Design of Digital Systems: Lab Assignment 1

The objective of this assignment is to recognize the different styles of writing in VHDL.

Background

Not every component is well suited to every style of VHDL. It is important to be able to recognize whether behavioral, structural, or dataflow would be most appropriate style to use. In other cases, especially with more complex circuits like multipliers, using FPGA resources is the better option. In this assignment, you will learn the different architectures to build a circuit.

Program Specification

The design must meet the following specifications and use all components listed at least once.

Two-bit Comparator

The Two-bit comparator has the following features:

- Inputs (Two bits each):
  - `X, Y`: Numbers to be compared
- Outputs (One bit):
  - `Z`: High if `X` = `Y`, otherwise low.

Notes

- There will be four of these comparators. See Listing 1, Listing 2, Listing 3 and Listing 4 for the architecture of these comparators.
- The fourth one is implemented using a primitive `Look-Up Table` (LUT), which is one of the FPGA's building blocks. Normally, the value that the LUTs use to determine the output, known as an INIT value, would be set by the synthesis tools. However, the value can also be manually set. Table 1 shows how to calculate the INIT value for a 4-input AND gate. Your objective is to compute the INIT value for the 2-bit comparator.

Table 1: INIT Table

W	X	Y	Z	Out	INIT
0	0	0	0	0	0000 = 0
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	0000 = 0
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	0000 = 0
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	1000 = 8
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

This gives an INIT value of 0x8000, as it is read from MSB to LSB.

Listing 1: If-Then-Else Behavioral

```
architecture behavioral of if_then_else is
begin
    P1: process(X,Y)
    begin
        if (X = Y) then
            Z <= '1';
        else
            Z <= '0';
        end if;
    end process;
end behavioral;
```

Listing 2: When-Else Dataflow

```
architecture dataflow of when_else is
begin
    Z <= '1' when (X = Y) else '0';
end dataflow;
```

Listing 3: Boolean Dataflow

```
architecture dataflow of boolean_equation is
    signal temp : std_logic_vector(3 downto 0);
begin
    temp(3) <= not X(1) and not X(0) and not Y(1) and not Y(0);
    temp(2) <= not X(1) and     X(0) and not Y(1) and     Y(0);
    temp(1) <=     X(1) and not X(0) and     Y(1) and not Y(0);
    temp(0) <=     X(1) and     X(0) and     Y(1) and     Y(0);
    Z <= temp(3) or temp(2) or temp(1) or temp(0);
end dataflow;
```

Listing 4: LUT Primitive

```
architecture primitive of LUT_primitive is
begin
-- Instantiate LUT here.
-- Go to: Tools > Language Templates
-- Expand: VHDL > Device Primitive Instantiation > Artix-7 >
-- Slice/CLB Primitives > LUTs >LUT4
-- Check Figure 1.
-- Follow the instructions for copying the library statements
-- and the instantiations.
end primitive;
```

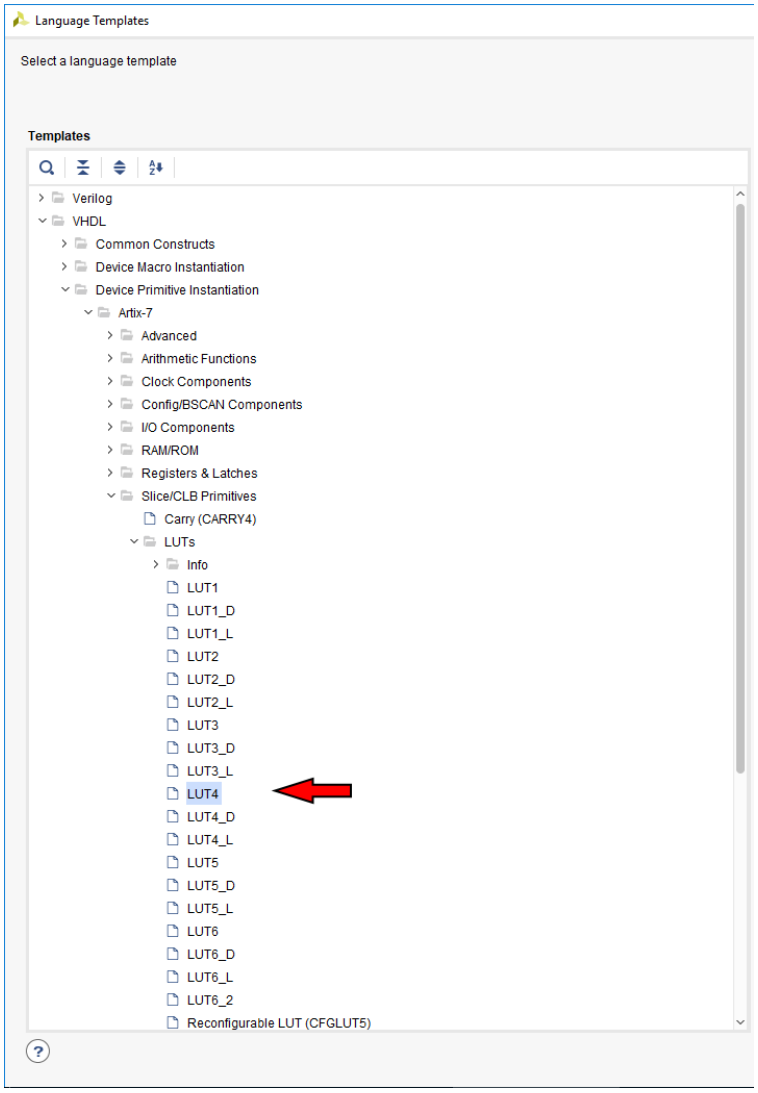

Figure 1: LUT Selection Menu

Wrapper

The top level wrapper is responsible for instantiating the different implementations of the comparator as shown in Figure 2. Listing 5 shows the architecture of the wrapper.

