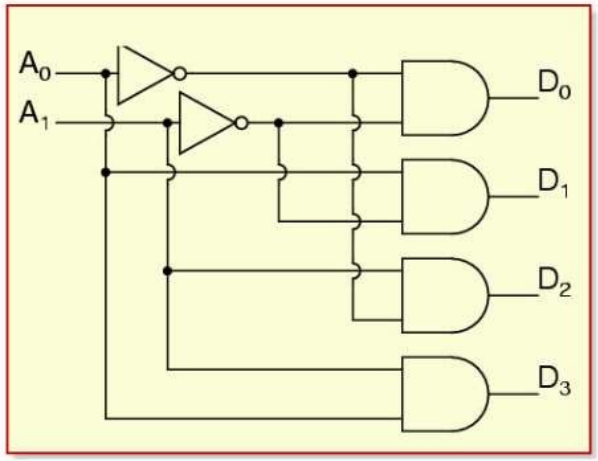
AIM: To simulate and synthesize 2 to 4 decoder using if statement.

SOFTWARE REQUIRED: XILINX VIVADO 2020.1

THEORY:

A decoder is a multiple input, multiple output logic circuit that changes codes inputs into coded outputs, where both the inputs and outputs are dissimilar for instance n-to-2n, and binary coded decimal decoders. Decoding is essential in applications like data multiplexing, memory address decoding, and 7 segment display. The best example of decoder circuit would be an AND-gate because when all its inputs are "High.", the output of this gate is "High" which is called "active High output". As an alternative to AND gate, the NAND gate is connected the output will be "Low" (0) only when all its inputs are "High". Such o/p is called "active low output".



2-to-4-Decoder Circuit

As a decoder, this circuit takes an n-bit binary number and generates an output on one of the 2n output lines. It is therefore usually described by the number of addressing input lines & the number of data o/p lines. In this type of decoders, decoders have two inputs namely A0, A1, and four outputs denoted by D0, D1, D2, and D3. As you can see in the following truth table – for every input combination, one o/p line is turned on.

A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

2-to-4-Decoder Truth Table

D0 =A1 A0, (minterm m0) which corresponds to input 00 D1 =A1 A0, (minterm m1) which corresponds to input 01 D2 =A1 A0, (minterm m2) which corresponds to input 10 D3 =A1 A0, (minterm m3) which corresponds to input 11.

PROCEDURE:

- Create New Project from File Menu.
- Give the file name.
- Select RTL project.
- Select VHDL language and mixed simulator. Then click next.
- Select add or create design sources. Then click next.
- Give port declarations as we do in entity. Then click Next. And then Finish.
- VHDL Code is presented on the screen. Now we just have to write the code in architecture.
- Save and run simulation
- Now we must give the values to the inputs using force constant.
- Select the type (hexadecimal, binary, etc.), force value, starting after time offset and cancel after time offset.

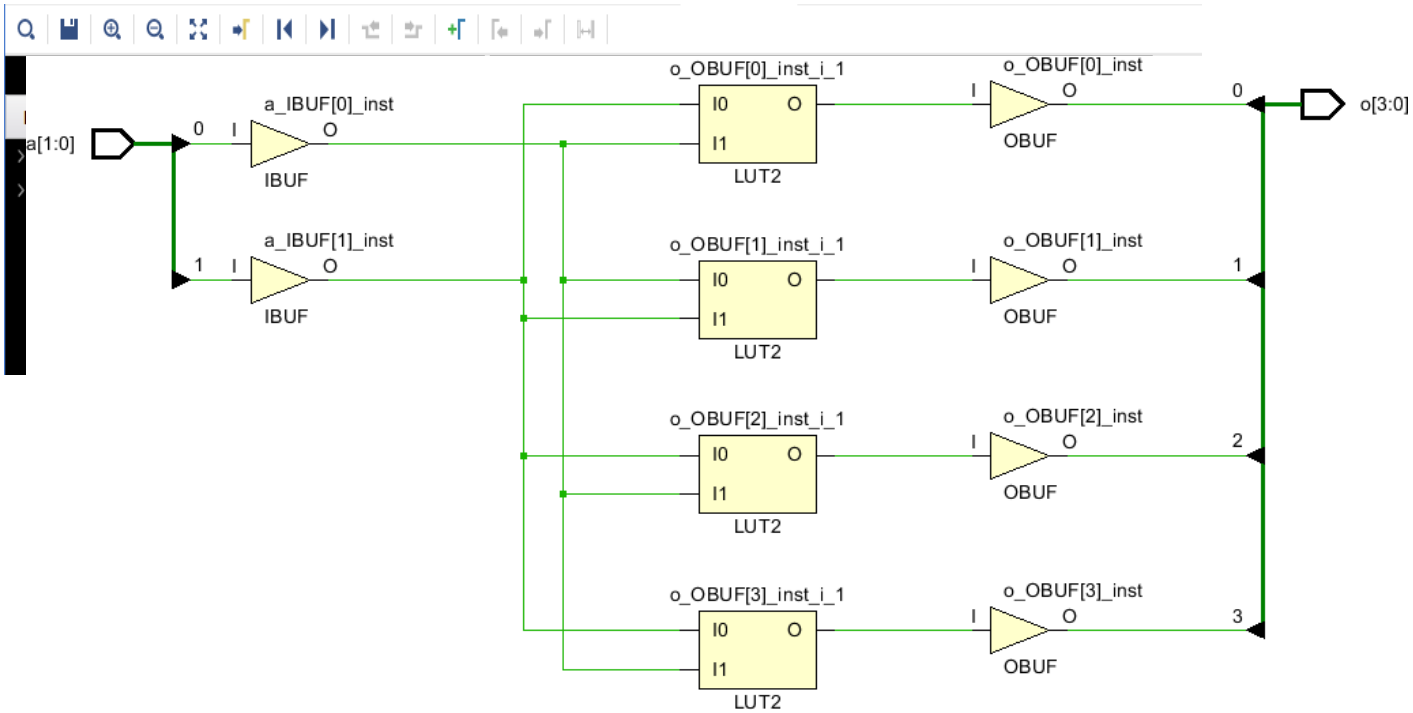
- Analyze the waveform.
- Synthesize it and do the RTL analysis.

