ECE-111: Advanced Digital Design Project: Homework 6

| **Designs** | Vending Machine (Moore and Mealy), Synchronous FIFO, UARTs |
| --- | --- |
| **Deadline** | Nov 13, 2024 at 11:59pm |
| **Max. late days** | 2 (20% grade reduction per day) |

# Overview

There will be two parts for this homework. In **homework-6a** you will design a synthesizable SystemVerilog Model of a Vending Machine. Here you will be designing the Vending Machine in both a Mealy and a Moore model. In **homework-6b**, you will develop a synthesizable SystemVerilog code for Synchronous FIFO. In homework-6c, you will develop a synthesizable SystemVerilog code for a UART system. For a UART system, you need to design UART Reciever FSM (UART\_RX), UART Top Module (UART Tx-Rx Communication System), and UART Control System.

We have provided a folder called **Lab6.zip** which contains the following:

**Homework-6a:**

1. vending\_machine\_moore.sv template code for design
2. vending\_machine\_moore\_testbench.sv full code
3. vending\_machine\_mealy.sv template code for design
4. vending\_machine\_mealy\_testbench.sv full code

### Homework-6b:

1. sync\_fifo.sv design template code
2. sync\_fifo\_testbench.sv full code
3. dual\_port\_ram.sv design template code

**Homework-6c:**

Homework-6c includes

1. uart\_top folder which includes

* uart\_rx sub folder which consists of uart\_rx design template code and full testbench. Students are required to complete this code.
* uart\_tx sub folder which consists of uart\_tx full design code and full testbench. Students should review this code and also simulate it for understanding the transmitter.

1. uart\_control system folder which includes

* uart\_tc\_control, uart\_rx\_control and uart\_control\_system design template code. Students are required to complete all of these codes.
* Full testbench code for uart\_control\_system. There is no separate testbench for uart\_tx\_control and uart\_rx\_control. Instead these modules will be verified as part of uart\_control\_system simulation as both uart\_tx\_control anduart\_rx\_control are instantiated inside uart\_control\_system.

# Assignment Tasks

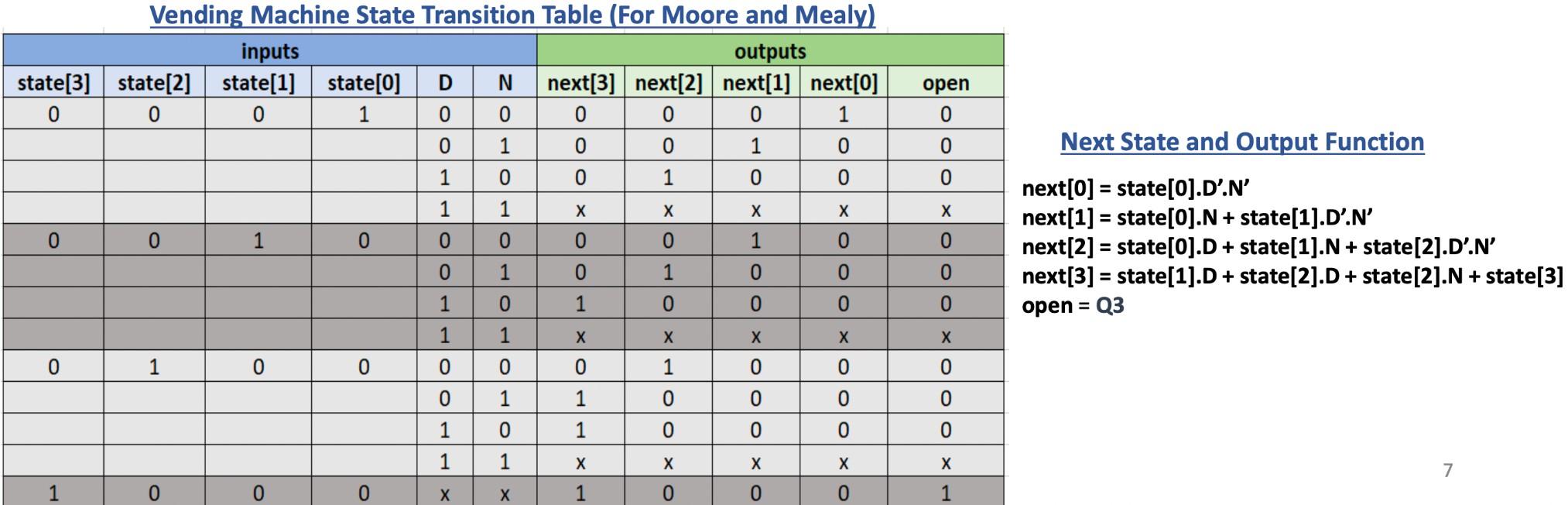
This assignment requires you to complete the following tasks:

