به نام خدا

عنوان:

تکلیف شماره سوم درس طراحی سیستم های دیجیتال

استاد:

دکتر مروستی

اعضا:

محمدعلی مجتهدسلیمانی- ۹۹۲۰۲۳۰۳۹

تاریخ:

۱۴۰۳/۰۳/۵

Q1. for this question first we need a counter that can handle a generic number of decimal digits. Then we use the "for-generate" statement to handle multiple digits. After that we create a test bench to verify 4bit counter. The entity will have this ingredient: CLK, RESET, ENABLE and COUNT. For implement this we need an array of signal to hold the value for each digit. Then we use "for-generate" statement. At the end, combine the individual digit values into the final COUNT output.

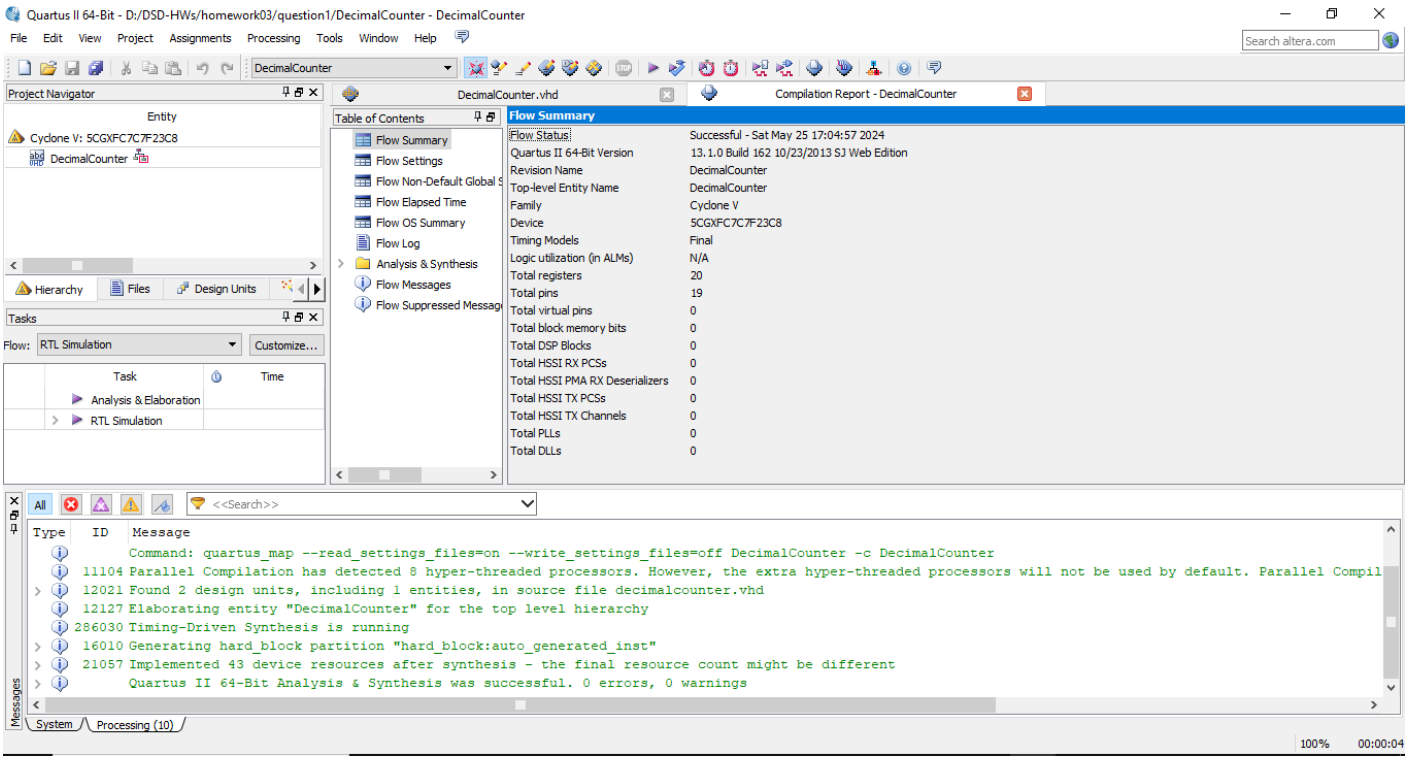
❖Code:

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


entity DecimalCounter is
   generic (
      NUM_DIGITS : integer := 4 -- Number of digits
   );
   Port (
      clk    : in  STD_LOGIC;
      rst    : in  STD_LOGIC;
      en     : in  STD_LOGIC;
      count     : out STD_LOGIC_VECTOR((NUM_DIGITS*4)-1 downto 0) --
Output for the counter
   );
```

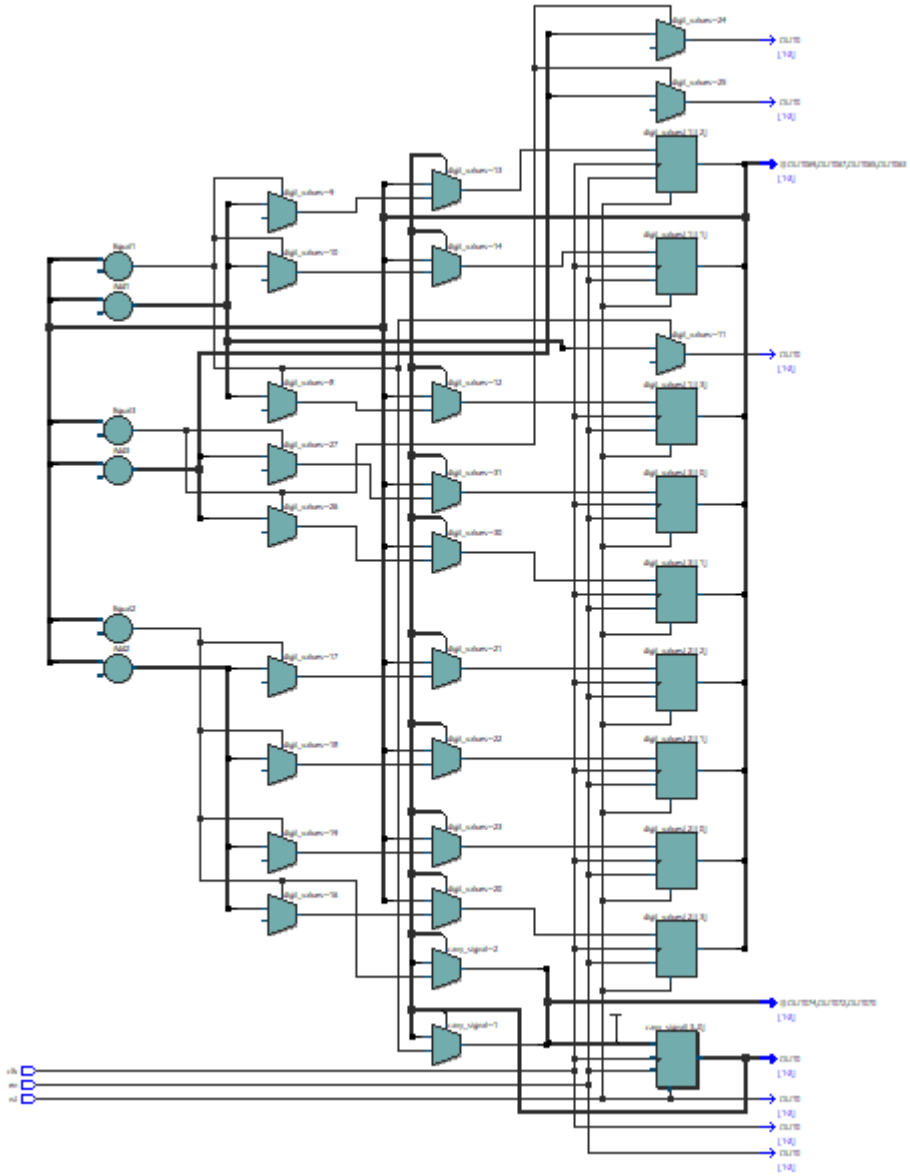
```
end DecimalCounter;


architecture Behavioral of DecimalCounter is
   type digit_array is array (NUM_DIGITS-1 downto 0) of STD_LOGIC_VECTOR(3 downto 0); -- Array for digits
   signal digit_values : digit_array := (others => (others => '0'));
   signal carry_signal : STD_LOGIC_VECTOR(NUM_DIGITS downto 0) := (others => '0');
begin


   process(clk, rst)
   begin
      if rst = '1' then
         digit_values <= (others => (others => '0')); -- Reset all digits
         carry_signal  <= (others => '0');
      elsif rising_edge(clk) then
         if en = '1' then
            carry_signal(0) <= '1'; -- Initial carry to start counting
            for i in 0 to NUM_DIGITS-1 loop
               if carry_signal(i) = '1' then
                  if digit_values(i) = "1001" then
                     digit_values(i) <= "0000";
                     carry_signal(i+1) <= '1';
                  else
                     digit_values(i) <= digit_values(i) + 1;
```

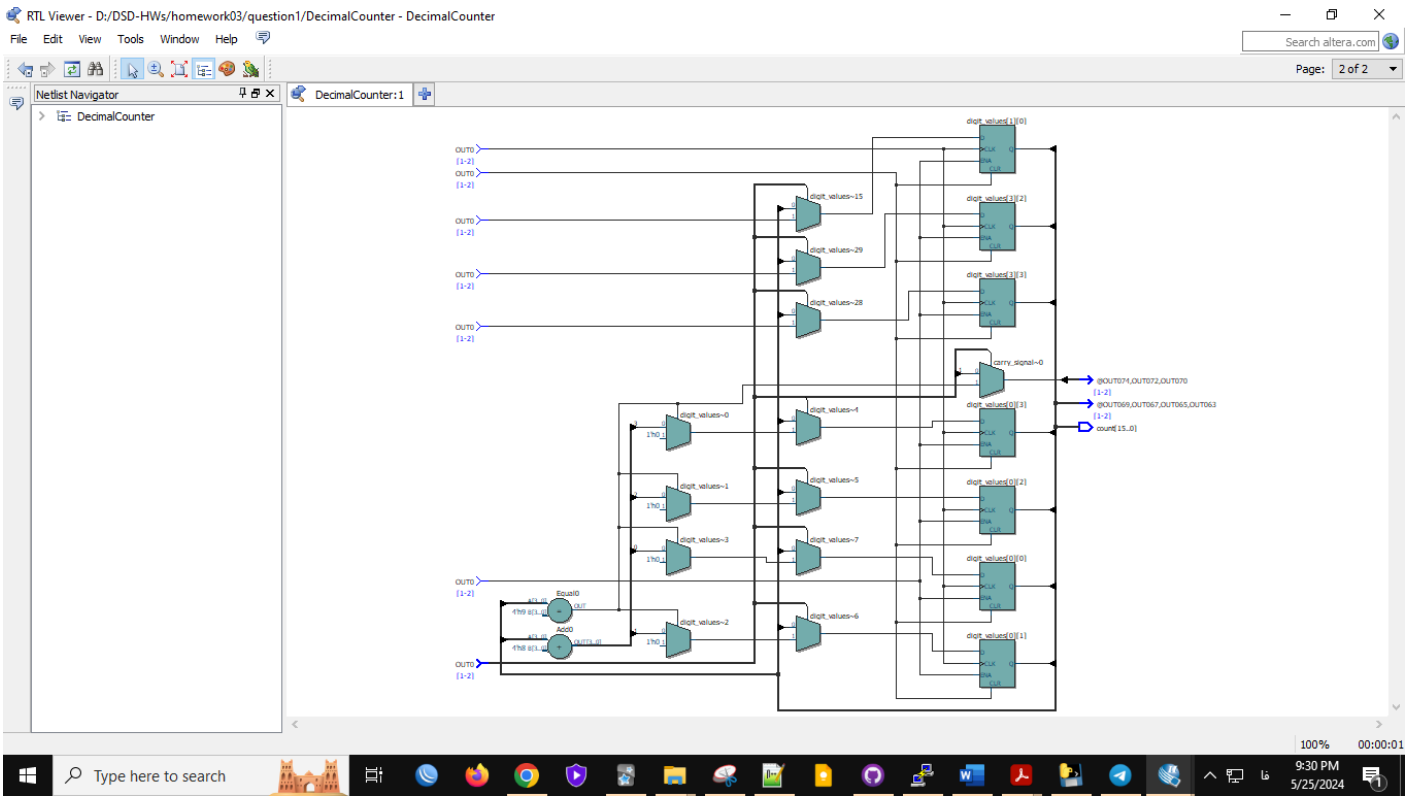
```
                    carry_signal(i+1) <= '0';
                end if;
            end if;
        end loop;
    end if;
end if;
end process;

-- Concatenate digits into count
process(digit_values)
begin
    for i in 0 to NUM_DIGITS-1 loop
        count((i*4)+3 downto i*4) <= digit_values(i);
    end loop;
end process;

