

Lab Report

|  |  |
| --- | --- |
| Name:Brandon Hoang | CWID: 889874371 |
| Course: EGCP-381 | Date: 02/10/19 |

Grading Criteria:

|  |  |  |
| --- | --- | --- |
| **Section** | **Earned Points** | **Possible Points** |
| Problem/Objective: |  | 5 |
| Method: |  | 15 |
| Question(s): |  | 10 |
| Program Code: |  | 25 |
| Test Bench: |  | 15 |
| Demo: |  | 30 |
| Total: | 0 | 100 |

Report Submission Instructions:

* Please, upload your report to TITANIum
* No paper submissions
* When showing your work, you can use MS Word’s equation tool. Or you can hand write your work, take a picture, and paste the image here. One app that I would suggest to easily do this is “CamScanner”.
* When giving the screenshots, please take a screenshot of the whole screen (i.e., include the OS taskbar, date, clock, etc.). No cropping.

# Problem/Objective

State the problem statement and/or objective of the lab. This must be a complete paragraph (i.e., at least 5 sentences).

This purpose of this lab is to revisit combinational circuits and refresh our memory on them. Combinational logic will be applied in the computer circuits we will be dealing with for the whole of this class. This lab will also be putting into practice Boolean algebra. VHDL will also be practiced in this lab, as it will be the basis for all programming tasks in Xilinx ISE. Lastly, we will be working with the Diligent S3 board and using it in simulations.

# Methodology

Read about and describe here each component listed below:

1. Multiplexers (MUX) - pg. 388 - Section 11.3

A multiplexer takes multiple input signals and selectively chooses which input signal to output based on a selection input.

1. Decoders - pg. 390 - Section 11.3

Decoders take n input signals and outputs a 2­nbit signal with one ‘1’ bit whose location in the output is based on the value of the input signal.

1. Adders - pg. 392 - Section 11.3

An adder functions as its name suggest, it takes 2 input signals and a carry in input, and outputs the sum and carry out values.

1. ALU - pg. 329 - Section 10.1

ALU’s perform basic arithmetic and specific bitwise calculations on input signals. It has 2 input operand signals, an input signal for the operation, and an output signal.

1. Conduct research via the internet on comparators and multipliers

A comparator takes two input signals and outputs a ‘1’ to one of its 3 outputs signals that represent an inequality outcome. Multipliers (self-explanatory function) perform a multiplication operation on two input signals, and output a signal that usually has more bits than the inputs.

# Question(s)

Describe in the differences between sequential and concurrent statements.

Sequential statements differ from concurrent statements mainly because of the inclusion of a clock, and thus a dependency on ‘states’ between clock cycles. Concurrent statements are purely logic with no attention to clocks.

# Program Code

Copy your code here. Please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your code as text, not as a screenshot/image**.

Full Adder(SUB COMPONENT of FOURADDER AND MULTIPLIER):

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity FullAdder is**

**Port ( a : in std\_logic;**

**b : in std\_logic;**

**cin : in std\_logic;**

**s : out std\_logic;**

**cout : out std\_logic);**

**end FullAdder;**

**architecture Behavioral of FullAdder is**

**begin**

**s <= (not(a) and not(b) and cin) or (not(a) and b and not(cin)) or (a and not(b) and not(cin)) or ( a and b and cin);**

**cout <= (b and cin) or (a and cin) or (a and b);**

**end Behavioral;**

Comparator:

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity Comparator is**

**port ( A : in std\_logic\_vector( 1 downto 0);**

**B : in std\_logic\_vector( 1 downto 0);**

**Less: out std\_logic;**

**Equal: out std\_logic;**

**More: out std\_logic);**

**end Comparator;**

**architecture Behavioral of Comparator is**

**begin**

**Less <= (not(A(1)) and not(A(0)) and B(0)) or (not(A(1)) and B(1)) or (B(1) and not(B(0)) and not(A(0)));**

**Equal <= (not(A(1) XOR B(1))) and (not(A(0) XOR B(0))) ;**

**More <= (A(1) and not(B(0))) or (not(B(0)) and not(B(1)) and A(0));**

**end Behavioral;**

4 bit Adder

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity FourAdder is**

**Port ( a : in std\_logic\_vector(3 downto 0);**

**b : in std\_logic\_vector(3 downto 0);**

**c : in std\_logic;**

**o : out std\_logic;**

**s : out std\_logic\_vector(3 downto 0));**

**end FourAdder;**

**architecture Behavioral of FourAdder is**

**component FullAdder**

**Port ( A : in std\_logic;**

**B : in std\_logic;**

**Cin : in std\_logic;**

**S : out std\_logic;**

**Cout : out std\_logic);**

**end component;**

**signal B0, B1, B2, B3, c1, c2, c3, c4: std\_logic;**

**begin**

**B0 <= b(0);**

**B1 <= b(1);**

**B2 <= b(2);**

**B3 <= b(3);**

**FA1: FullAdder port map ( a(0), B0, c, s(0), c1);**

**FA2: FullAdder port map ( a(1), B1, c1, s(1), c2);**

**FA3: FullAdder port map ( a(2), B2, c2, s(2), c3);**

**FA4: FullAdder port map ( a(3), B3, c3, s(3), c4);**

**o <= c4;**

**end Behavioral;**

Multiplier:

