LAB 2

Rachel Ruddy - 261162987
Natasha Lawford - 261178596
Rachel Bunsick - 261159677

Executive Summary (short description of what you have done in this VHDL assignment)
In this VHDL assignment (lab 2), we created the schematic for an AeqB and generated its testbench as per the lab's instructions, we then implemented a 2-to-1 multiplexer (MUX) and a 4-bit circular barrel shifter using structural and behavioral descriptions. The 2-to-1 MUX was tested to verify proper functionality across all possible input combinations. For the 4-bit circular barrel shifter, we used two layers of 2-to-1 MUXs to achieve logic that shifts inputs by a specified amount. Designs were simulated using ModelSim, and tests were conducted to ensure correctness.

(1) Briefly explain your VHDL code implementation of all circuit

Part 6:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity rachel_ruddy_MUX_structural is
     Port (A : in std_logic;
           B : in std_logic;
           S : in std_logic;
           Y : out std_logic);
end rachel_ruddy_MUX_structural;

architecture Structural of rachel_ruddy_MUX_structural is
     signal not_S : std_logic;
     signal A_and_not_S : std_logic;
     signal B_and_S : std_logic;
begin
     not_S <= not S;
     A_and_not_S <= A and not_S;
     B_and_S <= B and S;

     Y <= A_and_not_S or B_and_S;
end Structural;
```

Structural architecture of 2 to 1 MUX: created signals to represent each of the individual signals for the 2-to-1 MUX logic: Y= (S'A)+(SB) in order to give the appropriate value to the output Y. In accordance with structural programming, we are connecting different parts to get the overall design.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity rachel_ruddy_MUX_behavioral is
     Port (A : in std_logic;
                    B : in std_logic;
                    S : in std_logic;
                    Y : out std_logic);

end rachel_ruddy_MUX_behavioral;

architecture Behavioral of rachel_ruddy_MUX_behavioral is
begin
     with S select
          Y <= A when '0',
               B when '1';
              ‘0’ when others;
end Behavioral;
```

The code above describes the behavioral architecture for a 2-to-1 MUX. Using “with/select”, we assign the value of Y in specific cases, describing how Y behaves in relation to other inputs and their values rather than connecting different parts.

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

entity rachel_ruddy_MUX_tst is
end rachel_ruddy_MUX_tst;

architecture Testbench of rachel_ruddy_MUX_tst is
     component MUX_2to1 is
          Port ( A : in std_logic;
                    B : in std_logic;
                    S : in std_logic;
                    Y : in std_logic);
end component;

signal A, B, S : std_logic := '0';
signal Y_structural, Y_behavioral : std_logic;

begin
i1 : entity work.rachel_ruddy_MUX_structural
      port map (A => A, B => B, S => S, Y => Y_structural);
i2 : entity work.rachel_ruddy_MUX_behavioral
     port map (A => A, B => B, S => S, Y => Y_behavioral);

process
begin
      --test case 1: expect Y=0
      A <= '0'; B <= '0'; S <= '0';
      wait for 10 ns;

      --test case 2: expect Y=0
      A <= '0'; B <= '0'; S <= '1';
      wait for 10 ns;

      --test case 3: expect Y=0
      A <= '0'; B <= '1'; S <= '0';
      wait for 10 ns;

      --test case 4: expect Y=1
```

```
        A <= '0'; B <= '1'; S <= '1';
        wait for 10 ns;

        --test case 5: expect Y=1
        A <= '1'; B <= '0'; S <= '0';
        wait for 10 ns;

        --test case 6: expect Y=0
        A <= '1'; B <= '0'; S <= '1';
        wait for 10 ns;

        --test case 7: expect Y=1
        A <= '1'; B <= '1'; S <= '0';
        wait for 10 ns;

        --test case 8: expect Y=1
        A <= '1'; B <= '1'; S <= '1';
        wait for 10 ns;

        wait;
    end process;