Code

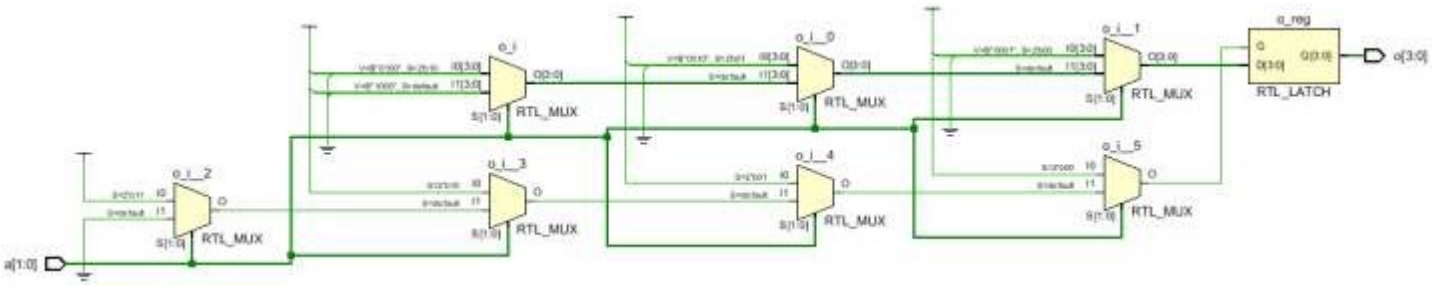
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dec is
    Port ( a : in STD_LOGIC_VECTOR (1 downto 0);
           o : out STD_LOGIC_VECTOR (3 downto 0));
end dec;

architecture Behavioral of dec is
begin
    process(a)
    begin
        if (a = "00") then
        o <= "0001";
        elsif (a="01") then
        o <= "0010";
        elsif (a="10") then
        o <= "0100";
        elsif (a="11") then
        o <= "1000";
        end if;
    end process;