end Behavioral;
```

❖Compilation Report:

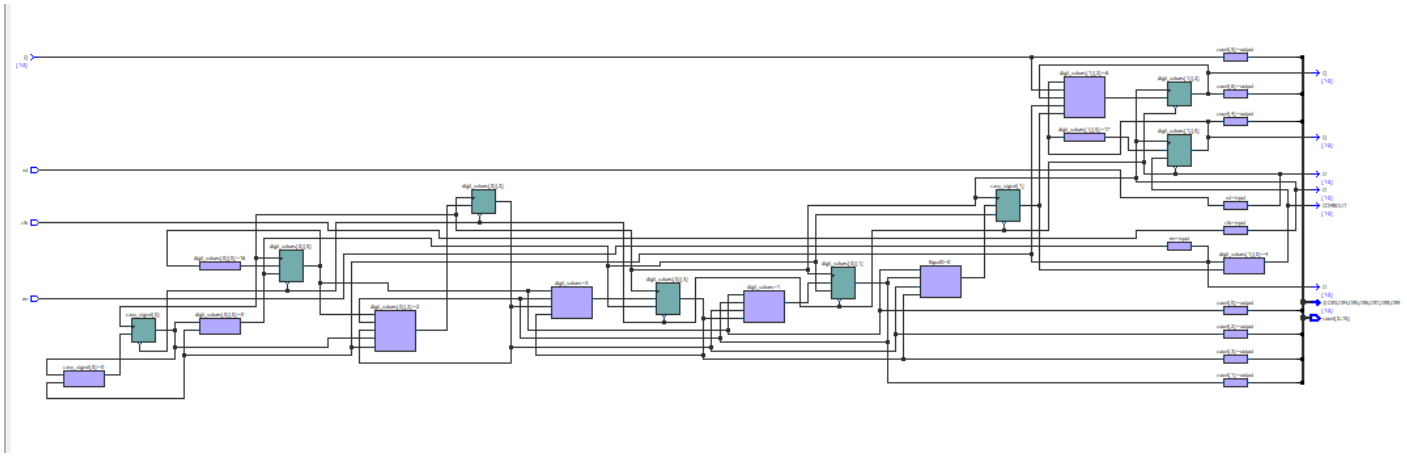


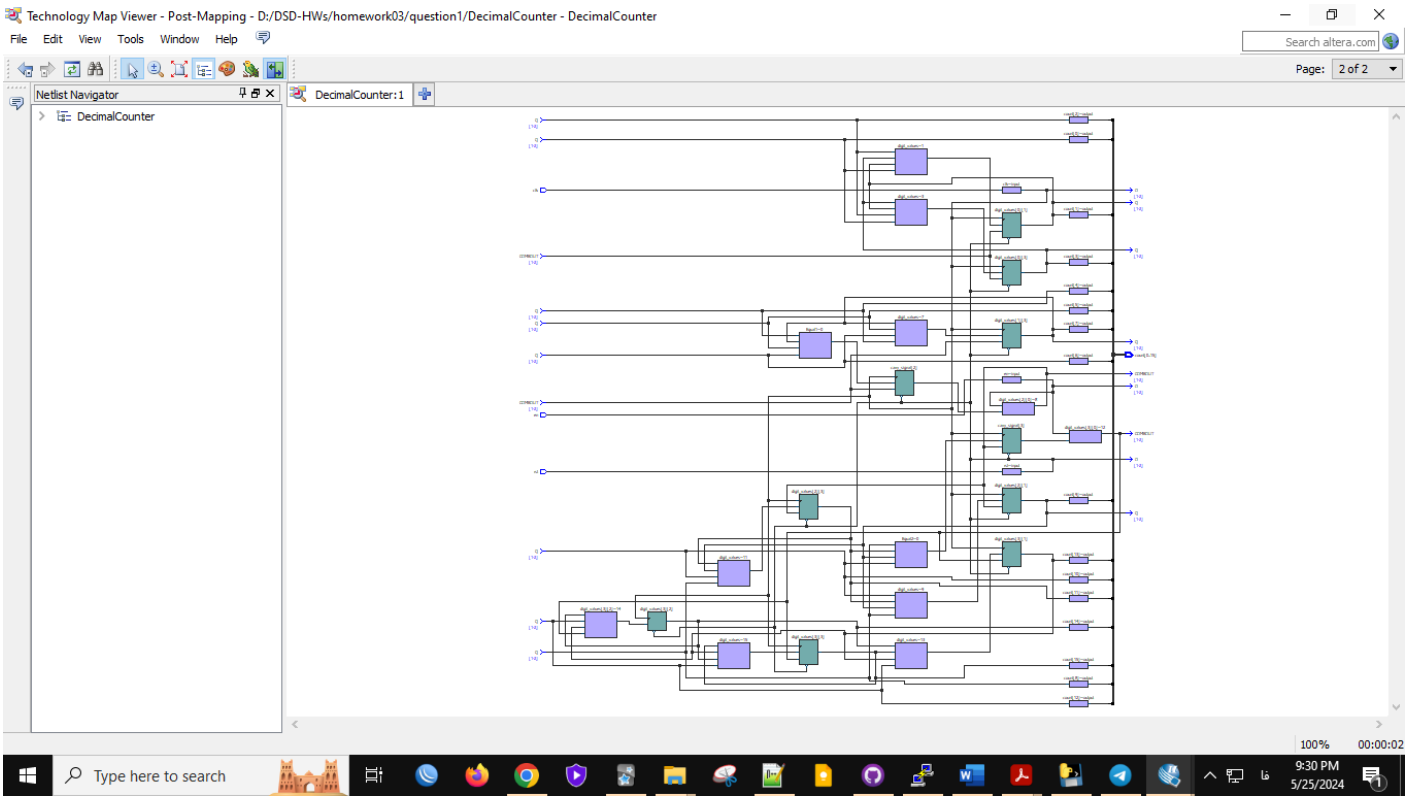
❖RTL View:



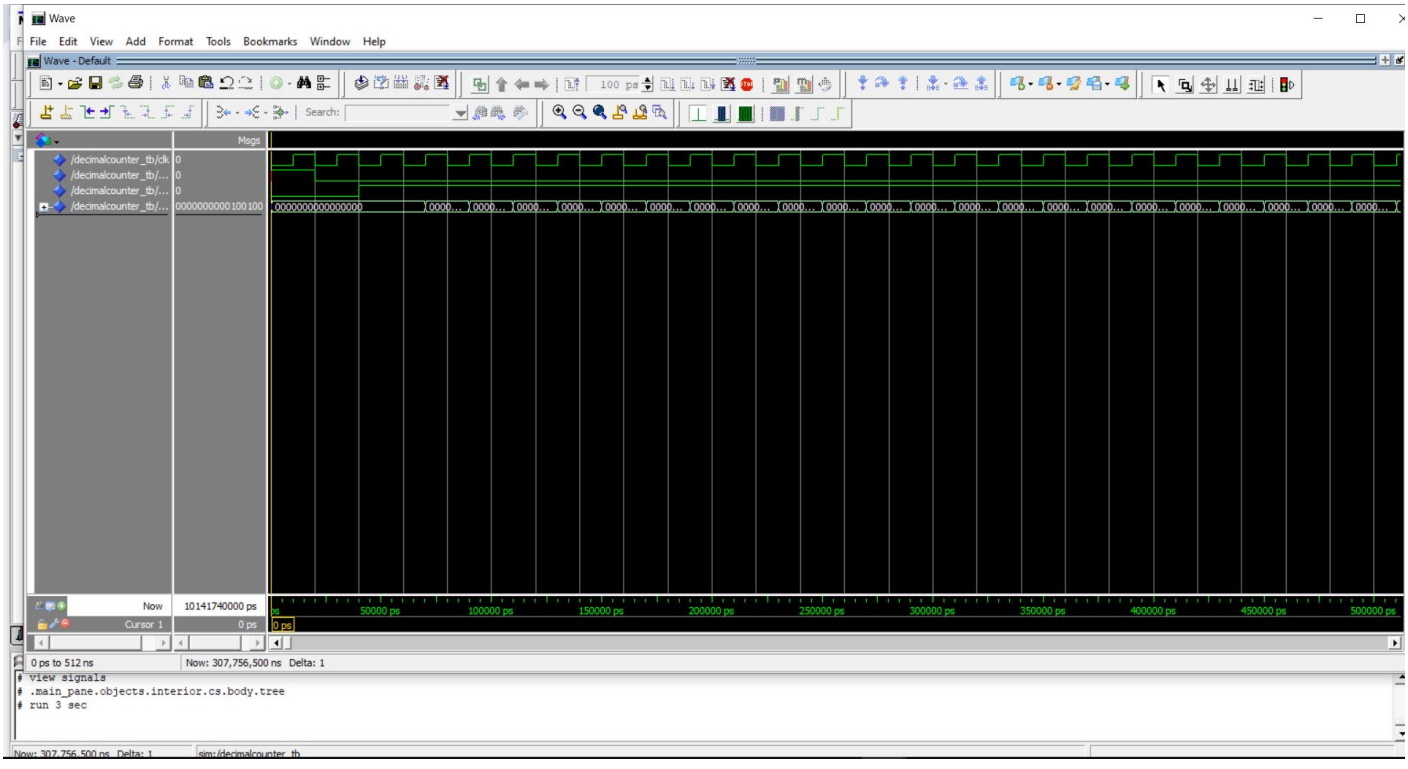


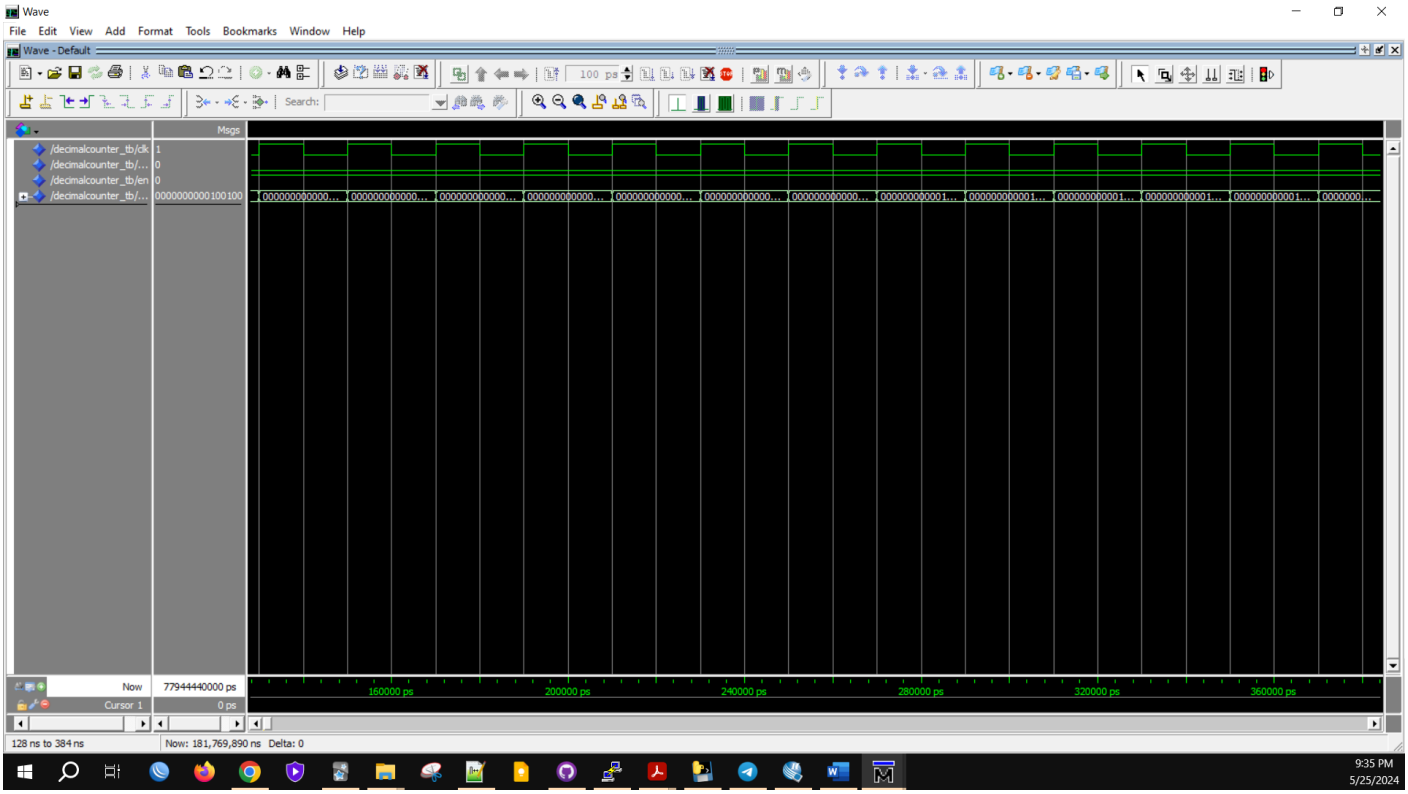
❖Post-mapping:

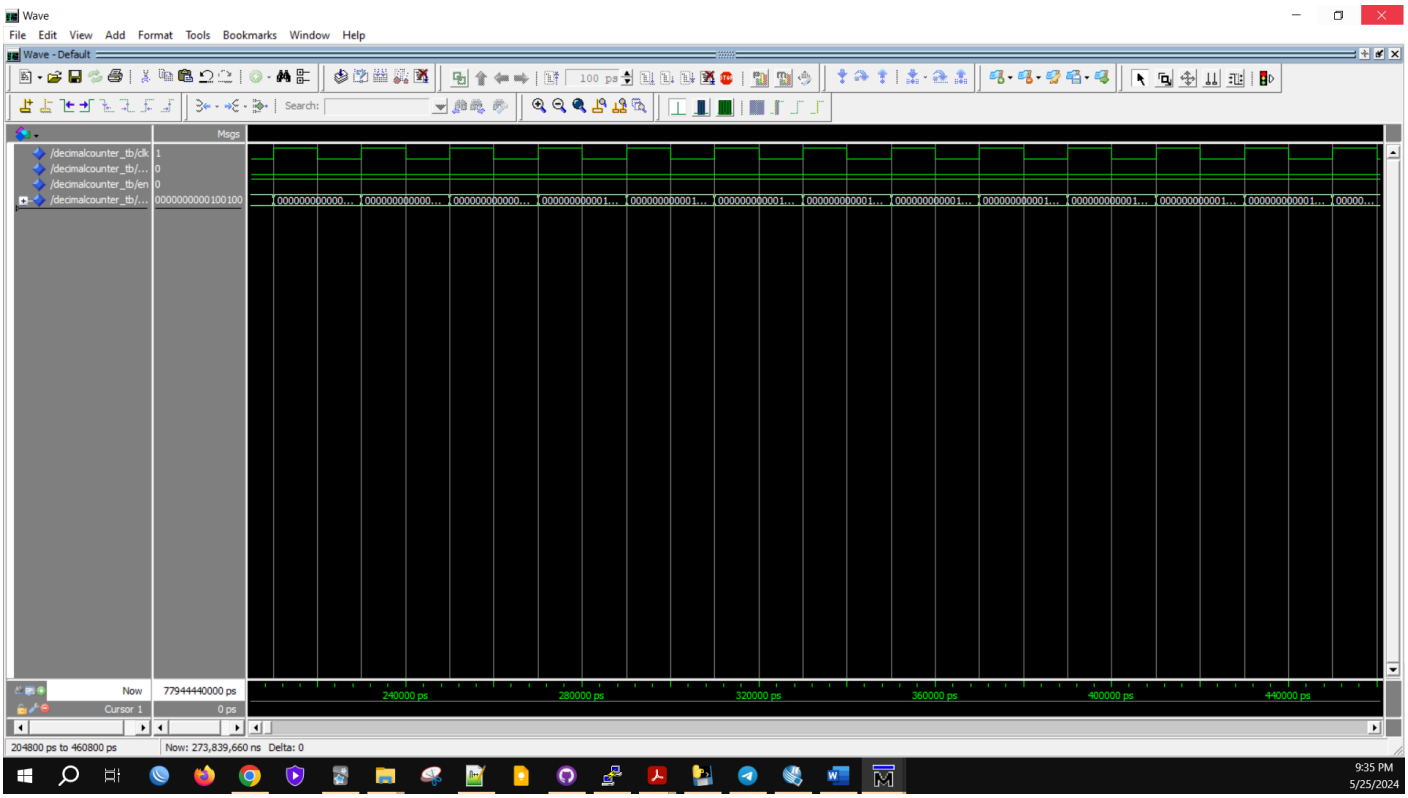


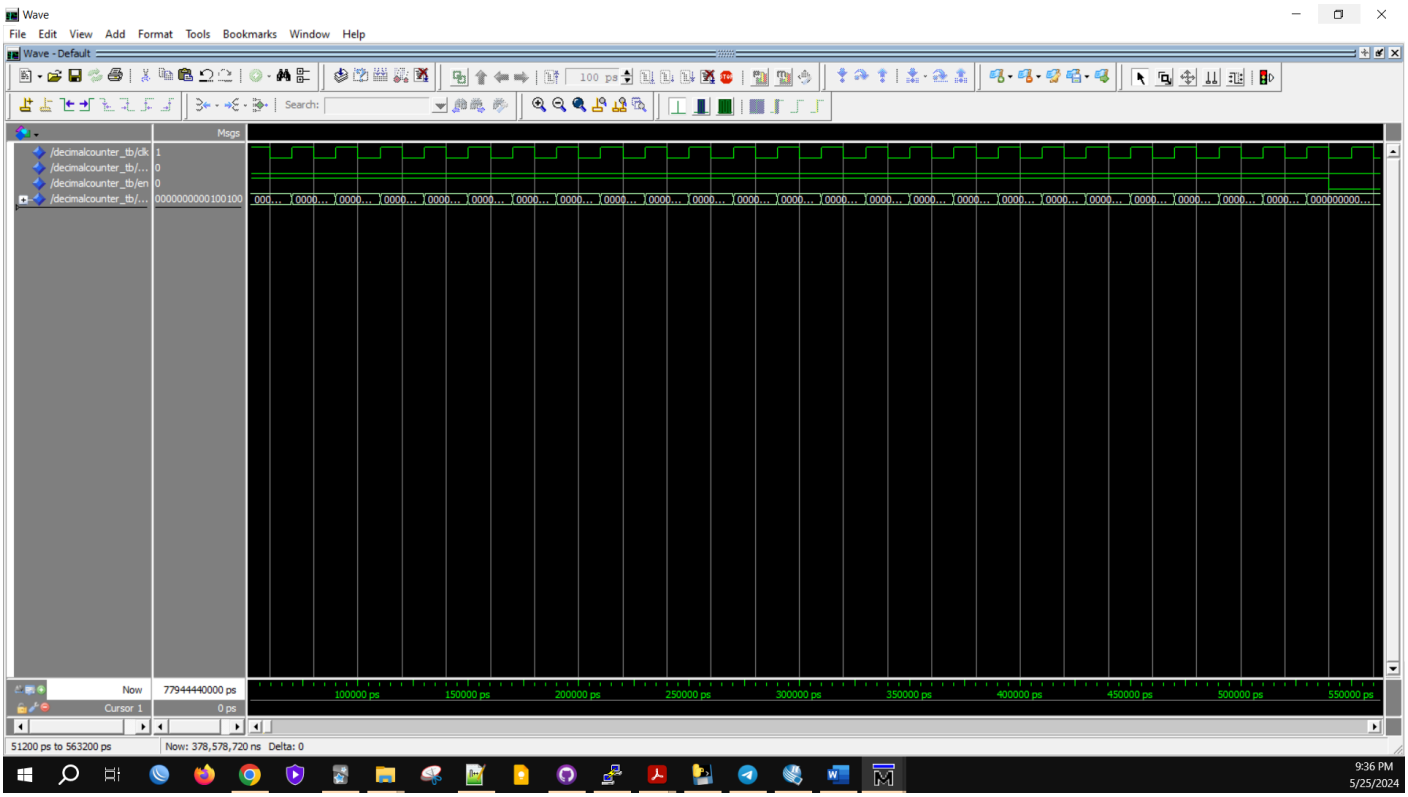

Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question1/DecimalCounter - DecimalCounter
File Edit View Tools Window Help
Search altera.com
Page: 2 of 2
Netlist Navigator
DecimalCounter
DecimalCounter:1
100%
00:00:02
Type here to search
9:30 PM
5/25/2024

❖Wave Form:


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/decimalcounter_tb/clk
/decimalcounter_tb/...
/decimalcounter_tb/...
/decimalcounter_tb/...
Now
10141740000 ps
Cursor 1
0 ps
50000 ps
100000 ps
150000 ps
200000 ps
250000 ps
300000 ps
350000 ps
400000 ps
450000 ps
500000 ps
0 ps to 512 ns
Now: 307,756,500 ns Delta: 1
view signals
.main_pane.objects.interior.cs.body.tree
run 3 sec
Now: 307,756,500 ns Delta: 1
sim:/decimalcounter_tb


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
/decimalcounter_tb/clk
/decimalcounter_tb/...
/decimalcounter_tb/en
/decimalcounter_tb/...
Msgs
Now
77944440000 ps
Cursor 1
0 ps
160000 ps
200000 ps
240000 ps
280000 ps
320000 ps
360000 ps
128 ns to 384 ns
Now: 181,769,890 ns Delta: 0
9:35 PM
5/25/2024


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
/decimalcounter_tb/clk
/decimalcounter_tb/...
/decimalcounter_tb/en
/decimalcounter_tb/...
Msgs
Now
77944440000 ps
Cursor 1
0 ps
240000 ps
280000 ps
320000 ps
360000 ps
400000 ps
440000 ps
204800 ps to 460800 ps
Now: 273,839,660 ns Delta: 0
9:35 PM
5/25/2024



❖Test bench code:

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

entity DecimalCounter_tb is
end DecimalCounter_tb;

architecture Behavioral of DecimalCounter_tb is
    constant NUM_DIGITS : integer := 4; -- Number of digits for testing
    signal clk    : STD_LOGIC := '0';
```

```
    signal rst     : STD_LOGIC := '0';
    signal en      : STD_LOGIC := '0';
    signal count    : STD_LOGIC_VECTOR((NUM_DIGITS*4)-1 downto
0);

    component DecimalCounter
        generic (
            NUM_DIGITS : integer
        );
        Port (
            clk     : in  STD_LOGIC;
            rst     : in  STD_LOGIC;
            en      : in  STD_LOGIC;
            count       :   out   STD_LOGIC_VECTOR((NUM_DIGITS*4)-1
downto 0)
        );
    end component;

begin

    UUT: DecimalCounter
        generic map (
            NUM_DIGITS => 4 -- Specify 4 digits
```

```
    )
    port map (
        clk => clk,
        rst => rst,
        en => en,
        count => count
    );

-- Clock generation
clk_process: process
begin
    while True loop
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
    end loop;
end process;

-- Stimulus process
stimulus: process
begin
```

```
        -- Apply reset
        rst <= '1';
        wait for 20 ns;
        rst <= '0';
        wait for 20 ns;

        -- Enable counting
        en <= '1';
        wait for 500 ns;

        -- Disable counting
        en <= '0';
        wait for 20 ns;

        wait;
    end process;

end Behavioral;
```

Q2. So for this question we should design a FIFO buffer for 64-bit system. We need 8 slots, with each slot capable of holding an 8-byte (64 bit) word. Two main components: Register File and FIFO Controller. It should handle write "WR" and read "RD" operations. The FIFO should be able to indicate when it is full "FULL" or empty "EMPTY".

