Homework 3 (10p)

!!! IMPORTANT !!!

We can call you to explain your solution. If you cannot explain your answer, the homework will be invalid!

KTH has a zero-tolerance policy against cheating.

https://www.kth.se/en/student/stod/studier/fusk-1.997287

If you have questions or need clarification, you can ask in the discussion forum in Canvas.

You should upload the code and testbench to GitHub and upload your report to Canvas.

QUESTIONS

1.1 QUESTION 1 (2P)

Design an FSM that can detect an *input_sequence* of 5 consecutive ones. Assume that a new input comes at every clock cycle. The output *detected* should be asserted at the same cycle when the 5th '1' is given to the input. It should remain asserted as long as the input remains '1'.

Module pinout

Name	Direction	Width	Control/Data	Description
clk	in	1	Control	Clock signal
rstn	in	1	Control	Asynchronous active low reset
input_sequence	in	1	Data	Input
detected	out	1	Data	1 if 5 consecutive ones are detected

```
module sequence_detector_structural (
    input logic clk,
    input logic rst_n,
    input logic input_sequence,
    output logic detected
);
module sequence_detector_behavioral (
    input logic clk,
    input logic rst_n,
    input logic input_sequence,
    output logic detected
);
```

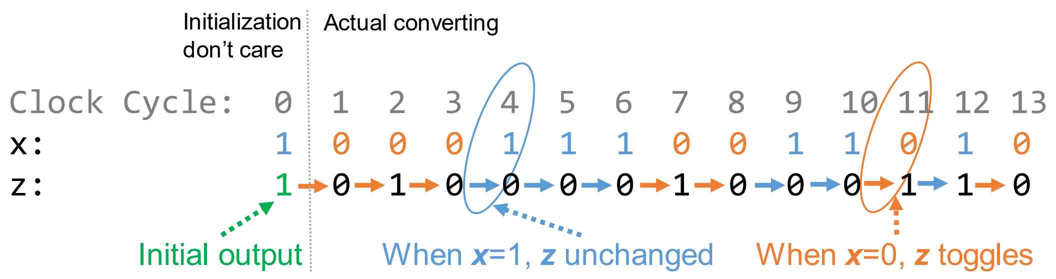
1. What should be the type of FSM machine?
2. Draw the state diagram for the FSM machine.
3. Derive the next state logic using D flops.
4. Model the logic that you derived in the previous part in SystemVerilog structurally.
5. Model the FSM in a behavioral manner with SystemVerilog.
6. Verify both structural and behavioral models and compare their results and waveforms together in your report.

1.2 Question 2 (2P)

Design a message conversion system with the conversion rule below:

For every cycle, if the input message bit is '0', toggle the current output bit; if the input message bit is '1', keep the output bit unchanged.

For example, let the input message be x, and the output message be z. Assuming z initially is '1'. Given the sequence of x = 10001110011010, the output sequence of z should be: 10100001000110. See the figure below.


Initialization don't care
Actual converting
Clock Cycle: 0 1 2 3 4 5 6 7 8 9 10 11 12 13
x: 1 0 0 0 1 1 1 0 0 1 1 0 1 0
z: 1 0 1 0 0 0 0 1 0 0 0 1 1 0
Initial output
When x=1, z unchanged
When x=0, z toggles

Module pinout

Name	Direction	Width	Control/Data	Description
clk	in	1	Control	Clock signal
rstn	in	1	Control	Asynchronous active low reset
x	in	1	Data	Input
z	out	1	Data	Output

```
module conversion_system_mealy (
    input logic clk,
    input logic rst_n,
    input logic x,
    output logic z
);

module conversion_system_moore (
    input logic clk,
    input logic rst_n,
    input logic x,
    output logic z
);
```

(1) Draw the state diagram of a **Mealy FSM** with the same behavior as described above.

(2) Draw the state diagram of a **Moore FSM** with the same behavior as described above.

(3) Model both FSMs in a behavioral manner using SystemVerilog.

(4) Write a testbench for both FSMs and compare the timing, number of states, and the output waveforms of Mealy and Moore FSMs in your report.