end Behavioral;
```

Simulation

Technology View



RTL View



Result

The 2 to 4 decoder has been simulated and synthesized successfully.

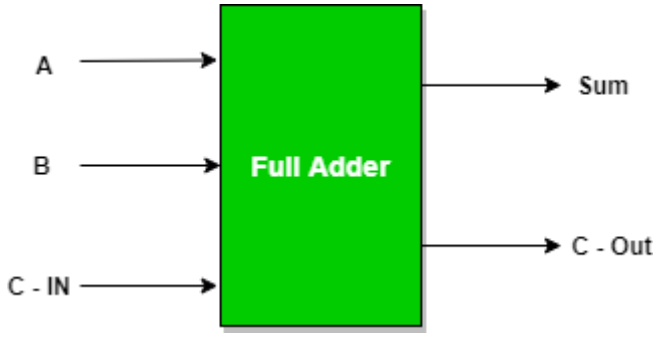
AIM: To simulate and synthesize full adder using structural modeling.

SOFTWARE REQUIRED: XILINX VIVADO 2020.1

THEORY:

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C- OUT and the normal output is designated as S which is SUM.

A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to the another.



Full Adder Truth Table:

Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Logical Expression for SUM:

= A' B' C-IN + A' B C-IN' + A B' C-IN' + A B C-IN

= C-IN (A' B' + A B) + C-IN' (A' B + A B')

= C-IN XOR (A XOR B)

Logical Expression for C-OUT:

= A' B C-IN + A B' C-IN + A B C-IN' + A B C-IN

= A B + B C-IN + A C-IN

Another form in which C-OUT can be implemented:

= A B + A C-IN + B C-IN (A + A')

= A B C-IN + A B + A C-IN + A' B C-IN

= A B (1 +C-IN) + A C-IN + A' B C-IN

= A B + A C-IN + A' B C-IN

= A B + A C-IN (B + B') + A' B C-IN

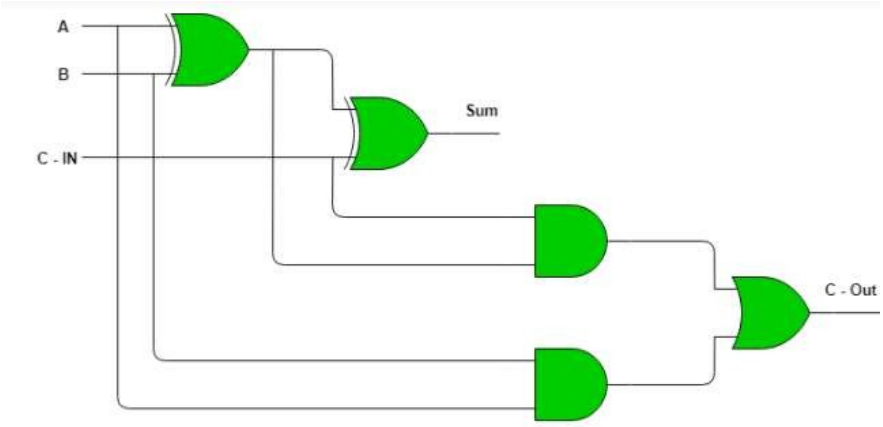
= A B C-IN + A B + A B' C-IN + A' B C-IN

= A B (C-IN + 1) + A B' C-IN + A' B C-IN

= A B + A B' C-IN + A' B C-IN

= AB + C-IN (A' B + A B')

Therefore COUT = AB + C-IN (A EX – OR B)



Full Adder logic circuit.

STRUCTURAL MODELLING

In this modelling, an entity is described as a set of interconnected components. A component instantiation statement is a concurrent statement. Therefore, the order of these statements is not important. The structural style of modelling describes only an interconnection of components (viewed as black boxes), without implying any behaviour of the components themselves nor of the entity that they collectively represent.

In Structural modelling, architecture body is composed of two parts − the declarative part (before the keyword begin) and the statement part (after the keyword begin).

PROCEDURE:

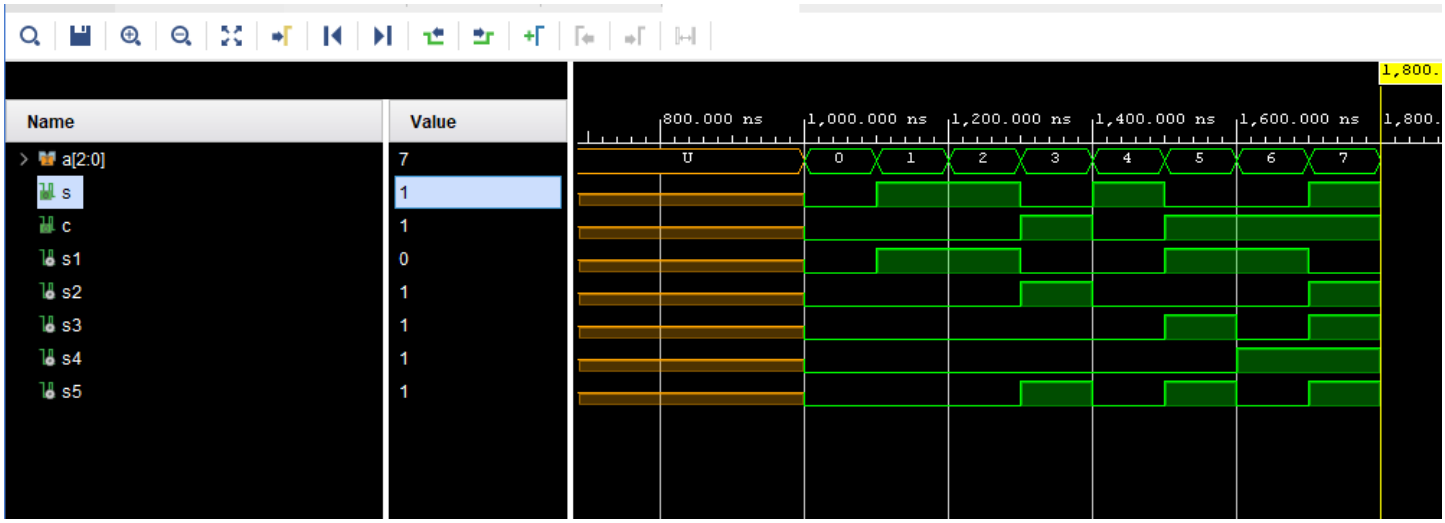
- Create New Project from File Menu.
- Give the file name.
- Select RTL project.
- Select VHDL language and mixed simulator. Then click next.
- Select add or create design sources. Then click next.
- Give port declarations as we do in entity. Then click Next. And then Finish.
- VHDL Code is presented on the screen. Now we just must write the code in architecture.
- Save and run simulation

- Now we must give the values to the inputs using force constant.
- Select the type (hexadecimal, binary, etc.), force value, starting after time offset and cancel after time offset.
- Analyze the waveform.
- Synthesize it and do the RTL analysis.

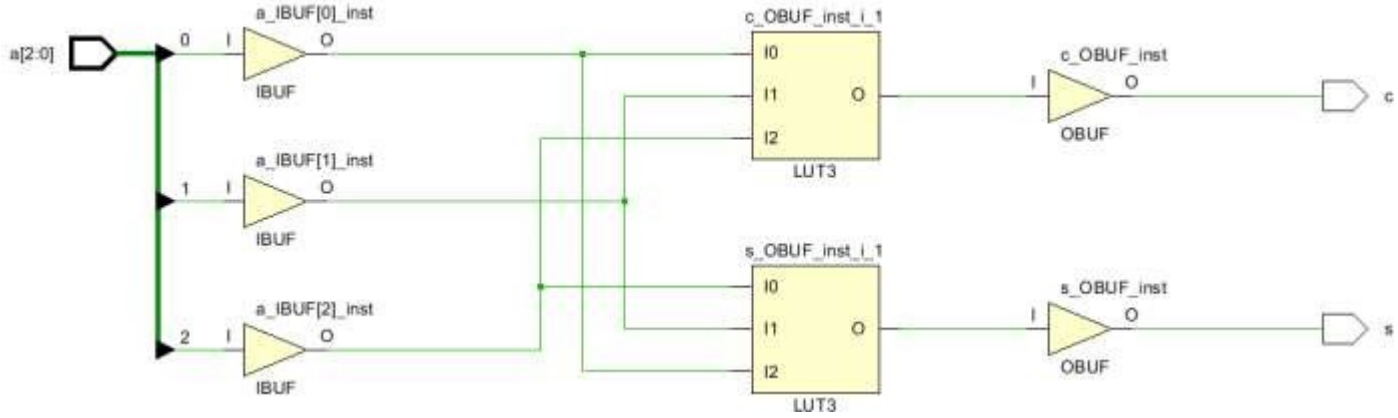
CODE:

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

entity fa_struct is
    Port ( a : in STD_LOGIC_VECTOR (2 downto 0);
           s : out STD_LOGIC;
           c : out STD_LOGIC);
end fa_struct;

architecture struct of fa_struct is

component xor1 is
   port(i : in STD_LOGIC_VECTOR(1 downto 0); y: out STD_LOGIC);
end component;

component and1 is
    port(i : in STD_LOGIC_VECTOR(1 downto 0); y: out STD_LOGIC);
end component;

component or1 is
    port(i : in STD_LOGIC_VECTOR(1 downto 0); y: out STD_LOGIC);
end component;

signal s1, s2, s3, s4, s5: STD_LOGIC;
begin

    x1: xor1 port map(i(0) => a(0), i(1) => a(1), y => s1);
    x2: xor1 port map(i(0) => a(2), i(1) => s1, y => s);

    a1: and1 port map(i(0) => a(0), i(1) => a(1), y => s2);
    a2: and1 port map(i(0) => a(0), i(1) => a(2), y => s3);
    a3: and1 port map(i(0) => a(1), i(1) => a(2), y => s4);

    o1: or1 port map(i(0) => s2, i(1) => s3, y => s5);
    o2: or1 port map(i(0) => s4, i(1) => s5, y => c);

end struct;
```

