CSc 21100 - Fundamentals of Computer Systems

Instructor: Prof. Zheng Peng

Project: 02

Name: Kristina Ilyovska

Section: CC1

Date: November 03, 2021

Task 1: Inhibit Gate

The VHDL module implements an Inhibit gate that returns 1 if X input is 1 and Y input is 0. It returns 0 in all other cases.

Entity Declaration

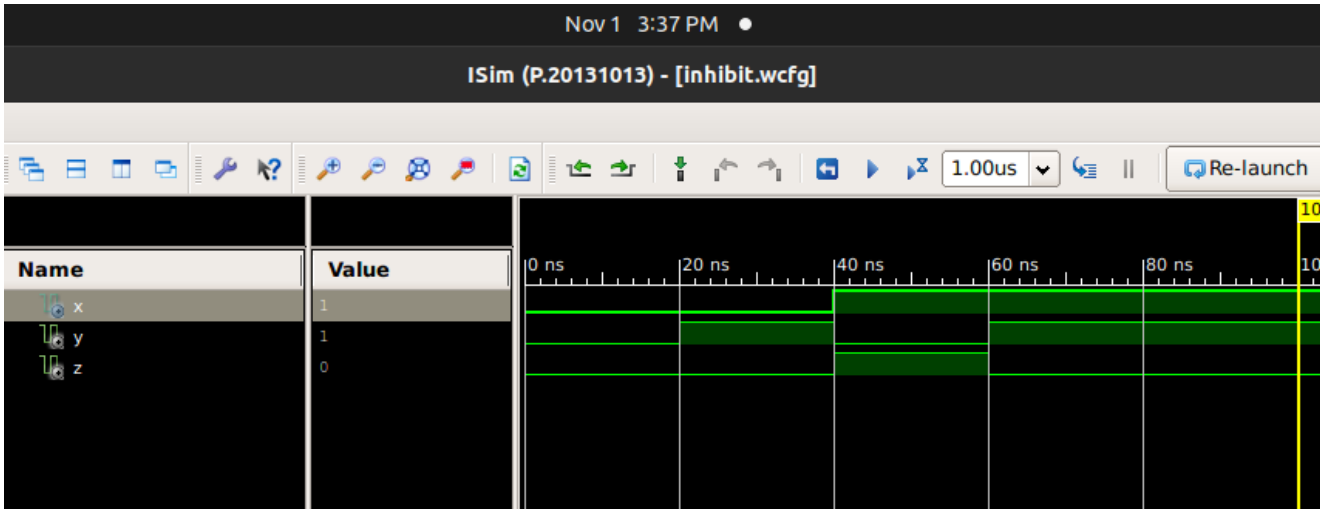
```
entity inhibit_gate is      --also known as "BUT-NOT"
   Port ( x : in  BIT;          --as in x BUT NOT y
          y : in  BIT;
          z : out  BIT);
end inhibit_gate;
```

Architecture Definition

```
architecture Inhibit_arch of inhibit_gate is
begin
   z <= '1' when x='1' and y='0' else '0';
end Inhibit_arch;
```

Waveform

The simulation results are as follow:



The waveform above covers all test cases. Below is an explanation for each test case in a given ns range.

- [0 ns, 20 ns], Input: x = 0, y = 0; output: z = 0
- [20 ns, 40 ns], Input: x = 0, y = 1; output: z = 0
- [40 ns, 60 ns], Input: x = 1, y = 0; output: z = 1

- [60 ns, 80 ns], Input: x = 1, y = 1; output: z = 0

The gate works as expected.

Task 2: Prime Number Detector

The digital circuit detects whether a 4-bit unsigned binary number is prime or not, storing a single bit F as the result.

