Loughborough University

# ELECTRONIC SYSTEM DESIGN WITH FPGAs

23WSC354

Semester 2 In-Person Exam paper

---

This examination is to take place in-person at a central University venue under exam conditions. The standard length of time for this paper is **2 hours**.

You will not be able to leave the exam hall for the first 30 or final 15 minutes of your exam. Your invigilator will collect your exam paper when you have finished.

Help during the exam

Invigilators are not able to answer queries about the content of your exam paper. Instead, please make a note of your query in your answer script to be considered during the marking process.

If you feel unwell, please raise your hand so that an invigilator can assist you.

---

Answer **ALL THREE** questions.

Questions carry the marks shown.

Use of a calculator is permitted - It must comply with the University's Calculator Policy for In-Person exams, in particular that it must not be able to transmit or receive information (e.g. mobile devices and smart watches are not allowed).

---

**1.**

a) Write a PROCESS that uses a IF-THEN-ELSE statement to represent the circuit illustrated in Figure Q1a . [2 marks]

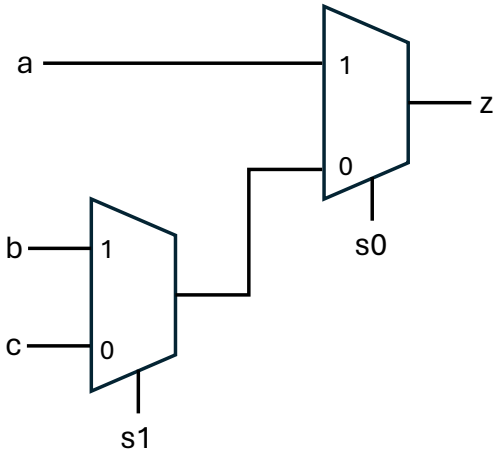



**Figure Q1a: Circuit diagram.**

b) Write a VHDL entity and architecture for a 2-input exclusive OR gate. Use a PROCESS that uses a CASE statement and the VHDL concatenate operator to represent the circuit behaviour. [3 marks]

c) Enter "T" or "F" for each question below:

i. Synthesis is the process of converting VHDL code at RTL-level into a physical circuit. [1 mark]

ii. In VHDL, signal assignment with $<=$ can be used for both sequential and concurrent statements. [1 mark]

iii. In VHDL array elements are considered to be ordered from right to left, in the same direction as index range. [1 mark]

iv. A PROCESS without a sensitivity list starts running at time zero in simulation. [1 mark]

v. A constraint file is used to set time constraints of HDL designs validated in FPGA. [1 mark]

vi. The initial keyword in SystemVerilog is used to define a procedural block that executes continuously. [1 mark]

vii. In SystemVerilog, logic and bit data types are interchangeable and can be used interchangeably in all contexts. [1 mark]

viii. The assert statement in SystemVerilog is used to check conditions during simulation and halt the simulation if the condition is false. [1 mark]

ix. Parameters in SystemVerilog modules can be overridden when instantiating the module. [1 mark]

x. The unique keyword in SystemVerilog is used to specify that a variable can only have one driver. [1 mark]

d) Write a "structural" program in SystemVerilog to implement the circuit shown in Figure Q1d with four 8-bit inputs and one 8-bit output. [3 marks]



/continued

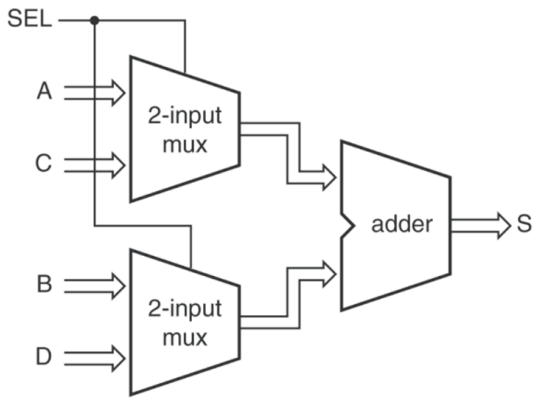



**Figure Q1d: Circuit diagram.**

e) Write a SystemVerilog program to model a 16-bit shift register. At every clock edge, it shifts “three” bits to the left. The three most significant bits shift circularly to become the three least significant bits. The design includes a synchronous reset. [2 marks]

**2.**

a) Figure Q2a(A) shows the partial code of a comparator with two 2-bit input *A* and *B*, and four outputs.