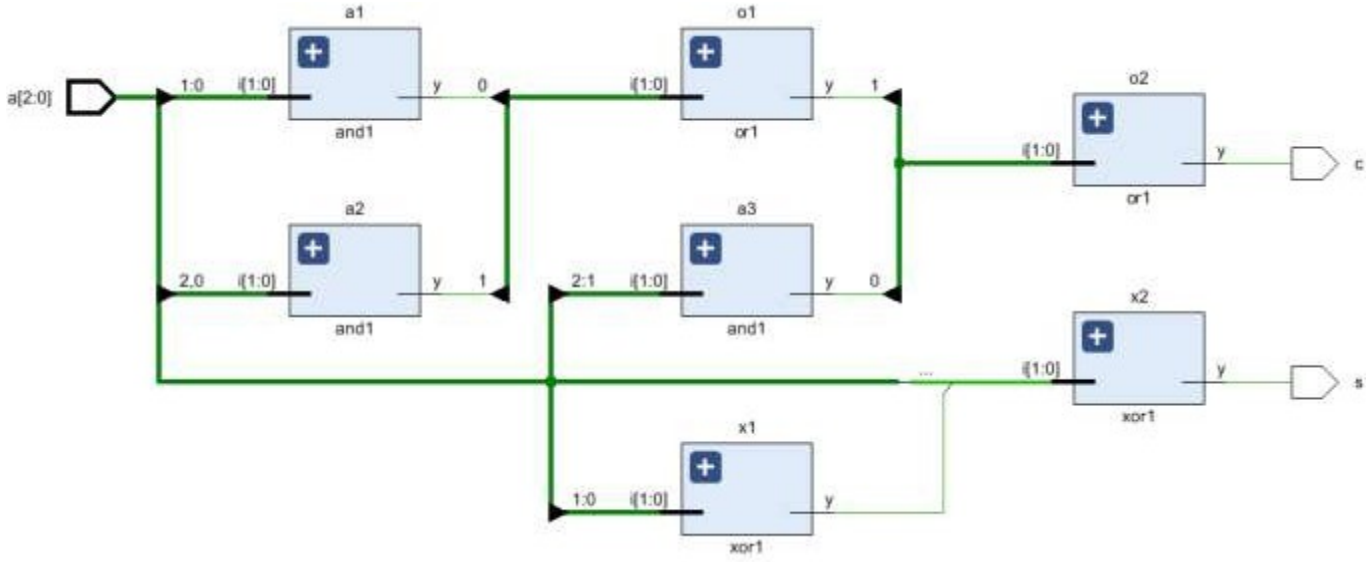
SIMULATION:



