Assignment 5 – Extra

1) The attached design contains design issues that can be identified early in the design process. Examine the code and make a list of potential issues that the linting tool can detect. Some of these design checks we covered in class, while others did not, so do your best to detect potential synthesis issues that can be seen in the design.

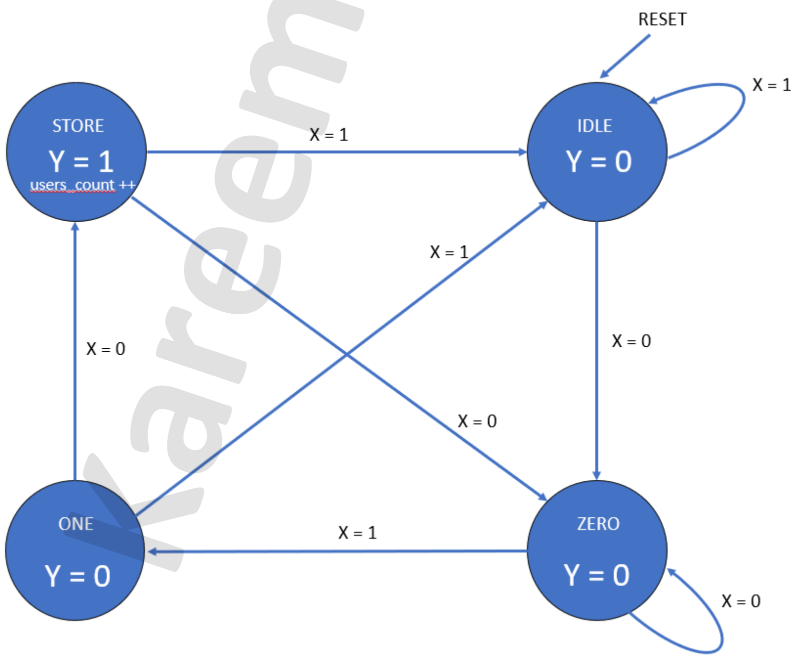
- The design is attached as Verilog file in the classroom assignment 5.

2) Requirements:

1- Design Moore FSM that detects “010” non-overlapped pattern.
2- Use sequential encoding (2’b00, 2’b01, 2’b10, 2’b11)
3- Store how many time the pattern was detected (user_count is a sequential output that increments when the state is STORE)

Ports:

Name	Type	Size	Description
x			Input sequence
clk	Input	1 bit	Clock
rst			Active high asynchronous reset
y	Output	1 bit	Output that is HIGH when the sequence 010 is detected
count		10 bits	Outputs the number of time the pattern was detected



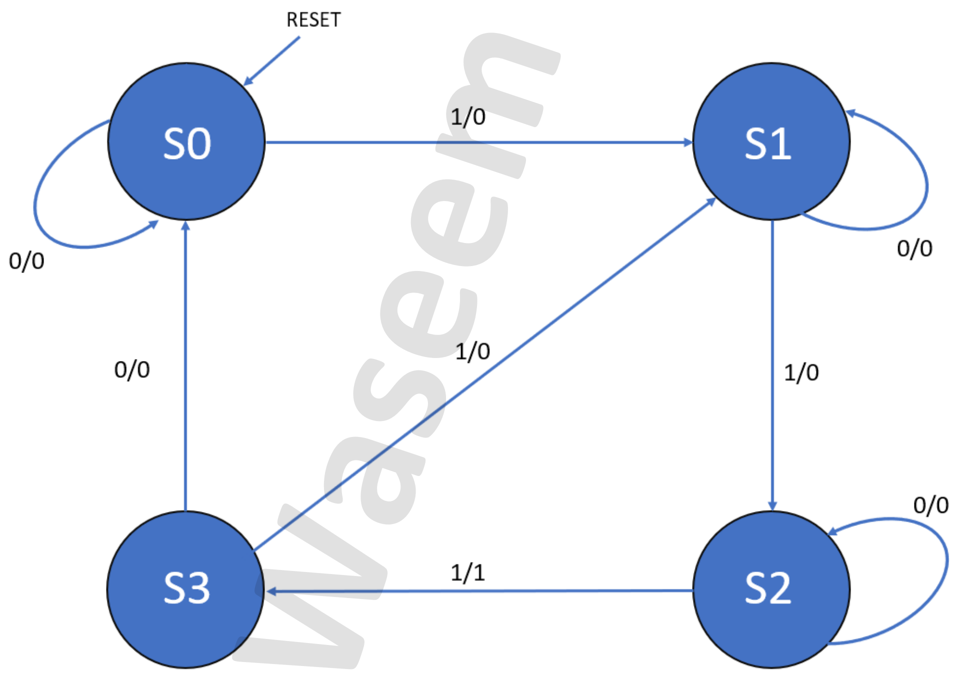
Deliverables:

- Code snippets & simulation snippets

3) Create a sequence detector using a Mealy state machine. There is one input (in) and one output (y) in the Mealy state machine. If and only if the total number of 1s received is divisible by 3, the output yout is 1. The rst of the design is active high async.