### Recommended Tasks:

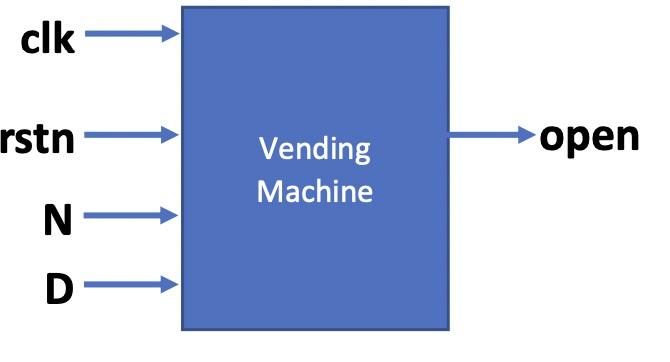
* Go over discussion video and discussion slides that go over this homework.

## For homework-6a:

### For Vending Machine develop SystemVerilog code for Moore and Mealy FSM.

* + Use one-hot encoding for state variables.
  + Review Vending state transition table and Moore FSM state transition diagram(Discussion 6) for FSM code development.
  + Design state transition diagram for Mealy implementation.
  + Synthesize and review RTL netlist schematic, state machine viewer and resource usage.
  + Simulate both Moore and Mealy implementations using the testbenches provided and review waveforms.
  + Design’s top SystemVerilog module name should be vending\_machine\_moore and vending\_machine\_mealy.

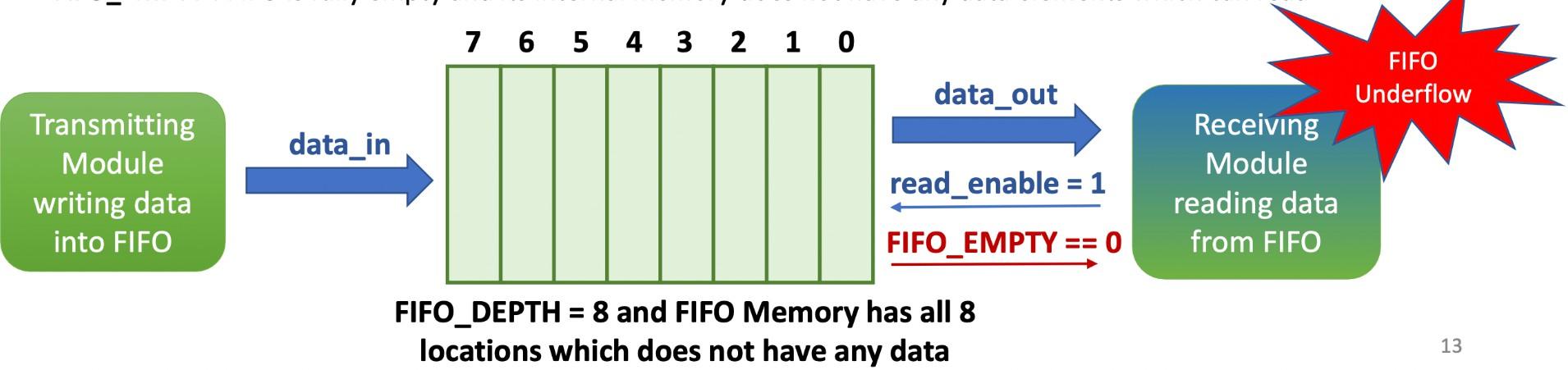
### Assume below mentioned primary Ports for Vending Machine