TECHNOLOGY VIEW:



RTL VIEW:



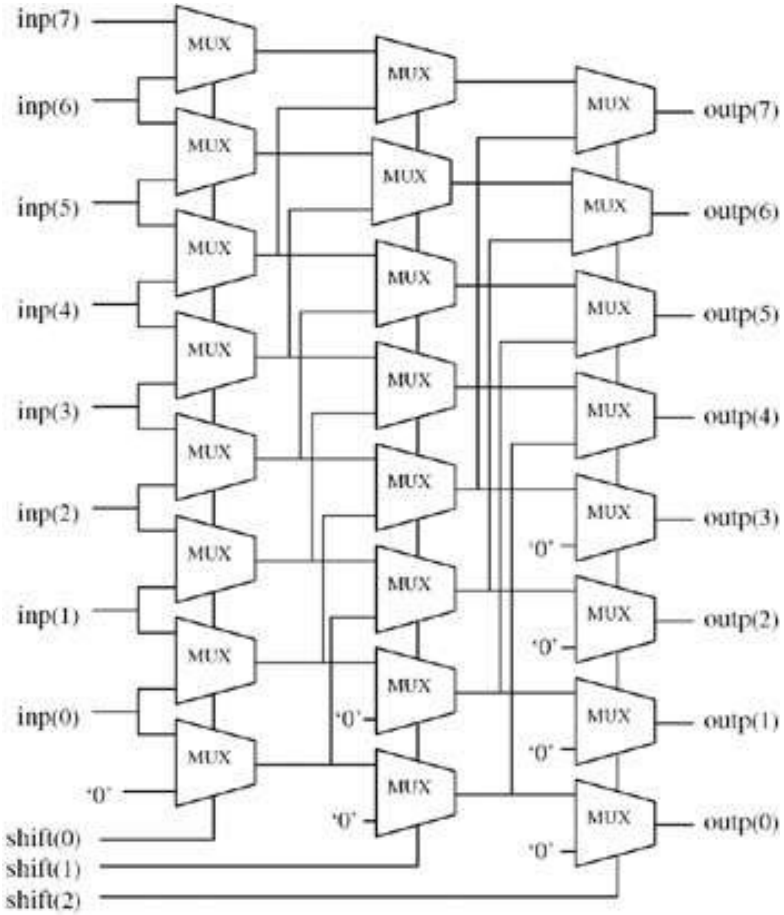
RESULT: The full adder using structural modelling has been simulated and synthesized successfully

AIM: To simulate and analyze barrel shifter using structural modelling.