❖ **For this FIFO buffer:**

- First, we define entity:

We need these ports: "CLK", "RESET", "WR", "RD", "W_DATA", "R_DATA", "FULL" AND "EMPTY"

❖ **For Register File:**

We need an array of 8 slots, each 64 bits wide.

And for FIFO Controller we need handle read and write signals.

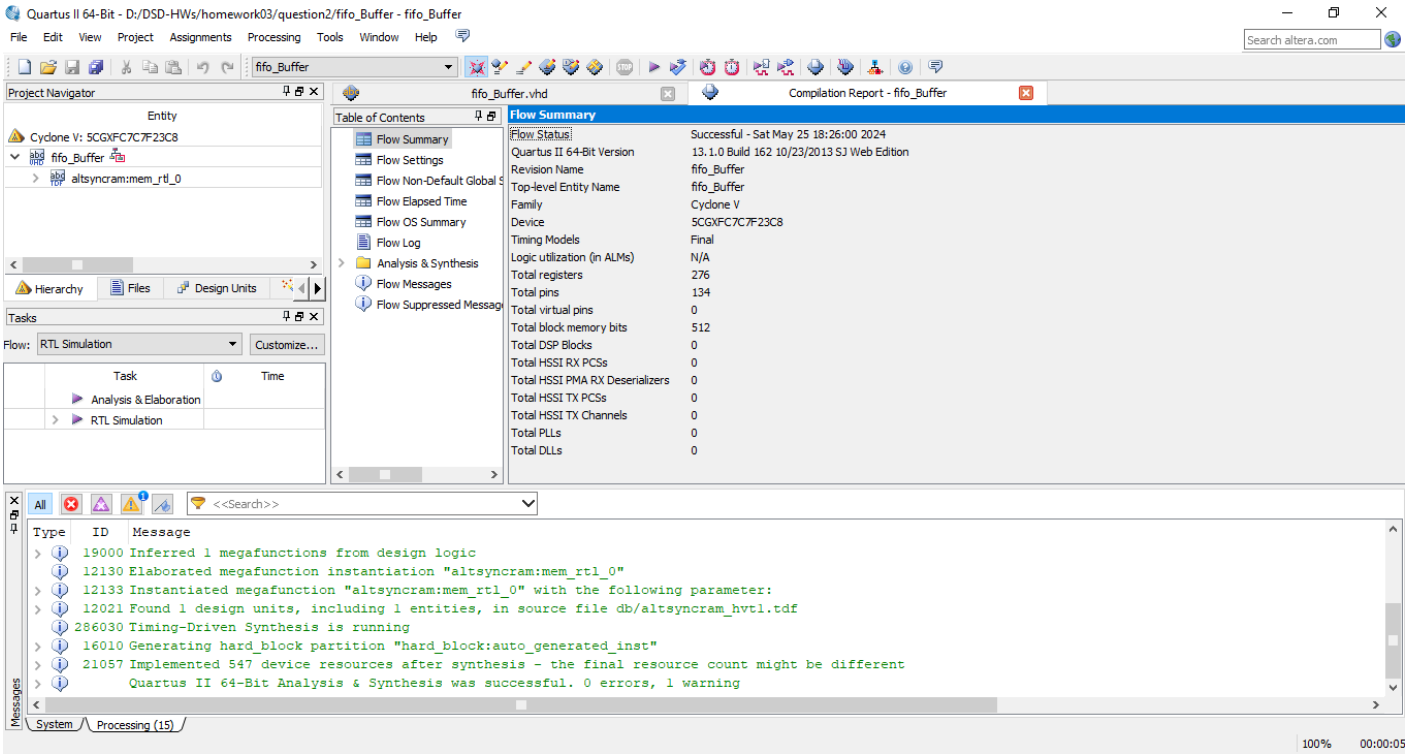
❖Code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity fifo_Buffer is
    port (
        clk: in std_logic;
        reset: in std_logic;
        wr: in std_logic;
        rd: in std_logic;
        w_data: in std_logic_vector (63 downto 0);
        r_data: out std_logic_vector (63 downto 0);
        full: out std_logic;
        empty: out std_logic
    );
end fifo_Buffer;

architecture Behavioral of fifo_Buffer is
```

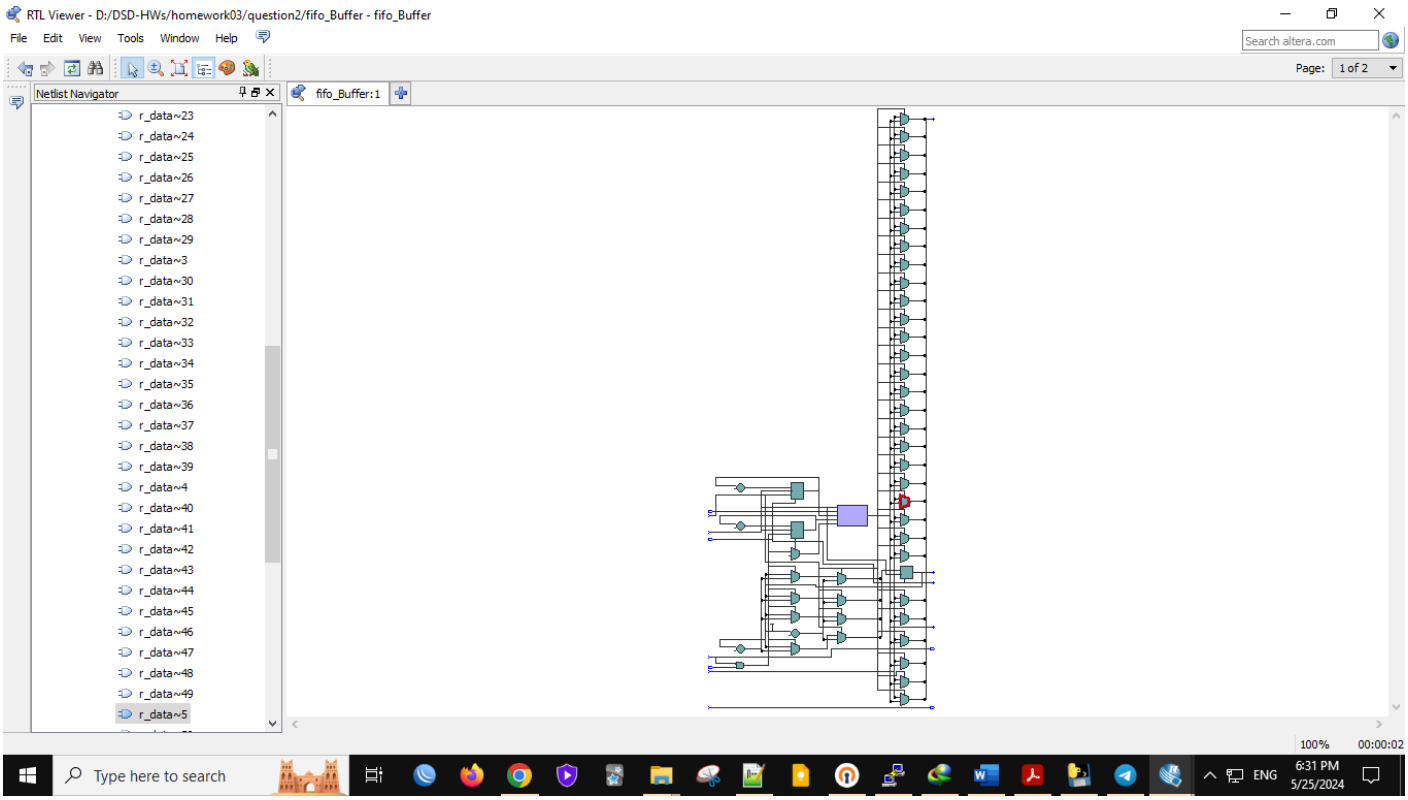
```
    type mem_type is array (0 to 7) of std_logic_vector(63 downto 0);
    signal mem: mem_type := (others => (others => '0'));
    signal w_ptr, r_ptr: integer range 0 to 7 := 0;
    signal count: integer range 0 to 8 := 0;
    signal full_signal, empty_signal: std_logic;
begin
    full <= full_signal;
    empty <= empty_signal;

    process(clk, reset)
    begin
        if reset = '1' then
            w_ptr <= 0;
            r_ptr <= 0;
            count <= 0;
            full_signal <= '0';
            empty_signal <= '1';
        elsif rising_edge(clk) then
            if wr = '1' and full_signal = '0' then
                mem(w_ptr) <= w_data;
                w_ptr <= (w_ptr + 1) mod 8;
                count <= count + 1;
```

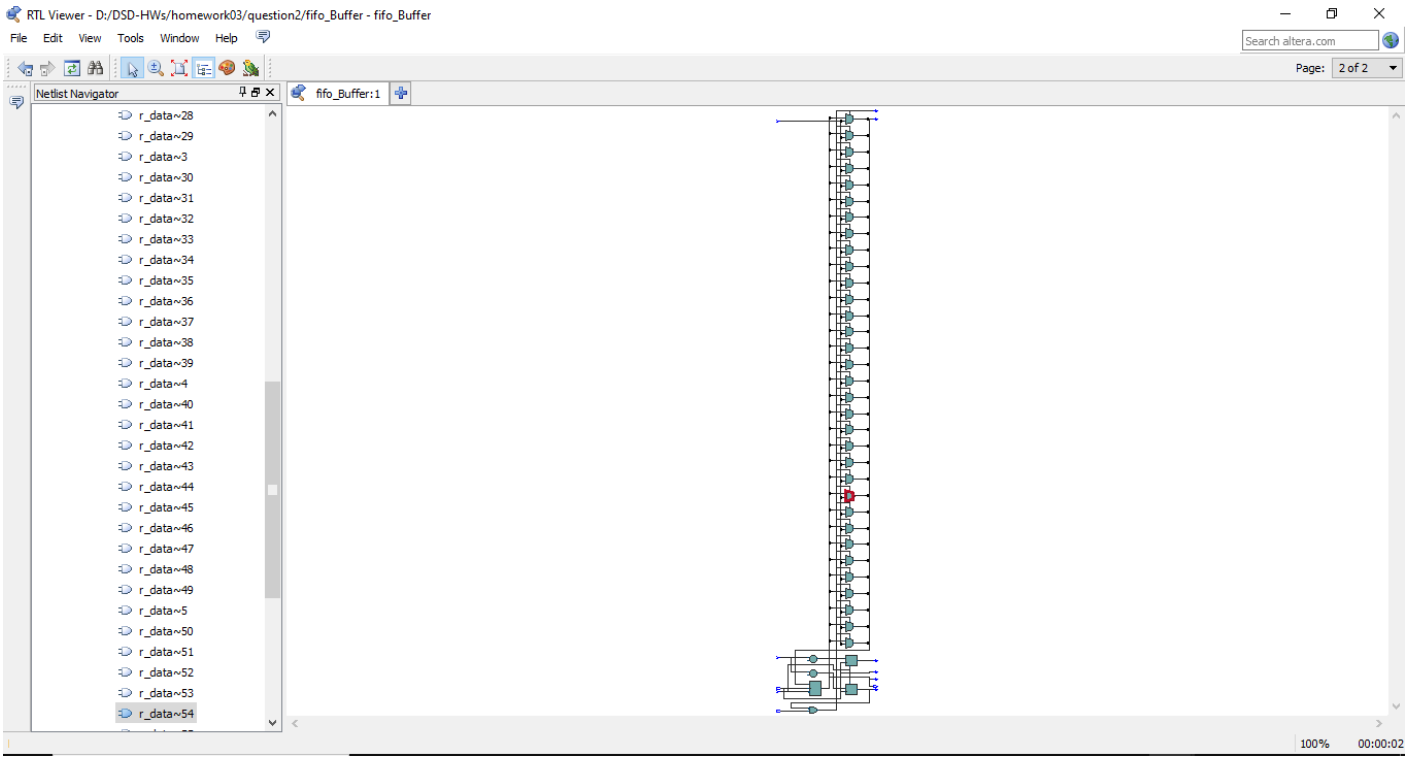
```
        end if;

        if rd = '1' and empty_signal = '0' then
            r_data <= mem(r_ptr);
            r_ptr <= (r_ptr + 1) mod 8;
            count <= count - 1;
        end if;

        if count = 8 then
            full_signal <= '1';
        else
            full_signal <= '0';
        end if;