    end Testbench;
```

Above is the testbench for the 2-to-1 MUX. Commented out are the expected values for every possible value for A, B, S. Two objects i1, and i2, are instances of the behavioral and the structural MUX architectures. This allows us to see the equivalent outputs for both architectures when seeing the waveform.

Part 7:

Structural

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


entity rachel_ruddy_barrel_shifter_structural is
    Port (X          : in std_logic_vector (3 downto 0);
          sel        : in std_logic_vector (1 downto 0);
          stage1     : inout std_logic_vector (3 downto 0);
          Y          : out std_logic_vector (3 downto 0));
end rachel_ruddy_barrel_shifter_structural;


architecture Structural of rachel_ruddy_barrel_shifter_structural is

     component rachel_ruddy_MUX_structural
          Port (A,B,S : in std_logic;
                      Y : out std_logic);
     end component;

begin

    mux0_stage1: rachel_ruddy_MUX_structural
        port map (
            A => X(0),
            B => X(2),
            S => sel(1),
            Y => stage1(0)
        );

    mux1_stage1: rachel_ruddy_MUX_structural
        port map (
            A => X(1),
            B => X(3),
            S => sel(1),
            Y => stage1(1)
```

```
        );

    mux2_stage1: rachel_ruddy_MUX_structural
        port map (
            A => X(2),
            B => X(0),
            S => sel(1),
            Y => stage1(2)
        );

    mux3_stage1: rachel_ruddy_MUX_structural
        port map (
            A => X(3),
            B => X(1),
            S => sel(1),
            Y => stage1(3)
        );
-------------------------------------------------------------


    mux0_stage2: rachel_ruddy_MUX_structural
        port map (
            A => stage1(0),
            B => stage1(3),
            S => sel(0),
            Y => Y(0)
        );

    mux1_stage2: rachel_ruddy_MUX_structural
        port map (
            A => stage1(1),
            B => stage1(0),
            S => sel(0),
            Y => Y(1)
        );

    mux2_stage2: rachel_ruddy_MUX_structural
        port map (
            A => stage1(2),
```

```
            B => stage1(1),
            S => sel(0),
            Y => Y(2)
        );

      mux3_stage2: rachel_ruddy_MUX_structural
       port map (
                A => stage1(3),
                B => stage1(2),
                S => sel(0),
                Y => Y(3)
       );

end Structural;
```

The structural implementation instantiates 8 of the 2 to 1 mux that we created and maps the values of our 4 bit input and 2 bit selector accordingly. To implement the ports, we separated the mux into 2 stages. For the first stage (4 mux), we followed the diagram provided in the lab report, mapping each bit of the input value x to the appropriate place on the mux. Then, we mapped the outputs of each of the mux in the first stage to the appropriate mux in the second stage. We also assigned each mux in the first stage to follow the most significant bit of sel and each mux in the second stage to the least significant value of sel. Finally, we assigned each output from the mux to the appropriate spot in the 4 bit output Y.

Behavioral:

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

entity rachel_ruddy_barrel_shifter_behavioral is
    Port (
        X   : in std_logic_vector(3 downto 0);
        sel : in std_logic_vector(1 downto 0);
        Y   : out std_logic_vector(3 downto 0));
end rachel_ruddy_barrel_shifter_behavioral;

architecture Behavioral of rachel_ruddy_barrel_shifter_behavioral is
begin
```

```
    Y <= X when sel = "00" else
         X(2 downto 0) & X(3) when sel = "01" else
         X(1 downto 0) & X(3 downto 2) when sel = "10" else
         X(0) & X(3 downto 1);
end Behavioral;
```

The behavioral implementation of the 4-bit circular barrel shifter describes the functionality of the 4 bit barrel shifter. It takes the input bit X and assigns the output to be the shifted value based on the value of the sel. Sel may have a value between 0-3 as a 2-bit value. The output is then assigned to be the values of X shifted to the left by sel digits, and since it is circular the digits that get pushed off are looped around from the right side. For example, 1101 will become 1011 when sel = 1 (01 as a 2 bit vector) and 0111 when sel = 2 (10 as a 2 bit vector).

Testbench:

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

entity rachel_ruddy_barrel_shifter_tb is
end rachel_ruddy_barrel_shifter_tb;

architecture Testbench of rachel_ruddy_barrel_shifter_tb is
--
--signal X, stage1 : std_logic_vector(3 downto 0) := "0000";
--signal sel: std_logic_vector(1 downto 0) := "00";
--signal Y_behavioral: std_logic_vector(3 downto 0);


signal X, stage1 : std_logic_vector(3 downto 0);
signal sel: std_logic_vector(1 downto 0);
signal Y_behavioral: std_logic_vector(3 downto 0);

     component rachel_ruddy_barrel_shifter_behavioral
          Port (
               X : in std_logic_vector(3 downto 0);
               sel : in std_logic_vector(1 downto 0);
```

```
                Y : out std_logic_vector(3 downto 0)
          );
     end component;
--
--    component rachel_ruddy_barrel_shifter_structural
--         Port (
--              X : in std_logic_vector(3 downto 0);
--              sel : in std_logic_vector(1 downto 0);
--              stage1 : inout std_logic_vector (3 downto 0);
--              Y : out std_logic_vector(3 downto 0)
--         );
--    end component;


begin
--i1: rachel_ruddy_barrel_shifter_structural port map (X => X, sel =>
sel, stage1 => stage1, Y => Y_structural);

i2: rachel_ruddy_barrel_shifter_behavioral port map (X => X, sel =>
sel, Y => Y_behavioral);


generate_test: PROCESS
BEGIN
     FOR i IN 0 to 4 LOOP
          sel <= std_logic_vector(to_unsigned(i,2));
               FOR j IN 0 to 16 LOOP
                    X <= std_logic_vector(to_unsigned(j,4));
               WAIT FOR 10 ns ;
               END LOOP;
     END LOOP;
     WAIT FOR 10 ns;
     WAIT;
END PROCESS generate_test;

end Testbench;
```

