WRIGHT STATE UNIVERSITY

EE4620/6620, CEG4324/6324
Digital Integrated Circuit Design
with PLDs and FPGAs

---

# Lab 2 (Spring 2024)

# Xilinx Vivado Sequential Circuit Design, Simulation, and Test in FPGA

---

## A. Objective

The main objective of this lab is for you to explore the design space for sequential circuit designs and to become familiar with the Xilinx Vivado simulation environment. Read all directions carefully. You will design a VHDL project in Xilinx Vivado and functionally verify and validate your design along with the implementation steps to upload the design through the Vivado simulation environment. Read through all of the steps before you begin the lab.

## B. Instruction

Explore the design space for sequential circuit designs by constructing four different sequence detectors using Finite State Machines (FSMs). The four different sequential design cases are listed:

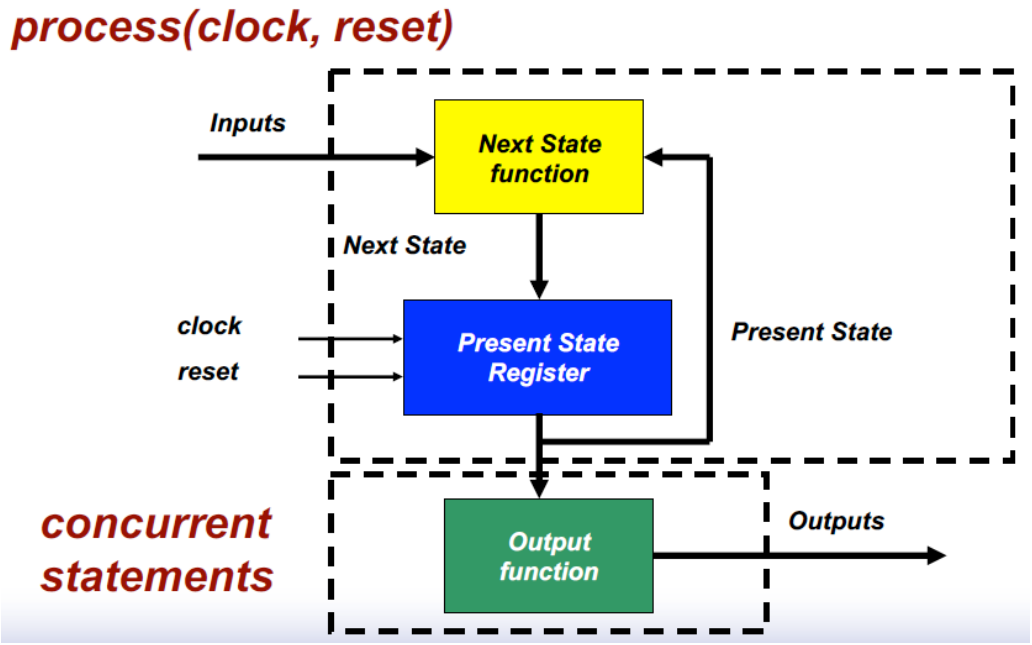
| | |
|---|---|
| **Case 1** | Moore machine to detect sequence '101' |
| **Case 2** | Mealy machine to detect sequence '101' |
| **Case 3** | Moore machine to detect sequence '0010' |
| **Case 4** | Mealy machine to detect sequence '0010' |

All flip-flops used in the FSM design have:

- Synchronous reset
- Clock has a higher priority than reset

1. Moore machine 101

Let's construct the sequence detector for sequence 101 using the Moore state machine. The Output of the State machine depends only on the present state. The output of the state machine is only updated at the clock edge.

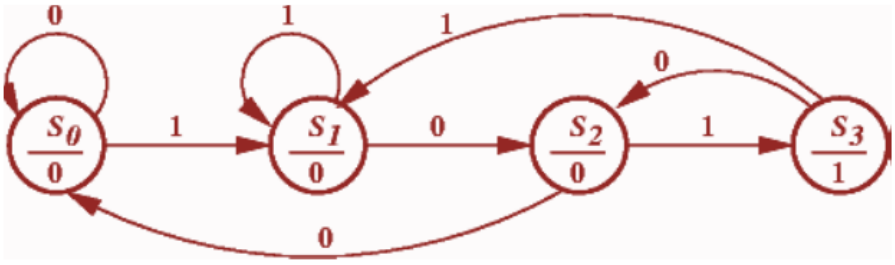



Moore state machine requires four states st0,st1,st2, st3 to detect the sequence 101. The state transition graph (STG) is shown below.

VHDL for Moore machine 101 is in Appendix A.