Entity Declaration

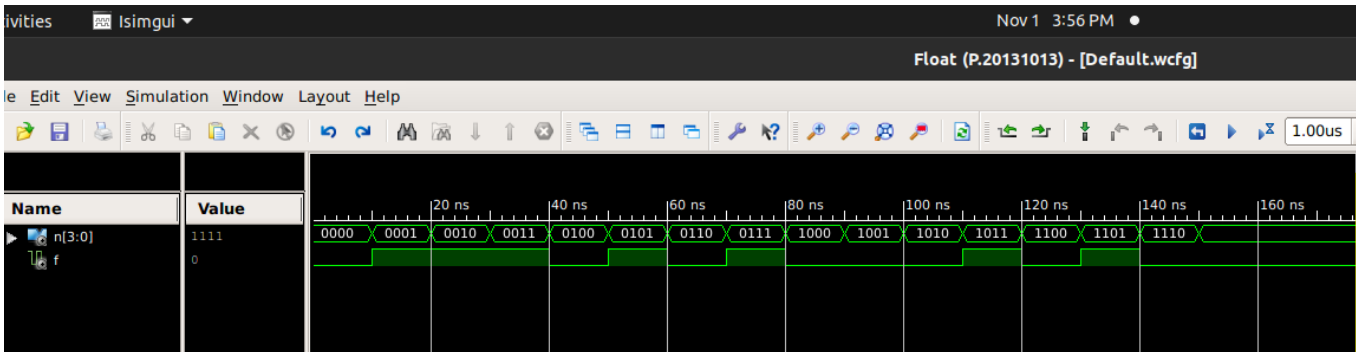
```
entity prime is
        Port ( N : in  STD_LOGIC_VECTOR (3 downto 0);          --little endian, MSB first
        F : out  STD_LOGIC);
end prime;
```

Architecture Definition

```
architecture prime_arch1 of prime is

signal N3_L, N2_L, N1_L: STD_LOGIC;                                          --variables
signal N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0: STD_LOGIC;

component INV port (I: in STD_LOGIC; O: out STD_LOGIC); end component;       --gates used
component AND2 port (I0, I1: in STD_LOGIC; O: out STD_LOGIC); end component;
component AND3 port (I0, I1, I2: in STD_LOGIC; O: out STD_LOGIC); end component;
component OR4 port (I0, I1, I2, I3: in STD_LOGIC; O: out STD_LOGIC); end component;

begin

   U1: INV port map (N(3), N3_L);                                  --negate MSB, N3
   U2: INV port map (N(2), N2_L);                                  --negate N2
   U3: INV port map (N(1), N1_L);                                   --negate N1
   U4: AND2 port map (N3_L, N(0), N3L_N0);                          --AND of 2 inputs
   U5: AND3 port map (N3_L, N2_L, N(1), N3L_N2L_N1);                --AND of 3 inputs
   U6: AND3 port map (N2_L, N(1), N(0), N2L_N1_N0);                 --AND of 3 inputs
   U7: AND3 port map (N(2), N1_L, N(0), N2_N1L_N0);                 --AND of 3 inputs
   U8: OR4 port map (N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0, F);  --OR of 4 inputs,
                                                                     save result in F
end prime_arch1;
```

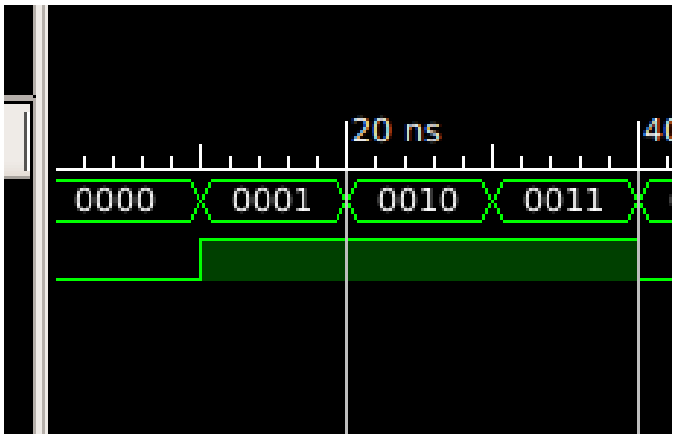
Waveform

The results of the simulation can be seen in the full waveform below:



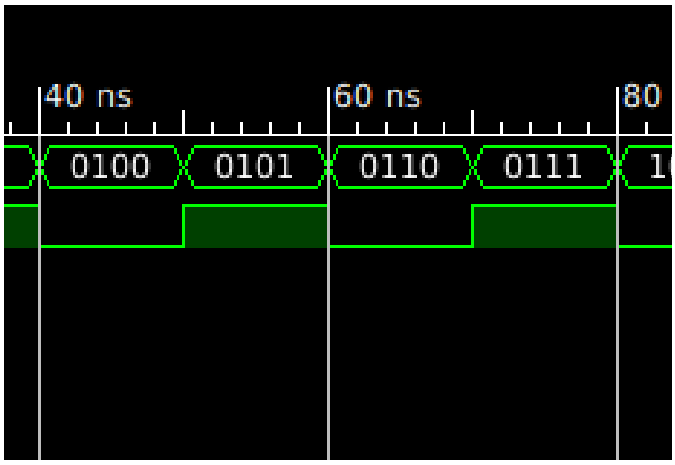
Here are partial screenshots of the waveform to make it more readable, below each is an explanation of the test cases captured in it:

Pt.1 - First 4 cases



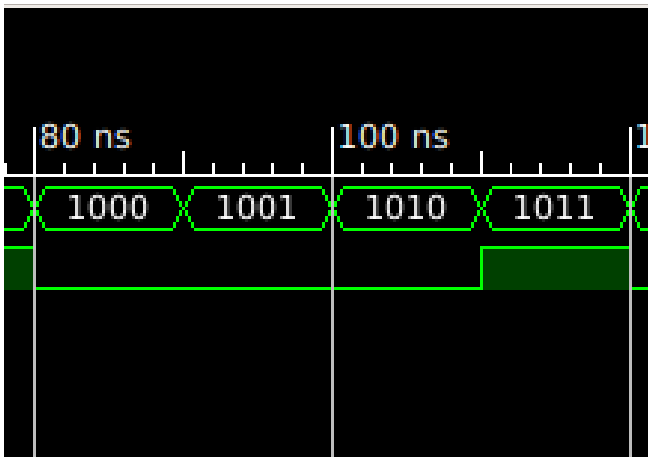
- 0: input 0000, output: 0 → 0 is not prime
- 1: input 0001, output: 1 → 1 is prime
- 2: input 0010, output: 1 → 2 is prime
- 3: input 0011, output: 1 → 3 is prime

Pt. 2



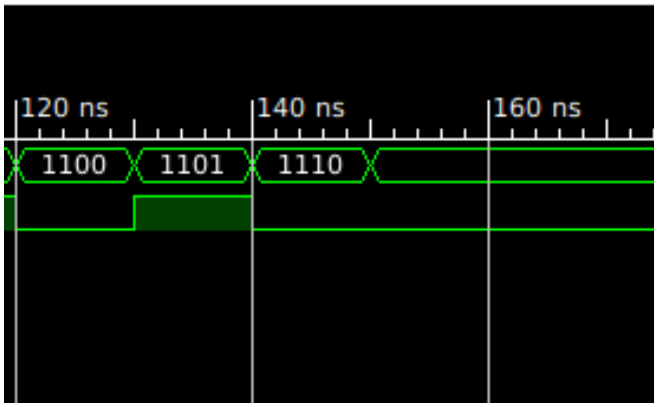
- 4: input 0100, output: 0 → 4 is not prime
- 5: input 0101, output: 1 → 5 is prime
- 6: input 0110, output: 0 → 6 is not prime
- 7: input 0111, output: 1 → 7 is prime

Pt. 3



- 8: input 1000, output: 0 → 8 is not prime
- 9: input 1001, output: 0 → 9 is not prime
- 10: input 1010, output: 0 → 10 is not prime
- 11: input 1011, output: 1 → 11 is prime