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity FourMultiplier is**

**Port (a: in std\_logic\_vector ( 3 downto 0);**

**b: in std\_logic\_vector ( 3 downto 0);**

**Y: out std\_logic\_vector ( 7 downto 0));**

**end FourMultiplier;**

**architecture Behavioral of FourMultiplier is**

**component FourAdder**

**PORT(**

**A : IN std\_logic\_vector(3 downto 0);**

**B : IN std\_logic\_vector(3 downto 0);**

**C : IN std\_logic;**

**O : out std\_logic;**

**S : OUT std\_logic\_vector(3 downto 0)**

**);**

**end component;**

**signal c2, c3, c4, c5, s0, s1, s2, s3, x2, x3: std\_logic;**

**signal IA1, IA2, IA3, IB1, IB2, IB3, IC1, IC2, IC3: std\_logic\_vector(3 downto 0);**

**begin**

**-- based on diagram design based off of, right input Adder input 1, left 2, sum of level 3, level ABC**

**Y(0) <= a(0) and b(0);**

**IA1(0) <= a(0) and b(1);**

**IA1(1) <= a(1) and b(1);**

**IA1(2) <= a(2) and b(1);**

**IA1(3) <= not(a(3) and b(1));**

**IA2(0) <= a(1) and b(0);**

**IA2(1) <= a(2) and b(0);**

**IA2(2) <= not(a(3) and b(1));**

**IA2(3) <= '1';**

**IB1(0) <= a(0) and b(2);**

**IB1(1) <= a(1) and b(2);**

**IB1(2) <= a(2) and b(2);**

**IB1(3) <= not(a(3) and b(2));**

**x2 <= '0';**

**x3 <= '1';**

**IC1(0) <= not(a(0) and b(3));**

**IC1(1) <= not(a(1) and b(3));**

**IC1(2) <= not(a(2) and b(3));**

**IC1(3) <= a(3) and b(3);**

**FA1: FourAdder port map (IA1, IA2, x2, c2, IA3 );**

**Y(1) <= IA3(0);**

**IB2(0) <= IA3(1);**

**IB2(1) <= IA3(2);**

**IB2(2) <= IA3(3);**

**IB2(3) <= c2;**

**FA2: FourAdder port map (IB1, IB2, x2, c3, IB3 );**

**Y(2) <= IB3(0);**

**IC2(0) <= IB3(1);**

**IC2(1) <= IB3(2);**

**IC2(2) <= IB3(3);**

**IC2(3) <= c3;**

**FA3: FourAdder port map (IC1, IC2, x2, c4, IC3 );**

**Y(3) <= IC3(0);**

**Y(4) <= IC3(1);**

**Y(5) <= IC3(2);**

**Y(6) <= IC3(3);**

**Y(7) <= not(c4);**

**end Behavioral;**

ALU:

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity TwoALU is**

**Port(A: in std\_logic\_vector (1 downto 0);**

**B: in std\_logic\_vector (1 downto 0);**

**Sel: in std\_logic\_vector ( 1 downto 0);**

**Y: out std\_logic\_vector ( 1 downto 0));**

**end TwoALU;**

**architecture Behavioral of TwoALU is**

**begin**

**with Sel select**

**Y <= A+B when "00",**

**A + (not(B)) +"01" when "01",**

**(A and B) when "10",**

**(A or B) when others;**

**end Behavioral;**

2-4 Decoder

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity TwoFourDecoder is**

**Port ( A : in std\_logic\_vector(1 downto 0);**

**Z : out std\_logic\_vector(3 downto 0));**

**end TwoFourDecoder;**

**architecture Behavioral of TwoFourDecoder is**

**begin**

**Z <= "0001" when A = "00" else**

**"0010" when A = "01" else**

**"0100" when A = "10" else**

**"1000" when A = "11" else**

**"0000";**

**end Behavioral;**

Multiplexer:

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**-- Uncomment the following lines to use the declarations that are**