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5    ADD THE MISSING LIBRARY HERE
6   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8   entity Comparator is
9   port (
10        A,B: in std_logic_vector(1 downto 0);
11        A_less_B: out std_logic;
12        A_equal_B: out std_logic;
13        A_greater_B: out std_logic;
14        A_greater_B2: out std_logic
15  );
16  end Comparator;
17
18  architecture comparator_structural of Comparator is
19  signal tmp1,tmp2,tmp3,tmp4,tmp5, tmp6, tmp7, tmp8: std_logic;
20  begin
21      -- A = B combinational logic circuit
22      tmp1 <= A(1) xnor B(1);
23      tmp2 <= A(0) xnor B(0);
24      A_equal_B <= tmp1 and tmp2;
25
26      -- A < B combinational logic circuit
27      tmp3 <= (not A(0)) and (not A(1)) and B(0);
28      tmp4 <= (not A(1)) and B(1);
29      tmp5 <= (not A(0)) and B(1) and B(0);
30      A_less_B <= tmp3 or tmp4 or tmp5;
31
32      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33      ADD THE A > B combinational logic here
34      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35
36      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37      ADD THE A > B PROCESS() HERE
38      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39
40  end comparator_structural;
```

**Figure Q2a(A) : Partial code of a 2-bit comparator.**

i. Write a combinational logic that uses the *tmp6*, *tmp7* and *tmp8* signals to define when $A > B$. [3 marks]

ii. Write a process that uses a IF-THEN-ELSE statement, the UNSIGNED() type-cast and the TO_INTEGER function to verify when $A > B$. [2 marks]

iii. Cite one IEEE library that you could include in line 5 (Figure Q2a(A)) to support the Q2a(ii) conversions. [1 mark]

iv. Complete the process *stim_proc* described in lines 35-50 (Figure Q2a(C)). This process implements 3 FOR-LOOP statements to set the test vectors for each case: $A = B$, $A < B$ and $A > B$. Note that your process must generate the same number sequence for *A* and *B*, respecting the time intervals illustrated in Figure Q2a(B) - i.e., each number is generated every 20ns. [4 marks]



/continued

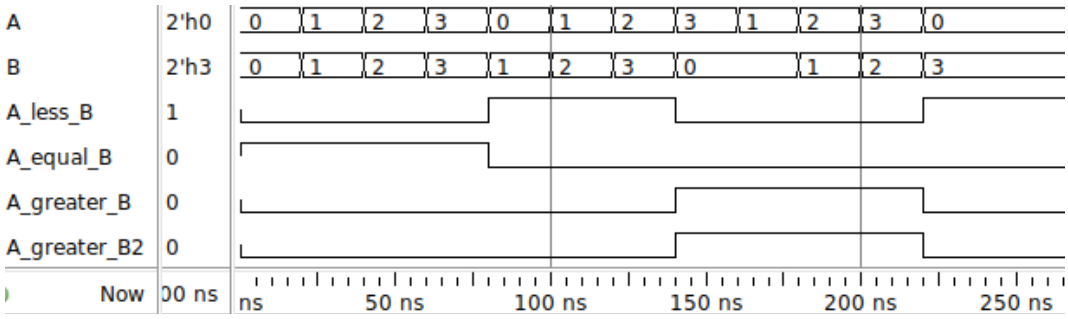



**Figure Q2a(B): Scenario validation waveform.**

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY Comparator_TB IS
5   END Comparator_TB;
6
7   ARCHITECTURE behavior OF Comparator_TB IS
8       COMPONENT Comparator
9       PORT(
10              A : IN  std_logic_vector(1 downto 0);
11              B : IN  std_logic_vector(1 downto 0);
12              A_less_B : OUT  std_logic;
13              A_equal_B : OUT  std_logic;
14              A_greater_B : OUT  std_logic;
15              A_greater_B2 : OUT  std_logic
16            );
17      END COMPONENT;
18
19     signal A : std_logic_vector(1 downto 0) := (others => '0');
20     signal B : std_logic_vector(1 downto 0) := (others => '0');
21     signal A_less_B, A_equal_B, A_greater_B, A_greater_B2 : std_logic;
22
23  BEGIN
24   -- Instantiate the comparator in VHDL
25     uut: Comparator PORT MAP (
26              A => A,
27              B => B,
28              A_less_B => A_less_B,
29              A_equal_B => A_equal_B,
30              A_greater_B => A_greater_B,
31              A_greater_B2 => A_greater_B2
32            );
33     -- Stimulus process
34     stim_proc: process
35     begin
36
37       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38       % ADD THE TEST CASES for A = B
39       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42       % ADD THE TEST CASES for A < B
43       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
44
45     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
46       % ADD THE TEST CASES for A > B
47       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48
49     end process;
50  END;
```

**Figure Q2a(C): Pseudo testbench code.**



/continued

b) Use a CASE-WHEN statement to produce a more compact version of the behaviour illustrated in Figure Q2b. Your solution must have only two WHEN choices to set the output *F* signal assignment. [3 marks]

```
1   Library IEEE;
2   USE IEEE.Std_logic_1164.all;
3
4   entity CircuitX is
5     port(
6         A, B, C: in bit;
7         F      : out bit
8     );
9   end CircuitX;
10
11  architecture Behavioral of CircuitX is
12  begin
13    process (A, B, C)
14      begin
15       if    (A = '0' and B = '0' and C= '0') then F <= '1';
16       elsif (A = '0' and B = '0' and C= '1') then F <= '0';
17       elsif (A = '0' and B = '1' and C= '0') then F <= '1';
18       elsif (A = '0' and B = '1' and C= '1') then F <= '0';
19       elsif (A = '1' and B = '0' and C= '0') then F <= '0';
20       elsif (A = '1' and B = '0' and C= '1') then F <= '0';
21       elsif (A = '1' and B = '1' and C= '0') then F <= '1';
22       elsif (A = '1' and B = '1' and C= '1') then F <= '0';
23      end if;
24    end process;
25  end Behavioral;
```

**Figure Q2b: CircuitX code.**

c) Figure Q2c shows the entity/architecture pair of a circuit, which behaviour is described with a PROCESS and two signal assignments.