SOFTWARE REQUIRED: XILINX VIVADO 2020.1

THEORY:

Barrel shifters are often utilized by embedded digital signal processors and general-purpose processors to manipulate data. This paper examines design alternatives for barrel shifters that perform the following functions: shift right logical, shift right arithmetic, rotate right, shift left logical, shift left arithmetic, and rotate left. Four different barrel shifter designs are presented and compared in terms of area and delay for a variety of operand sizes. This paper also examines techniques for detecting results that over ow and results of zero in parallel with the shift or rotate operation. Several Java programs are developed to generate structural VHDL models for each of the barrel shifters. Synthesis results show that data-reversal barrel shifters have less area and mask-based data-reversal barrel shifters have less delay than other designs. Mask-based data-reversal barrel shifters are especially attractive when over ow and zero detection is also required, since the detection is performed in parallel with the shift or rotate operation.



It shows the block diagram of an 8-bit logical right shifter, which uses three stages with 4-bit, 2- bit, and 1-bit shifts. To optimize the design, each multiplexor that has '0' for one of its inputs can be replaced by a 2-input and gate.

STRUCTURAL MODELLING

In this modelling, an entity is described as a set of interconnected components. A component instantiation statement is a concurrent statement. Therefore, the order of these statements is not important. The structural style of modelling describes only an interconnection of components (viewed as black boxes), without implying any behaviour of the components themselves nor of the entity that they collectively represent.In Structural modelling, architecture body is composed of two parts − the declarative part (before the keyword begin) and the statement part (after the keyword begin).

PROCEDURE:

- Create New Project from File Menu.
- Give the file name.

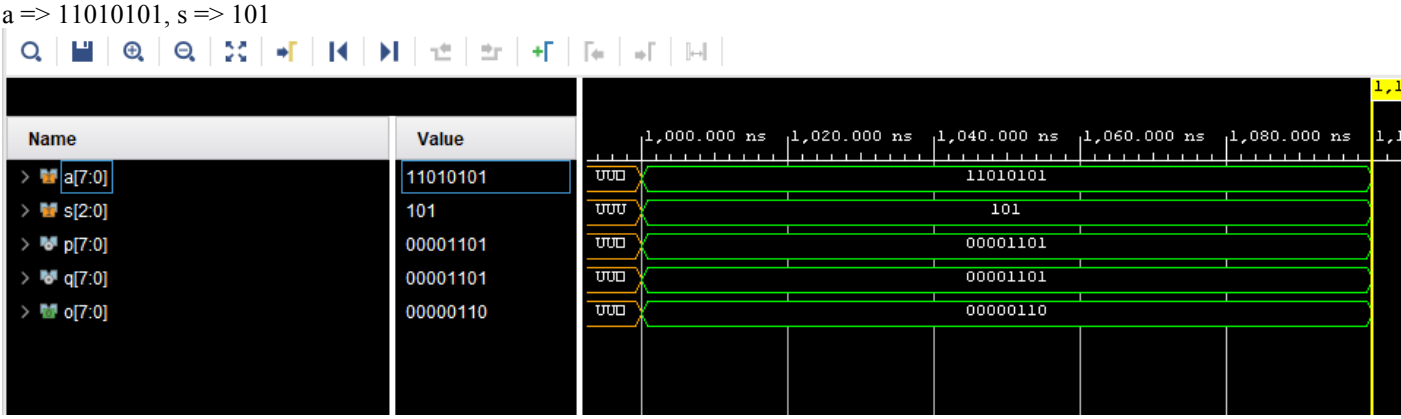
- Select RTL project.
- Select VHDL language and mixed simulator. Then click next.
- Select add or create design sources. Then click next.
- Give port declarations as we do in entity. Then click Next. And then Finish.
- VHDL Code is presented on the screen. Now we just have to write the code in architecture.
- Save and run simulation
- Now we must give the values to the inputs using force constant.
- Select the type (hexadecimal, binary, etc.), force value, starting after time offset and cancel after time offset.
- Analyze the waveform.
- Synthesize it and do the RTL analysis.