Pt. 4

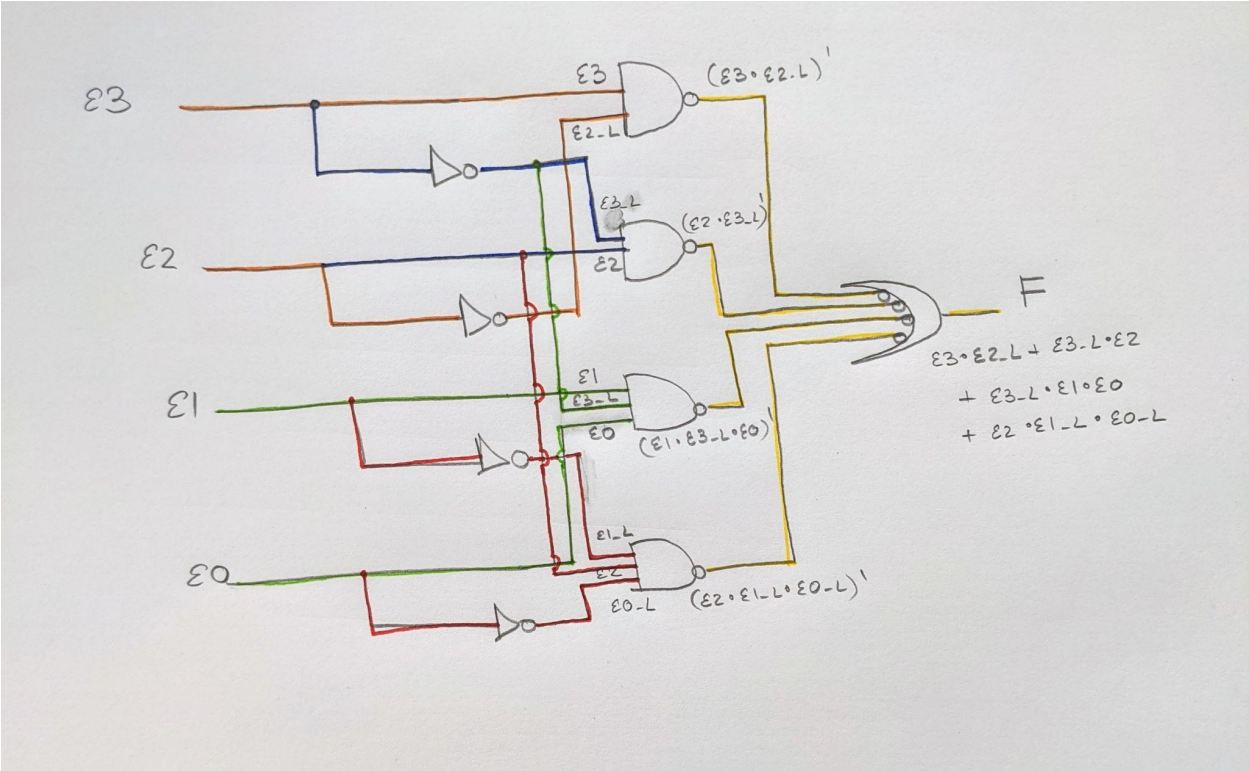


- 12: input 1100, output: 0 → 12 is not prime
- 13: input 1101, output: 1 → 13 is prime
- 14: input 1110, output: 0 → 14 is not prime
- 15: input 1111, output: 0 → 115 is not prime (1111 label in screenshot was cut out)

Task 3: Excess-3 Code Detector

The digital circuit detects whether a 4-bit unsigned binary number is valid Excess-3 code word or not. The result is stored in F.

Circuit Diagram



Entity Declaration

```
entity ex3_detector is
        port ( E : in  STD_LOGIC_VECTOR(3 downto 0);   --little endian, MSB first
        F : out  STD_LOGIC);
end ex3_detector;
```

Architecture Declaration

```
architecture ex3_detector_arch of ex3_detector is

signal E3_L, E2_L, E1_L, E0_L: STD_LOGIC;                                   --variables
signal E3_E2L, E3L_E2, E3L_E1_E0, E2_E1L_E0L: STD_LOGIC;

component INV port (I: in STD_LOGIC; O: out STD_LOGIC); end component;      --gates used
component NAND2 port (I0, I1: in STD_LOGIC; O: out STD_LOGIC); end component;
component NAND3 port (I0, I1, I2: in STD_LOGIC; O: out STD_LOGIC); end component;
component NAND4 port (I0, I1, I2, I3: in STD_LOGIC; O: out STD_LOGIC); end component;

begin
```

```
   U1: INV port map (E(3), E3_L);                                    --invert E3
   U2: INV port map (E(2), E2_L);                                    --invert E2
   U3: INV port map (E(1), E1_L);                                    --invert E1
   U4: INV port map (E(0), E0_L);                                    --invert E0
   U5: NAND2 port map (E(3), E2_L, E3_E2L);                          --NAND of 2 inputs
   U6: NAND2 port map (E3_L, E(2), E3L_E2);                          --NAND of 2 inputs
   U7: NAND3 port map (E3_L, E(1), E(0), E3L_E1_E0);                 --NAND of 3 inputs
   U8: NAND3 port map (E(2), E1_L, E0_L, E2_E1L_E0L);                --NAND of 3 inputs
   U9: NAND4 port map (E3_E2L, E3L_E2, E3L_E1_E0, E2_E1L_E0L, F); --NAND of 4 inputs,
                                                                       store in F
end ex3_detector_arch;
```