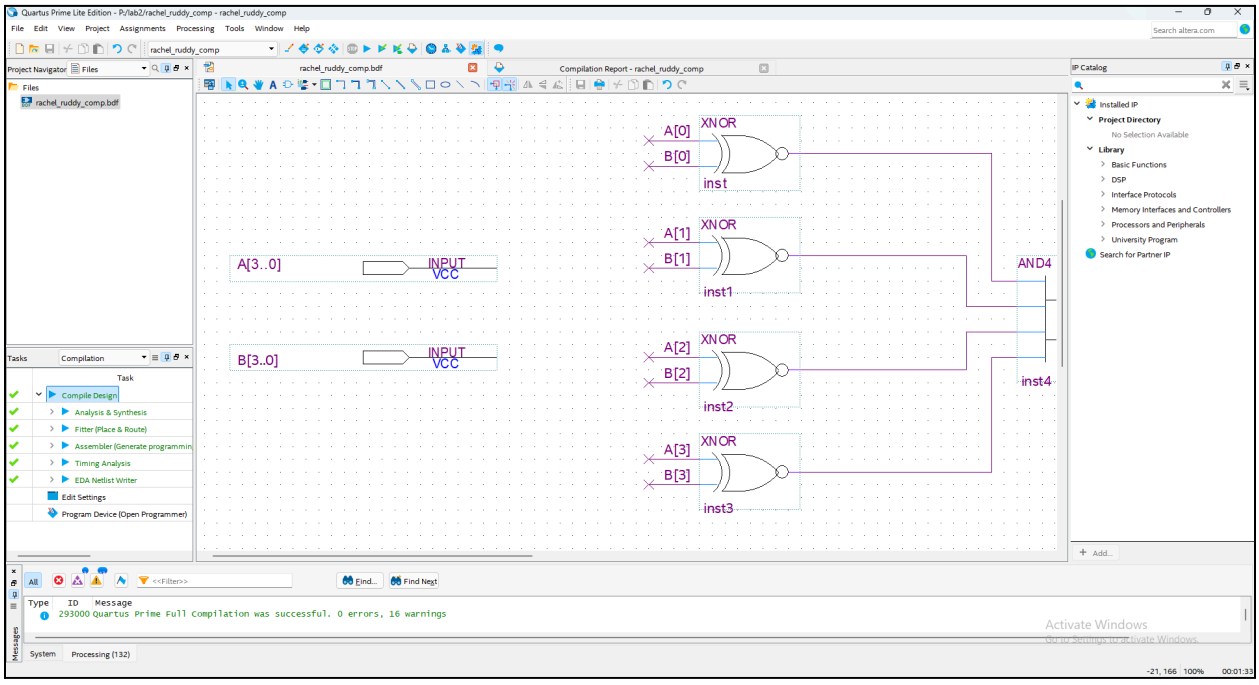
Above is the testbench code for the 4 bit barrel shifter. It loops through each value of select, and