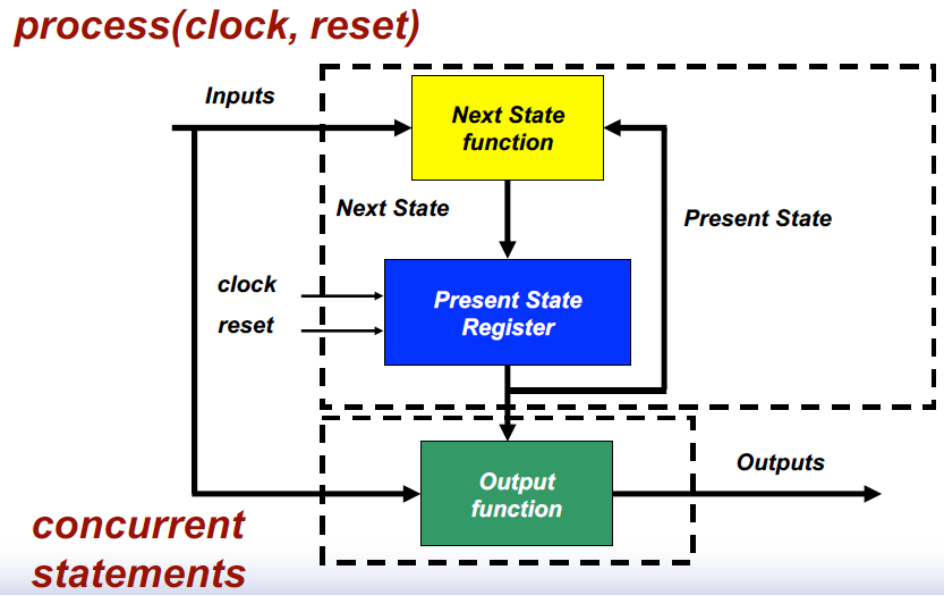
VHDL testbench for Moore machine 101 is in Appendix B in which the input bit sequence is "01010101"

**Note: You need to modify the testbench using the input bit sequence "01010001010010" for all 4 sequence detector cases.**

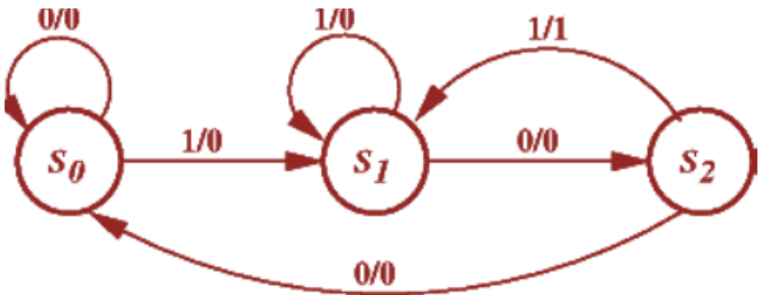



2. Mealy machine 101

Let's construct the sequence detector for sequence 101 using the Mealy state machine. The output of the state machine depends on both the present state and current input. When the input changes, the output of the state machine updated without waiting for a change in clock input.




Mealy state machine requires only three states st0,st1,st2 to detect the sequence 101. The state transition graph (STG) is shown below. The VHDL for Mealy machine 101 is shown in Appendix C.




3. Moore machine to detect sequence '0010'

Let's construct the sequence detector for *'0010'* using the Moore state machine. The output of the State machine depends only on the present state. The output of the state machine is only updated at the clock edge.

4. Mealy machine to detect sequence '0010'

Let's construct the sequence detector for *'001'* using the Mealy state machine. The output of the state machine depends on both the present state and current input. When the input changes, the output of the state machine updated without waiting for a change in clock input.

**C. Report [100 pts] (Submitted to your Lab Pilot Dropbox by 11:30 pm, Sunday, March 17, 2024)**

Turn in a written report that captures your approach to the 4 FSM cases: 1) *Moore machine 101*, 2) *Mealy machine 101*, 3) *Moore machine 0010*, and 4) *Mealy machine 0010.* It should include:

1. [20 pts] Case 1: *Moore machine 101*
   - [10 pts] The sequence detector VHDL code and testbench. Mark your verification in the output waveform showing a successful operation, using the bit sequence provided in the above instruction.
   - [10 pts] The post-synthesis logic design schematic and hardware report.
2. [20 pts] Case 2: *Mealy machine 101*
   - [10 pts] The sequence detector VHDL code and testbench. Mark your verification in the output waveform showing a successful operation, using the bit sequence provided in the above instruction.
   - [10 pts] The post-synthesis logic design schematic and hardware report.
3. [30 pts] Case 3: *Moore machine 0010*
   - [10 pts] The STG.
   - [10 pts] The sequence detector VHDL code and testbench. Mark your verification in the output waveform showing a successful operation, using the bit sequence provided in the above instruction.
   - [10 pts] The post-synthesis logic design schematic and hardware report.
4. [30 pts] Case 4: *Mealy machine 0010*
   - [10 pts] The STG.
   - [10 pts] The sequence detector VHDL code and testbench. Mark your verification in the output waveform showing a successful operation, using the bit sequence provided in the above instruction.
   - [10 pts] The post-synthesis logic design schematic and hardware report.