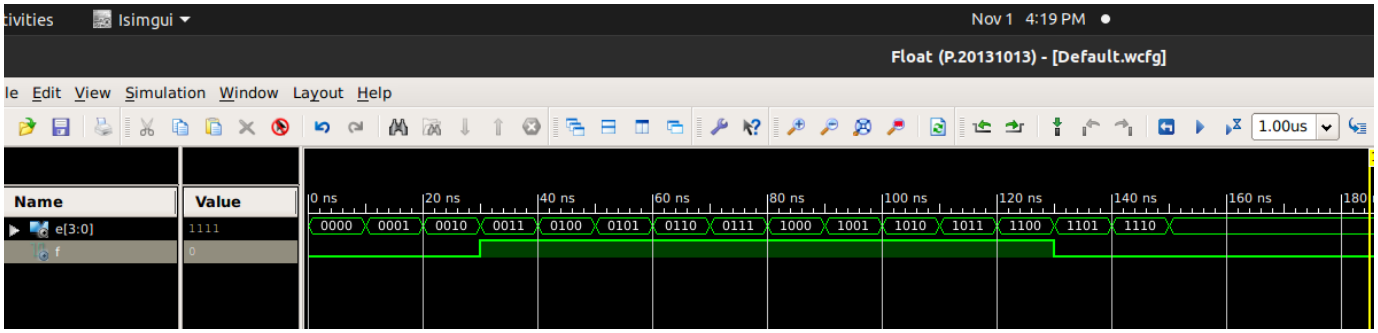
Testbench Program

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

ENTITY ex3_detector_tb IS
END ex3_detector_tb;

ARCHITECTURE behavior OF ex3_detector_tb IS
        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT ex3_detector
        PORT(
        E : IN  std_logic_vector(3 downto 0);
        F : OUT  std_logic
        );
        END COMPONENT;

  --Inputs
  signal E : std_logic_vector(3 downto 0) := (others => '0');
         --Outputs
  signal F : std_logic;

BEGIN
   -- Instantiate the Unit Under Test (UUT)
  uut: ex3_detector PORT MAP (
        E => E,
        F => F
        );