        if count = 0 then
            empty_signal <= '1';
        else
            empty_signal <= '0';
        end if;
    end if;
  end process;
end Behavioral;
```

❖Compilation Report:



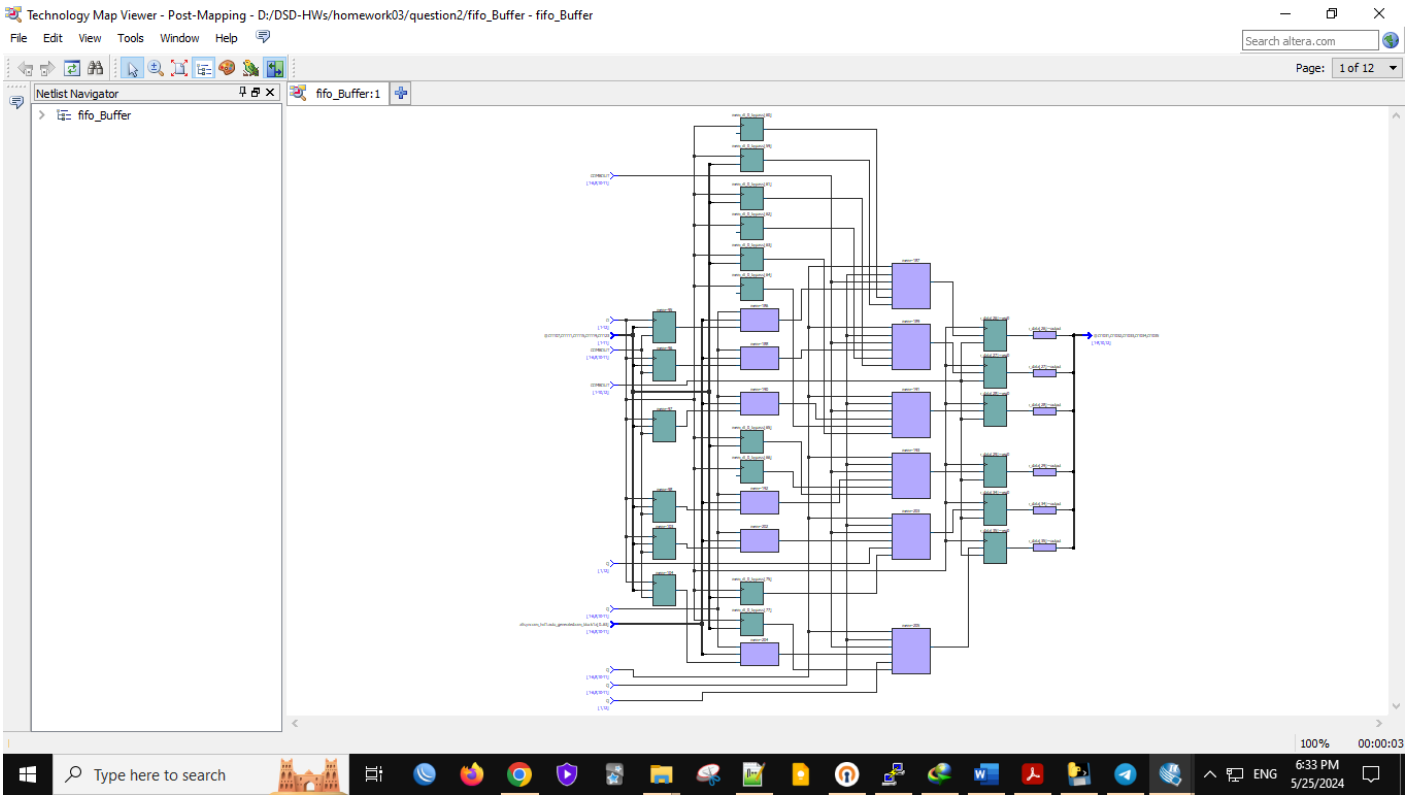
❖RTL View:


RTL Viewer - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 1 of 2
Netlist Navigator
fifo_Buffer:1
r_data~23
r_data~24
r_data~25
r_data~26
r_data~27
r_data~28
r_data~29
r_data~3
r_data~30
r_data~31
r_data~32
r_data~33
r_data~34
r_data~35
r_data~36
r_data~37
r_data~38
r_data~39
r_data~4
r_data~40
r_data~41
r_data~42
r_data~43
r_data~44
r_data~45
r_data~46
r_data~47
r_data~48
r_data~49
r_data~5
100%
00:00:02
Type here to search
ENG
6:31 PM
5/25/2024

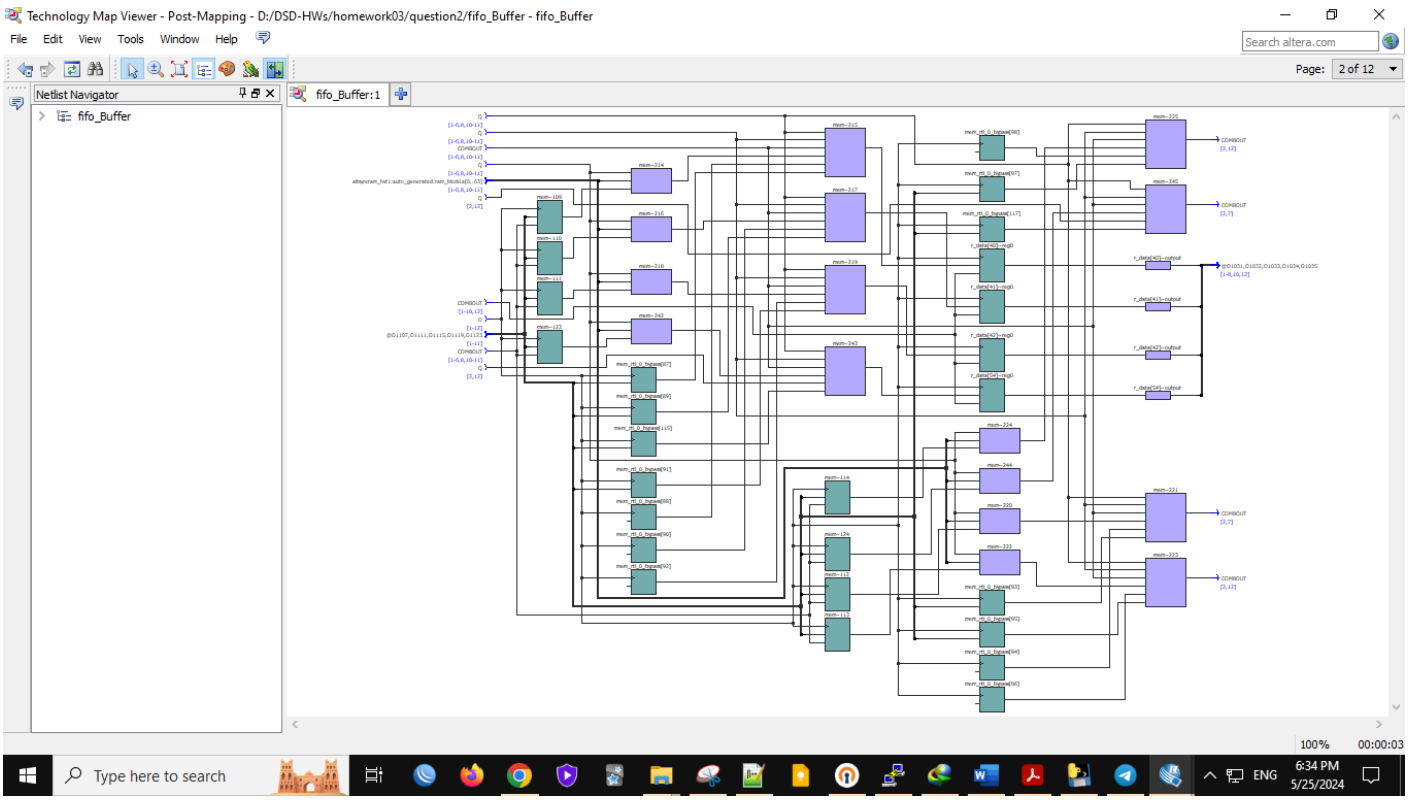

RTL Viewer - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 2 of 2
Netlist Navigator
fifo_Buffer:1
r_data~28
r_data~29
r_data~3
r_data~30
r_data~31
r_data~32
r_data~33
r_data~34
r_data~35
r_data~36
r_data~37
r_data~38
r_data~39
r_data~4
r_data~40
r_data~41
r_data~42
r_data~43
r_data~44
r_data~45
r_data~46
r_data~47
r_data~48
r_data~49
r_data~5
r_data~50
r_data~51
r_data~52
r_data~53
r_data~54
100%
00:00:02

Post-Mapping View:

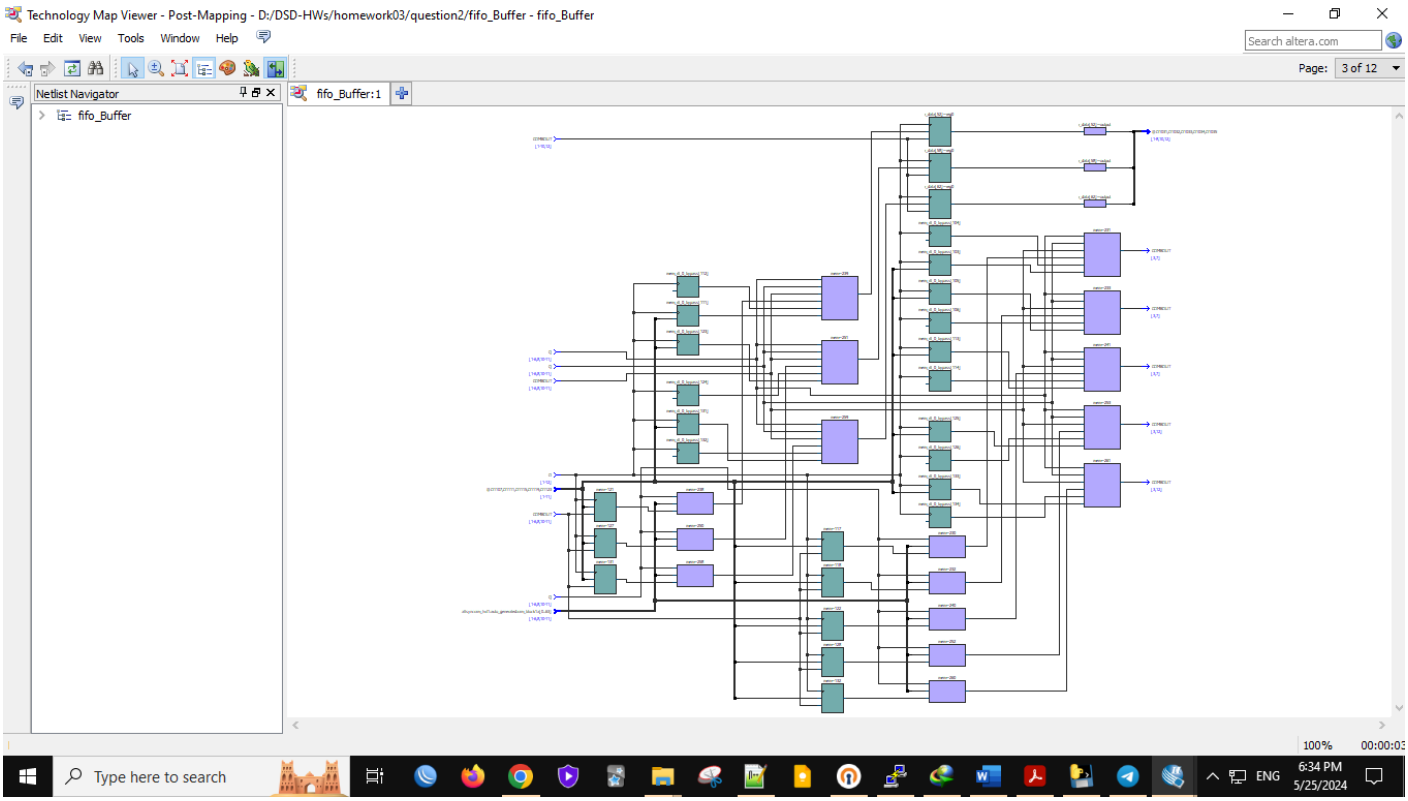
1:



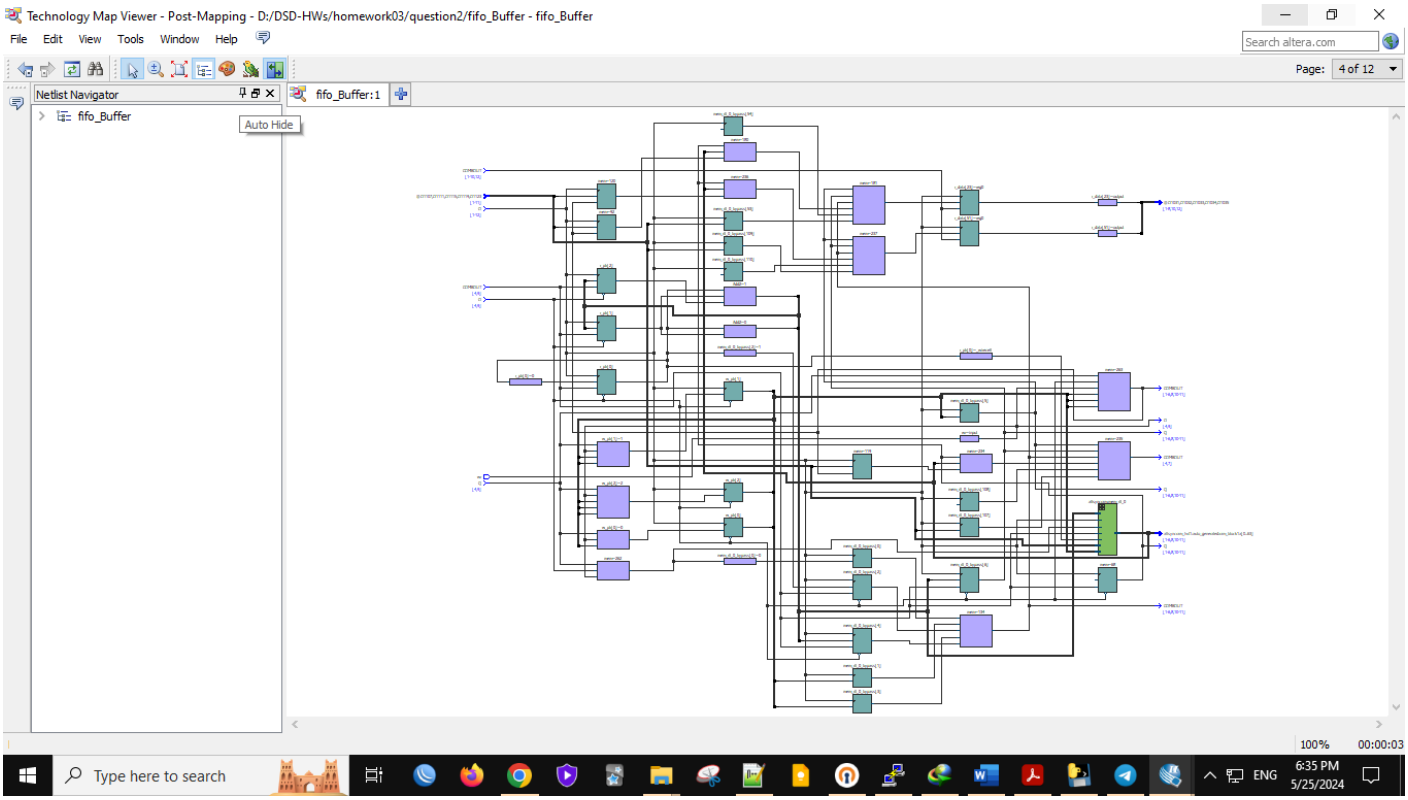
2:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 2 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:34 PM
5/25/2024

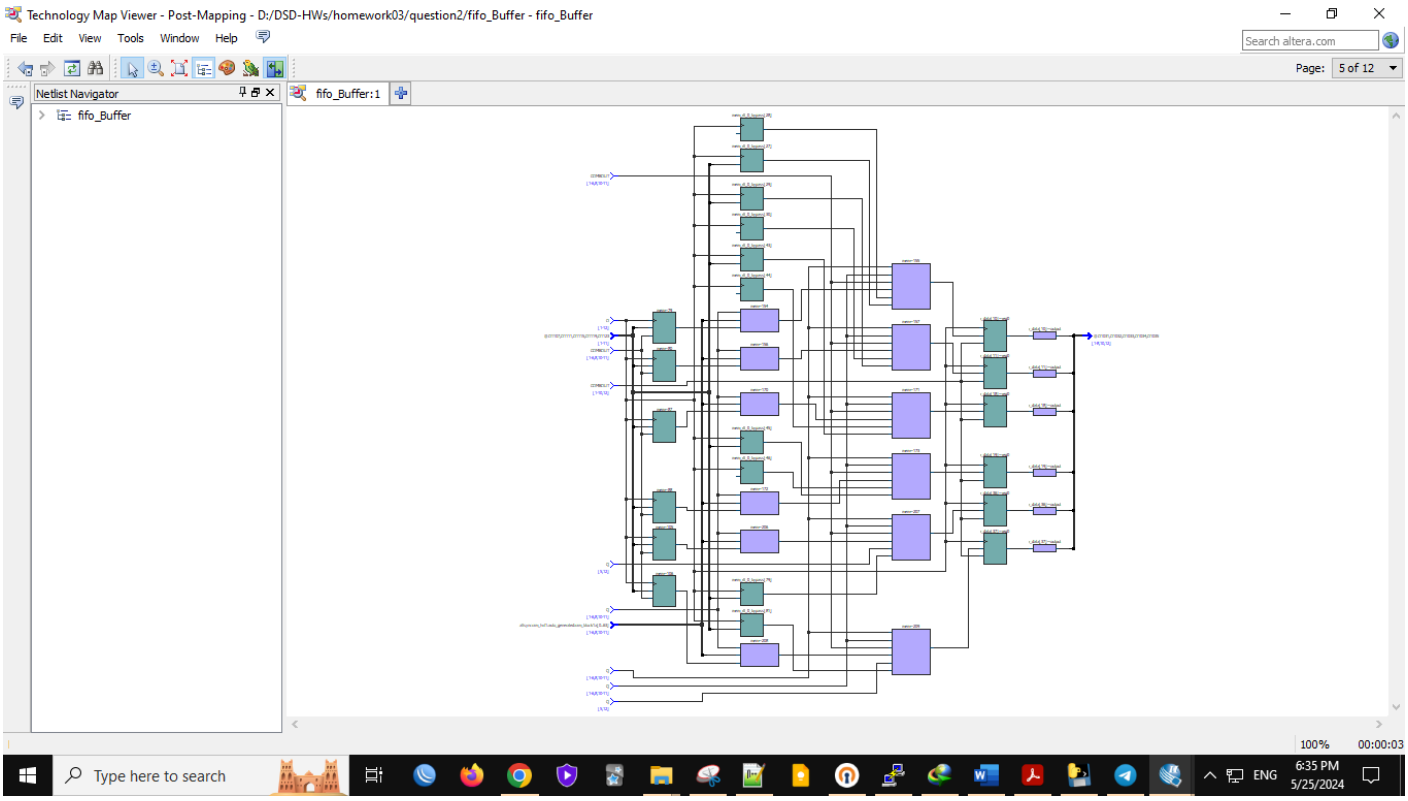
3:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 3 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:34 PM
5/25/2024

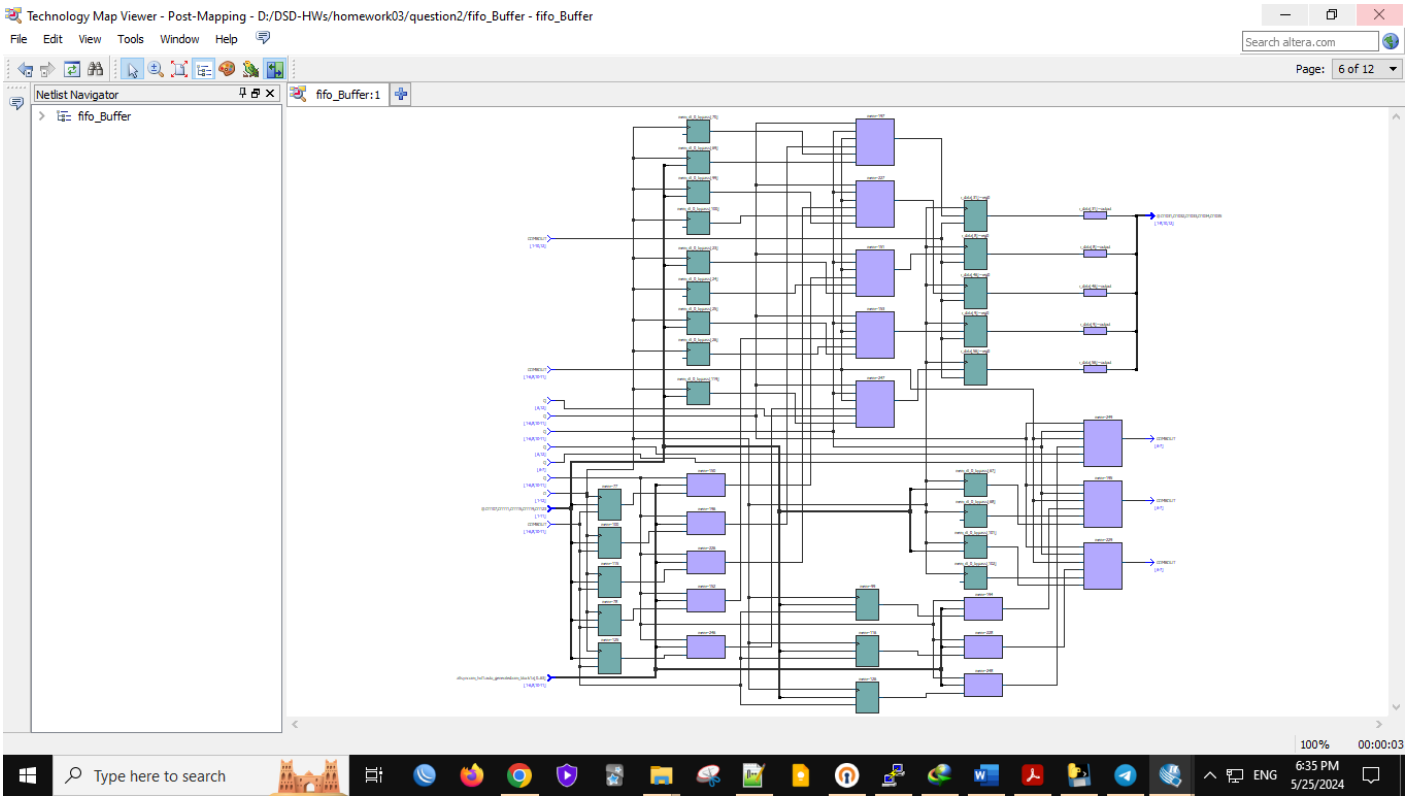
4:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 4 of 12
Netlist Navigator
fifo_Buffer
Auto Hide
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:35 PM
5/25/2024

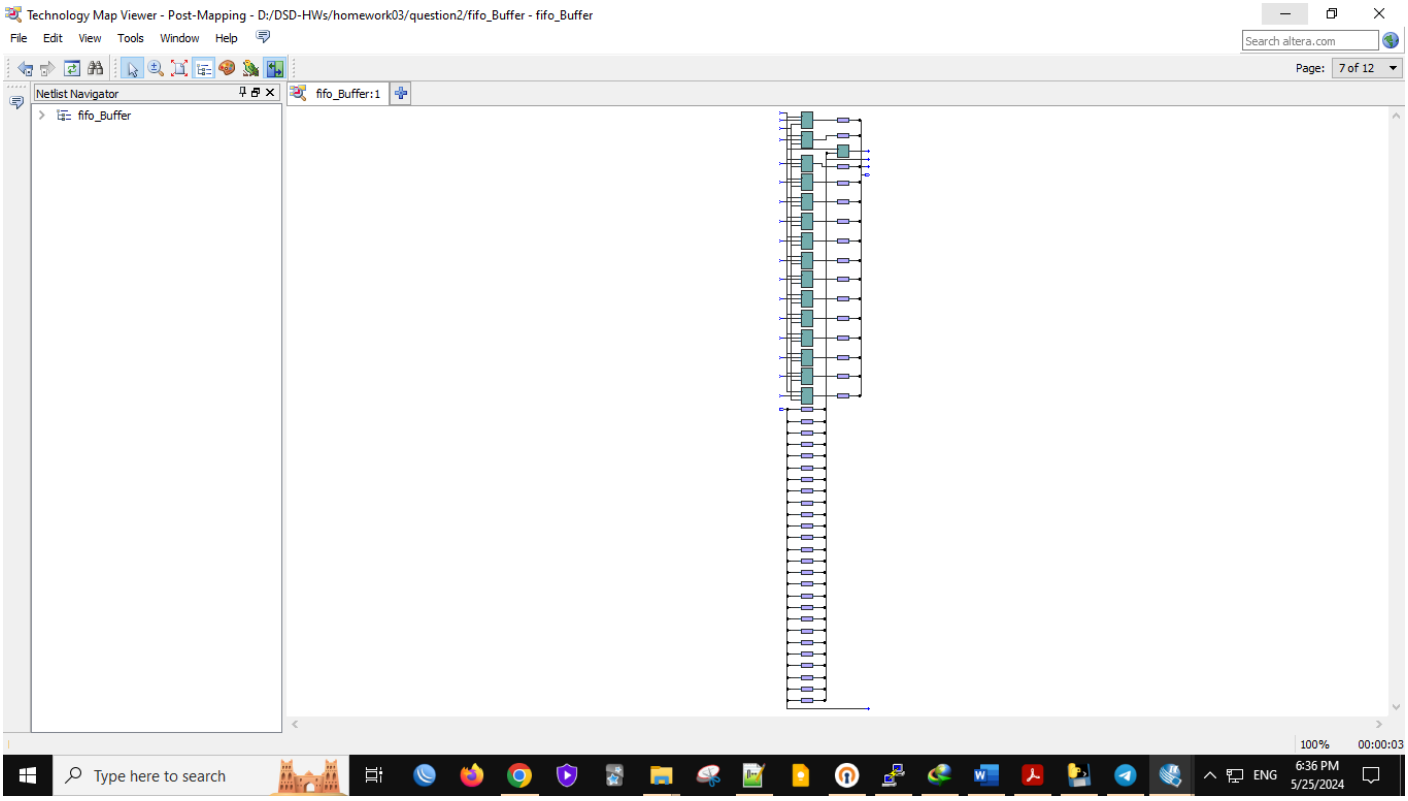
5:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 5 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:35 PM
5/25/2024

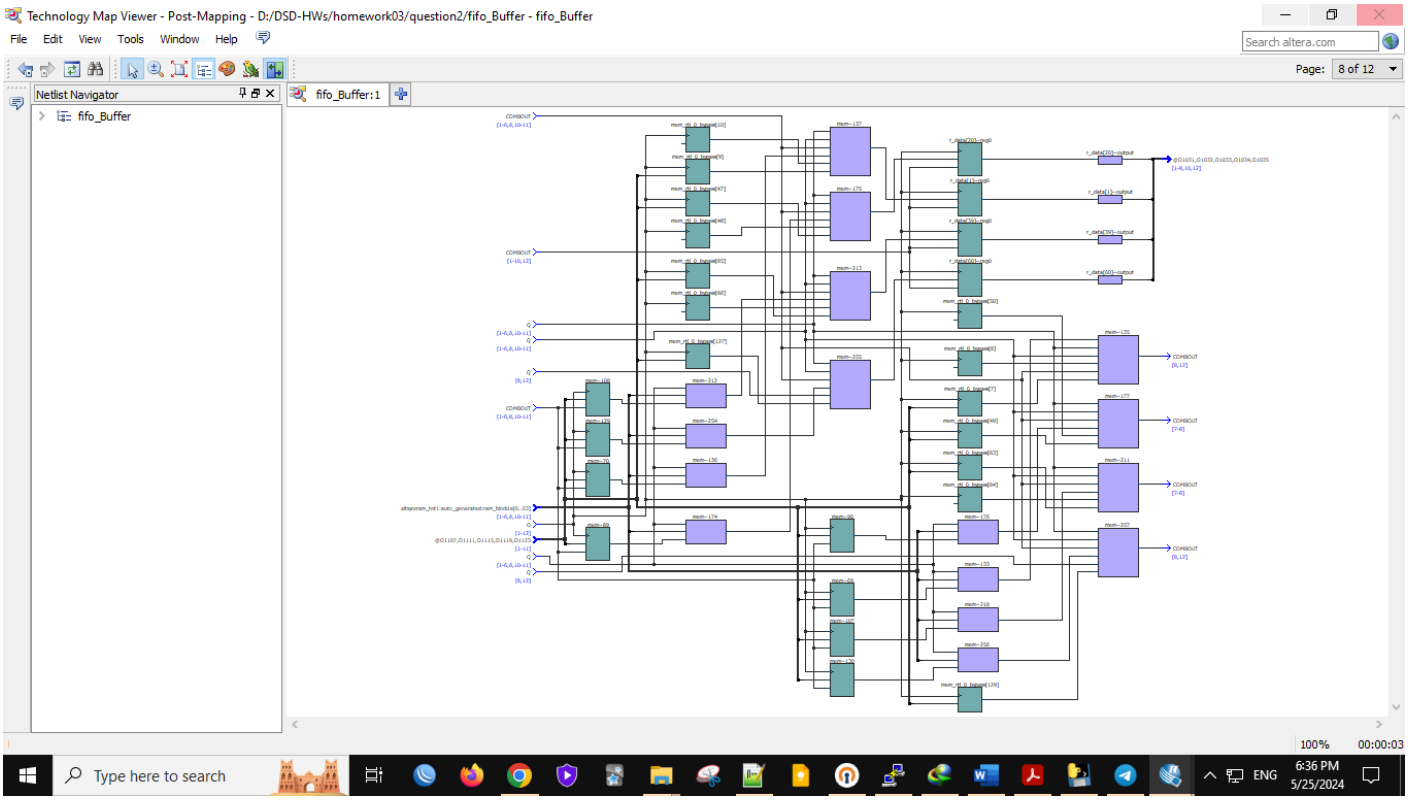
6:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 6 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:35 PM
5/25/2024

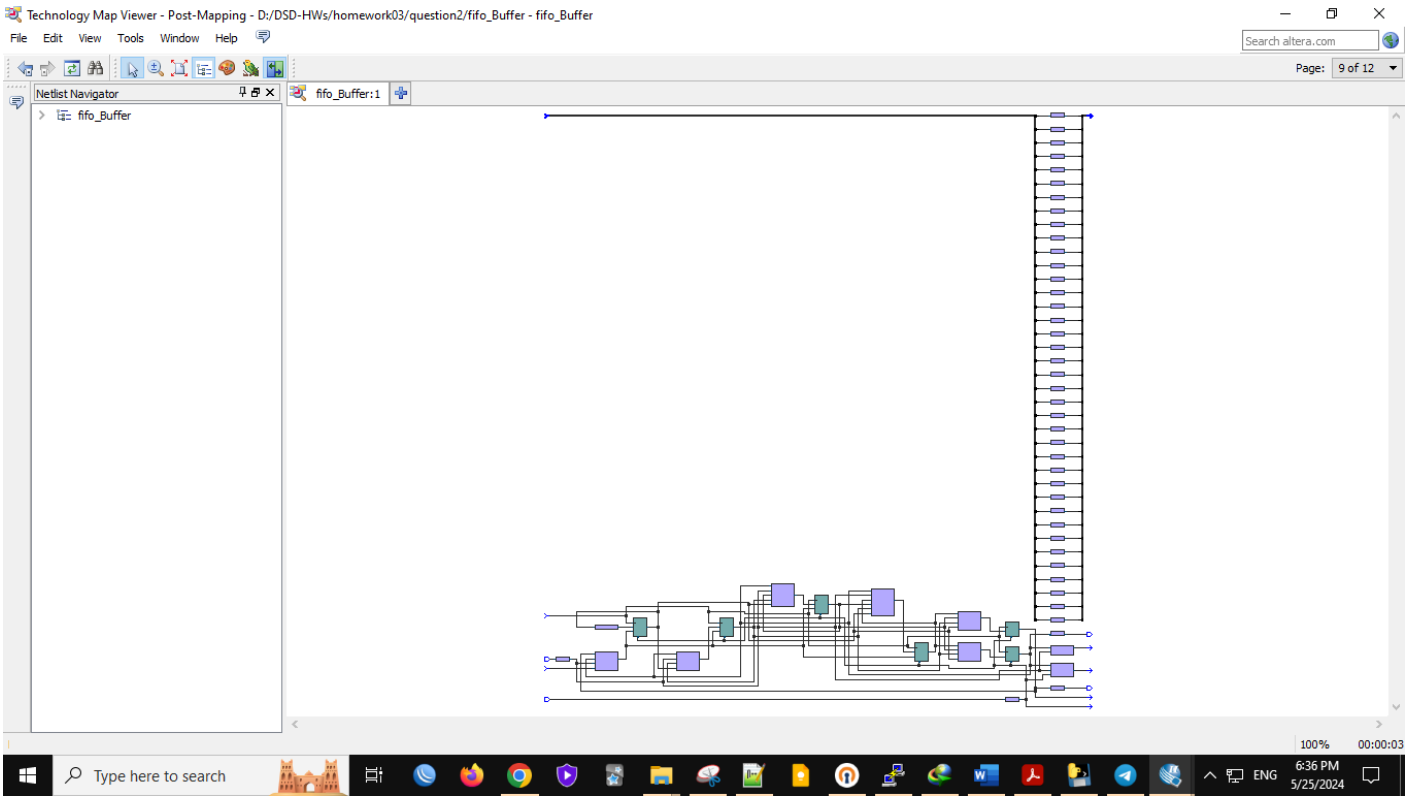
7:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 7 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:36 PM
5/25/2024

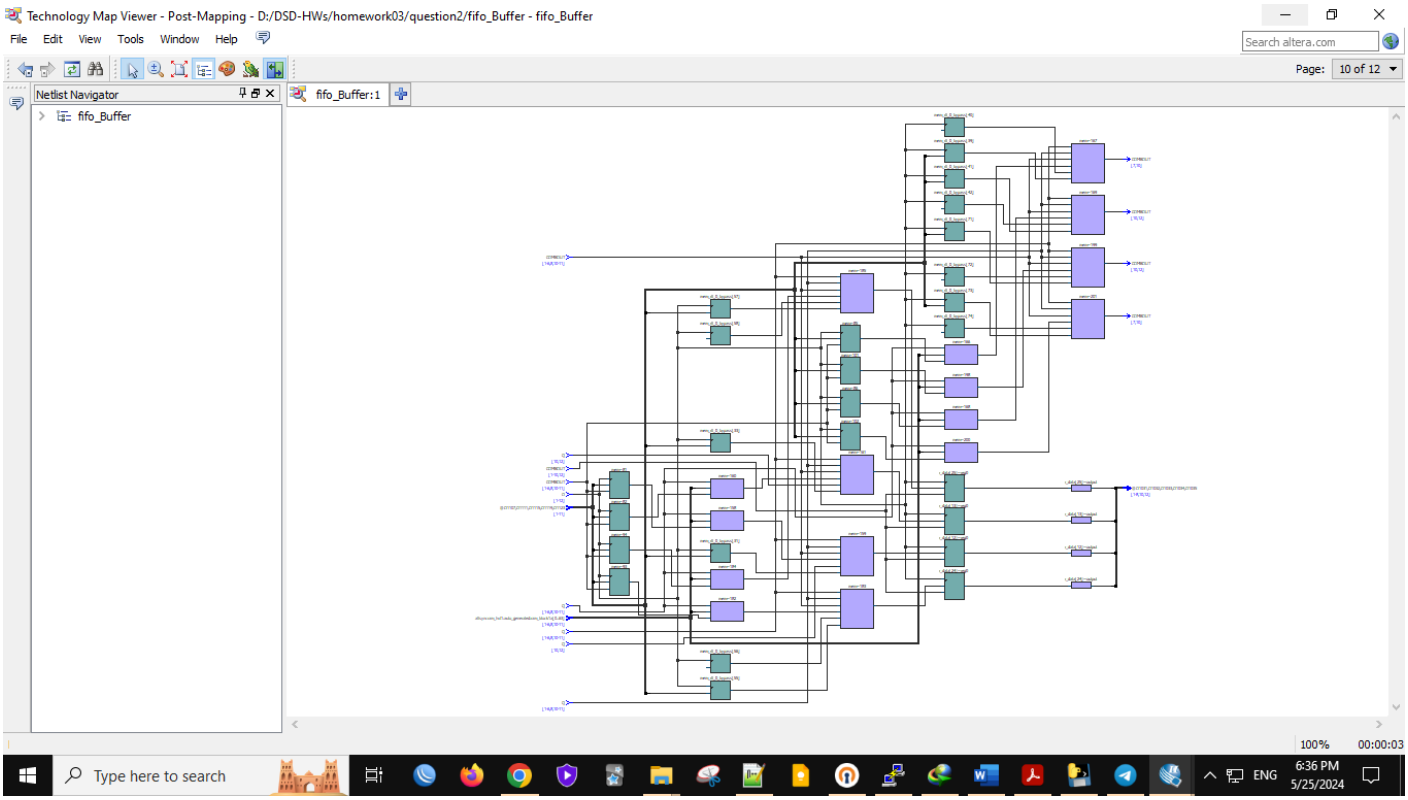
8:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 8 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:36 PM
5/25/2024

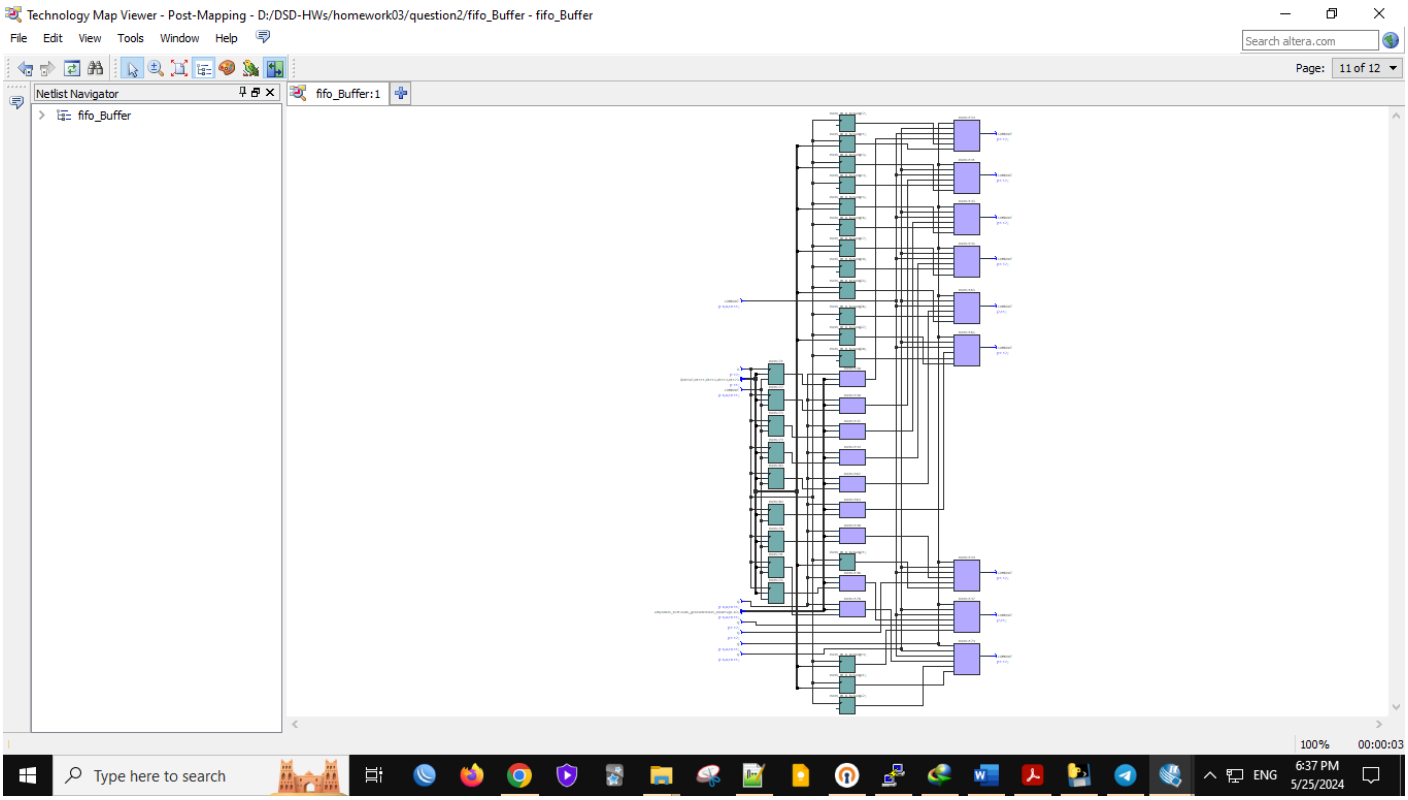
9:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 9 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:36 PM
5/25/2024

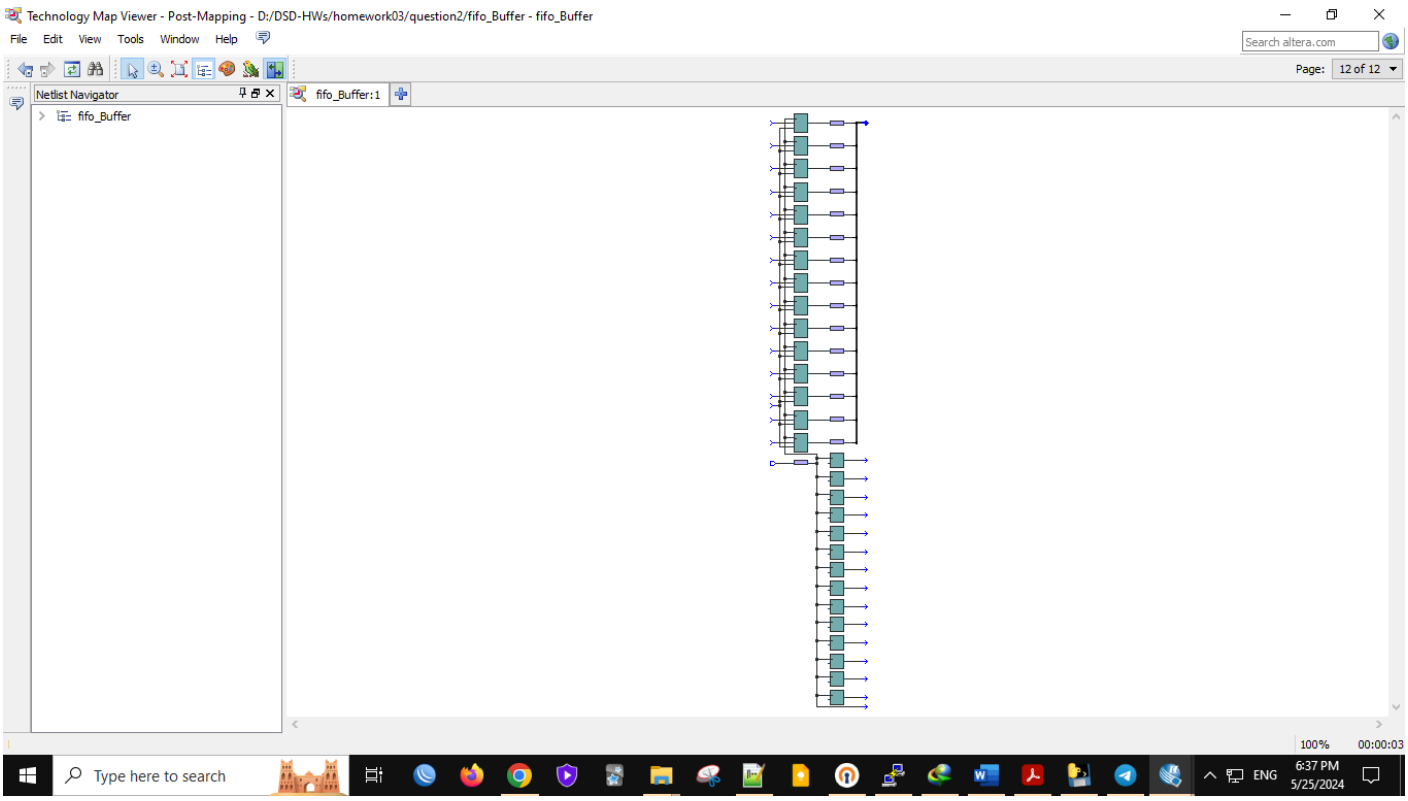
10:


Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 10 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:36 PM
5/25/2024

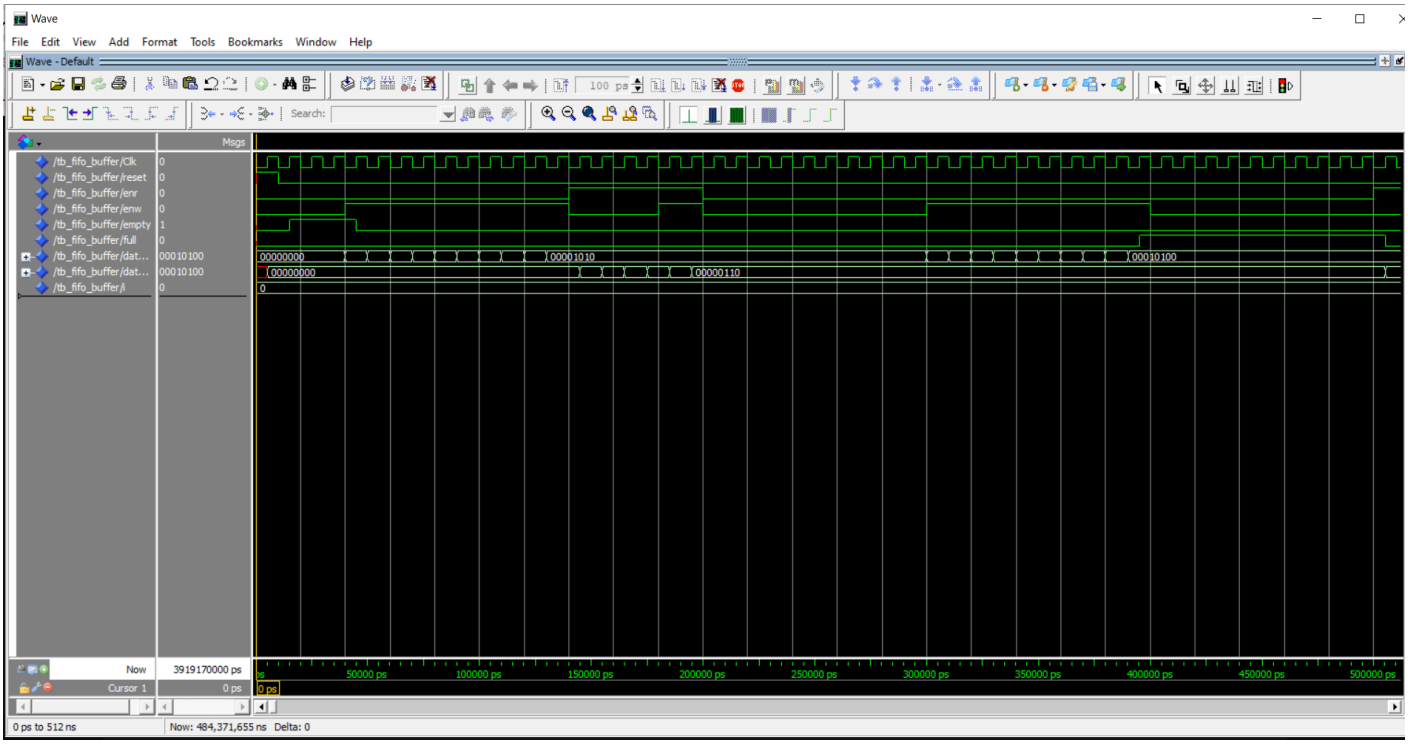
11:

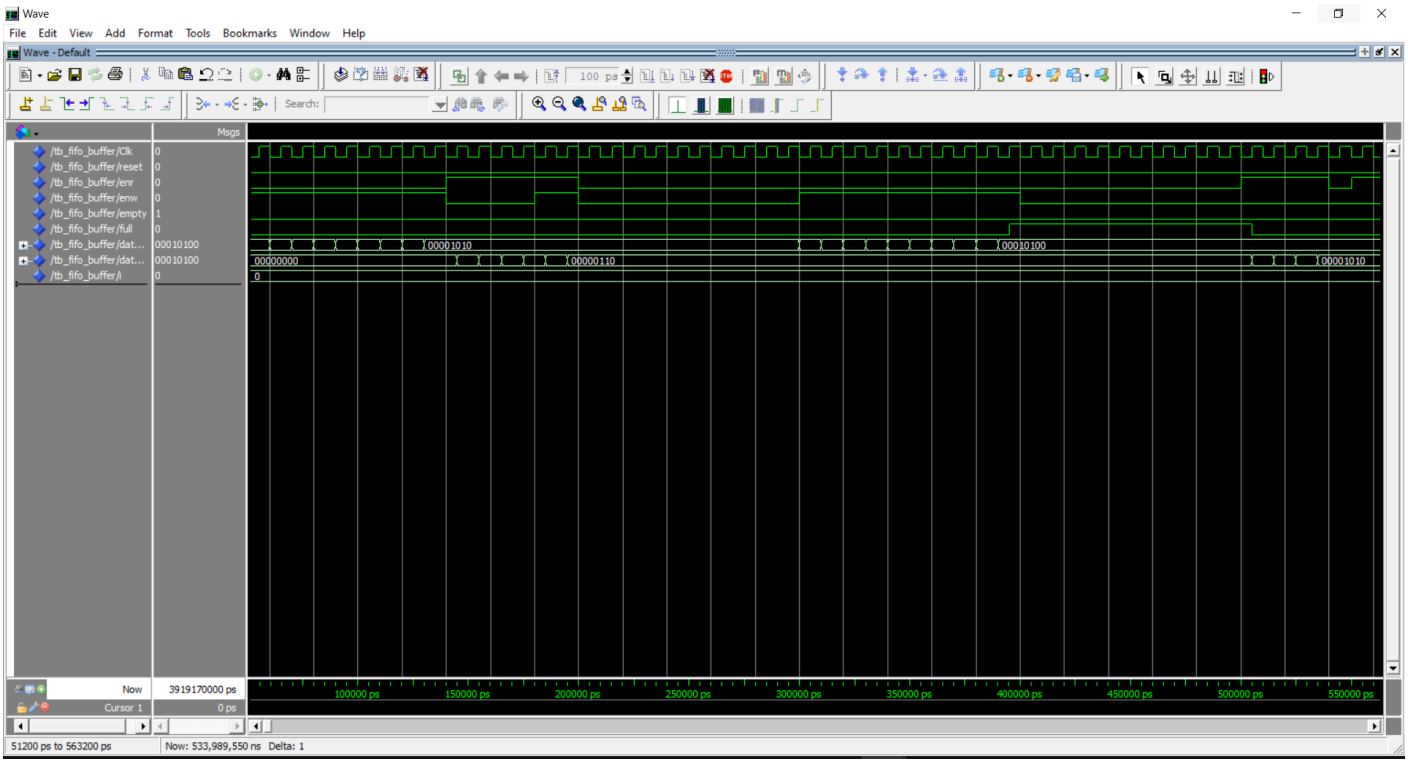

Technology Map Viewer - Post-Mapping - D:/DSD-HWs/homework03/question2/fifo_Buffer - fifo_Buffer
File Edit View Tools Window Help
Search altera.com
Page: 11 of 12
Netlist Navigator
fifo_Buffer
fifo_Buffer:1
100%
00:00:03
Type here to search
ENG
6:37 PM
5/25/2024

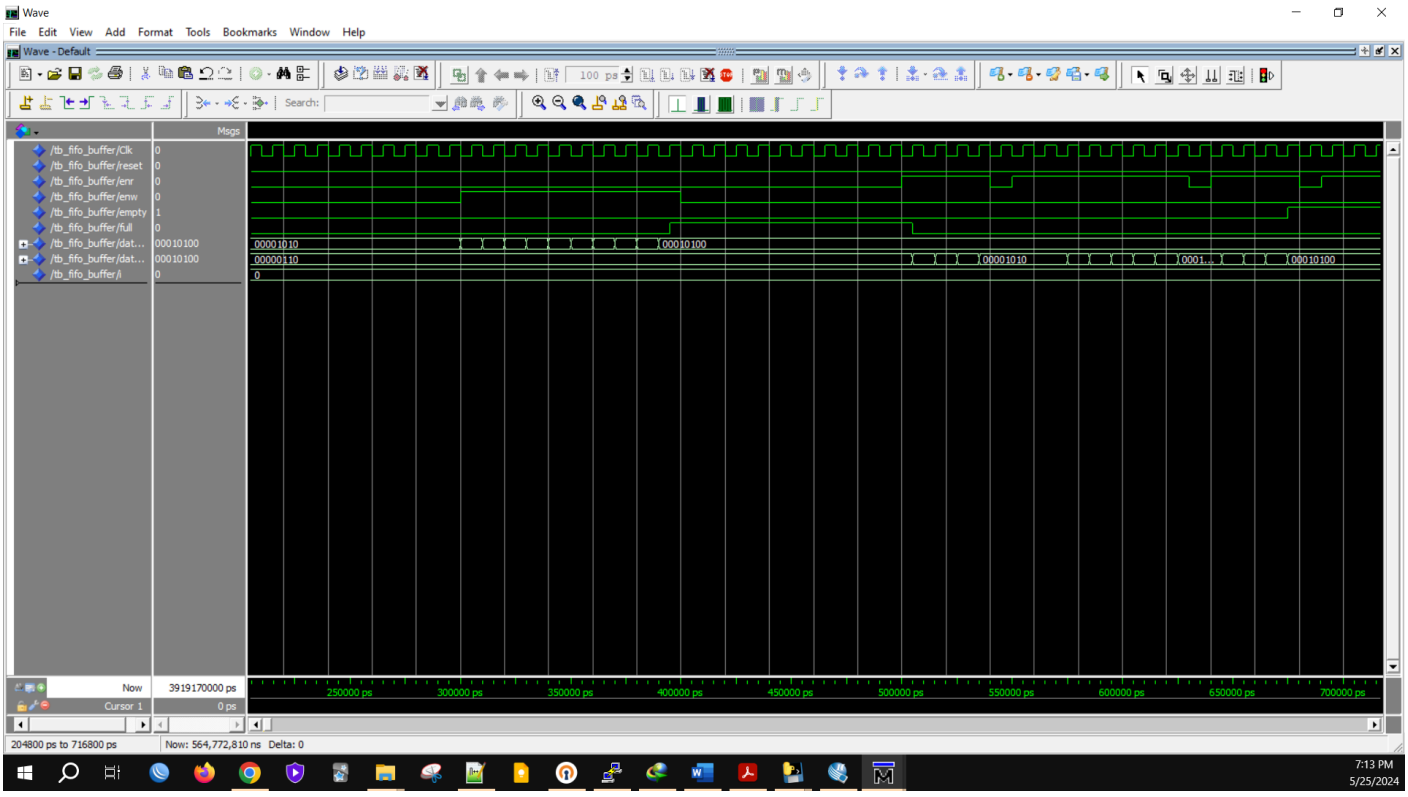
12:

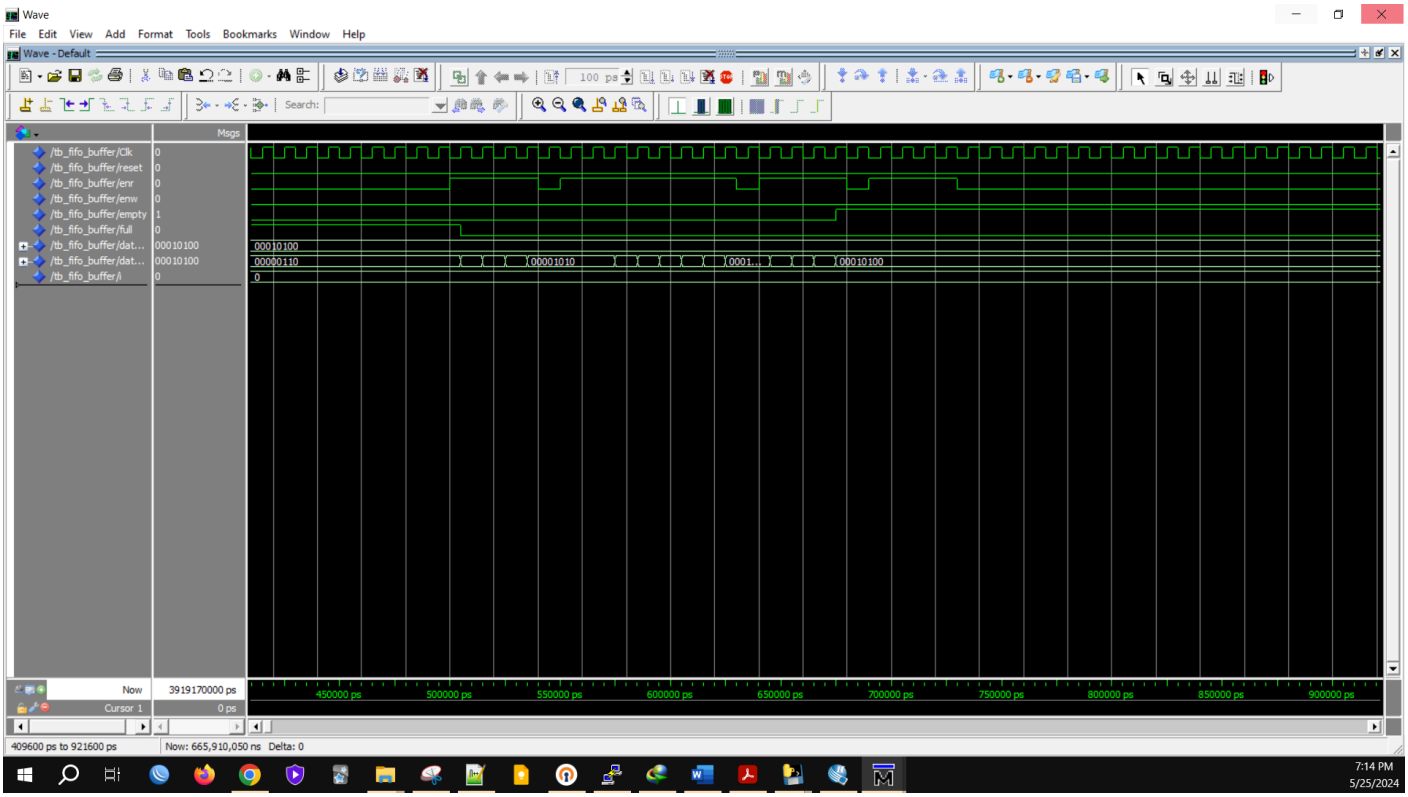


❖Wave Form Code:




Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/tb_fifo_buffer/Clk
/tb_fifo_buffer/reset
/tb_fifo_buffer/enr
/tb_fifo_buffer/enw
/tb_fifo_buffer/empty
/tb_fifo_buffer/full
/tb_fifo_buffer/dat...
/tb_fifo_buffer/dat...
/tb_fifo_buffer/i
00001010
00010100
00000000
00000110
Now
3919170000 ps
Cursor 1
0 ps
100000 ps
150000 ps
200000 ps
250000 ps
300000 ps
350000 ps
400000 ps
450000 ps
500000 ps
550000 ps
51200 ps to 563200 ps
Now: 533,989,550 ns Delta: 1


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/tb_fifo_buffer/Clk
/tb_fifo_buffer/reset
/tb_fifo_buffer/enr
/tb_fifo_buffer/enw
/tb_fifo_buffer/empty
/tb_fifo_buffer/full
/tb_fifo_buffer/dat...
/tb_fifo_buffer/dat...
/tb_fifo_buffer/i
00001010
00010100
00000110
Now
3919170000 ps
Cursor 1
0 ps
250000 ps
300000 ps
350000 ps
400000 ps
450000 ps
500000 ps
550000 ps
600000 ps
650000 ps
700000 ps
204800 ps to 716800 ps
Now: 564,772,810 ns Delta: 0
7:13 PM
5/25/2024



❖Test Bench Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;


ENTITY tb_fifo_Buffer IS
END tb_fifo_Buffer;


ARCHITECTURE behavior OF tb_fifo_Buffer IS


  --Inputs and outputs
```

```
  signal Clk,reset,enr,enw,empty,full : std_logic := '0';
  signal data_in,data_out : std_logic_vector(7 downto 0) := (others => '0');
  --temporary signals
  signal i : integer := 0;
  -- Clock period definitions
  constant Clk_period : time := 10 ns;
  constant DEPTH : integer := 16;  --specify depth of fifo here.

BEGIN

  -- Instantiate the Unit Under Test (UUT)
  uut: entity work.fifo_Buffer
        generic map(DEPTH => DEPTH)
        PORT MAP (clk => clk,
              reset => reset,
              enr => enr,
              enw => enw,
              data_in => data_in,
              data_out => data_out,
              fifo_empty => empty,
              fifo_full => full);
```