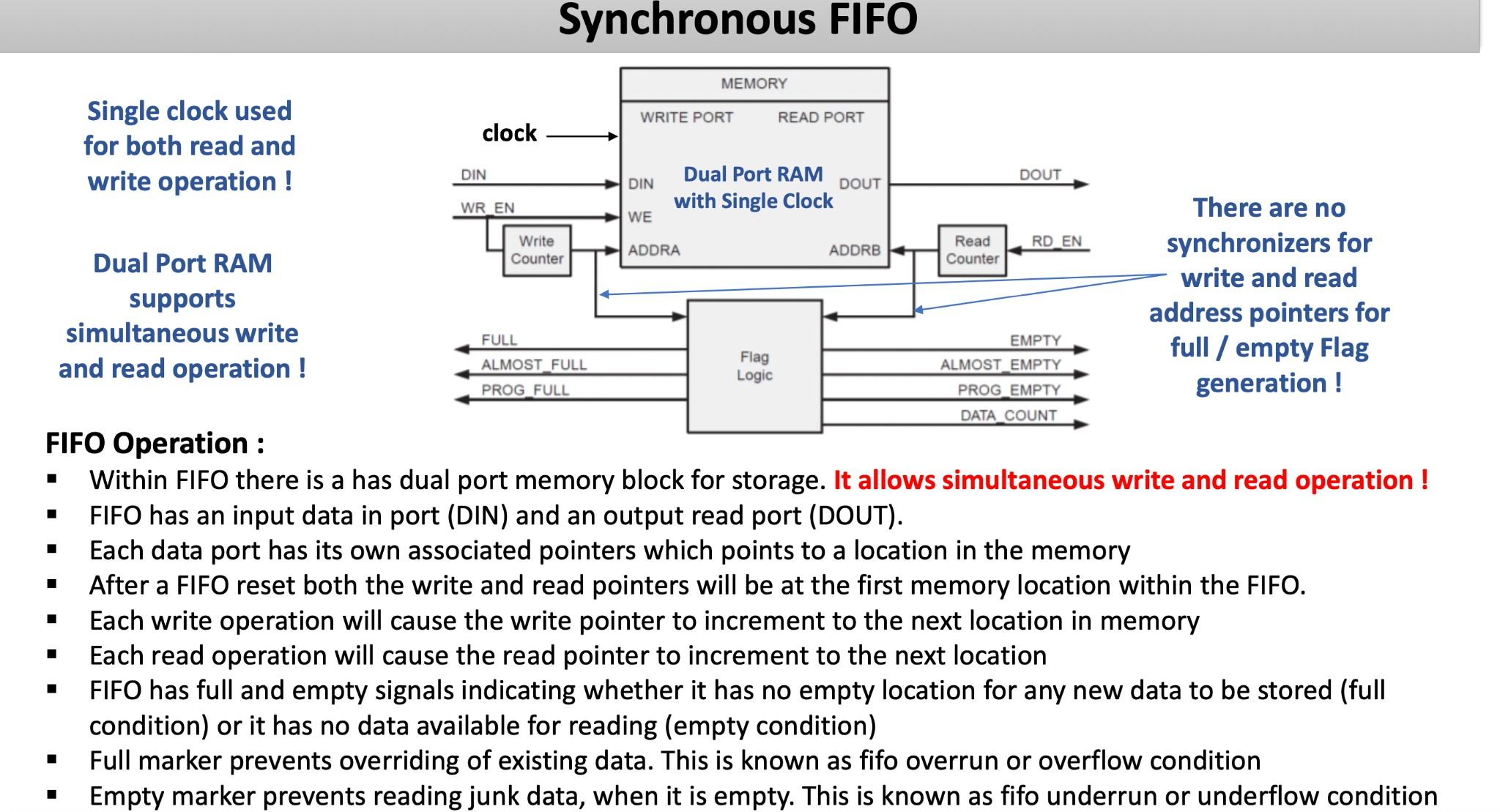
* + Input clk : posedge clock
  + Input rstn : reset should be synchronous negedge reset.
  + Input N, D : 1-bit Nickel and Dime inputs indicating Nickel and Dime are deposited if values are '1' .
  + Output open : 1-bit open signal indicating vending machine is dispatching candy.
* **Block Diagram:**
* **Reference Moore FSM Simulation Waveform:**