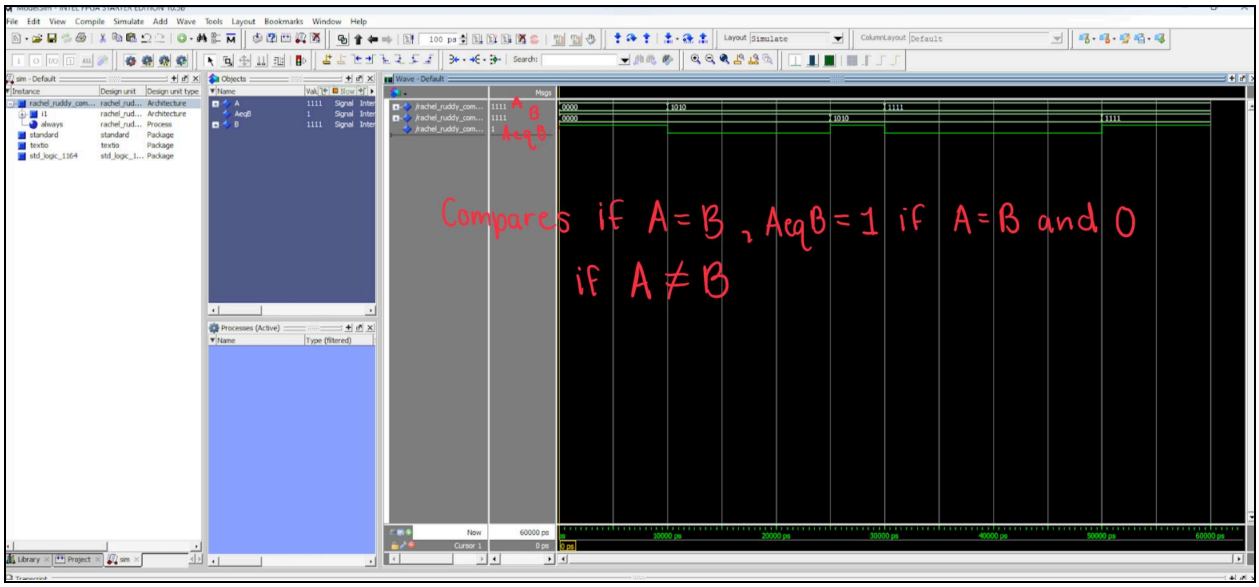
then tests each of the 16 possible 4 bit inputs of X. This exhaustively tests all possible combinations of input for our barrel shifter.

(2) Report the number of pins and logic modules used to fit your designs on the FPGA board. These results can be obtained from the flow summary tab in the table of contents menu in Quartus.

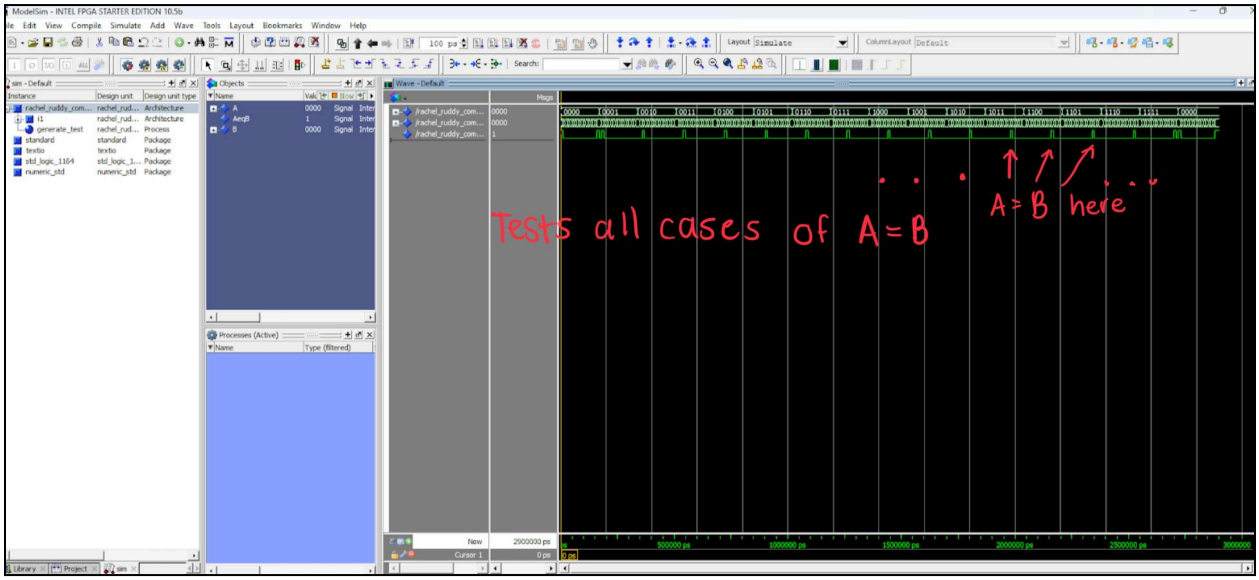
	AeqB	2-to-1 MUX		4-bit circular barrel shifter	
	Schematic	Structural	Behavioral	Structural	Behavioral
Logic Utilization (in ALMs)	2/32,070	1/32,070	1/32,070	7 / 56,480	7 / 56,480
Total pins	9/314	4/314	4/314	14/268	14/268

(3) Show a representative simulation plot for the introductory testing example. You can simply include a snapshot from the waveform that you obtained from ModelSim. In order to fully capture all the signals from the waveform, you can adjust the display range using the magnifier icons.