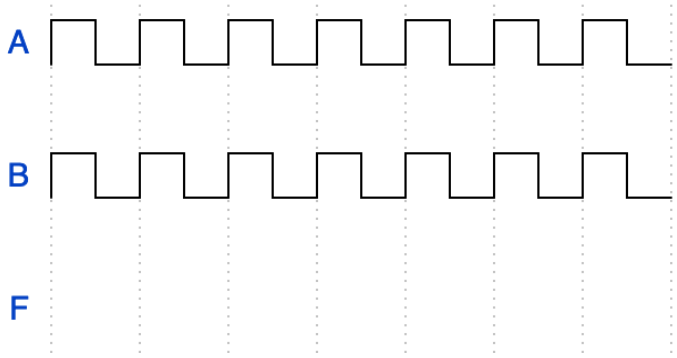
```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   entity CircuitAF is
4   port (
5         A: in bit;
6         F: out bit
7    );
8   end CircuitAF;
9
10  architecture behaviourAF of CircuitAF is
11  signal B : bit;
12     begin
13        process (A)
14           begin
15              B <= A;
16              F <= not B;
17        end process;
18  end architecture;
```

**Figure Q2c: CircuitAF entity/architecture pair in VHDL.**

i. Draw the waveform for the *F* output signal of the CircuitAF (Figure Q2c). [1 mark]



/continued




ii. Explain the *F* output signal. [2 marks]

d) Figure Q2d(A) shows a 4-input multiplexer.

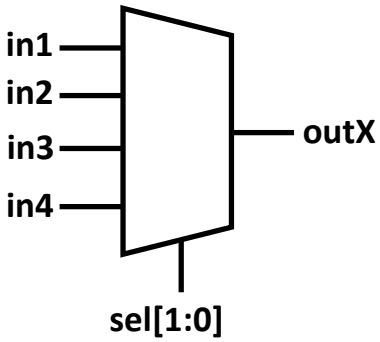



**Figure Q2d(A) : 4-input multiplexer diagram.**

Consider the pseudo-code illustrated in Figure Q2d(B) to complete the 4-input multiplexer implementation.

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

-----------------------------------------------
entity mux is
     port(
          in1, in2, in3, in4: in BIT;
          sel: signal type declaration;
          outX: out STD_LOGIC
     );
end mux;
-----------------------------------------------
ARCHITECTURE arch1 OF mux IS
BEGIN

Multiplexer Architecture ();

END arch1;
-----------------------------------------------
```

**Figure Q2d(B) : Pseudo-code of a 4-input multiplexer.**

i. Declare the *sel* signal type. [1 mark]

ii. Write a PROCESS that uses the CASE statement to implement the underlying multiplexer behaviour. [3 marks]

**3.**

a) Consider the SystemVerilog code shown in Figure Q3a, which contains a long combinational path.

```
module Add4 (output logic [7:0] Result,
             input logic [7:0] A, B, C, D);
always_comb
    Result = A + B + C + D;

endmodule
```

**Figure Q3a: SystemVerilog code.**

i. Write the pipelined version of this algorithm that cuts the long path in half by inserting a register. [3 marks]

ii. Explain how the performance of the new pipelined version is different, compared to the original version (Figure Q3a), in terms of throughput and total latency. [2 marks]

b) The code of a multiplier is given in Figure Q3b.

```
module Multiplier (output logic [15:0] Result,
                   input logic [7:0] A, B);

always_comb
        Result = A * B;

endmodule
```

**Figure Q3b: SystemVerilog code of a multiplier.**

Verify all possible "odd" input values of A and B. Each multiplication operation should complete within 20ns. Write a simple testbench to facilitate this testing. The testbench should verify the multiplier output values and display an appropriate message indicating the success or failure of each operation. [5 marks]

c) Write a SystemVerilog module for a clocked synchronous state machine with a single input, X, and a single output, UNLK. The output UNLK must be set to 1 if and only if X is 0 and the sequence of seven inputs received on X at the preceding seven clock ticks matches 0101001, considering the least significant bit as the most recent input. Your implementation should use a shift register to maintain a record of the previous values of X. [5 marks]

d) A majority circuit outputs a high signal when the majority of the bits in an input word are high.

i. Write a SystemVerilog module for a majority circuit that receives a 4-bit input. [2 marks]

ii. Write a "parameterised" SystemVerilog module for a majority circuit that determines if the majority of bits in an "N-bit" input word are high. [3 marks]

**L. Ost**
**S. Amiri**