**-- provided for instantiating Xilinx primitive components.**

**--library UNISIM;**

**--use UNISIM.VComponents.all;**

**entity FourOneMUX is**

**Port(A: in std\_logic\_vector ( 3 downto 0);**

**B: in std\_logic\_vector ( 3 downto 0);**

**C: in std\_logic\_vector ( 3 downto 0);**

**D: in std\_logic\_vector ( 3 downto 0);**

**Sel: in std\_logic\_vector ( 1 downto 0);**

**Y: out std\_logic\_vector ( 3 downto 0)**

**);**

**end FourOneMUX;**

**architecture Behavioral of FourOneMUX is**

**begin**

**Y <= A when Sel = "00" else**

**B when Sel = "01" else**

**C when Sel = "10" else**

**D when Sel = "11" else**

**"0000";**

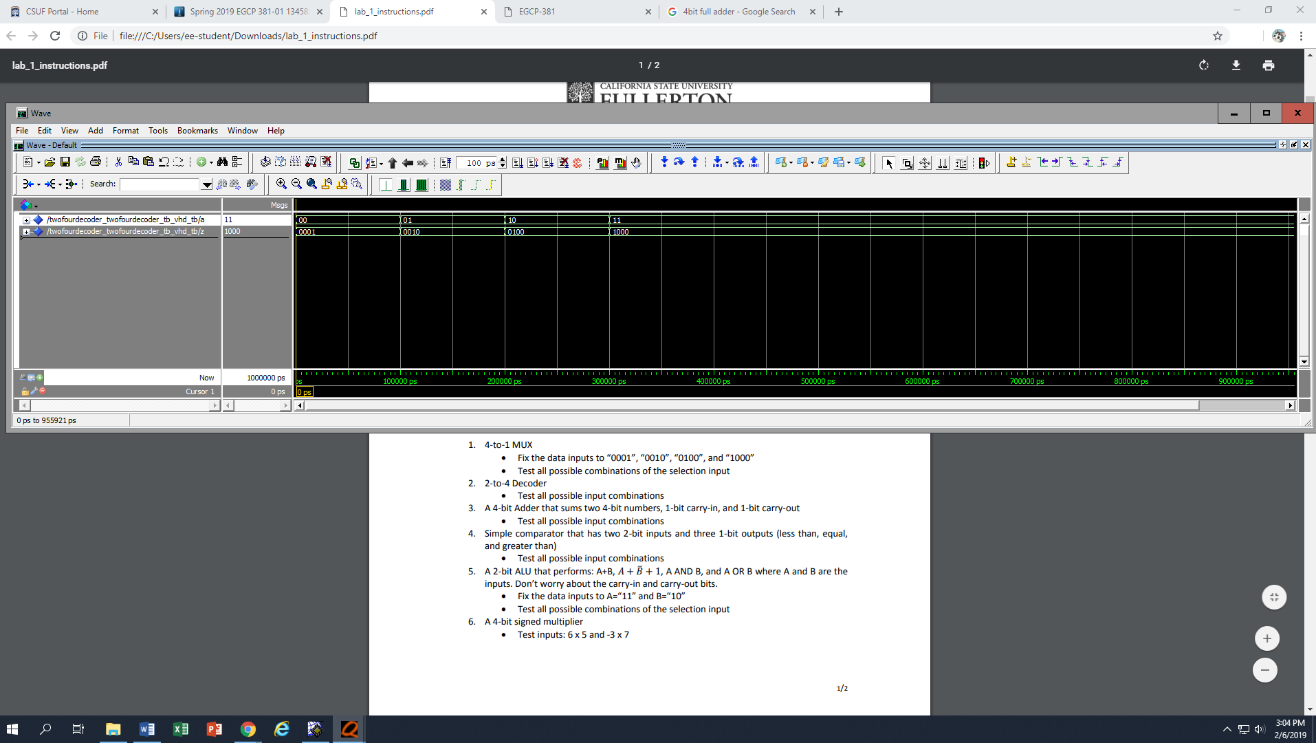
**end Behavioral;**

# Test Bench

Copy your test bench code here. Again, please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your test bench code as text, not as a screenshot/image**.

Also in this section, provide the waveform output of your test benches. Please, properly label each waveform. They must be images embedded in the document.