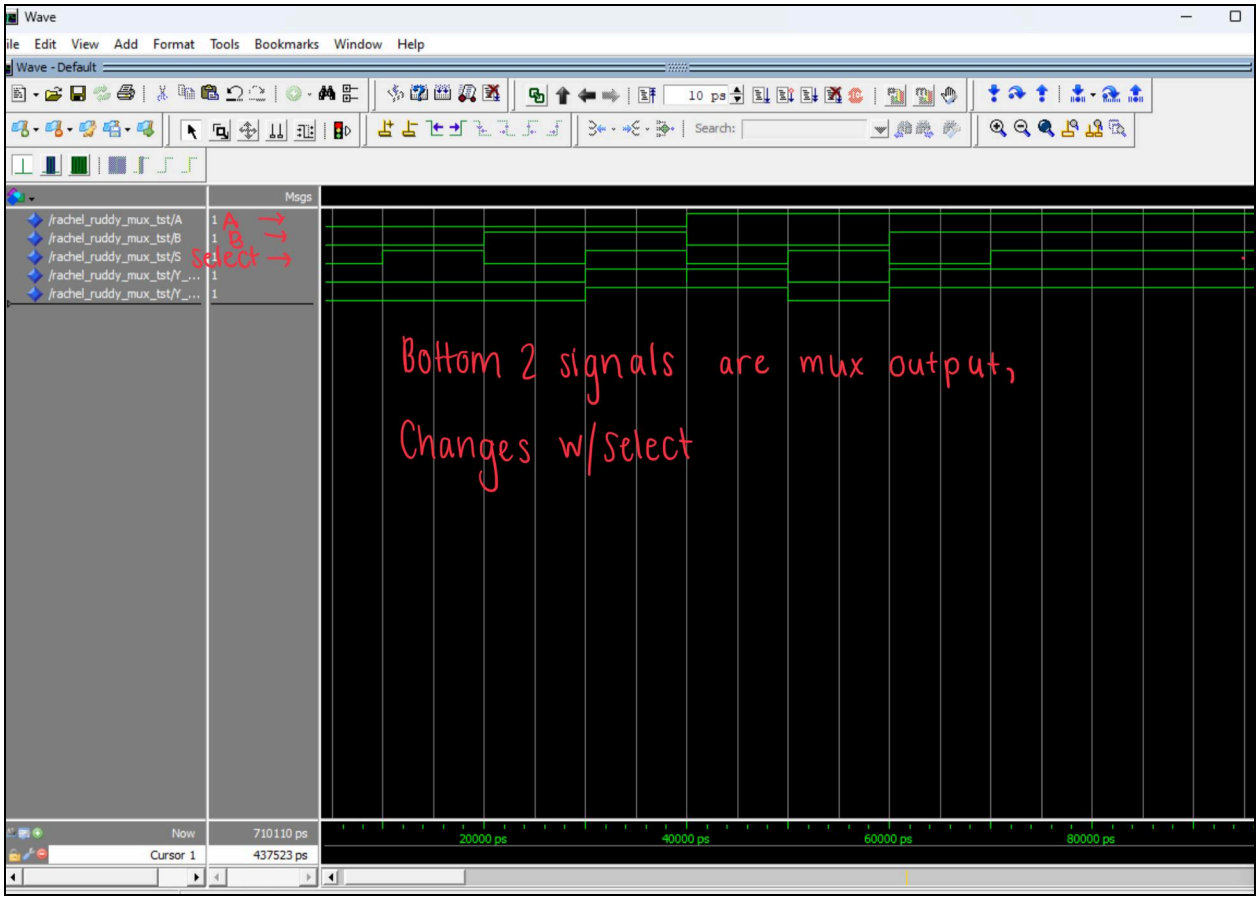




(4) Show representative simulation plots for the exhaustive test.

