



MSCV/ESIREM

Real-Time Imaging and Control Practice

Antoine Lavault
antoine.lavault@u-bourgogne.fr

[VHDL 1]

Most of the exercises will require you to write VHDL on paper. Pay attention to the syntax.

▮ Problem 1 ▮

Create a D flip-flop with the following:

1. 2 inputs: D, clk
2. 1 output: q

▮ Problem 2 ▮

Create a synchronous 8-bit counter with asynchronous reset :

1. 2 inputs: reset, clk
2. 1 output: s

▮ Problem 3 ▮

Create a NOR gate with a generic number of inputs greater or equal to 4.
You can use the following entity:

```
entity NORgate is
  generic (
    W : positive := 4 -- number of inputs of the NOR gate
  );
  port (
    input : in std_logic_vector(W - 1 downto 0);
    output : out std_logic
  );
end NORgate;
```

Problem 4

Create an 8-bit shift register:

NB: In digital circuits, a shift register is a cascade of D flip flops, sharing the same clock, in which the output of each flip-flop is connected to the 'data' input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the 'bit array' stored in it, 'shifting in' the data present at its input and 'shifting out' the last bit in the array, at each transition of the clock input.

Problem 5

Create a multiplexer (mux) with the following:

1. 2 inputs : data(8), sel(3)
2. 1 output : s(1)
3. 1 CLK for synchronous behavior

Problem 6

Create a VHDL Design that allows the same behavior of the common-cathode BCD¹-decoder (figure 2), the 74LS48 IC².

A BCD-decoder displays the value on a 7-segment display (see figure 1).

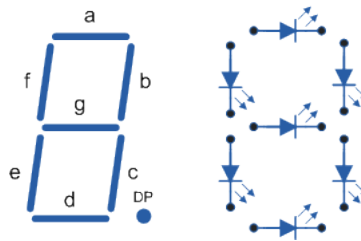


Figure 1: Example of 7 segments display

NB: 7-segment LED (Light Emitting Diode) or LCD (Liquid Crystal Display) type displays provide a very convenient way of displaying information or digital data in the form of numbers, letters, or even alpha-numerical characters.

Typically, 7-segment displays consist of seven individual colored LEDs (called segments) within one display package. To produce the required numbers or HEX characters from 0 to 9 and A to F, respectively, on the display, the correct combination of LED segments needs to be illuminated and BCD to 7-segment Display Decoders.

A standard 7-segment LED display generally has 8 input connections, one for each LED segment and

¹Binary-Coded Decimal

²<http://pdf.datasheetcatalog.com/datasheet/motorola/74LS48.pdf>

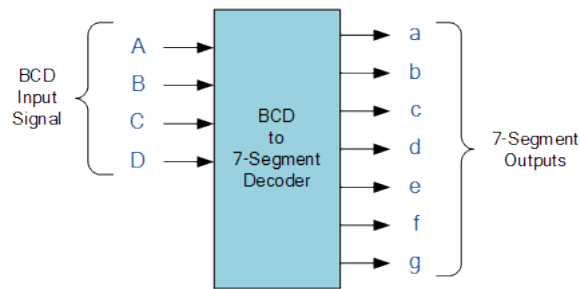


Figure 2: BCD-decoder

one as a common terminal or connection for all the internal display segments. Some single displays also have an additional input pin to display a decimal point in their lower right or left-hand corner.

Problem 7

Propose a VHDL component with a N -bit input x ($N \geq 8$) with output at a high logic level when there are at least 4 '1' and 4 '0' in x .

Problem 8

The following code describes a finite state machine.

```
-- A Mealy machine has outputs that depend on both the state and
-- the inputs.          When the inputs change, the outputs are updated
-- immediately, without waiting for a clock edge. The outputs
-- can be written more than once per state or clock cycle.
```

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

ENTITY mealy_4s IS
    PORT (
        clk : IN STD_LOGIC;
        data_in : IN STD_LOGIC;
        reset : IN STD_LOGIC;
        data_out : OUT STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
END ENTITY;

ARCHITECTURE rtl OF mealy_4s IS
    -- Build an enumerated type for the state machine
    TYPE state_type IS (s0, s1, s2, s3);
    -- Register to hold the current state
    SIGNAL state : state_type;
BEGIN
    PROCESS (clk, reset)
```

