\EE 214\Labs\5 October>
```

**Some outputs from out.txt**

```
0001001 0 Success
0001000 0 Success
0001001 0 Success
0001100 0 Success
0001101 0 Success
0001100 0 Success
0001101 0 Success
0001100 0 Success
0001101 0 Success
0001100 0 Success
0001101 0 Success
0001100 0 Success
0001101 0 Success
0001100 0 Success
0001101 0 Success
0000100 0 Success
0000101 0 Success
0011000 0 Success
0011001 0 Success
```

**Observations:**

The main observation and learning outcome from this experiment was how to write the logic of a Mealy-type FSM. And also how to write different concurrent processes and how to assign them their task.

**References:**

My primary reference was our course webpage; it contained many valuable things, such as a sample code and many other specifications. It also had one handout for this experiment which was very helpful. This I used as my reference