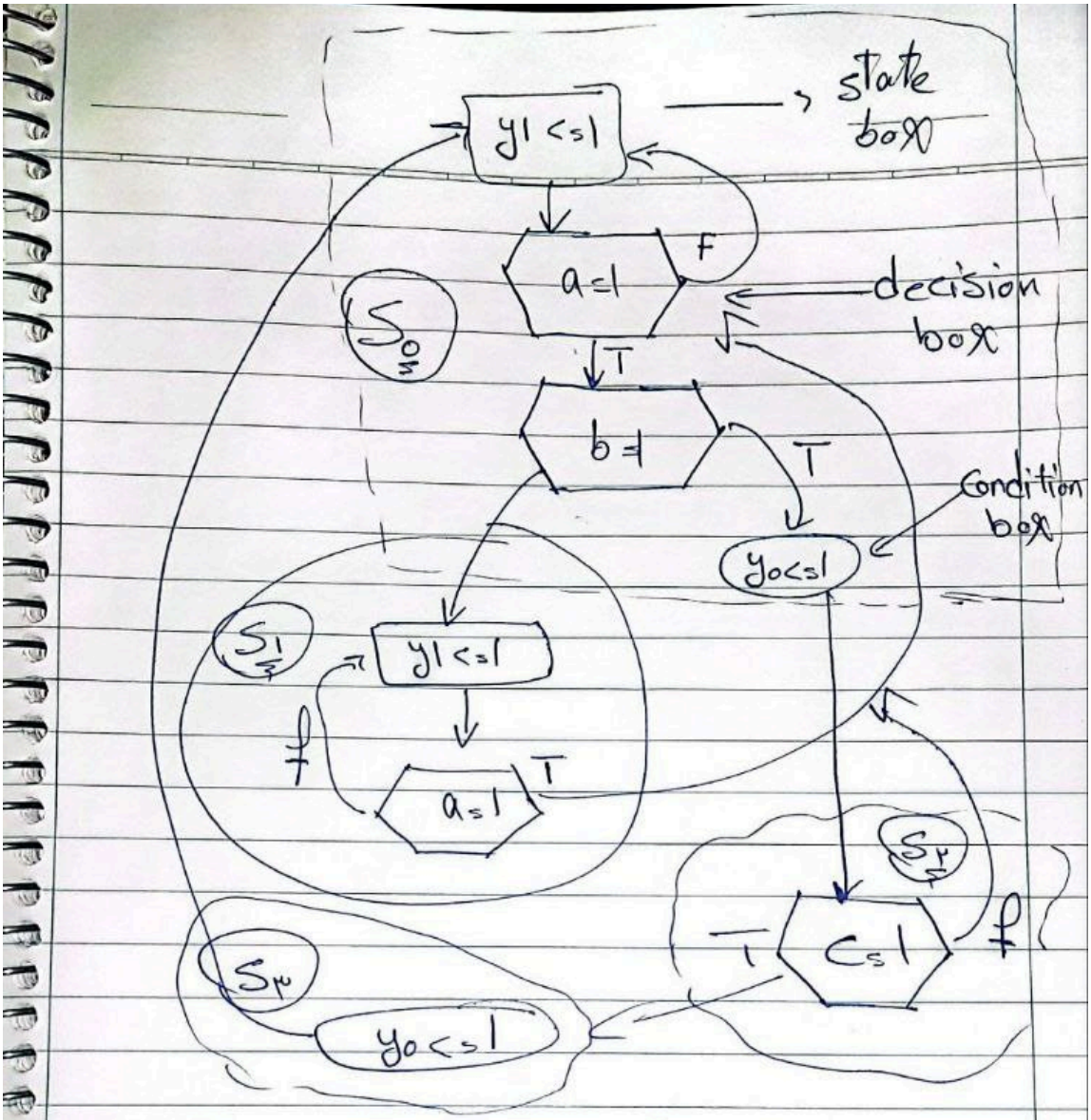
```
-- Clock process definitions
Clk_process :process
begin
   Clk <= '0';
   wait for Clk_period/2;
   Clk <= '1';
   wait for Clk_period/2;
end process;


-- Stimulus process
stim_proc: process
begin
   reset <= '1';  --apply reset for one clock cycle.
   wait for clk_period;
   reset <= '0';
   wait for clk_period*3;  --wait for 3 clock periods(simply)
   enw <= '1';     enr <= '0';         --write 10 values to fifo.
   for i in 1 to 10 loop
      data_in <= conv_std_logic_vector(i,8);
      wait for clk_period;
   end loop;
   enw <= '0';     enr <= '1';         --read 4 values from fifo.
```

```
    wait for clk_period*4;
    enw <= '1';     enr <= '1';         --read and write at the same time
for 2 clock cycles
    wait for clk_period*2;
    enw <= '0';     enr <= '0';                 --neither read nor write
    wait for clk_period*10;  --wait for some clock cycles.
    enw <= '1';     enr <= '0';         --write 10 values to fifo.
    for i in 11 to 20 loop
      data_in <= conv_std_logic_vector(i,8);
      wait for clk_period;
    end loop;
    enw <= '0';     enr <= '0';                 --neither read nor write
    wait for clk_period*10;  --wait for some clock cycles.
    enw <= '0';     enr <= '1';         --read 4 values from fifo.
    wait for clk_period*4;
    enw <= '0';     enr <= '0';      --neither read nor write
    wait for clk_period;
    enw <= '0';     enr <= '1';         --read 4 values from fifo.
    wait for clk_period*8;
    enw <= '0';     enr <= '0';          --neither read nor write
    wait for clk_period;
    enw <= '0';     enr <= '1';         --read 8 values from fifo.
    wait for clk_period*4;
```

```
    enw <= '0';     enr <= '0';        --neither read nor write
    wait for clk_period;
    enw <= '0';     enr <= '1';        --read 4 values from fifo.
    wait for clk_period*4;
    enw <= '0';     enr <= '0';         --neither read nor write
    wait;
  end process;


END;
```