- Inputs: Same as Two-bit comparator.
- Outputs:
  - `z` (4 bits): Result of each comparison.

Listing 5: Wrapper

```
architecture structural of comparator_top is
    -- Component Declarations
    component if_then_else is
        port( X,Y : in std_logic_vector(1 downto 0);
                Z : out std_logic);
    end component;

    component when_else is
        port( X,Y : in std_logic_vector(1 downto 0);
                Z : out std_logic);
    end component;

    component boolean_equation is
        port( X,Y : in std_logic_vector(1 downto 0);
                Z : out std_logic);
    end component;

    component LUT_primitive is
        port( X,Y : in std_logic_vector(1 downto 0);
                Z : out std_logic);
    end component;

begin
    -- Component Port Maps
    if_then_else0 : if_then_else port map(
        X => X,
        Y => Y,
        Z => Z(3));
    when_else0 : when_else port map(
        X => X,
        Y => Y,
        Z => Z(2));
    boolean_equation0 : boolean_equation port map(
        X => X,
        Y => Y,
        Z => Z(1));
    LUT_primitive0 : LUT_primitive port map(
        X => X,
        Y => Y,
        Z => Z(0));
end structural;
```

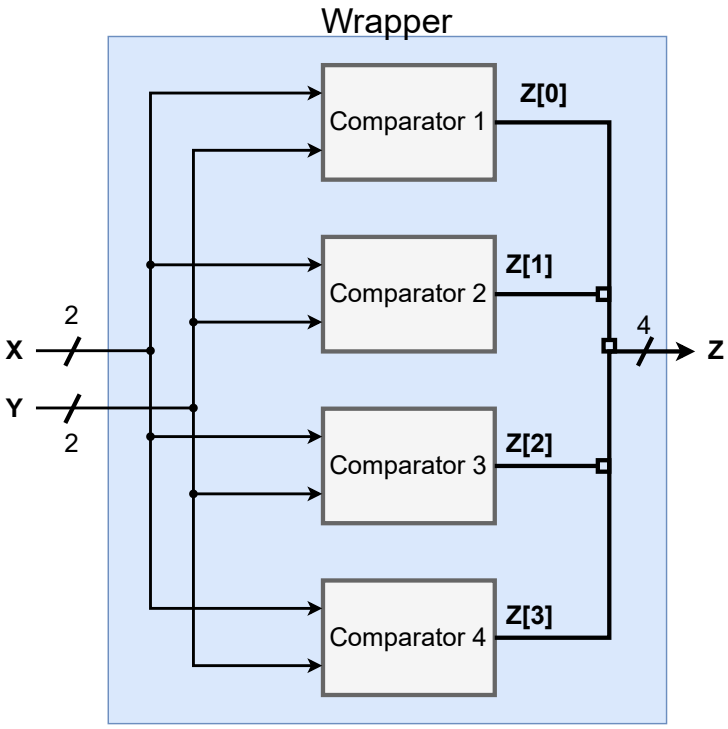

Figure 2: Wrapper schematic diagram

Testbench

Use Listing 6 to validate the two-bit comparators. Notice that we added the `std_logic_arith` package to use the `conv_std_logic_vector` function.

Listing 6: Test bench

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

-- Entity Declaration
entity comparator_tb is
    -- Notice that no inputs and outputs in testbenches
end comparator_tb;

-- Behavioral Architecture
architecture behavioral of comparator_tb is
    -- Component Declaration
    component comparator_top port(
        X, Y : in  std_logic_vector(1 downto 0);
        Z    : out std_logic_vector(3 downto 0)
    );
    end component;

    -- Signals are used instead of inputs and outputs
    signal X_in, Y_in : std_logic_vector(1 downto 0);
    signal Z_out : std_logic_vector(3 downto 0);

begin
  -- Component Instantiation
  uut: comparator_top
  port map (
    X => X_in,
    Y => Y_in,
    Z => Z_out
  );

  -- Test Bench Statements
  tb : process
  begin
    for i in 0 to 3 loop
      for j in 0 to 3 loop
        X_in <= conv_std_logic_vector(i,2);
        Y_in <= conv_std_logic_vector(j,2);
        wait for 100 ns;
      end loop;
    end loop;
    wait;
  end process;

end behavioral;
```

FPGA port map

IO	FPGA port
X[0]	SW0
X[1]	SW1
Y[0]	SW2
Y[1]	SW3
Z[0]	LED0
Z[1]	LED1
Z[2]	LED2
Z[3]	LED3