2-4 Decoder



**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.ALL;**

**USE ieee.numeric\_std.ALL;**

**ENTITY twofourdecoder\_TwoFourDecoder\_tb\_vhd\_tb IS**

**END twofourdecoder\_TwoFourDecoder\_tb\_vhd\_tb;**

**ARCHITECTURE behavior OF twofourdecoder\_TwoFourDecoder\_tb\_vhd\_tb IS**

**COMPONENT twofourdecoder**

**PORT(**

**A : IN std\_logic\_vector(1 downto 0);**

**Z : OUT std\_logic\_vector(3 downto 0)**

**);**

**END COMPONENT;**

**SIGNAL a : std\_logic\_vector(1 downto 0);**

**SIGNAL z : std\_logic\_vector(3 downto 0);**

**BEGIN**

**uut: twofourdecoder PORT MAP(**

**A => a,**

**Z => z**

**);**

**-- \*\*\* Test Bench - User Defined Section \*\*\***

**tb : PROCESS**

**BEGIN**

**a <= "00";**

**wait for 100ns;**

**a <= "01";**

**wait for 100ns;**

**a <= "10";**

**wait for 100ns;**

**a <= "11";**

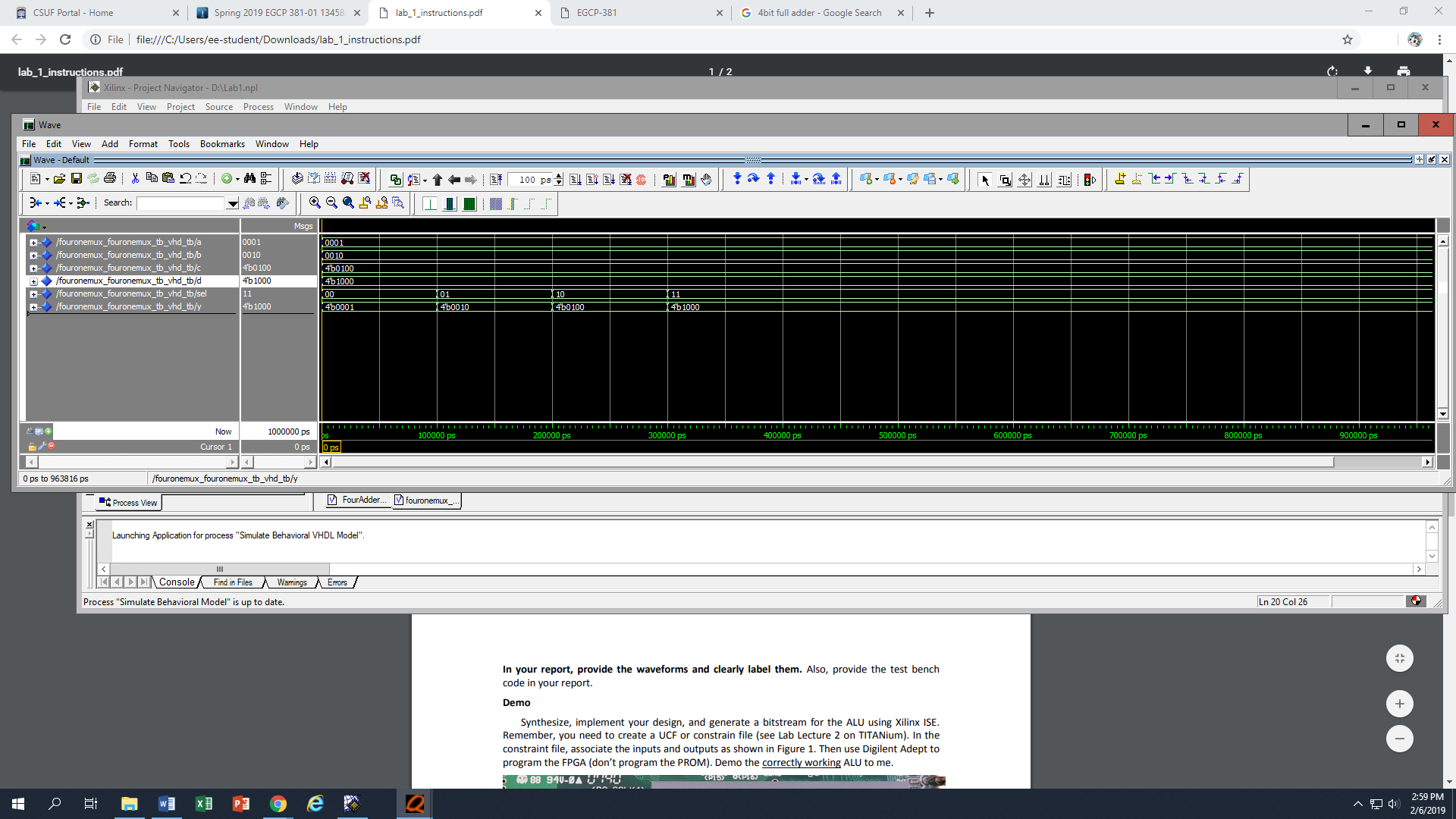
**wait for 100ns;**

**wait; -- will wait forever**

**END PROCESS;**

**-- \*\*\* End Test Bench - User Defined Section \*\*\***

**END;**

4-1Multiplexer

**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.ALL;**

**USE ieee.numeric\_std.ALL;**

**ENTITY fourmultiplier\_fourmultiplier\_tb\_vhd\_tb IS**

**END fourmultiplier\_fourmultiplier\_tb\_vhd\_tb;**

**ARCHITECTURE behavior OF fourmultiplier\_fourmultiplier\_tb\_vhd\_tb IS**

**COMPONENT fourmultiplier**

**PORT(**

**a : IN std\_logic\_vector(3 downto 0);**

**b : IN std\_logic\_vector(3 downto 0);**

**Y : OUT std\_logic\_vector(7 downto 0)**

**);**

**END COMPONENT;**

**SIGNAL A : std\_logic\_vector(3 downto 0);**

**SIGNAL B : std\_logic\_vector(3 downto 0);**

**SIGNAL y : std\_logic\_vector(7 downto 0);**

**BEGIN**

**uut: fourmultiplier PORT MAP(**

**a => A,**

**b => B,**

**Y => y**

**);**

**-- \*\*\* Test Bench - User Defined Section \*\*\***

**tb : PROCESS**

**BEGIN**

**A <= "0110";**

**B <= "0101";**

**wait for 100ns;**

**A <= "0111";**

**B <= "1101";**

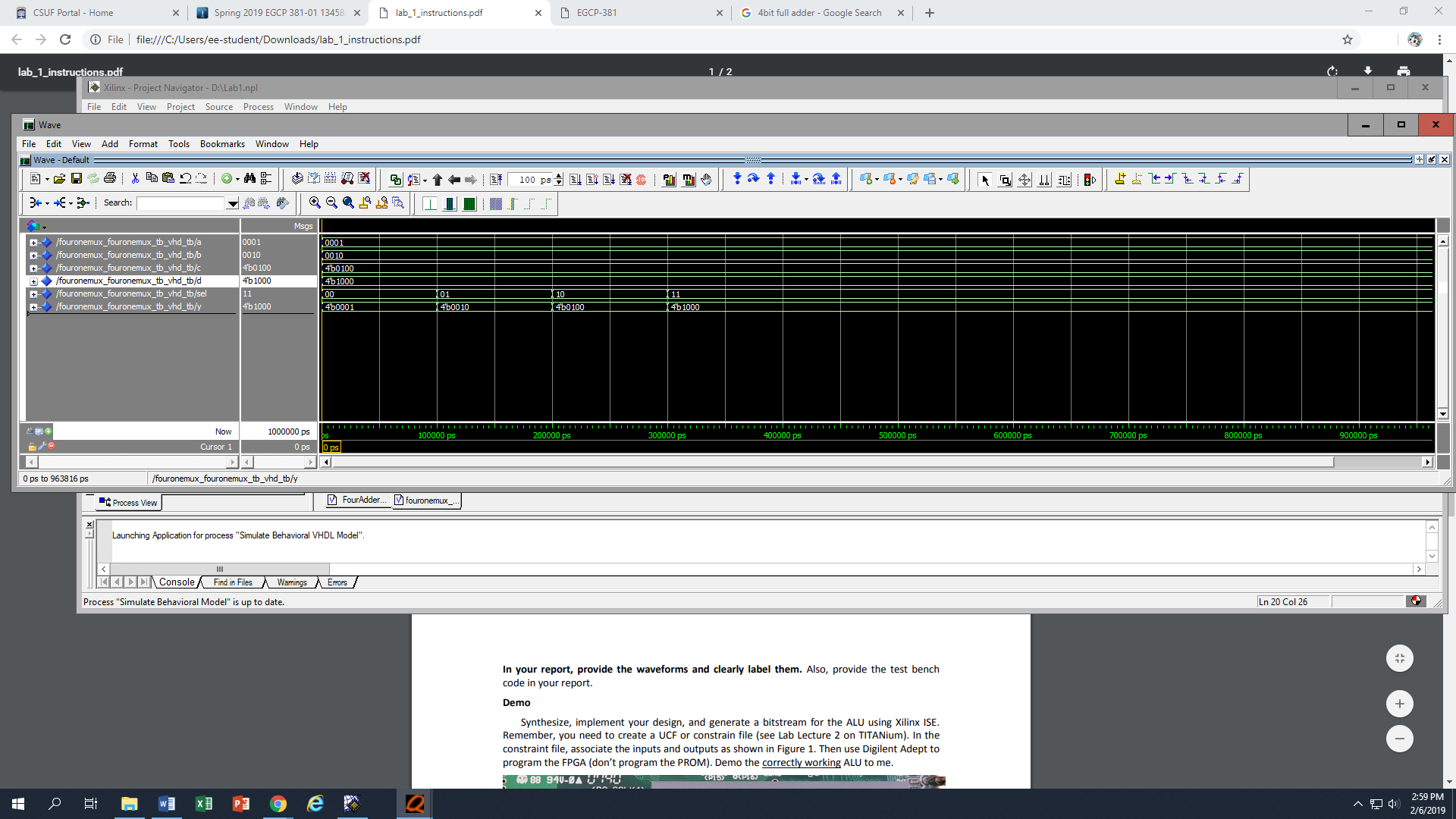
**wait; -- will wait forever**

**END PROCESS;**

**-- \*\*\* End Test Bench - User Defined Section \*\*\***

**END;**

ALU