Q3. To Implement the FSM and verify its correctness we first need FSM module so we declare the input and outputs then describe its behavior. After that we need Look-ahead output buffer to make sure that glitch- free moore outputs, implement a buffer that update outputs on the clock edge.

❖ASM chart:



States = S_0, S_1, S_2, S_3

Conditions and transitions based on input a, b and c

AYLAR

❖ State s0:

- Output: y0 = 1, y1 = 0, y2 = 0
- Decision: If a = 1 -> go to s1
- Decision: If a = 0 -> stay in s0

❖ State s1:

- Output: y0 = 0, y1 = 1, y2 = 0
- Decision: If a = 1 and b = 0 -> go to s2
- Decision: If a = 0 -> stay in s1

❖ State s2:

- Output: y0 = 0, y1 = 0, y2 = 1
- Decision: If a = 0 and b = 1 -> go to s0
- Decision: If c = 1 -> go to s3

❖ State s3:

- Output: y0 = 0, y1 = 0, y2 = 1
- Decision: If c = 0 -> go to s0
- Decision: If c = 1 -> stay in s3

❖Code:

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


entity FSM is
    Port ( clk : in STD_LOGIC;
```

```
        reset : in STD_LOGIC;
        a : in STD_LOGIC;
        b : in STD_LOGIC;
        c : in STD_LOGIC;
        y0, y1, y2 : out STD_LOGIC);
end FSM;

architecture Behavioral of FSM is
    type state_type is (s0, s1, s2, s3);
    signal state, next_state : state_type;
    signal y0_reg, y1_reg, y2_reg : STD_LOGIC := '0';
begin
    process(clk, reset)
    begin
        if reset = '1' then
            state <= s0;
        elsif rising_edge(clk) then
            state <= next_state;
        end if;
    end process;
```

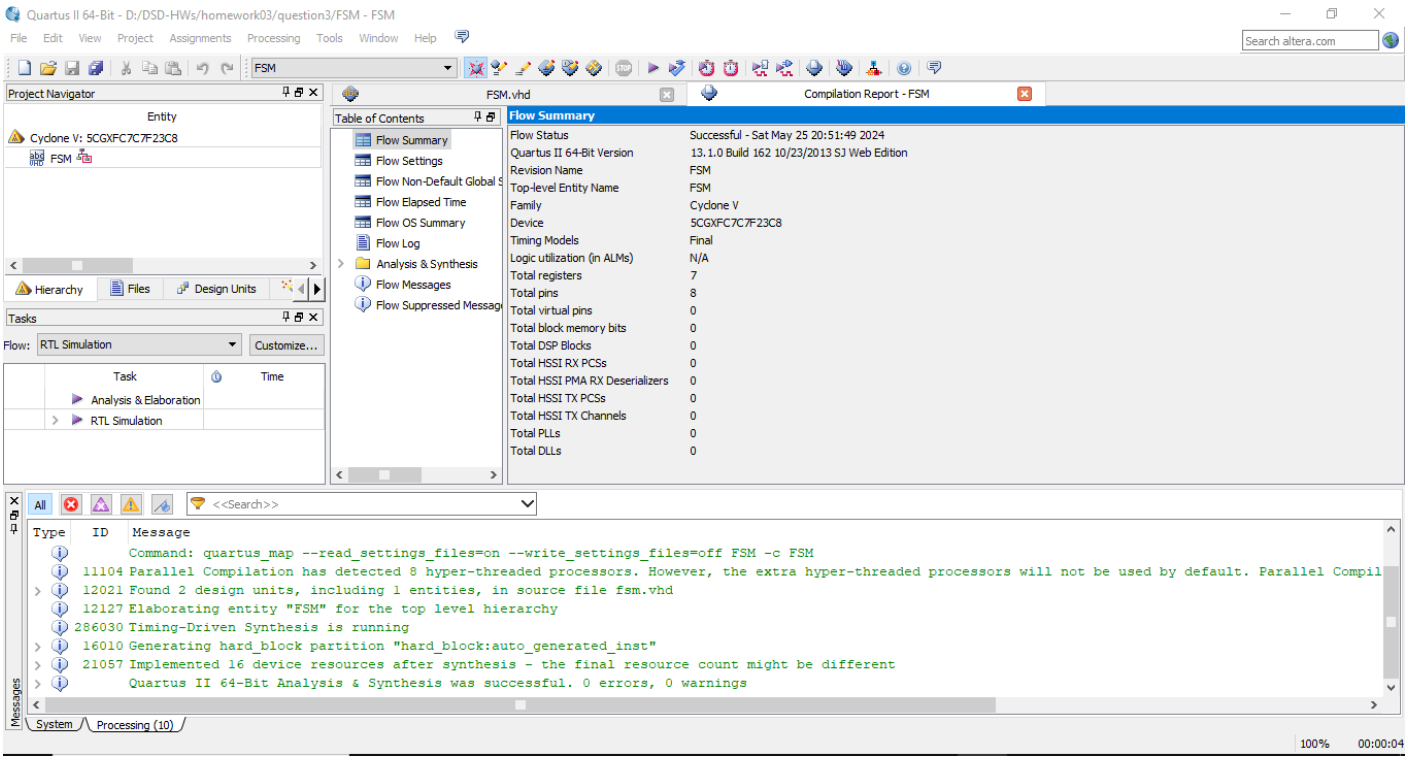
```
process(state, a, b, c)
begin
    case state is
        when s0 =>
            if a = '1' then
                next_state <= s1;
            else
                next_state <= s0;
            end if;
            y0_reg <= '1';
            y1_reg <= '0';
            y2_reg <= '0';

        when s1 =>
            if a = '1' and b = '0' then
                next_state <= s2;
            elsif a = '0' then
                next_state <= s1;
            else
```

```
            next_state <= s0;
        end if;
        y0_reg <= '0';
        y1_reg <= '1';
        y2_reg <= '0';

    when s2 =>
        if a = '0' and b = '1' then
            next_state <= s0;
        elsif c = '1' then
            next_state <= s3;
        else
            next_state <= s2;
        end if;
        y0_reg <= '0';
        y1_reg <= '0';
        y2_reg <= '1';

    when s3 =>
        if c = '0' then
```

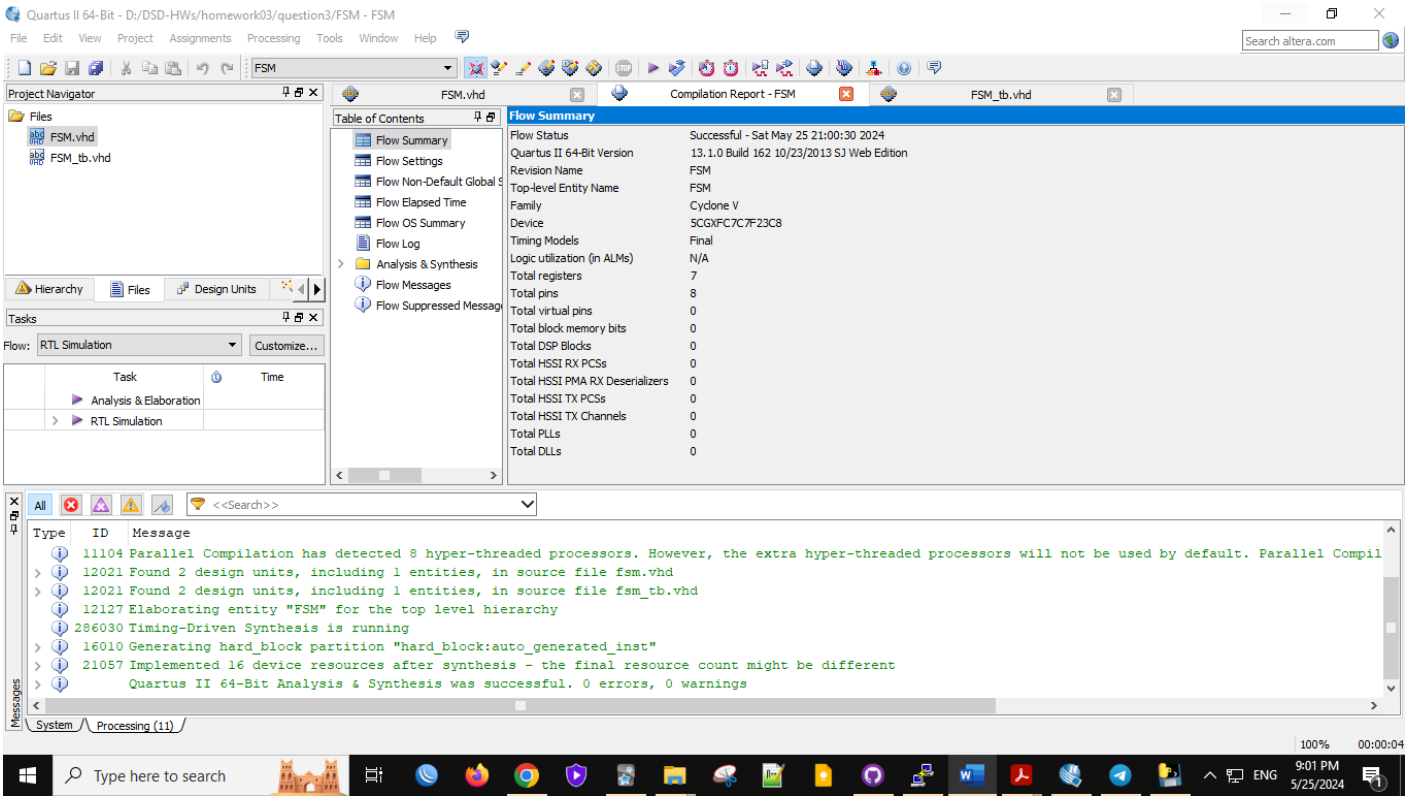
```
                next_state <= s0;
            else
                next_state <= s3;
            end if;
            y0_reg <= '0';
            y1_reg <= '0';
            y2_reg <= '1';

        when others =>
            next_state <= s0;
    end case;
end process;