CODE:

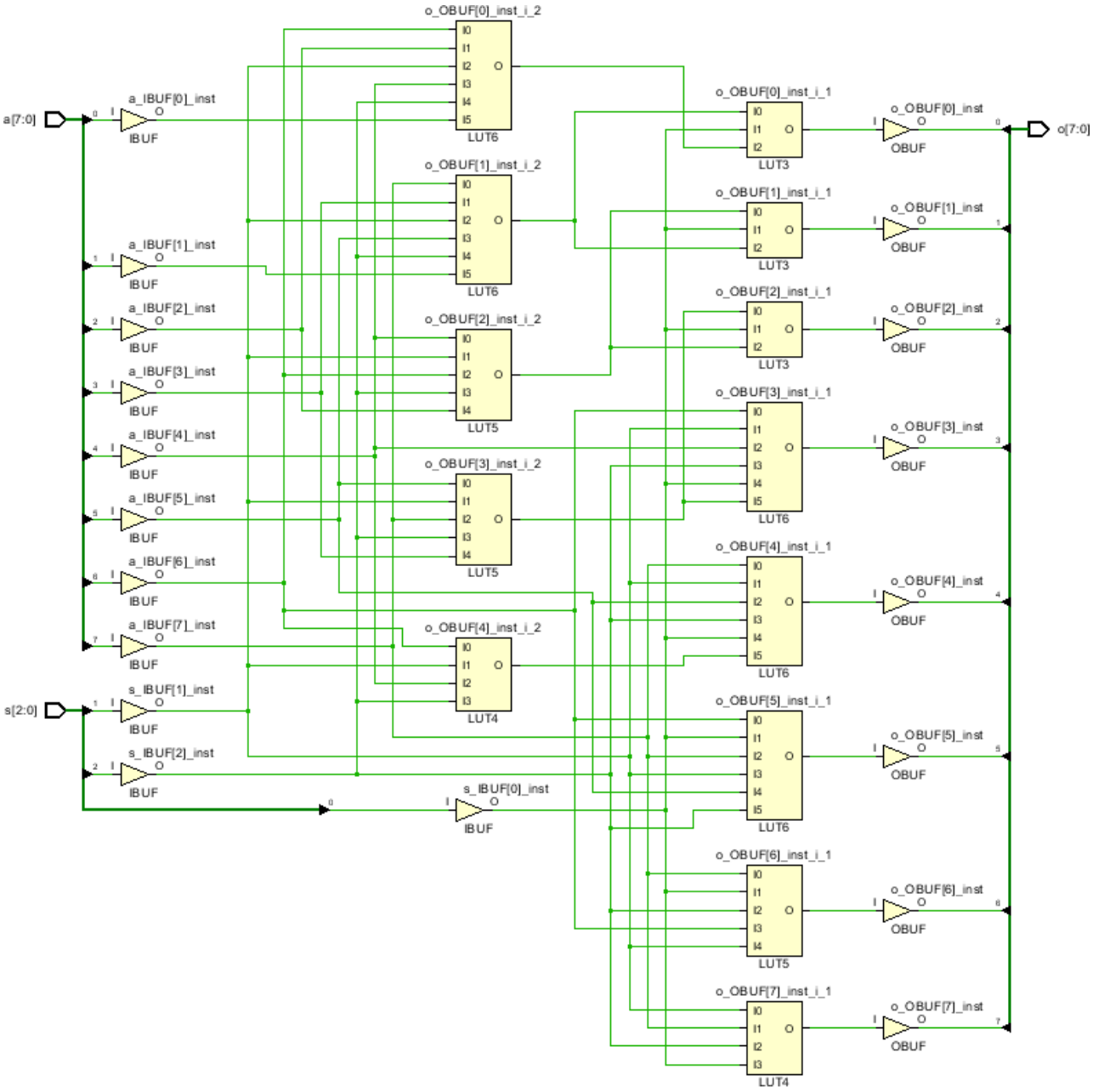
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity barrel_shifter is
    Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
           s : in STD_LOGIC_VECTOR (2 downto 0);
           o : out STD_LOGIC_VECTOR (7 downto 0));
end barrel_shifter;

architecture structural of barrel_shifter is

component mux2x1 is
   port(i0, i1 : in STD_LOGIC; s : in STD_LOGIC; o: out STD_LOGIC);
end component;

signal p, q : STD_LOGIC_VECTOR (7 downto 0);

begin

    L1: for k in 0 to 7 generate
        C1: if (k>3) generate
            M1: mux2x1 port map(a(k), '0', s(2), p(k));
        end generate C1;
        C2: if (k<=3) generate
            M2: mux2x1 port map(a(k), a(k+4), s(2), p(k));
        end generate C2;
     end generate L1;