## Appendix

### A. VHDL for Moore machine 101

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity moore is
Port (   clk : in STD_LOGIC;
         din : in STD_LOGIC;
         rst : in STD_LOGIC;
         dout : out STD_LOGIC);
end moore;

architecture Behavioral of moore is
type state is (st0, st1, st2, st3);
signal present_state, next_state : state;

begin

next_state_decoder : process(present_state, din)
begin
        case (present_state) is
                when st0 =>
                        if (din = '1') then
                                next_state <= st1;
                        else
                                next_state <= st0;
                        end if;
                when st1 =>
                        if (din = '1') then
                                next_state <= st1;
                        else
                                next_state <= st2;
                        end if;
                when st2 =>
                        if (din = '1') then
                                next_state <= st3;
                        else
                                next_state <= st0;
                        end if;
                when st3 =>
                        if (din = '1') then
                                next_state <= st1;
                        else
                                next_state <= st2;
                        end if;
                when others =>
                        next_state <= st0;
        end case;
```

```
end process;

synchronous_process: process(clk, rst)
begin
-- Synchronous reset; clock has a higher priority than reset.
        if (clk = ‘1’ AND clk’event) then
                if (rst = '1') then
                        present_state <= st0;
                else
                        present_state <= next_state;
                end if;
        end if;
end process;

        dout <= ‘1’ WHEN present_state = st3 ELSE ‘0’;
end Behavioral;
```

**B. TestBench VHDL for Moore machine 101**

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_moore IS
END tb_moore;

ARCHITECTURE behavior OF tb_moore IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT moore
PORT(
        clk : IN std_logic;
        din : IN std_logic;
        rst : IN std_logic;
        dout : OUT std_logic
        );
END COMPONENT;

--Inputs
signal clk : std_logic := '0';
signal din : std_logic := '0';
signal rst : std_logic := '0';

--Outputs
signal dout : std_logic;

-- Clock period definitions
constant clk_period : time := 20 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)
uut: moore PORT MAP (clk => clk,
                     din => din,
                     rst => rst,
                     dout => dout);
-- Clock process definitions
clk_process :process
begin
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end process;

-- Stimulus process
stim_proc: process
```

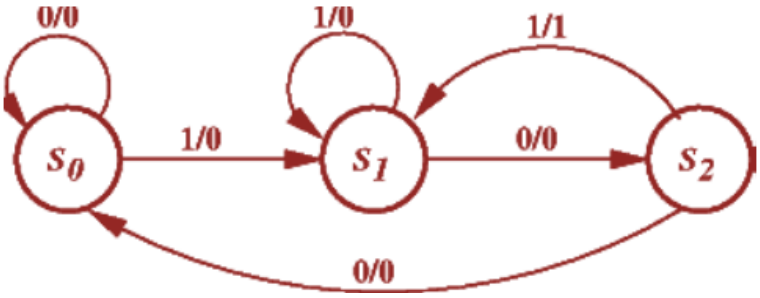
```
begin
rst <= '1';
wait for 100 ns;
-- input sequence “01010101”
rst <= '0';
din <= '0';
wait for 20 ns;
din <= '1';
wait for 20 ns;
din <= '0';
wait for 20 ns;
din <= '1';
wait for 20 ns;
….
….
….
wait for 20 ns;
end process;
END;
```

## C. VHDL for Mealy machine 101

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mealy is
Port (   clk : in STD_LOGIC;
         din : in STD_LOGIC;
         rst : in STD_LOGIC;
         dout : out STD_LOGIC);
end mealy;
architecture Behavioral of mealy is
type state is (st0, st1, st2);
signal present_state, next_state : state;

begin
```




```
next_state_decoder : process(present_state, din)
begin
        case (present_state) is
                when st0 =>
                        if (din = '1') then
                                next_state <= st1;
                                d_out <= '0';
                        else
                                next_state <= st0;
                                d_out <= '0';
                        end if;
                when st1 =>
                        if (din = '1') then
                                next_state <= st1;
                                d_out <= '0';
                        else
                                next_state <= st2;
                                d_out <= '0';
                        end if;
                when st2 =>
                        if (din = '1') then
                                next_state <= st1;
                                d_out <= '1';
                        else
```

```
                                    next_state <= st0;
                                    d_out <= '0';
                            end if;
                    when others =>
                                    next_state <= st0;
                                    d_out <= '0';
        end case;
end process;

synchronous_process: process(clk)
begin
        if (clk = '1' AND clk'event) then
                if (rst = '1') then
                        present_state <= st0;
                else
                        present_state <= next_state;
                end if;
        end if;
end process;

end Behavioral;
```