Lab 7: VHDL 16-Bit Shifter

Objectives:
Design a 16-bit shifter which needs to implement eight shift operations: logic shift right, logic shift left, arithmetic shift right, arithmetic shift left, rotate right, rotate left, rotate left with carry, rotate left with carry.

16-Bit Shifter specification:
<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>Shift Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Logic Shift</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Arithmetic shift</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Rotate</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Rotate with carry</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S_3$</th>
<th>$S_2$</th>
<th>Shift direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Parallel Load</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Shift Right with $I_L$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Shift Left with $I_R$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>No Change</td>
</tr>
</tbody>
</table>

In the above logic diagram, for the unit “16-bit basic shifter” design you may refer to the available code for the Universal Shift Register with parallel load (three different style codes are given).

**Universal Shifter Register with parallel Load**

![Figure 3. Universal Shift Register Symbol](image)

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>Next State after rising edge of clk</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$Q[3..0] &lt;= Q[3..0]$</td>
<td>No Change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$Q[3..0] &lt;= I[3..0]$</td>
<td>Parallel Load</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$Q[3..1] &lt;= Q[2..0], Q[0] &lt;= I_R$</td>
<td>Shift Left with $I_R$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$Q[3] &lt;= I_L, Q[2..0] &lt;= Q[3..1]$</td>
<td>Shift Right with $I_L$</td>
</tr>
</tbody>
</table>