     L2: for k in 0 to 7 generate
        C3: if (k>5) generate
            M3: mux2x1 port map(p(k), '0', s(1), q(k));
        end generate C3;
        C4: if (k<=5) generate
            M4: mux2x1 port map(p(k), p(k+2), s(1), q(k));
        end generate C4;

     end generate L2;

    L3: for k in 0 to 7 generate
        C5: if (k>6) generate
            M5: mux2x1 port map(q(k), '0', s(0), o(k));
        end generate C5;
        C6: if (k<=6) generate
            M6: mux2x1 port map(q(k), q(k+1), s(0), o(k));
        end generate C6;
     end generate L3;

end structural;
```

SIMULATION:

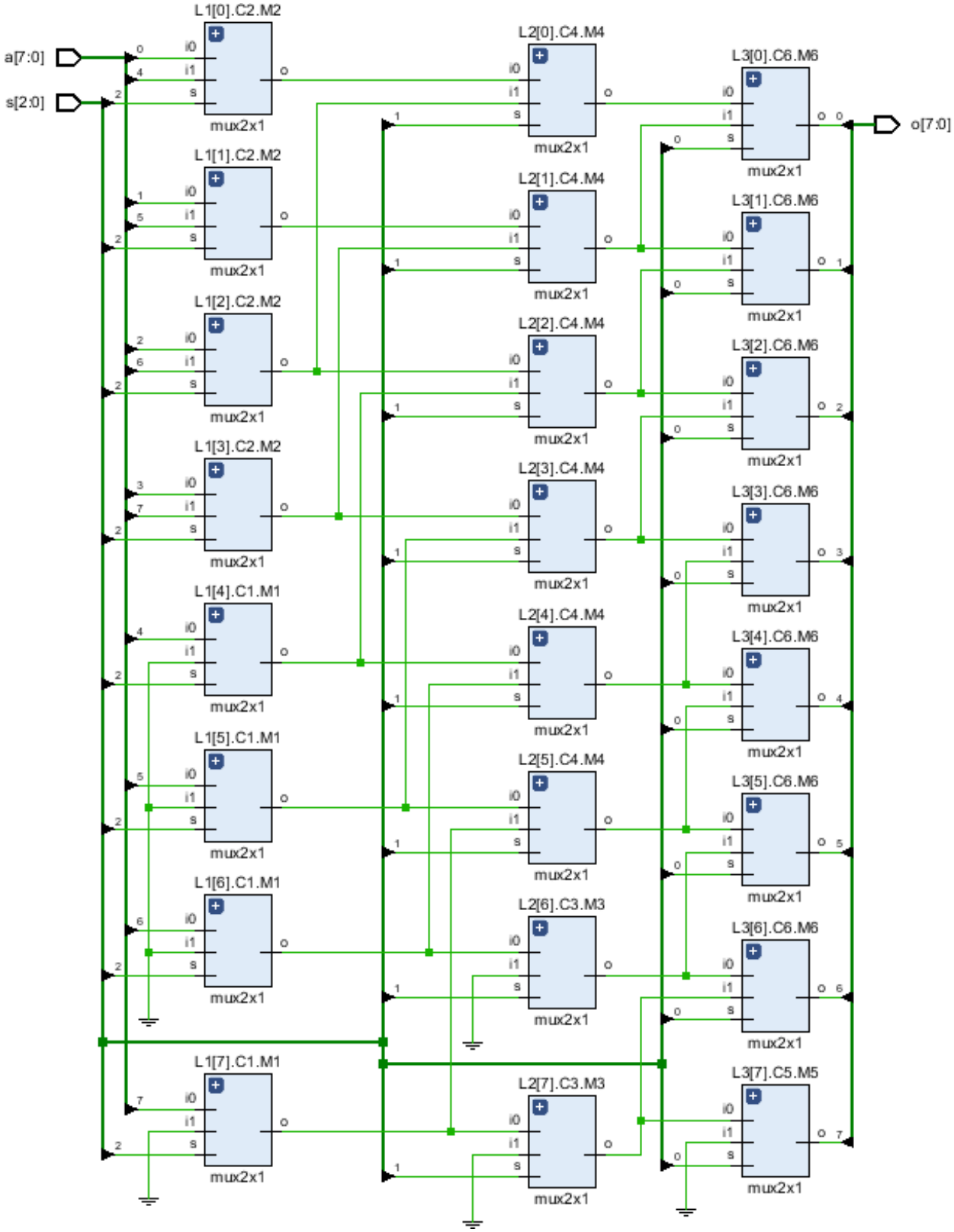
a => 11010101, s => 101



TECHNOLOGY VIEW:



RTL VIEW:



RESULT: The barrel shifter using structural modelling has been simulated and synthesized successfully.