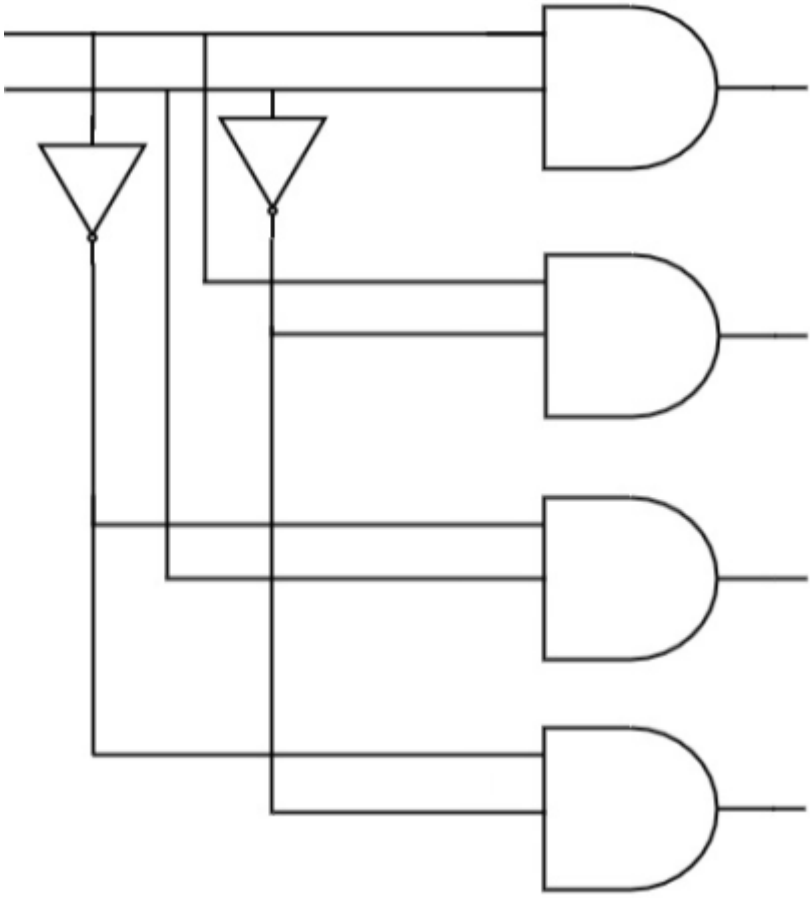
Lab 1 Report

Comparators & 2x4 Decoder

CENG 342 Digital Systems



Source: www.tutorialspoint.com

Samuel Donovan

2021-01-18

OVERVIEW

The purpose of this lab is to become familiar with Vivado and VHDL, by writing and simulating simple one- and two-bit comparators, and a 2x4 binary decoder.

--- 1-BIT IDENTITY COMPARATOR ---

A 1-bit identity comparator takes two inputs, i_1 and i_2, and outputs **true** if they are equal. **Table 1** is the truth table for a 1-bit identity comparator.

Table 1 1-bit Identity Comparator

i_1	i_0	eq
0	0	1
0	1	0
1	0	0
1	1	1

Equation 1 1-bit Identity Comparator Logic Equation

$$eq = i_1 i_0 + \overline{i_1 i_0}$$

--- 2-BIT IDENTITY COMPARATOR ---

A 2-bit identity comparator is identical to **and**ing the outputs of two 1-bit identity comparators. Here, however, the inputs are denoted a_n and b_n , with the respective *n*th inputs being compared. **Table 2** is the truth table for a 2-bit identity comparator.

Table 2 2-bit Identity Comparator

a_1	a_0	b_1	b_0	eq
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Equation 2 2-bit Identity Comparator Logic Equation

$$eq = \overline{a_1 a_0 b_1 b_0} + \overline{a_1} a_0 \overline{b_1} b_0 + a_1 \overline{a_0} b_1 \overline{b_0} + a_1 a_0 b_1 b_0$$

or

$$eq = \left(a_1 b_1 + \overline{a_1 b_1}\right) \cdot \left(a_0 b_0 + \overline{a_0 b_0}\right)$$

--- DECODER ---

Decoders take n inputs and map them to 2^n outputs. The n inputs represent a binary number that determines which of the outputs is **true**. For example, a 8-to-256 decoder given an input of `0010111` will set the 23rd output to **true**, as $0010111_2 = 23_{10}$. **Table 3** is the truth table for a 2-to-4 decoder. The input is a and the output Y.

Table 3 2x4 Decoder Truth Table

a_1	a_0	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

NOTE: Decoders often have active low outputs (0==true)

Equation 3 2x4 Decoder Logic Equations

$$Y_0 = a_1 + a_0$$

$$Y_1 = a_1 + \overline{a_0}$$

$$Y_2 = \overline{a_1} + a_0$$

$$Y_3 = \overline{a_0} + \overline{a_0}$$

DESIGN

All designs were made for a Nexys A7-100T (xc7a100tcsg324-1) FPGA board in VHDL using Vivado 2020.2.