**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.ALL;**

**USE ieee.numeric\_std.ALL;**

**ENTITY twoalu\_TwoALU\_tb\_vhd\_tb IS**

**END twoalu\_TwoALU\_tb\_vhd\_tb;**

**ARCHITECTURE behavior OF twoalu\_TwoALU\_tb\_vhd\_tb IS**

**COMPONENT twoalu**

**PORT(**

**A : IN std\_logic\_vector(1 downto 0);**

**B : IN std\_logic\_vector(1 downto 0);**

**Sel : IN std\_logic\_vector(1 downto 0);**

**Y : OUT std\_logic\_vector(1 downto 0)**

**);**

**END COMPONENT;**

**SIGNAL a : std\_logic\_vector(1 downto 0);**

**SIGNAL b : std\_logic\_vector(1 downto 0);**

**SIGNAL sel : std\_logic\_vector(1 downto 0);**

**SIGNAL y : std\_logic\_vector(1 downto 0);**

**BEGIN**

**uut: twoalu PORT MAP(**

**A => a,**

**B => b,**

**Sel => sel,**

**Y => y**

**);**

**-- \*\*\* Test Bench - User Defined Section \*\*\***

**tb : PROCESS**

**BEGIN**

**a <= "11";**

**b <= "10";**

**sel <= "00";**

**wait for 100ns;**

**sel <= "01";**

**wait for 100ns;**

**sel <= "10";**

**wait for 100ns;**

**sel <= "11";**

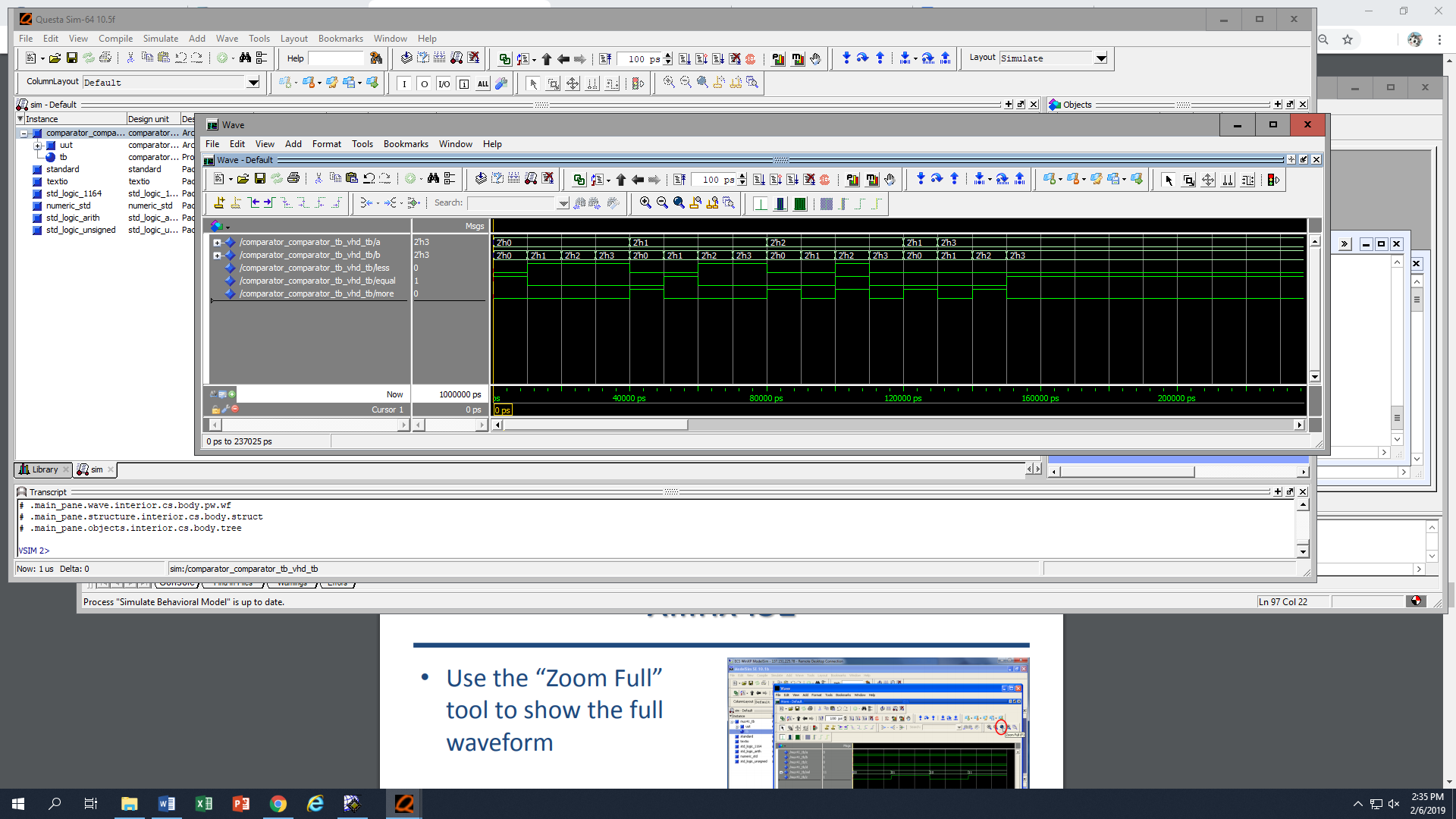
**wait; -- will wait forever**

**END PROCESS;**

**-- \*\*\* End Test Bench - User Defined Section \*\*\***

**END;**

Comparator



**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.ALL;**

**USE ieee.numeric\_std.ALL;**

**ENTITY comparator\_comparator\_tb\_vhd\_tb IS**

**END comparator\_comparator\_tb\_vhd\_tb;**

**ARCHITECTURE behavior OF comparator\_comparator\_tb\_vhd\_tb IS**

**COMPONENT comparator**

**PORT(**

**A : IN std\_logic\_vector(1 downto 0);**

**B : IN std\_logic\_vector(1 downto 0);**

**Less : OUT std\_logic;**

**Equal : OUT std\_logic;**

**More : OUT std\_logic**

**);**

**END COMPONENT;**

**SIGNAL a : std\_logic\_vector(1 downto 0);**

**SIGNAL b : std\_logic\_vector(1 downto 0);**

**SIGNAL less : std\_logic;**

**SIGNAL equal : std\_logic;**

**SIGNAL more : std\_logic;**

**BEGIN**

**uut: comparator PORT MAP(**

**A => a,**

**B => b,**

**Less => less,**

**Equal => equal,**

**More => more**

**);**

**-- \*\*\* Test Bench - User Defined Section \*\*\***

**tb : PROCESS**

**BEGIN**

**a <= "00";**

**b <= "00";**

**wait for 10ns;**

**a <= "00";**