A screen shot of a computer

Description automatically generated

## For Homework-6b:

* **FIFO Underflow:**
  + When FIFO internal memory is fully empty and any attempt to read any location of FIFO memory is made, it is called FIFO underflow condition.
  + Underflow expression is : (FIFO\_EMPTY == 0) && (read\_enable == 1). See the below mentioned Figure.
  + It is the responsibility of the receiving module to ensure it does not read FIFO memory when it is empty.
  + Typically FIFO designs provide a FIFO\_EMPTY signal which gives an indication to the receiving module if FIFO has any data element to read or not.
  + Additionally FIFO designs provides such more flags to indicate memory occupancy information
    - **FIFO\_ALMOST\_EMPTY** : FIFO internal memory has 1 data. element remaining to be read before it is fully empty. Acts like an early empty indication.
    - **FIFO\_HALF\_EMPTY** : FIFO internal memory has 50% occupancy
    - **FIFO\_EMPTY** : FIFO is fully empty and its internal memory does not have any data elements which can be read.



### Review and Study SystemVerilog RTL model for M-bit width and N-depth Synchronous FIFO:

* + Review System Verilog code for Synchronous FIFO provided in Lab folder.
  + Review SystemVerilog code for dual port ram in Lab folder.
  + Synthesize sync\_fifo model and run simulation using testbench provided.
  + Review synthesis results (resource usage and RTL netlist/schematic).
  + Review input and output signals in simulation waveform.
    - Review dual\_port\_ram input and output signals in waveform to understand how write and read operations are performed
    - Review sync\_fifo input and output signals in waveform and understand write and read operations are performed

**For Homework-6c**

**Design UART Receiver (UART-Rx) Module using Finite State Machine in SystemVerilog** :

* Develop FSM for UART receiver (uart\_rx). Use 1 always\_ff block FSM implementation with non-blocking assignment statement
* Develop state transition diagram. Use binary state encoding for states
* Testbench will send 8-bit serial data to UART RX.
* Uart RX will convert 8-bit serial data (rx) and generate 8-bit parallel data (dout)
* done will be set to '1' when 8-bit dout parallel data is available otherwise set to '0'
* Baud rate requirement for UART-RX is 115200 bps. Which means set NUM\_CLKS\_PER\_BIT=434 for UART RX clock period = 20ns.
  + Bit period = (1 / 1152000 bps) = 8.68us, NUM\_CLKS\_PER\_BIT = 8.68us/ 20ns = 434
* However for faster simulation results design UART RX with NUM\_CLKS\_PER\_BIT=16 instead of 434
* Use testbench provided for uart\_rx receiver and simulate FSM to confirm its behavior.
* Assume below mentioned primary port list for uart\_rx module :
* **input** clk, rstn : posedge clk and synchronous active low reset
* **input** rx : 1-bit serial data input
* **output** [7:0] dout, **output** done.**A green rectangular object with black text

  Description automatically generated**

**How may states are required to design UART-RX FSM ?**

* 1 state for IDLE
* 1 state bit to receive start bit (wait for rx == 0). "rx" is input serial port of uart\_rx FSM.
* 8 states to receive 8 data bits serially (i.e. 1 data bit state to receive 1-bit of data)
* 1 state bit to receiver stop bit (wait for rx == 1)
* Total number of states = 1 IDLE state + start bit state + 8 data bit states + 1 stop bit sate = 10 states for FSM