1.3 Question 3 (2P)

A serial communication device has a begin-sequence (011010) that leads to the transmission of 32 bits on its `serData` input. After receiving the begin-sequence, the `outValid` output becomes 1, and the next 32 bits on `serData` are regarded as valid serial data of the communication device. After 32 clock cycles, `outValid` becomes 0 and the device returns to the first state, where it searches for the begin-sequence again.

Module pinout

Name	Direction	Width	Control/Data	Description
clk	in	1	Control	Clock signal
rstn	in	1	Control	Asynchronous active low reset
serData	in	1	Data	Input
outValid	out	1	Data	Output

```
module serial_communication(
    input logic clk,
    input logic rst_n,
    input logic serData,
    output logic outValid
);
```

(1) Create a state diagram to represent the moore FSM for this system, clearly showing all states, transitions, and places where the output is issued. Draw the FSM in your report.
(2) Write a SystemVerilog description of this communication device, including the FSM and the counter.
(3) Write a testbench and verify the functionality of your design.

1.4 Question 4 (2P)

Design and implement an *average calculator* that computes the average of n m-bit input numbers, where n=2^k with $\boldsymbol{k} > \mathbf{1}$, and $\boldsymbol{m} > \mathbf{1}$. The module should begin computation when it receives a positive pulse on the `start` signal and assert the `done` signal once the average is ready.

Module parameters

parameter	Description
m	input bit width
n = 2^k , $k > 1$	number of inputs

Module pinout

Name	Direction	Width	Control/Data	Description
clk	in	1	Control	Clock signal
rstn	in	1	Control	Asynchronous active low reset
start	in	1	Control	Input control signal
inputx	in	m	Data	Input data
done	out	1	Control	Output
result	out	m	Data	Output result

```
module average_calculator #(parameter m = 8, parameter n = 4) (
    input logic clk,
    input logic rst_n,
    input logic start,
    input logic [m-1:0] inputx,
    output logic [m-1:0] result,
    output logic done
);
```

1. Draw a schematic of your datapath in your report, including the components, their interfaces, and necessary control signals.
2. Draw a state diagram showing your controller's behavior in your report. In each state, show the control signals that are issued.
3. Show the wiring between the datapath and controller within the top-level module in your report.
4. Model the datapath and controller in SystemVerilog.
5. Model the average calculator (top-level) by connecting the datapath and controller in SystemVerilog.
6. Verify the functionality of your module with a testbench.

1.5 Question 5 (2P)

The Taylor series expansion of *sinx* is shown below.

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

Design an accelerator that calculates the sin of x based on the Taylor series expansion of sinx. The x signal is a 15-bit fraction and 1-bit integer, $x \epsilon [0, \frac{\pi}{2})$. The accelerator starts calculating the sin of x when it receives a positive pulse on the `start` signal. Accelerator should issue 1 on the `done` signal when the result is calculated after eight iterations. The accelerator has a 16-bit fractional output signal called `result`, $result \epsilon [0,1)$.

Module pinout

Name	Direction	Width	Control/Data	Description
clk	in	1	Control	Clock signal
rstn	in	1	Control	Asynchronous active low reset
start	in	1	Control	Input control signal
x	in	16	Data	Input data
done	out	1	Control	Output
result	out	16	Data	Output result

```
module sin (
    input  logic        clk,
    input  logic        rst_n,
    input  logic [15:0] x,
    input  logic        start,
    output logic [15:0] result,
    output logic        done
);
```

1. Draw a schematic of your datapath in your report, including the components, interfaces, and necessary control signals. (Use the sin_coeff_lut module that is given for the coefficients.)
2. Draw a state diagram that shows the behavior of your controller in your report. In each state, show the control signals that are issued.
3. Show the wiring between the datapath and controller within the top-level module in your report.
4. Model the datapath and controller in SystemVerilog.
5. Model the sinx accelerator by connecting the datapath and controller in SystemVerilog.
6. Verify the functionality of your accelerator with a testbench.