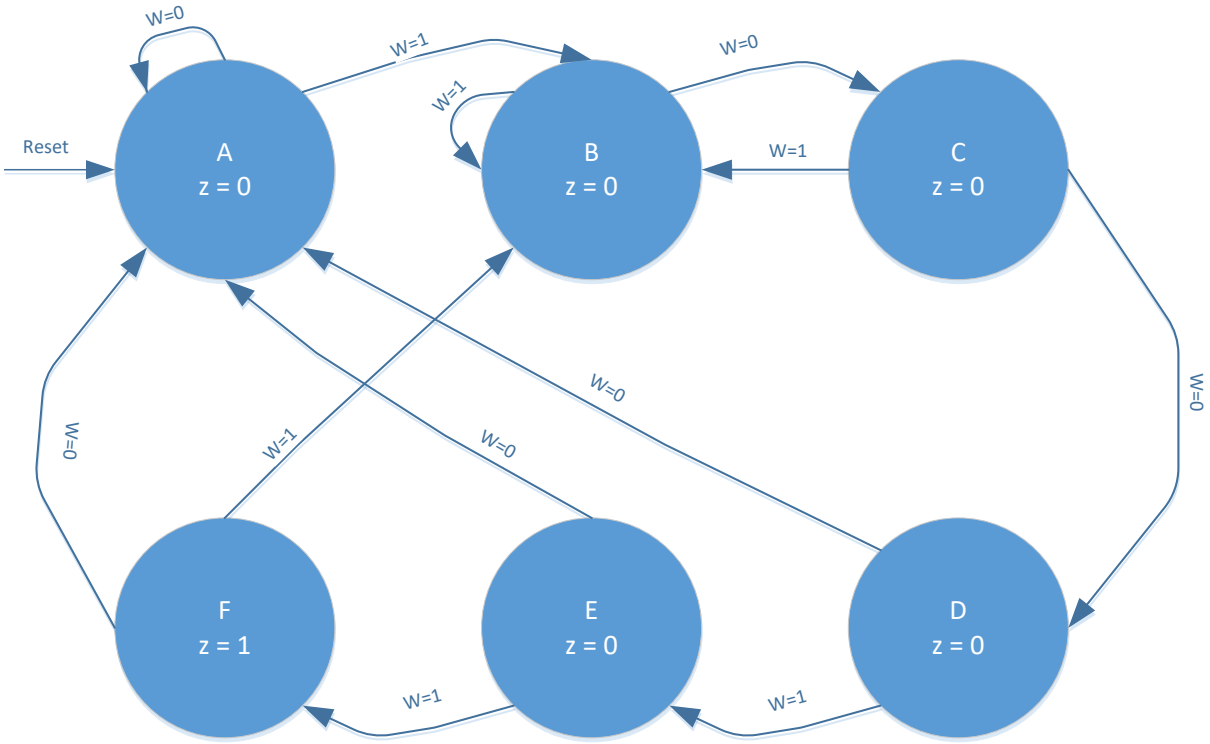
Scarlett Jones
CSCE 3730
Homework #5
Due 11/22/2016, 11:55 PM

1. Design a Moore FSM (only state diagram and state table) that can detect the sequence 1, 0, 0, 1, 1. Write a behavioral VHDL code that implements the FSM. Write a VHDL test bench to test the FSM using ten 1-bit test inputs.
    a. State diagram



    b. State table

Present State	Present State $Q_2Q_1Q_0$	Input W	Next State	Next State $q_2q_1q_0$	D_2	D_1	D_0	Output z
A	000	0	A	000	0	0	W	0
		1	B	001				
B	001	0	C	010	0	$\overline{W}$	W	0
		1	B	001				
C	010	0	D	011	0	$\overline{W}$	1	0
		1	B	001				
D	011	0	A	000	W	0	0	0
		1	E	100				
E	100	0	A	000	W	0	W	0
		1	E	101				
F	101	0	A	000	0	0	W	1
		1	B	001				
X	110	X	X	X	X	X	X	X
X	111	X	X	X	X	X	X	X

Scarlett Jones
CSCE 3730
Homework #5
Due 11/22/2016, 11:55 PM

c. VHDL code

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity moore is
Port ( Clock, Reset, w : in std_logic;
        z : out std_logic );
End moore;