  -- Stimulus process
  stim_proc: process
```

```
  begin
        E <= "0000"; wait for 10 ns;
         E <= "0001"; wait for 10 ns;
         E <= "0010"; wait for 10 ns;
         E <= "0011"; wait for 10 ns;
         E <= "0100"; wait for 10 ns;
         E <= "0101"; wait for 10 ns;
         E <= "0110"; wait for 10 ns;
         E <= "0111"; wait for 10 ns;
         E <= "1000"; wait for 10 ns;
         E <= "1001"; wait for 10 ns;
         E <= "1010"; wait for 10 ns;
         E <= "1011"; wait for 10 ns;
         E <= "1100"; wait for 10 ns;
         E <= "1101"; wait for 10 ns;
         E <= "1110"; wait for 10 ns;
         E <= "1111"; wait for 10 ns;
        wait;
  end process;
END;
```

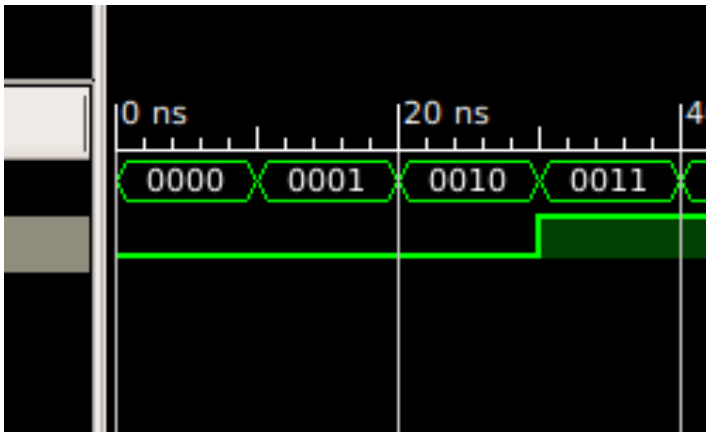
Waveform

The results of the simulation can be seen in the full waveform below:



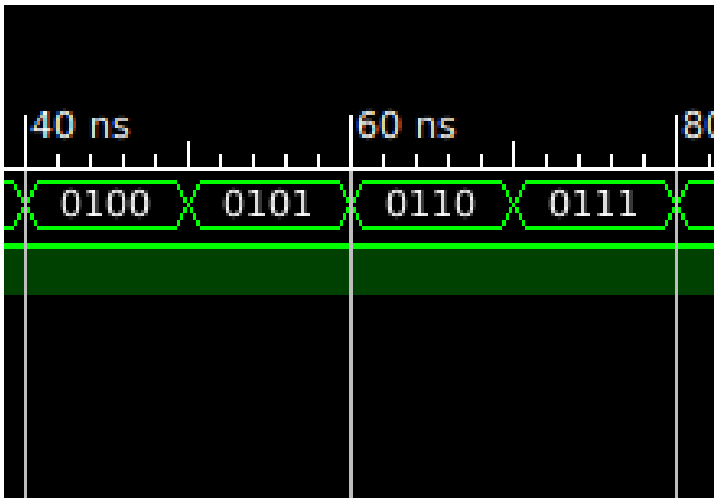
Here are partial screenshots of the waveform to make it more readable, below each is an explanation of the test cases captured in it:

Pt.1 - First 4 cases



- 0: input 0000, output: 0 → 0000 is not valid excess-3 code word
- 1: input 0001, output: 0 → 0001 is not valid excess-3 code word
- 2: input 0010, output: 0 → 0010 is not valid excess-3 code word
- 3: input 0011, output: 1 → 0011 is valid excess-3 code word

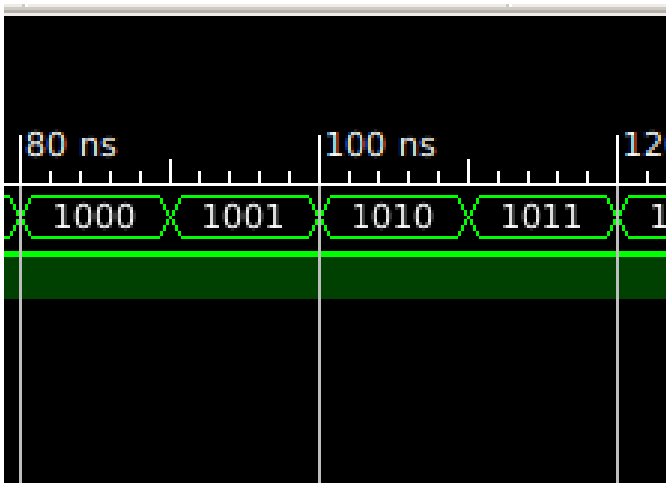
Pt. 2



- 4: input 0100, output: 1 → 0100 is valid excess-3 code word
- 5: input 0101, output: 1 → 0101 is valid excess-3 code word
- 6: input 0110, output: 1 → 0110 is valid excess-3 code word
- 7: input 0111, output: 1 → 0111 is valid excess-3 code word

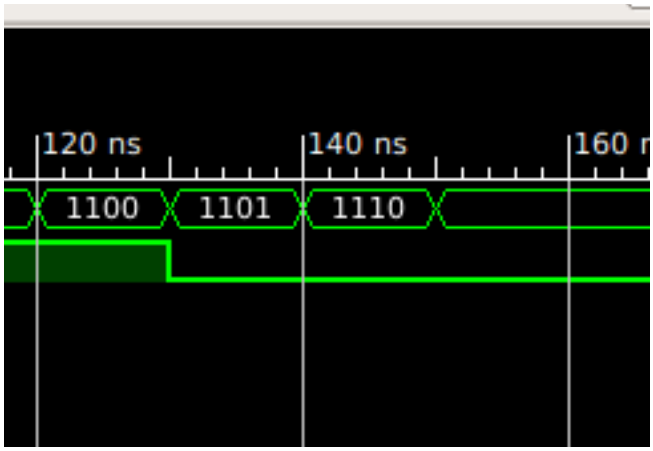
Kristina Ilyovska
VHDL Intro
10

Pt. 3



- 8: input 1000, output: 1 → 1000 is valid excess-3 code word
- 9: input 1001, output: 1 → 1001 is valid excess-3 code word
- 10: input 1010, output: 1 → 1010 is valid excess-3 code word
- 11: input 1011, output: 1 → 1011 is valid excess-3 code word

Pt. 4



- 12: input 1100, output: 1 → 1100 is valid excess-3 code word
- 13: input 1101, output: 0 → 1101 is not valid excess-3 code word
- 14: input 1110, output: 0 → 1110 is not valid excess-3 code word
- 15: input 1111, output: 0 → 1111 is not valid excess-3 code word