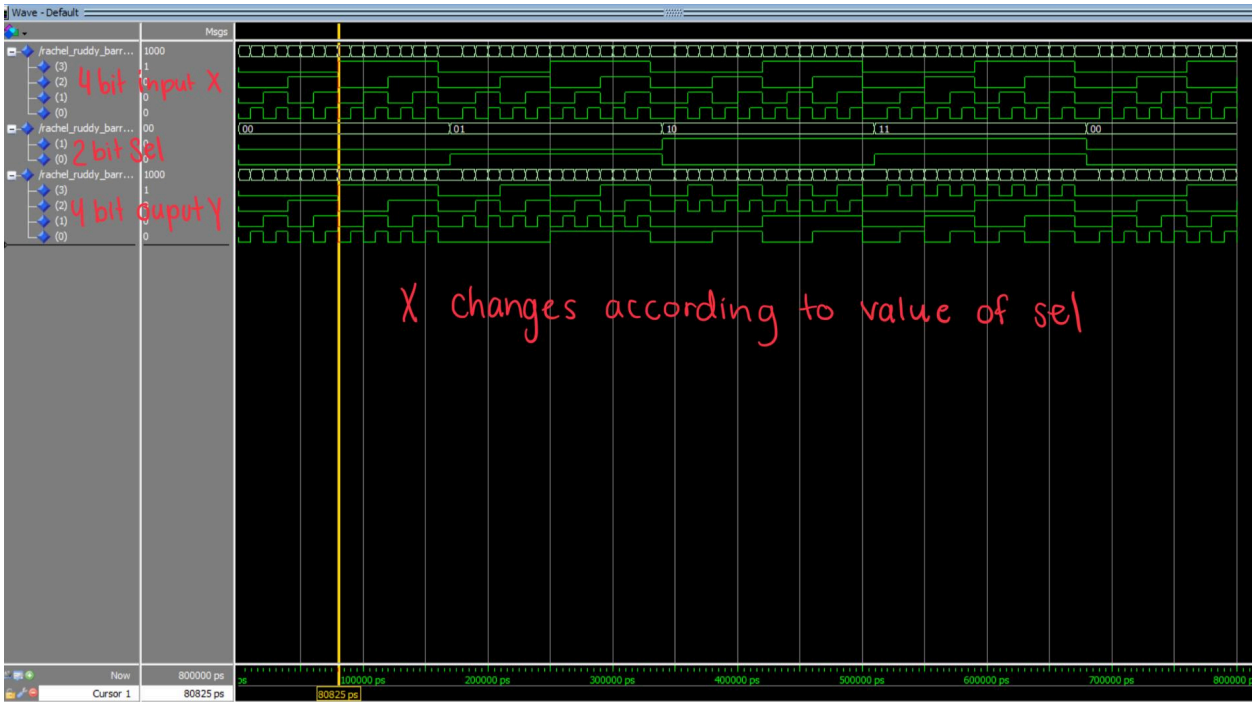


(5) Show representative simulation plots of the 2-to-1 MUX circuits for all the possible input values.



(6) Show representative simulation plots of the 4-bit circular shift register circuits for a given input sequence,e.g., “1011” (X = “1011”), for all the possible shift amounts.