Architecture Behavioral of moore is
        Type State_Type Is (A, B, C, D, E, F);
        Signal y: State_Type;
begin
        Process (Clock, Reset)
        Begin
                If Reset = '0' Then
                        y <= A;
                Elsif (Clock'Event AND Clock = '1') then
                        CASE y IS
                        When A =>
                                IF w = '0' then
                                        y <= A;
                                Else
                                        y <= B;
                                End If;
                        When B =>
                                IF w = '0' then
                                        y <= C;
                                Else
                                        y <= B;
                                End If;
                        When C =>
                                IF w = '0' then
                                        y <= D;
                                Else
                                        y <= B;
                                End If;
                        When D =>
                                IF w=’0’ then
                                        y <= A;
                                Else
                                        y <= E;
```

```
                End If;
            When E =>
                IF w = '0' then
                    y <= A;
                Else
                    y <= F;
                End If;
            When F =>
                IF w='0' then
                    y <= A;
                Else
                    y <= B;
                End If;

            End CASE;
        End If;
    End Process;
z <= '1' When y = F Else '0';
End Behavioral;
```

d. VHDL test bench

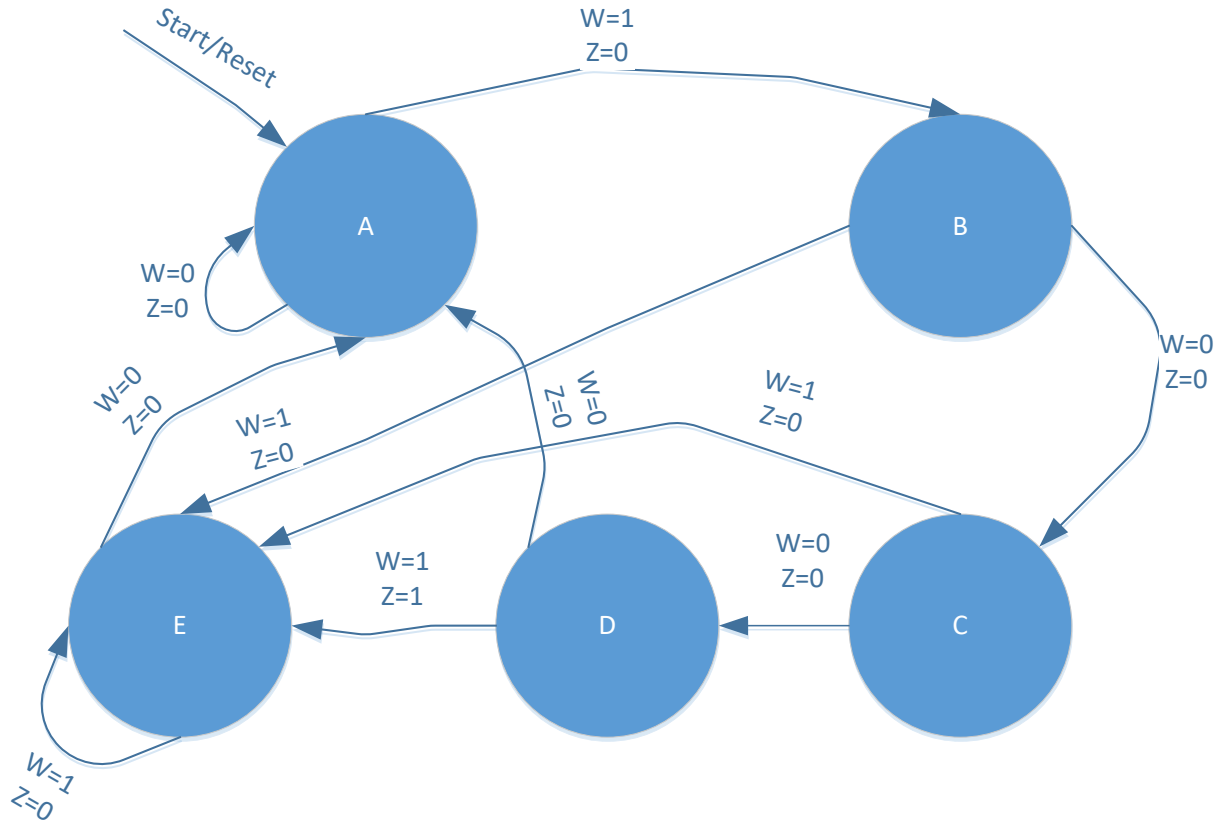
```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