1. Universal shift register using array slicing and concatenation method.

**LISTING 1: Save as Filename=“univ_shiftreg1.vhd”**

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.NUMERIC_BIT.ALL;
```
ENTITY univ_shiftreg1 IS
  PORT(clk, il, ir : IN BIT;
       s: IN BIT_VECTOR(1 DOWNTO 0);
       i : IN BIT_VECTOR(3 DOWNTO 0);
       q : OUT BIT_VECTOR(3 DOWNTO 0));
END univ_shiftreg1;

ARCHITECTURE beh1 OF univ_shiftreg1 IS
  SIGNAL qtmp : BIT_VECTOR(3 DOWNTO 0);
BEGIN
  PROCESS(clk)
  BEGIN
    IF (clk = '1' AND clk'EVENT) THEN
      CASE s IS
        WHEN "00" => qtmp <= qtmp;
        WHEN "01" => qtmp <= i;
        WHEN "10" => qtmp<=qtmp(2 DOWNTO 0) & ir;
        WHEN "11" => qtmp<=il & qtmp(3 DOWNTO 1);
        WHEN OTHERS => NULL;
      END CASE;
    END IF;
  END PROCESS;
  q <= qtmp;
END beh1;


LISTING 2: Save as Filename="univ_shiftreg2.vhd"

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_bit.all;

entity univ_shiftreg2 is
  port(clk,il,ir :in bit;
       s :in bit_vector(1 downto 0);
       i : in bit_vector(3 downto 0);
       q : out bit_vector(3 downto 0));
end univ_shiftreg2;

architecture beh2 of univ_shiftreg2 is
  signal qtmp: bit_vector(5 downto 0);
BEGIN
  process(clk)
  begin
    if (clk = '1' and clk'EVENT) then
      case s is
        when "00" =>
          qtmp <= qtmp;
      when "01" =>
          qtmp <= i;
      when "10" =>
          qtmp<=qtmp(2 DOWNTO 0) & ir;
      when "11" =>
          qtmp<=il & qtmp(3 DOWNTO 1);
      when OTHERS => NULL;
    end case;
  end if;
  END PROCESS;
  q <= qtmp;
END beh2;
when "01" =>
qtmp <= i & i & ir;
when "10" =>
qtmp <= (i & qtmp(4 downto 1) & ir) sll 1;
when "11" =>
qtmp <= (i & qtmp(4 downto 1) & ir) srl 1;
when others =>
nul;
end case;
end if;
end process;
q <= qtmp(4 downto 1);
end beh2;


![Universal Shift Register Implementation](image)

<table>
<thead>
<tr>
<th>MUX Sel</th>
<th>NEXT STATE after rising_edge of clk</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 S0</td>
<td>Q3 Q2 Q1 Q0</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Q3 Q2 Q1 Q0</td>
<td>No Change</td>
</tr>
<tr>
<td>01</td>
<td>I3 I2 I1 I0</td>
<td>Parallel Load</td>
</tr>
<tr>
<td>10</td>
<td>Q2 Q1 Q0 IR</td>
<td>Shift left</td>
</tr>
<tr>
<td>11</td>
<td>I2 I1 Q0 Q1</td>
<td>Shift right</td>
</tr>
</tbody>
</table>

Write a vhdl code to model (entity and architecture) a four to one multiplexer called "MUX41".

LISTING 3: Save as Filename="univ_shiftreg3.vhd"
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX41 IS
    PORT (i3, i2, i1, i0 : IN BIT;
    ...
s: IN BIT_VECTOR(1 DOWNTO 0);
o: OUT BIT);
END MUX41;
ARCHITECTURE arch_mux41 OF MUX41 IS
BEGIN
  PROCESS(i3, i2, i1, i0, s)
  BEGIN
    CASE s IS
    WHEN "00" => o <= i0;
    WHEN "01" => o <= i1;
    WHEN "10" => o <= i2;
    WHEN "11" => o <= i3;
    WHEN OTHERS => NULL;
    END CASE;
  END PROCESS;
END arch_mux41

Write a vhdl code to model (entity and architecture) d flip-flop called “DFF”.
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY DFF IS
  PORT(d, clk : IN BIT;
       q, qb : OUT BIT);
END DFF;
ARCHITECTURE arch_dff OF DFF IS
BEGIN
  PROCESS(clk)
  VARIABLE q_temp : BIT;
  BEGIN
    IF(clk'EVENT AND clk='1')THEN
      q_temp := d;
    END IF;
    q <= q_temp;
    qb <= NOT q_temp;
  END PROCESS;
END arch_dff;

Using DFF and MUX41 components, generates the structural model of the “univ_shiftreg”. That is, show the port map for each of the components shown in Fig 2.

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY univ_shiftreg3 IS
  PORT(clk, il, ir : IN BIT;
       s: IN BIT_VECTOR(1 DOWNTO 0);
       i : IN BIT_VECTOR(3 DOWNTO 0);
       q : OUT BIT_VECTOR(3 DOWNTO 0));
END univ_shiftreg3;
ARCHITECTURE struct OF univ_shiftreg3 IS
COMPONENT MUX41
  PORT (i3, i2, i1, i0 : IN BIT;
        s: IN BIT_VECTOR(1 DOWNTO 0);
        o: OUT BIT);
END COMPONENT;
COMPONENT DFF
  PORT(d, clk : IN BIT;
q, qb : OUT BIT);
END COMPONENT;
FOR U1, U2, U3, U4: MUX41 USE ENTITY WORK.MUX41(arch_mux41);
FOR U5, U6, U7, U8: DFF USE ENTITY WORK.DFF(arch_dff);

SIGNAL o: BIT_VECTOR(3 DOWNTO 0);
SIGNAL qb: BIT_VECTOR(3 DOWNTO 0);
SIGNAL qt:BIT_VECTOR(3 DOWNTO 0);
BEGIN
  U1:MUX41 PORT MAP(il,qt(2), i(3), qt(3), s, o(3));
  U2:MUX41 PORT MAP(qt(3), qt(1), i(2), qt(2), s, o(2));
  U3:MUX41 PORT MAP(qt(2), qt(0), i(1), qt(1), s, o(1));
  U4:MUX41 PORT MAP(qt(1), ir, i(0), qt(0), s, o(0));
  U5:DFF PORT MAP(o(3), clk, qt(3), qb(3));
  U6:DFF PORT MAP(o(2), clk, qt(2), qb(2));
  U7:DFF PORT MAP(o(1), clk, qt(1), qb(1));
  U8:DFF PORT MAP(o(0), clk, qt(0), qb(0));
  q <= qt;
END struct;
---------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------
Combinatorial Implementation of Basic 4-bit Shifter*

A combinatorial implementation of basic 4 bit shifter is given in Fig 5. Please expand it to 16 bit shifter and use it in the implementation of Figure 2 instead of universal shift register. (S is 2 bit selection signal, S = S_1S_0)

*NOTE: This is 4-bit shifter implementation to be used in the project