-- Look-ahead output buffer to ensure glitch-free Moore output
process(clk)
begin
    if rising_edge(clk) then
        y0 <= y0_reg;
        y1 <= y1_reg;
        y2 <= y2_reg;
```

```
        end if;
    end process;


end Behavioral;
```

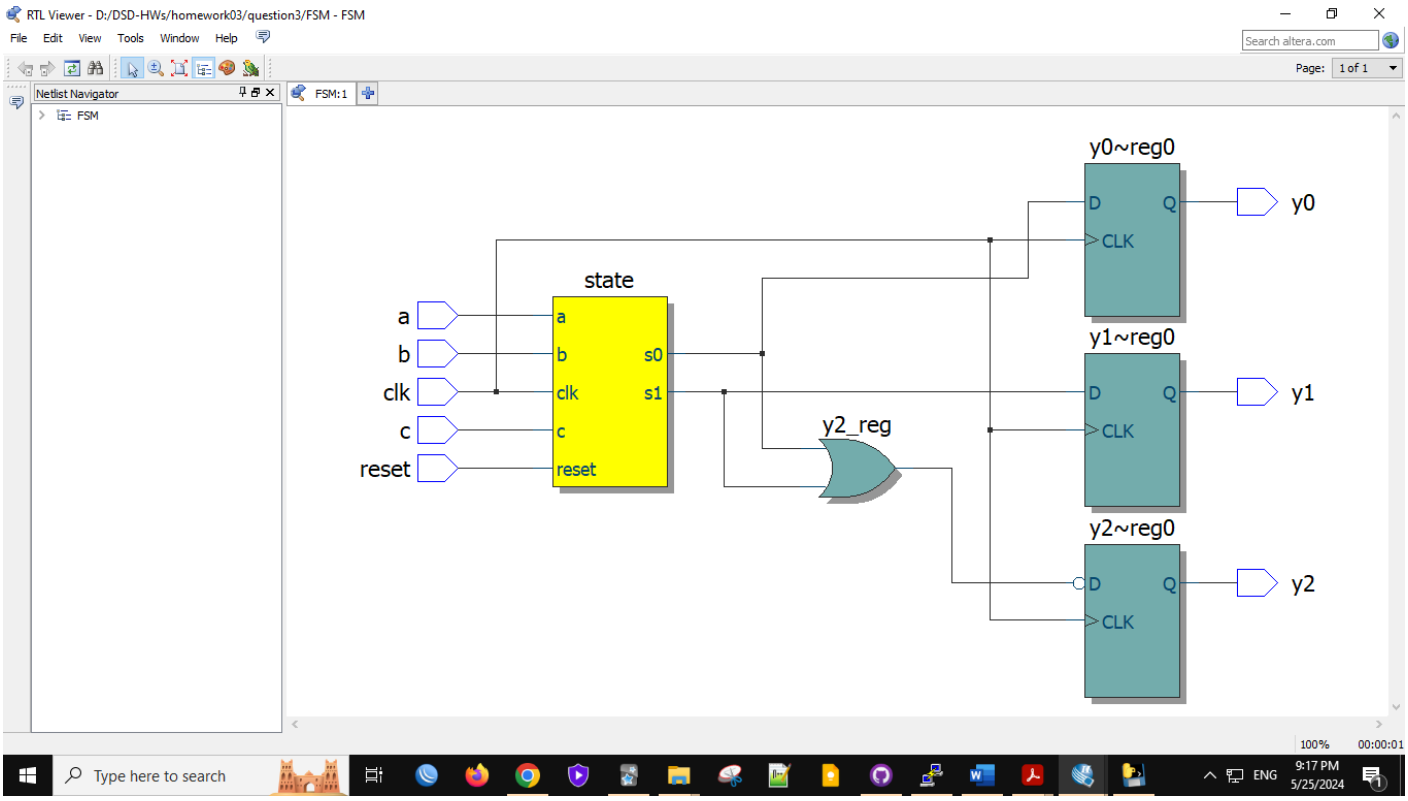
❖Compilation Report:



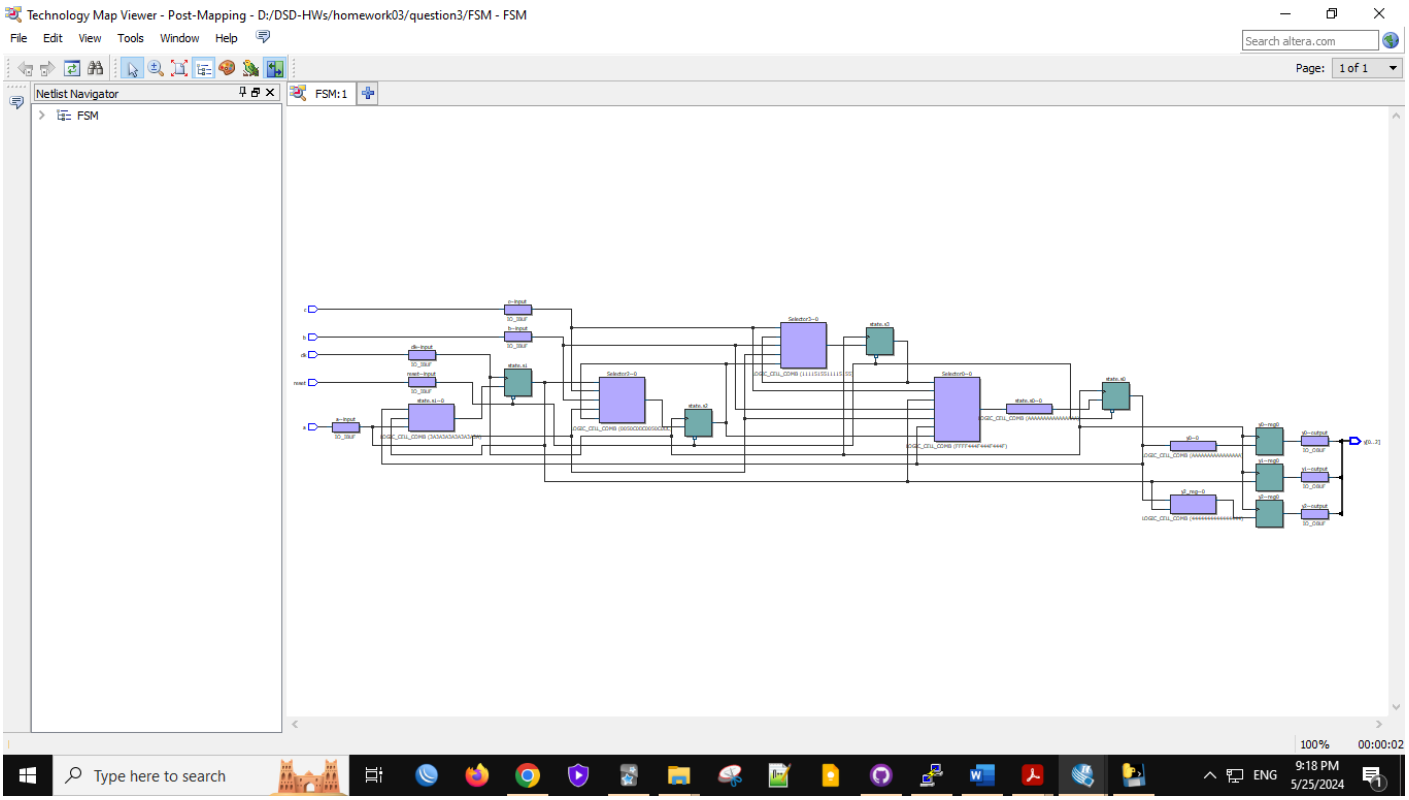
❖Compilation Report for Testbench Code:



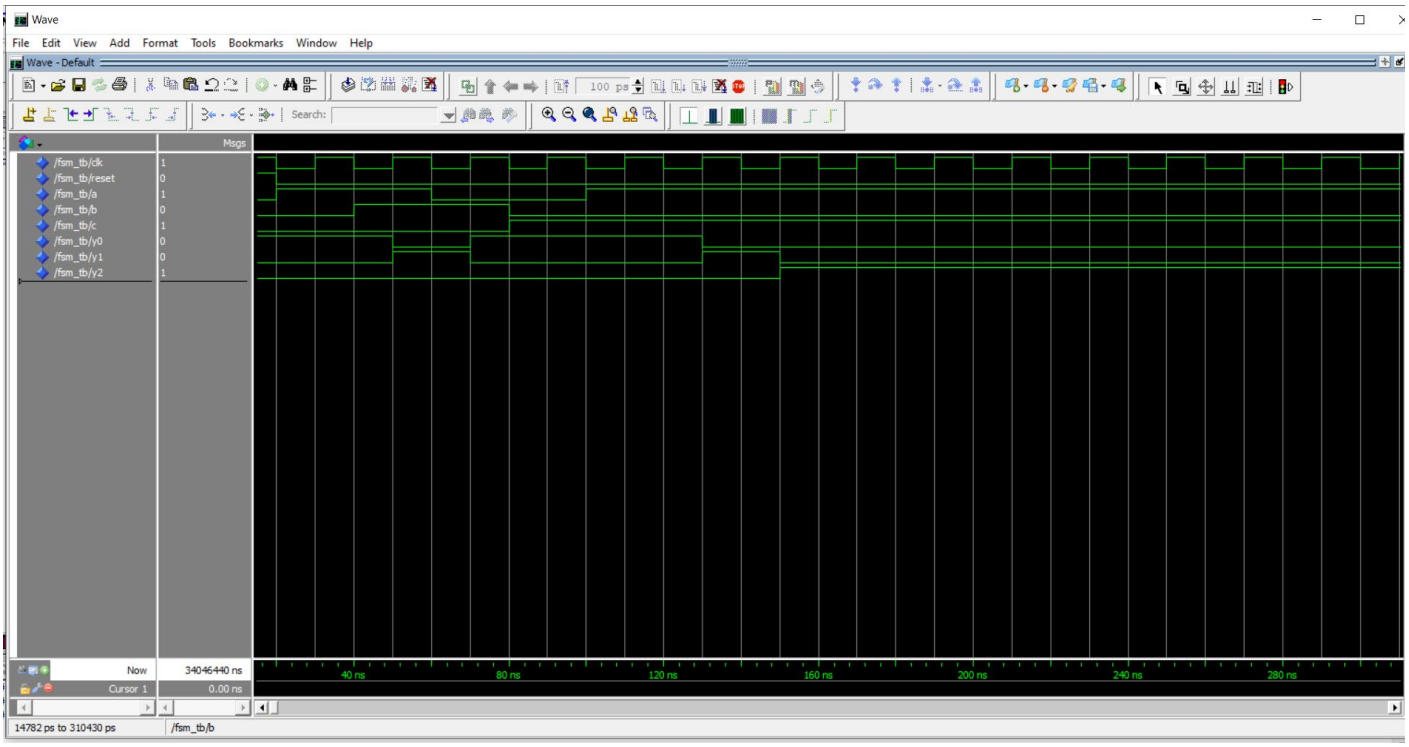
❖RTL View:

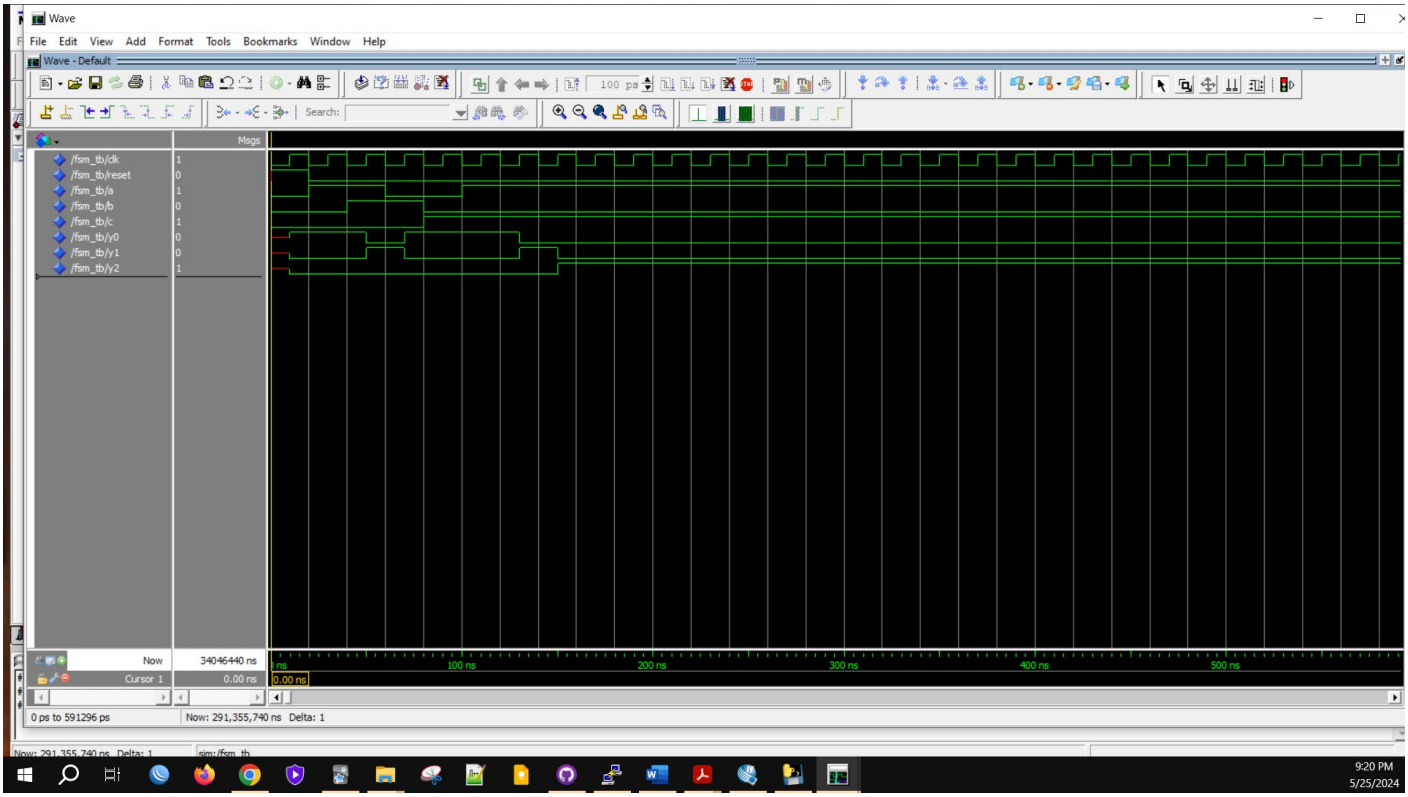


❖Post- Mapping View:



❖Wave View:





❖Testbench code:

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

entity FSM_tb is
end FSM_tb;

architecture Behavioral of FSM_tb is
    signal clk : STD_LOGIC := '0';
```

```
    signal reset : STD_LOGIC := '0';
    signal a : STD_LOGIC := '0';
    signal b : STD_LOGIC := '0';
    signal c : STD_LOGIC := '0';
    signal y0, y1, y2 : STD_LOGIC;

    component FSM
        Port ( clk : in STD_LOGIC;
               reset : in STD_LOGIC;
               a : in STD_LOGIC;
               b : in STD_LOGIC;
               c : in STD_LOGIC;
               y0, y1, y2 : out STD_LOGIC);
    end component;

begin
    uut: FSM Port map (
        clk => clk,
        reset => reset,
        a => a,
        b => b,
        c => c,
```

```
        y0 => y0,
        y1 => y1,
        y2 => y2
    );

    clk_process: process
    begin
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
    end process;

    stim_proc: process
    begin
        -- Initialize Inputs
        reset <= '1';
        wait for 20 ns;
        reset <= '0';

        -- Test sequence
        a <= '1'; b <= '0'; c <= '0';
```

```
        wait for 20 ns;
        a <= '1'; b <= '1'; c <= '0';
        wait for 20 ns;
        a <= '0'; b <= '1'; c <= '0';
        wait for 20 ns;
        a <= '0'; b <= '0'; c <= '1';
        wait for 20 ns;
        a <= '1'; b <= '0'; c <= '1';
        wait for 20 ns;

        -- Stop the simulation
        wait;
    end process;

end Behavioral;
```