USE ieee.numeric_std.ALL;

LIBRARY UNISIM;

USE UNISIM.Vcomponents.ALL;


ENTITY moore_tb IS

END moore_tb;


ARCHITECTURE behavioral OF moore_tb IS

    COMPONENT moore

    PORT( Clock, Reset, w : IN STD_LOGIC;

        z : OUT STD_LOGIC);

    END COMPONENT;
```

```
        SIGNAL Clock, Reset, w, z : STD_LOGIC;
        Constant Clk_period : time := 10 ns;
BEGIN
UUT: moore PORT MAP(
                        Clock=>Clock,
                        Reset => Reset,
                        w => w,
                        z => z);
--clock process definitions
Clk_process :process
 begin
                Clock <= '0';
                wait for Clk_period/2;
                Clock <= '1';
                wait for Clk_period/2;
  end process;

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
--hold reset state for 100 ns.
Wait for 100 ns;
Wait for Clk_period*10;

--insert stimulus here
w<='0';
wait for 10 ms;
w<='0';
```

```
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='0';
wait for 10 ms;
w <='0';
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='0';
wait for 10 ms;
WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
END behavioral;
```

2. Design a Mealy FSM (only state diagram and state table) that can detect the sequence 0, 1, 0, 0, 1. Write a behavioral VHDL code that implements the FSM. Write a VHDL test bench to test the FSM using ten 1-bit test inputs.
    a. State diagram

Scarlett Jones
CSCE 3730
Homework #5
Due 11/22/2016, 11:55 PM



b. State table

Present State	Present State $Q_2Q_1Q_0$	Input W	Next State	Next State $q_2q_1q_0$	D_2	D_1	D_0	Output z
A	000	0	A	000	0	0	W	0
		1	B	001				0
B	001	0	C	010	W	$\bar{W}$	0	0
		1	E	100				0
C	010	0	D	011	W	$\bar{W}$	$\bar{W}$	0
		1	E	100				0
D	011	0	A	000	W	0	0	0
		1	E	100				1
E	100	0	A	000	W	0	0	0
		1	E	100				0
X	101	X	X	X	X	X	X	X
X	110	X	X	X	X	X	X	X
X	111	X	X	X	X	X	X	X

c. VHDL code

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity mealy is
Port ( Clock, Reset, w : in std_logic;
        z : out std_logic );
End mealy;

Architecture Behavioral of mealy is
        Type State_Type Is (A, B, C, D, E);
        Signal y: State_Type;
begin
        Process (Clock, Reset)
        Begin
                If Reset = '0' Then
                        y <= A;
                Elsif (Clock'Event AND Clock = '1') then
                        CASE y IS
                        When A =>
                                IF w = '0' then
                                        y <= A;
                                Else
                                        y <= B;
                                End If;
                        When B =>
                                IF w = '0' then
                                        y <= C;
                                Else
                                        y <= E;
                                End If;
                        When C =>
                                IF w = '0' then
                                        y <= D;
                                Else
                                        y <= E;
                                End If;
                        When D =>
                                IF w=’0’ then
                                        y <= A;
                                Else
                                        y <= E;
                                End If;
                        When E =>
```

```
                        IF w = '0' then
                                y <= A;
                        Else
                                y <= E;
                        End If;
                End CASE;
        End If;
    End Process;

    Process(y, w)
    Begin
        CASE y IS
            WHEN A =>
                z<='0';
            WHEN B =>
                z<='0';
            WHEN C =>
                z<='0';
            WHEN D =>
                z<=w;
            WHEN E =>
                z<='0';
        END CASE;
    END PROCESS;
End Behavioral;
```

d. VHDL test bench

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

USE ieee.numeric_std.ALL;

LIBRARY UNISIM;

USE UNISIM.Vcomponents.ALL;


ENTITY mealy_tb IS

END mealy_tb;


ARCHITECTURE behavioral OF mealy_tb IS

    COMPONENT mealy
```

```
        PORT( Clock, Reset, w : IN STD_LOGIC;
                z : OUT STD_LOGIC);
        END COMPONENT;