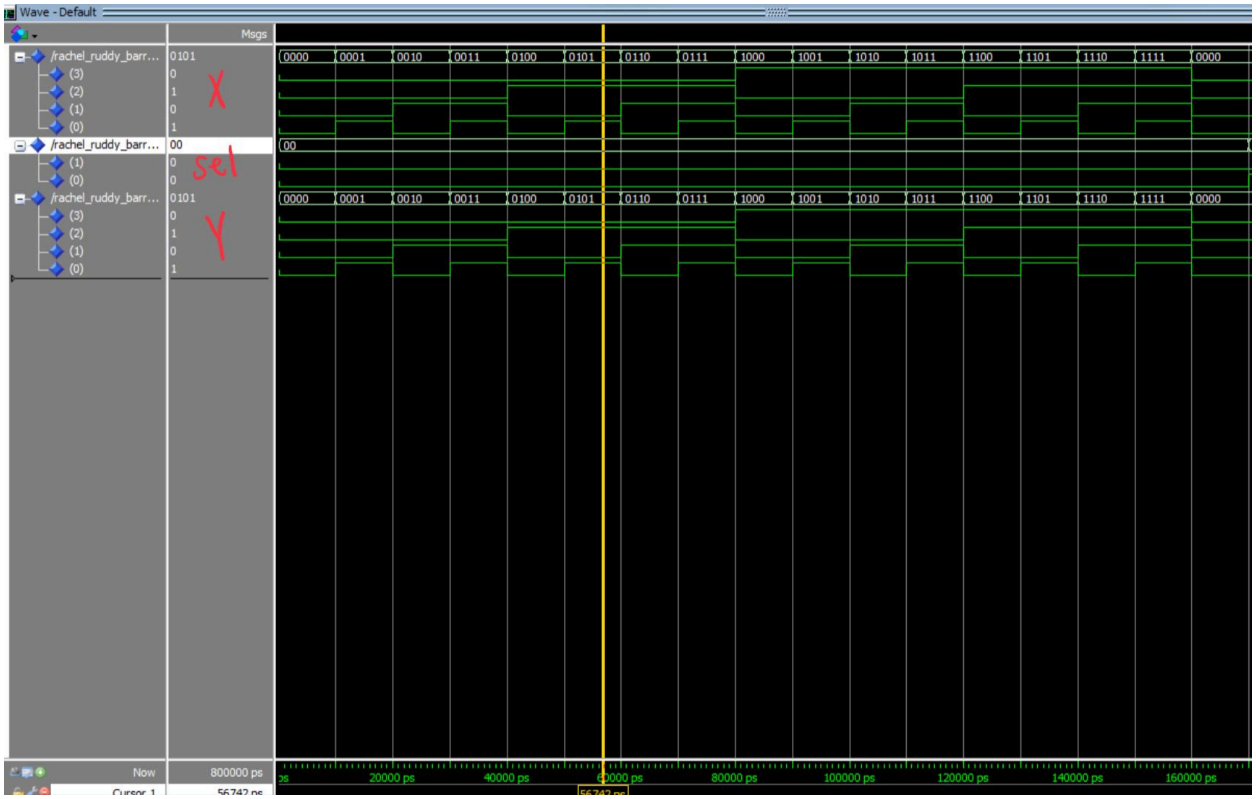
Image of All Tests:



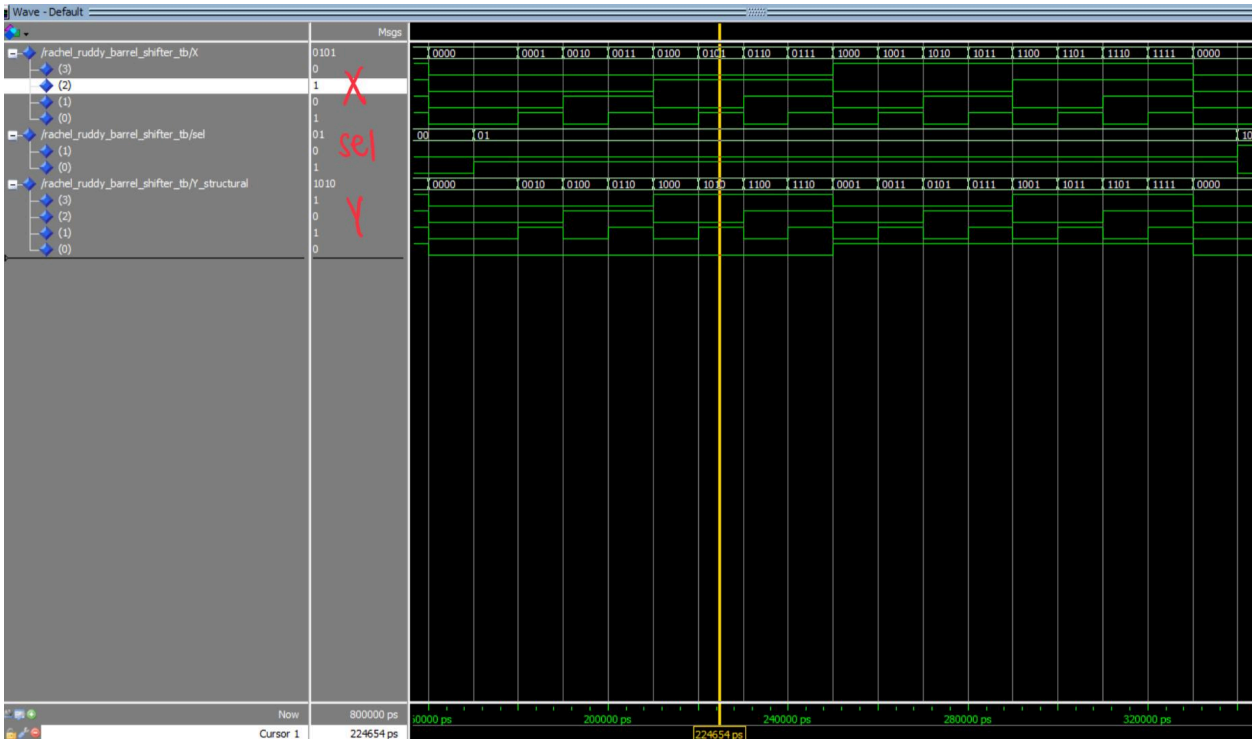
The figure above shows a representation of the exhaustive testing performed on the 4 bit circular barrel shifter. It shows the iteration of all 16 possible 4 bit inputs for each of the 4 possible selection values.

