1. A Moore model finite state machine that acts as a “1011” sequence detector is to be designed using behavioral VHDL. Your design should detect overlapping sequences. Assume the input is named \( A \), the output is named \( Z \) and that an active low reset signal (\( \text{reset}_n \)) asynchronously resets the machine. The VHDL ENTITY construct is given. Write the corresponding VHDL ARCHITECTURE construct to implement the FSM. Implement positive edge triggered flip-flops.

(a) Draw the Moore model state diagram for the FSM.

![State Diagram](image)

(b) Write the VHDL ARCHITECTURE construct to implement the FSM.

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY seqdetect_fsm IS
PORT (A  : IN STD_LOGIC;
      Clk : IN STD_LOGIC;
      reset_n : IN STD_LOGIC;
      Z   : OUT STD_LOGIC);
END seqdetect_fsm;

ARCHITECTURE Behavior OF seqdetect_fsm IS
  TYPE State_type IS (S0, S1, S2, S3, S4);
  SIGNAL y: State_type;
BEGIN
  PROCESS (reset_n,Clk)
  BEGIN
    IF reset_n = '0' THEN
      y <= S0;
    ELSIF RISING_EDGE(Clk) THEN
      CASE y IS
      WHEN S0 =>
        IF A='0' THEN
          y <= S0;
        ELSE
          y <= S1;
      WHEN S1 =>
        IF A='0' THEN
          y <= S1;
        ELSE
          y <= S2;
      WHEN S2 =>
        IF A='0' THEN
          y <= S2;
        ELSE
          y <= S3;
      WHEN S3 =>
        IF A='0' THEN
          y <= S3;
        ELSE
          y <= S4;
      WHEN S4 =>
        IF A='0' THEN
          y <= S4;
        ELSE
          y <= S0;
      END CASE;
    END IF;
  END PROCESS;
END Behavior;
```
2. Write a behavioral VHDL description for a 4-bit shift register. The shift register is to be negative edge triggered. Sin is a serial input to the most significant bit of the shift register. Sout is a serial output from the least significant bit of the shift register. En_n is an active low enable. sreg is the 4-bit register. Write only the VHDL ARCHITECTURE construct. The VHDL ENTITY construct is given below.

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY shiftreg is
PORT (  CLK  : IN STD_LOGIC;
   En_n  : IN STD_LOGIC;
      Sin  : IN STD_LOGIC;
      Sout  : OUT STD_LOGIC);
END shiftreg;

ARCHITECTURE behavior OF shiftreg IS
   SIGNAL sreg  : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN

   PROCESS(clk)
   BEGIN
      IF FALLING_EDGE(CLK) THEN
         IF (En_n='0') THEN
            Sout <= sreg(0);
            sreg <= Sin & sreg(3 DOWNTO 1);
         END IF;
      END IF;
   END PROCESS;

   Z <= '1' WHEN y=S4 ELSE '0';
END Behavior;
3. A given FPGA uses 2-input lookup tables to implement Boolean functions. Show a section of a programmed FPGA that implements the function \( f(x,y,z) = \sum m(0,1,2,5,7) \).

\[
f(x,y,z) = \sum m(0,1,2,5,7) = \bar{x}y + xz + \bar{x}z\]

4. Draw a diagram for typical output circuitry in a PLA. Describe each component.
The D flip-flop provides storage capability for the PLA block, allowing sequential circuits to be implemented. The multiplexer selects either registered or non-registered outputs from the PLA. The tri-state gate allows the output to have tri-state capabilities. Feedback into the programmable AND plane is provided as well.

5. What is the purpose of a sensitivity list in a VHDL PROCESS block?

The list contains all signals that can be used to activate the process.
6. Give the VHDL statement to implement the logic function in problem 4 using a selected assignment statement. Define any signals used in the statement.

```vhdl
SIGNAL xyz : STD_LOGIC_VECTOR(2 DOWNTO 0);

xyz <= x & y & z;
WITH xyz SELECT
  f <= '1' WHEN "000",
   '1' WHEN "001",
   '1' WHEN "010",
   '1' WHEN "101",
   '1' WHEN "111",
   '0' WHEN OTHERS;
```

7. Write a behavioral VHDL description for a 2-to-1 multiplexer. Model the multiplexer as a process block. Write only the VHDL ARCHITECTURE construct. The VHDL ENTITY construct is given below.

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux2to1 IS
  PORT ( w : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
        s : IN STD_LOGIC;
        f : OUT STD_LOGIC );
END mux2to1;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
  PROCESS (w,s)
  BEGIN
    IF s = '0' THEN
      f <= w(0);
    ELSE
      f <= w(1);
    END IF;
  END PROCESS;
END Behavior;
```