        SIGNAL Clock, Reset, w, z : STD_LOGIC;
        Constant Clk_period : time := 10 ns;
BEGIN
UUT: mealy PORT MAP(
                        Clock=>Clock,
                        Reset => Reset,
                        w => w,
                        z => z);
--clock process definitions
Clk_process :process
 begin
                Clock <= '0';
                wait for Clk_period/2;
                Clock <= '1';
                wait for Clk_period/2;
  end process;

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
--hold reset state for 100 ns.
Wait for 100 ns;
Wait for Clk_period*10;
```

```
--insert stimulus here
w<='0';
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='0';
wait for 10 ms;
w<='0';
wait for 10 ms;
w<='1';
wait for 10 ms;
w <='1';
wait for 10 ms;
w<='0';
wait for 10 ms;
w<='0';
wait for 10 ms;
w<='1';
wait for 10 ms;
w<='0';
wait for 10 ms;
WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
END behavioral;
```

3. What is System on a Chip (SoC)? What are its benefits?

   a. A system on a chip is an integrated circuit that integrates all of the components of an electronic system into a single chip. You can add whatever blocks your system calls for and connect them through a bus.
   b. The benefits are low power consumption and low cost. You can design any type of system on a chip that you need. If it is mass produced, it will be pretty cheap. If you have it custom designed, it will cost more.
4. Explain the three techniques used for bus interfaces?
   a. Synchronous logic: a global clock is used as a reference.
   b. Asynchronous logic: there is no global clock. Distributes timing information with values.
   c. Handshaking-oriented protocols: a handshake between two sides transmits data. There are values, control messages, and ack messages sent between two sides.
5. Differentiate (at least two differences) between a bus and a bridge?
   a. Bus
      i. Connects different components in a computer system together and sends data between these components.
      ii. Can be synchronous or asynchronous design.
   b. Bridge
      i. Connects two different bus types so that they can interface with each other.
      ii. Is synchronous design.
6. Discuss at least four Xilinx specific busses.
   a. Processor Local Bus (PLB)
      i. Can be 32 bits or 64 bits.
      ii. Is used to connect sub systems.
   b. Local Memory Bus (LMB)
      i. This is a fast bus.
      ii. It is used by Microblaze (CPU) data and instruction ports.
   c. Advanced eXtensible Interface 4 (AXI4)
      i. Designed to enhance the performance and utilization of the interconnect when used by multiple masters.
      ii. Has support for burst lengths of up to 256 beats.
      iii. Has quality of service signaling.
      iv. Has support for multiple region interfaces.
   d. AXI4-Lite
      i. This is a subset of the AXI4 protocol. It is intended for communication with simpler and smaller control register style interfaces in components.
      ii. All transactions have a burst length of one.
      iii. All data accesses are the same size as the width of the data.
      iv. Exclusive accesses are not supported.
   e. AXI4-Stream
      i. This is designed for unidirectional data transfers from master to slave with greatly reduced signal routing.
      ii. Has support for single and multiple data streams using the same set of shared wires.

   iii. Has support for multiple data widths within the same interconnect.
   iv. It is ideal for implementation in FPGAs.

7. Differentiate (at least two differences) between a block RAM and distributed RAM?
   a. Block RAM
      i. Block RAMs an be used as simple memory elements.
      ii. Available as IP resource on the FPGA.
      iii. Can be configured as single or dual port.
      iv. Hard IP memory.
   b. Distributed RAM
      i. LUTs can be used as simple memory elements.
      ii. LUTs can be combined to create a larger RAM.
      iii. Can be configured as single or dual port.
      iv. Soft IP memory.
8. What is UART? What does it do? Briefly explain the register module and the control module of Xilinx's UART Lite peripheral.
   a. UART stands for Universal Asynchronous Receiver Transmitter. This is a soft IP core designed to interface with the AXI4-Lite protocol.
   b. UART connects to the AXI through a bus and provides the controller interface for data transfer.
   c. The interface module provides the interface to the AXI and implements AXI protocol logic. The AXI interface module is a bidirectional interface between a user IP core and the AXI4-Lite interface standard.
   d. The register module interfaces to the AXI through the AXI interface module. It contains a status register, a control register, and a pair of transmit/receive FIFOs (both of 16-character depth). All registers are accessed directly from the AXI using the AXI interface module.
   e. The control module contains an RX module, a TX module, a parameterized baud rate generator (BRG), and a control unit. This module also contains the logic to generate the interrupts.