Next, we include screenshots of the 4 bit input “0101” being shifted by each of the 4 values of sel (00 - 0, 01 - 1, 10 - 2, and 11 - 3).

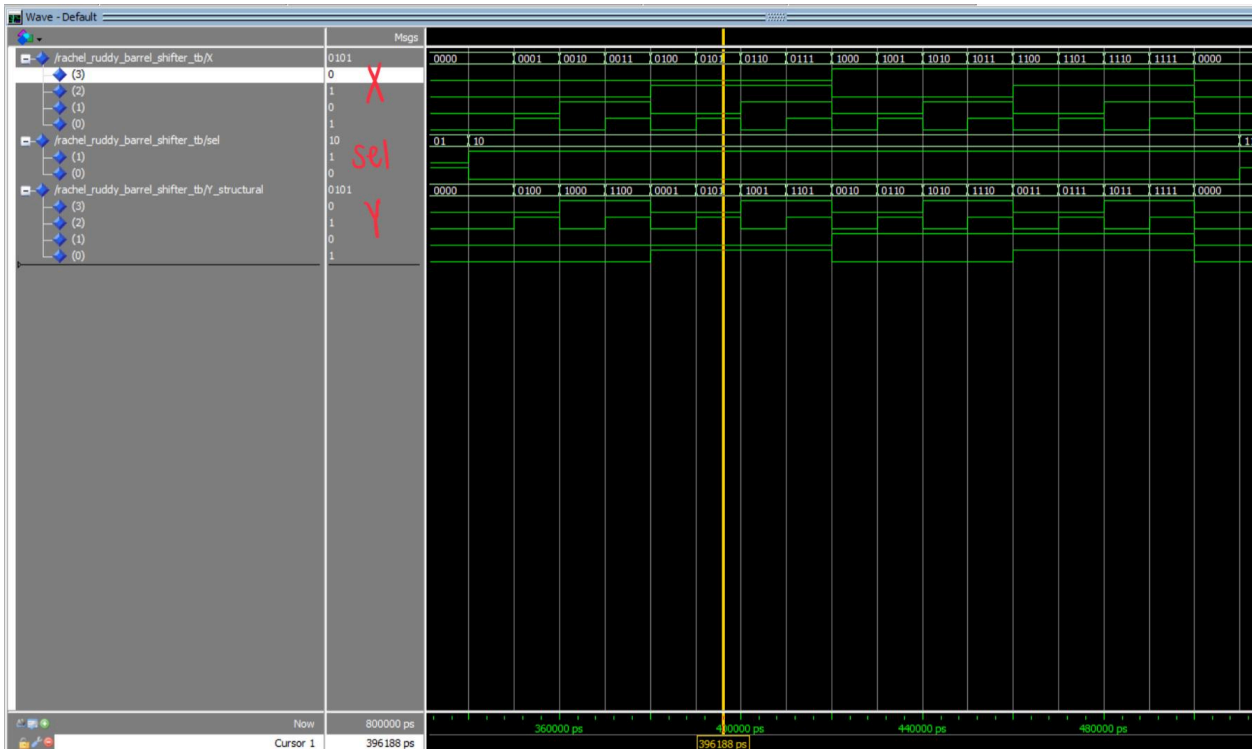
Shift when sel=00, This shifts the value of 0101 by 0 bits and we expect an outcome of 0101.



Shift when sel=01, This shifts the value of 0101 by 1 bit and we expect an outcome of 1010.



Shift when sel=10, This shifts the value of 0101 by 2 bits and we expect an outcome of 0101.



Shift when sel=11, This shifts the value of 0101 by 3 bits and we expect an outcome of 1010.