```

BEGIN
    IF reset = '1' THEN
        state <= s0;
    ELSIF (rising_edge(clk)) THEN
        -- Determine the next state synchronously, based on
        -- the current state and the input
        CASE state IS
            WHEN s0 =>
                IF data_in = '1' THEN
                    state <= s1;
                ELSE
                    state <= s0;
                END IF;
            WHEN s1 =>
                IF data_in = '1' THEN
                    state <= s2;
                ELSE
                    state <= s1;
                END IF;
            WHEN s2 =>
                IF data_in = '1' THEN
                    state <= s3;
                ELSE
                    state <= s2;
                END IF;
            WHEN s3 =>
                IF data_in = '1' THEN
                    state <= s3;
                ELSE
                    state <= s1;
                END IF;
        END CASE;
    END IF;
END PROCESS;

-- Determine the output based only on the current state
-- and the input (do not wait for a clock edge).
PROCESS (state, data_in)
BEGIN
    CASE state IS
        WHEN s0 =>
            IF data_in = '1' THEN
                data_out <= "00";
            ELSE
                data_out <= "01";
            END IF;
        WHEN s1 =>
            IF data_in = '1' THEN

```

```

                                data_out <= "01";
                                ELSE
                                data_out <= "11";
                                END IF;
WHEN s2 =>
    IF data_in = '1' THEN
        data_out <= "10";
    ELSE
        data_out <= "10";
    END IF;
WHEN s3 =>
    IF data_in = '1' THEN
        data_out <= "11";
    ELSE
        data_out <= "10";
    END IF;
END CASE;
END PROCESS;

END rtl;

```

What is the function described by this finite state machine?

Problem 9

Create the state-machine represented by the following pictures (figure: 3) :

You can assume the output is the number of the state (i.e., $S_0 \rightarrow 0$, $S_1 \rightarrow 1$, etc...)

Hint: you can use custom sub-types to encode the states like

```

type state_type is (idle, state1, state2, ...);
signal present_state, next_state: state_type;

```

Problem 10

The parity function is true if the number of bits equal to '1' at its input is even.

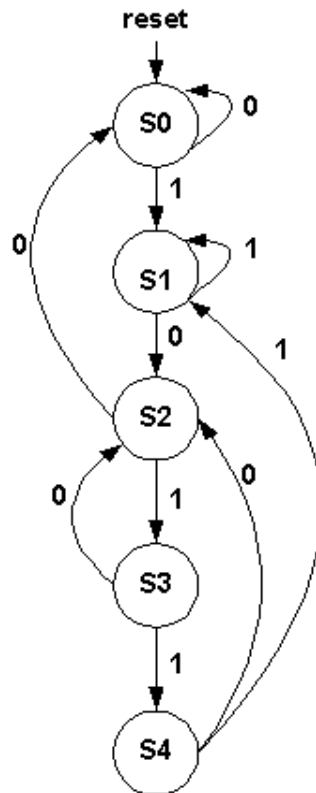
For instance, this function is true for 0000; 1111; 1001 and false for 0001; 0111.

```

entity parity is
    generic (
        W : positive := 4); -- number of input bits
    port (
        I : in std_logic_vector(W - 1 downto 0);
        F : out std_logic -- 1 if parity, 0 if not.
    );
end parity;

```

Write a testbench completely checking the parity module with an arbitrary number of bits.



State Transition Diagram

Figure 3: State machine

Remark 1. You can instantiate a UUT with the following snippet:

```
UUT: entity f (arch) generic map (W) port map (testVector, response);
```

VHDL functions or external files can get the expected test responses.

You can also use the `assert...report` statement.

Finally, it is easy to iterate over N bits with a `for` loop and get all possible values.

Problem 11

A billionaire, Bruce W., fears the privacy of his "special" affairs and wants to add a CCTV system to his manor.

The cameras used in the system are numbered $0, 1, 2, 3 \dots N$ and can detect intruders, whatever the time of day is. When an intruder is detected, the camera sends a signal on a dedicated data path.

To connect all these cameras, you are tasked to design a system that activates an alarm when at least one camera detects an intruder and gives the ID number of the activated camera (as a binary number). Since Mr. W. is feared by his enemies, and for good reasons, he is not expected to see more than one intruder at a time. But the system should raise an error flag if, by any chance, more than one camera detected an

intruder.

1. Give a VHDL code describing the case with 4 cameras
2. Same question with a generic number of cameras.