**b <= "01";**

**wait for 10ns;**

**a <= "00";**

**b <= "10";**

**wait for 10ns;**

**a <= "00";**

**b <= "11";**

**wait for 10ns;**

**a <= "01";**

**b <= "00";**

**wait for 10ns;**

**a <= "01";**

**b <= "01";**

**wait for 10ns;**

**a <= "01";**

**b <= "10";**

**wait for 10ns;**

**a <= "01";**

**b <= "11";**

**wait for 10ns;**

**a <= "10";**

**b <= "00";**

**wait for 10ns;**

**a <= "10";**

**b <= "01";**

**wait for 10ns;**

**a <= "10";**

**b <= "10";**

**wait for 10ns;**

**a <= "10";**

**b <= "11";**

**wait for 10ns;**

**a <= "01";**

**b <= "00";**

**wait for 10ns;**

**a <= "11";**

**b <= "01";**

**wait for 10ns;**

**a <= "11";**

**b <= "10";**

**wait for 10ns;**

**a <= "11";**

**b <= "11";**

**wait for 10ns;**

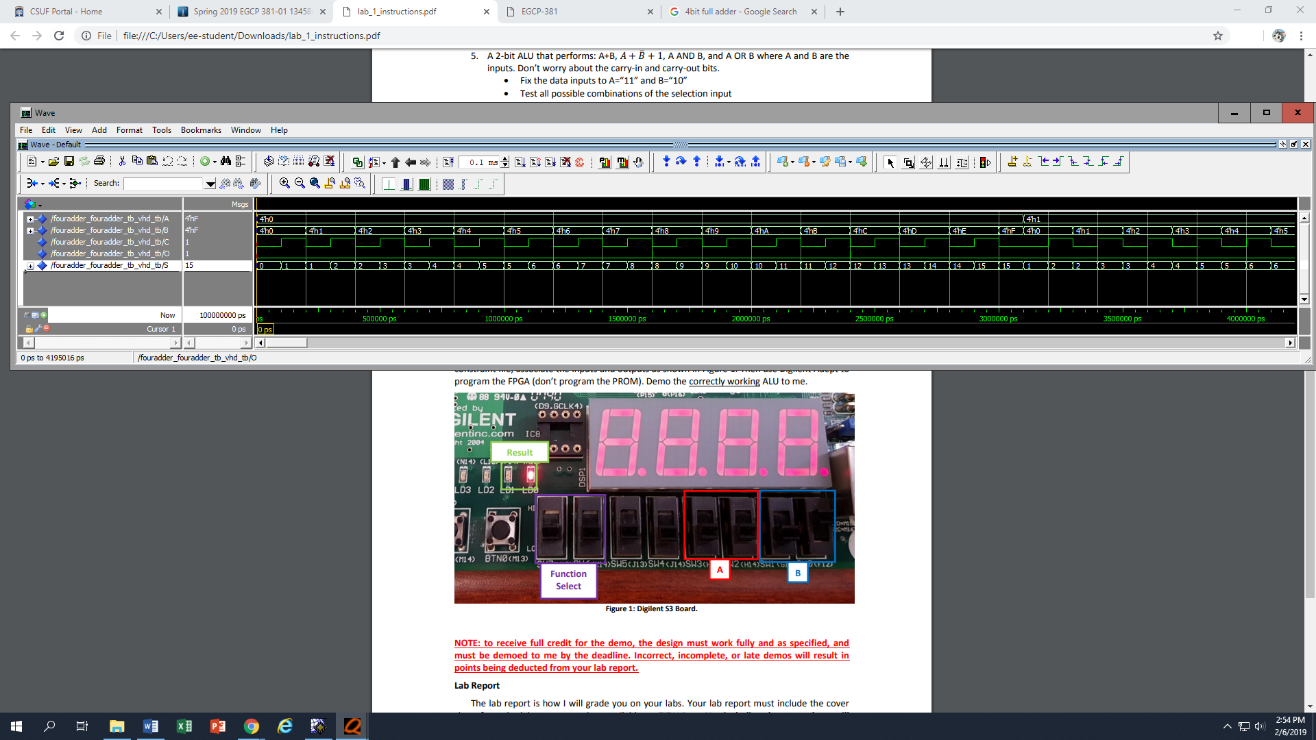
**wait; -- will wait forever**

**END PROCESS;**

**-- \*\*\* End Test Bench - User Defined Section \*\*\***

**END;**

4 bit Full Adder



(Pasted with super small text size for the sake of sanity, increase text size if too small)

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY fouradder\_fouradder\_tb\_vhd\_tb IS

END fouradder\_fouradder\_tb\_vhd\_tb;

ARCHITECTURE behavior OF fouradder\_fouradder\_tb\_vhd\_tb IS

COMPONENT fouradder

PORT(

a : IN std\_logic\_vector(3 downto 0);

b : IN std\_logic\_vector(3 downto 0);

c : IN std\_logic;

o : out std\_logic;

s : OUT std\_logic\_vector(3 downto 0)

);

END COMPONENT;

SIGNAL A : std\_logic\_vector(3 downto 0);

SIGNAL B : std\_logic\_vector(3 downto 0);

SIGNAL C : std\_logic;

SIGNAL O : std\_logic;

SIGNAL S : std\_logic\_vector(3 downto 0);

BEGIN

uut: fouradder PORT MAP(

a => A,

b => B,

c => C,

o => O,

s => S

);

-- \*\*\* Test Bench - User Defined Section \*\*\*

tb : PROCESS

BEGIN

A <= "0000" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0000" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "0001" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0001" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "0010" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0010" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "0011" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0011" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "0100" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "0111" ;

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0100" ;

B <= "1111";

C <= '0';

wait for 100ns ;

C <= '1';

A <= "0101" ;

B <= "0000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0101" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "0110" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0110" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "0111" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "0111" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "1000" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "0111" ;

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1000" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "1001" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1001" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "1010" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1010" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "1011" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1011" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1100" ;

B <= "1111";

C <= '0';

wait for 100ns ;

C <= '1';

A <= "1101" ;

B <= "0000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "0111" ;

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1101" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

A <= "1110" ;

B <= "0000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "0111" ;

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1110" ;

B <= "1111";

C <= '0';

wait for 100ns ;

C <= '1';

A <= "1111" ;

B <= "0000";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0001";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0010";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0011";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0100";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0101";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0110";

C <= '0';

wait for 100ns ;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "0111" ;