FSM Encoding:

- S0 = 2’b00
- S1 = 2’b01
- S2 = 2’b10
- S3 = 2’b11



Create a constraint file and go through the design flow where the input in is connected to a switch, output y is connected to a led and the reset is connected to a button and then generate a bitstream file.

Deliverables:

- Code snippets & simulation snippets
- Linting snippet free from warnings or errors
- Snippets from the schematic after the elaboration
- Snippet from the schematic after the synthesis and memories should be inferred in the synthesis schematic
- Snippet from the utilization report
- Snippet from the device after implementation

4) Verilog Implementation of a 4x4 Synchronous ROM

Background:

- Read-Only Memory (ROM) is a type of memory that stores predefined data and does not change during operation. In this task, you will design a 8x4 synchronous ROM in Verilog, where "8" refers to the number of addressable locations (3-bit address → $2^3 = 8$ locations) and "4" refers to the width of the data output (data_out is 4 bits). Data is read synchronously with a clock signal

The ROM should have the following specifications:

- The module has a 3-bit address input, a clock input, a reset input (sync active high), and a 4-bit data output.
- On every positive edge of the clock, if the reset signal is high, the output data should be reset to zero.
- If reset is low, the output data should be updated according to the address input as follows:
    - Address 0: Output 4'b0001
    - Address 1: Output 4'b0010
    - Address 2: Output 4'b0100
    - Address 3: Output 4'b1000
    - Address 4: Output 4'b0001
    - Address 5: Output 4'b0010
    - Address 6: Output 4'b0100
    - Address 7: Output 4'b1000

Write the Verilog code for this synchronous ROM module.

Module Interface:

- **Inputs:**
    - clk (Clock signal)
    - rst (active high sync)
    - address (3-bit input representing the ROM address)
- **Output:**
    - data_out

Design Requirements:

1. The output data_out should be updated on the rising edge of the clock.

Tasks:

- Complete the Verilog implementation of rom_8x4_sync.v
- Create a self-checking testbench to make sure the design is correct.

5)

The following design is an extension of the Linear Feedback Shift Register (LFSR) concept you learned earlier. While an LFSR is often used for pseudo-random sequence generation, the same shift-register-with-feedback structure can be adapted to perform polynomial division in hardware**,** which is the basis of the Cyclic Redundancy Check (CRC)**.**

The figure below shows a shift register with feedback connections through XOR gates.

This type of circuit is commonly used in communication systems to detect errors in transmitted data. The feedback connections are determined by a mathematical expression called a generator polynomial.

In this circuit:

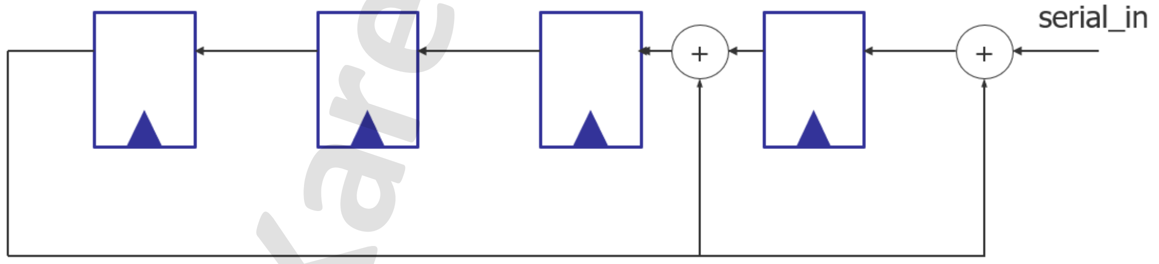
- The serial_in signal provides the input data bits one at a time (most significant bit first).
- On each clock cycle, the data bit is shifted into the register, and feedback from certain register stages is XORed back into the input stage.
- After all the data bits have been processed, the contents of the shift register represent a code that can be used to detect errors in the received data.

This process is the basis for the **Cyclic Redundancy Check (CRC)**, a widely used error detection method in digital communication.

To learn more about CRC. Please watch this video to explain the math behind it then watch this video to learn how hardware can be used to implement it. This video also has cool visual effects.

The given circuit implements the polynomial:

$$G(x) = x^4 + x + 1$$



Inputs:

- clk
- reset_n: Active-low asynchronous reset.
- data_in: Serial input data bit.
- data_valid: Data enable signal. When 1, the circuit processes data_in and updates the CRC register and when 0, the CRC value remains unchanged.

Output:

- crc: 4-bit CRC value. It contains the current remainder (CRC state) after processing the serial input data. After all message bits are processed, this holds the final CRC checksum for the data.

After writing the Verilog code, create a directed testbench where you will insert the message “b10110010000”. The expected CRC at the end is 1010.

<u>Deliverables:</u>

- Code snippets & simulation snippets.
- Run vivado synthesis, ROM is expected to be found in the RTL schematic. However, since AMD 7-Series FPGAs do not have dedicated ROM blocks, ROM is synthesized into LUTs (Look-Up Tables) or Block RAM (BRAM) with write ports disabled if the ROM size is big enough in the technology schematic.

The assignment should be submitted as a PDF file with this format <your_name>_Assignment5_Extra for example Kareem_Waseem_Assignment5_Extra.

Note that your document should be organized as 5 sections.