--- 1-BIT IDENTITY COMPARATOR ---

The VHDL code in **Appendix A** is representative of the logic equation for a 1-bit identity comparator. The testbench in **Appendix B** was used to simulate and test the base code. No intermittent signals were used, as this was a relatively simple digital circuit.

--- 2-BIT IDENTITY COMPARATOR ---

The VHDL code in **Appendix C** is representative of the logic equation for a 2-bit identity comparator. The testbench in **Appendix D** was used to simulate and test the base code. Two 1-bit comparator entities were used in the code in place of the operations within the parens.

--- 2-to-4 DECODER ---

The VHDL code in **Appendix E** is representative of the four logic equations for a 2-to-4 decoder. The testbench in **Appendix F** was used to simulate and test the base code. Again, no intermittent signals were used as this was a relatively simple set of digital circuits.

SIMULATION RESULTS

Figure 1 1-bit Identity Comparator Testbench Simulation Result

Figure 2 2-bit Identity Comparator Testbench Simulation Result

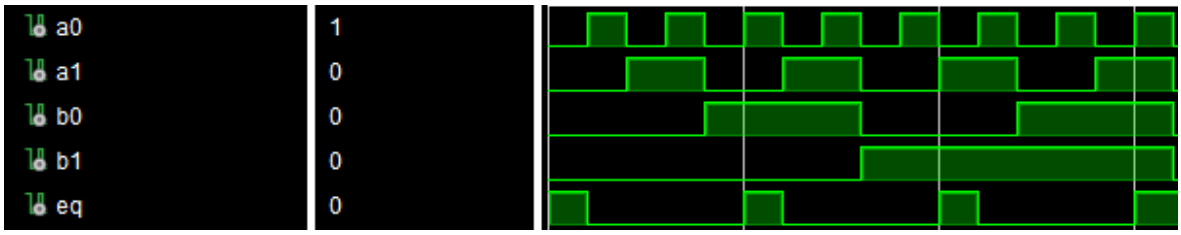
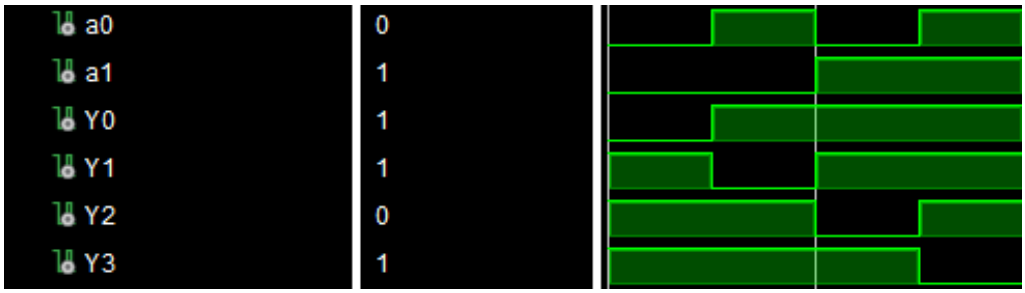


Figure 3 2x4 Decoder Testbench Simulation Result



CONCLUSIONS

There were a few issues with Vivado (the VHDL IDE used for this project) that had to be overcome.

The first issue was not having the target device needed, which was fixed by modifying the install to include the Nexys A7 board.

Secondly, the designed sources were not able to be accessed by the simulation sources (testbenches). This was fixed by checking the “Include all design sources for simulation” checkbox when creating a new simulation source.

Lastly, there were some setbacks during simulation, primarily due to our unfamiliarity with the software. By default, simulations run for 1 ms (1000 ns). Should the simulation need to run longer, the defaults may be changed, or on the toolbar exist the tools to continue the simulation. I am still unsure how to set the default “zoom” amount.

Other than these minor setbacks, everything functions as designed and was not difficult to implement.

APPENDIX A: 1-bit Identity Comparator VHDL Code (eq1.vhdl)

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


entity eq1 is
    Port ( i0 : in STD_LOGIC;
           i1 : in STD_LOGIC;
           eq : out STD_LOGIC);
end eq1;
-- TEST
architecture sop of eq1 is
begin
    eq <= (i1 AND i0) OR ((NOT i1) and (NOT i0));
end sop;
```

APPENDIX B: 1-bit Identity Comparator Testbench VHDL Code (eq1_testbench.vhdl)

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

entity eq1_testbench is
end eq1_testbench;

architecture behavioral of eq1_testbench is
  --inputs
  signal i0 : std_logic;
  signal i1 : std_logic;
  signal eq : std_logic;
begin

  -- Instantiate the Unit Under Test
  uut: entity work.eq1(sop) port map (
      i0 => i0,
      i1 => i1,
```

```
      eq => eq
      );

  -- Stimulus process
  stim_proc: process
  begin
    i0 <= '0';
    i1 <= '0';
    wait for 100 ns;
    i0 <= '1';
    i1 <= '0';
    wait for 100 ns;
    i0 <= '0';
    i1 <= '1';
    wait for 100 ns;
    i0 <= '1';
    i1 <= '1';
  end process;
end behavioral;
```