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1000";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1001";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1010";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1011";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1100";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1101";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1110";

C <= '0';

wait for 100ns;

C <= '1';

wait for 100ns;

A <= "1111" ;

B <= "1111";

C <= '0';

wait for 100ns;

C <= '1';

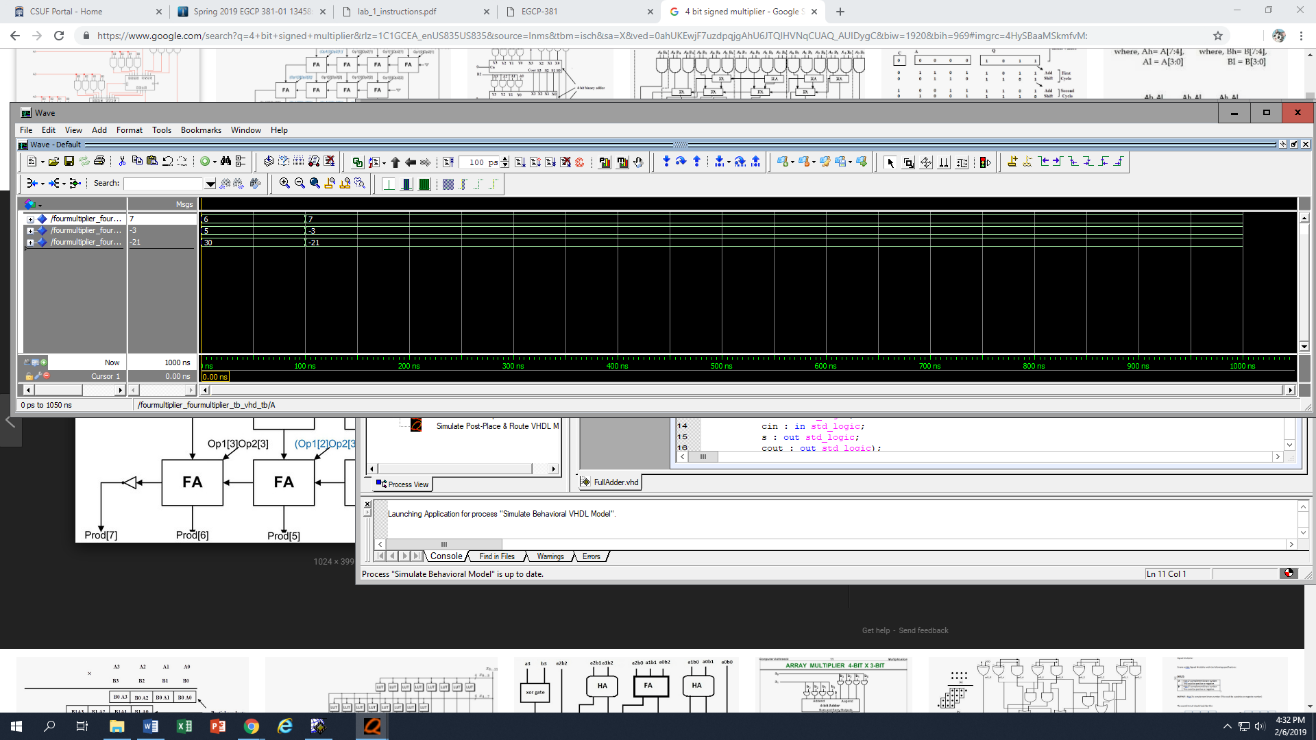
wait; -- will wait forever

END PROCESS;

-- \*\*\* End Test Bench - User Defined Section \*\*\*

END;

4 bit Multiplier



**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.ALL;**

**USE ieee.numeric\_std.ALL;**

**ENTITY fouronemux\_fouronemux\_tb\_vhd\_tb IS**

**END fouronemux\_fouronemux\_tb\_vhd\_tb;**

**ARCHITECTURE behavior OF fouronemux\_fouronemux\_tb\_vhd\_tb IS**

**COMPONENT fouronemux**

**Port(A: in std\_logic\_vector ( 3 downto 0);**

**B: in std\_logic\_vector ( 3 downto 0);**

**C: in std\_logic\_vector ( 3 downto 0);**

**D: 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;**

**SIGNAL a : std\_logic\_vector(3 downto 0);**

**SIGNAL b : std\_logic\_vector(3 downto 0);**

**SIGNAL c : std\_logic\_vector(3 downto 0);**

**SIGNAL d : std\_logic\_vector(3 downto 0);**

**SIGNAL sel : std\_logic\_vector(1 downto 0);**

**SIGNAL y : std\_logic\_vector(3 downto 0);**

**BEGIN**

**uut: fouronemux PORT MAP(**

**A => a,**

**B => b,**

**C => c,**

**D => d,**

**Sel => sel,**

**Y => y**

**);**

**-- \*\*\* Test Bench - User Defined Section \*\*\***

**tb : PROCESS**

**BEGIN**

**a <= "0001";**

**b <= "0010";**

**c <= "0100";**

**d <= "1000";**

**sel <= "00";**

**wait for 100ns;**

**sel <= "01";**

**wait for 100ns;**

**sel <= "10";**

**wait for 100ns;**

**sel <= "11";**

**wait; -- will wait forever**

**END PROCESS;**

**-- \*\*\* End Test Bench - User Defined Section \*\*\***

**END;**