APPENDIX C: 2-bit Identity Comparator VHDL Code (eq2.vhdl)

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

entity eq2 is
    Port ( a1 : in STD_LOGIC;
           a0 : in STD_LOGIC;
           b1 : in STD_LOGIC;
           b0 : in STD_LOGIC;
           eq : out STD_LOGIC);
end eq2;

architecture sop of eq2 is
    signal s0 : std_logic := '0';
    signal s1 : std_logic := '0';
begin
    eq0: entity work.eq1(sop) port map (
      i0 => a0,
```

```
      i1 => b0,
      eq => s0
      );

    eq1: entity work.eq1(sop) port map (
      i0 => a1,
      i1 => b1,
      eq => s1
      );

    eq <= s0 AND s1;
end sop;
```

APPENDIX D: 2-bit Identity Comparator Testbench VHDL Code (eq2_testbench.vhdl)

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

entity eq2_testbench is
end eq2_testbench;

architecture behavioral of eq2_testbench is
  --inputs
  signal a0 : std_logic;
  signal a1 : std_logic;
  signal b0 : std_logic;
  signal b1 : std_logic;
  signal eq : std_logic;
begin
  -- Instantiate the Unit Under Test
  uut: entity work.eq2(sop) port map (
      a0 => a0,
      a1 => a1,
      b0 => b0,
      b1 => b1,
      eq => eq
      );
```

```
-- Stimulus process
stim_proc: process
begin
  a0 <= '0';
  a1 <= '0';
  b0 <= '0';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '1';
  a1 <= '0';
  b0 <= '0';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '0';
  a1 <= '1';
  b0 <= '0';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '1';
  a1 <= '1';
  b0 <= '0';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '0';
  a1 <= '0';
  b0 <= '1';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '1';
  a1 <= '0';
  b0 <= '1';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '0';
  a1 <= '1';
  b0 <= '1';
  b1 <= '0';
  wait for 100 ns;
  a0 <= '1';
  a1 <= '1';
  b0 <= '1';
  b1 <= '0';
  wait for 100 ns;
```

```
    a0 <= '0';
    a1 <= '0';
    b0 <= '0';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '1';
    a1 <= '0';
    b0 <= '0';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '0';
    a1 <= '1';
    b0 <= '0';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '1';
    a1 <= '1';
    b0 <= '0';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '0';
    a1 <= '0';
    b0 <= '1';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '1';
    a1 <= '0';
    b0 <= '1';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '0';
    a1 <= '1';
    b0 <= '1';
    b1 <= '1';
    wait for 100 ns;
    a0 <= '1';
    a1 <= '1';
    b0 <= '1';
    b1 <= '1';
  end process;
end behavioral;
```

APPENDIX E: 2x4 Decoder VHDL Code (decoder.vhdl)

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

entity decoder is
    Port ( a0 : in STD_LOGIC;
           a1 : in STD_LOGIC;
           Y0 : out STD_LOGIC;
           Y1 : out STD_LOGIC;
           Y2 : out STD_LOGIC;
           Y3 : out STD_LOGIC);
end decoder;

architecture sop of decoder is

begin
    Y0 <= a0 OR a1;
    Y1 <= (NOT a0) OR a1;
    Y2 <= a0 OR (NOT a1);
    Y3 <= (NOT a0) OR (NOT a1);
end sop;
```

APPENDIX F: 2x4 Decoder Testbench VHDL Code (decoder_testbench.vhdl)

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

entity decoder_testbench is
end decoder_testbench;

architecture behavioral of decoder_testbench is
```

```
  --inputs
  signal a0 : std_logic;
  signal a1 : std_logic;
  --outputs
  signal Y0 : std_logic;
  signal Y1 : std_logic;
  signal Y2 : std_logic;
  signal Y3 : std_logic;
begin
    uut: entity work.decoder(sop) port map (
          a0 => a0,
          a1 => a1,
          Y0 => Y0,
          Y1 => Y1,
          Y2 => Y2,
          Y3 => Y3
          );

  -- Stimulus process
  stim_proc: process
  begin
    a0 <= '0';
    a1 <= '0';
    wait for 100 ns;
    a0 <= '1';
    a1 <= '0';
    wait for 100 ns;
    a0 <= '0';
    a1 <= '1';
    wait for 100 ns;
    a0 <= '1';
    a1 <= '1';
    wait for 100 ns;
  end process;
end behavioral;
```