A close-up of a text

Description automatically generated

A diagram of a diagram

Description automatically generated

UART Receiver (UART-RX) Simulation Waveform View 1

A screen shot of a computer

Description automatically generated

UART Receiver (UART-RX) Simulation Waveform View 2

A screen shot of a computer

Description automatically generated

UART Receiver (UART-RX) Simulation Waveform View 3

A computer screen with green lines and red text

Description automatically generated

**Requirements for UART Tx-Rx communication system (uart\_top)**

* Create uart\_top module with both uart\_tx and uart\_rx modules instantiated and make connections between these two
* modules
* Re-use uart\_rx module from Homework-7a and use uart\_tx module full implementation provided in Lab folderSynthesis
* uart\_top module along with uart\_tx and uart\_rx module
* Review uart\_tx design code provided in lab folder to understand UART transmitter design
* Review resource utilization including total flipflops and combination ALUT's in report
* Simulate UART Top including UART TX and RX implementation with testbench provided
* Review simulation waveform and explain results for UART Top module
* Primary port list for uart tx, uart\_rx and uart\_top modules :
* **uart\_top**: **input** tx\_clk, tx\_rstn, rx\_clk, rx\_rstn, tx\_start, **input** [7:0] tx\_din, **output** rx\_done, **output**[7:0] rx\_dout
* **uart\_rx module**: **input** clk, rstn, rx, **output** [7:0] dout, **output** done
* **uart\_tx module**: **input** clk, rstn, start, **input** [7:0] din, **output** done, **output** tx

**A diagram of a computer

Description automatically generated**

A diagram of a waveform

Description automatically generated

# Design UART Control System Module:

# Design UART TX Control FSM which will read 4 data bytes from ROM in testbench and send it to UART-Tx one by one

# Design UART RX Control FSM which will receive 4 data bytes one by one and writes each byte in RAM in testbench

# Instantiate UART\_TOP module, UART TX Control module and UART RX Control module within UART Control System Module

# A diagram of a computer system Description automatically generated

# UART TX Control FSM State Transition Diagram:

# 

# UART RX Conrol FSM State Transition Diagram

# A white sheet with black text Description automatically generated

# Requirements for UART Control System:

# Create uart\_tx\_control and uart\_rx\_control design modules using template code provided

# Create uart\_control\_system top level module with uart\_tx\_control, uart\_rx\_control and uart\_top modules instantiated

# and make connections between these modules

# Re-use uart\_top, uart\_tx and uart\_rx code from Homework7a and 7b

# Synthesis uart\_control\_system module and review source utilization including total flipflops/registers and combination

# ALUT's

# Simulate uart\_control\_system including uart\_tx\_control and uart\_rx\_control designs using testbench provided. Review

# simulation result.

# No separate testbench code provided for uart\_tx\_control and uart\_rx\_control designs. These will be verified as part

# of uart\_control\_system design simulation

# When creating uart\_control\_system project in Quartus and when simulating using Modelsim, add all these files : uart\_rx.sv, uart\_tx.sv, uart\_top.sv, uart\_tx\_control.sv, uart\_rx\_control.sv and uart\_control\_system.sv

# Primary port list for uart\_tx\_control module :

# input logic clock, rstn : posedge clock and synchronous active low reset

# output logic[3:0] mem\_write\_addr : memory write address generated to write RAM in testbench

# output logic[7:0] mem\_write\_data : memory write data generated to write data to RAM in testbench

# output logic mem\_write\_enable : memory write enable generated to enable writing to RAM in testbench

# input logic[7:0] mem\_read\_data : memory read data returned from ROM in testbench

# output logic[3:0] mem\_read\_addr : memory read address generated to read ROM in testbench

# output logic mem\_read\_enable : memory read enable generated to enable reading of ROM in testbench

# output logic transmission\_done : indicates all data bytes have been transmitted by uart tx control system

# output logic message\_received : indicates all data bytes have been received by uart rx control system

# Primary port list for uart\_rx\_control module:

* **input** logic clk, rstn : posedge clock, synchronous active low reset
* **output** logic[7:0] mem\_write\_data : output data byte to be written to RAM memory
* **output** logic [3:0] mem\_write\_addr : address to memory to write data byte received by uart\_tx fsm
* **output** logic mem\_write\_enable : if set to '1', write data byte to memory in testbench
* **input** logic uart\_rx\_done : comes from uart\_rx FSM as indication that parallel data byte is received and available to be
* written in testbench RAM memory
* **input** logic [7:0] uart\_rx\_data : parallel data byte received from uart\_rx FSM
* **output** logic message\_received : indicates that all data bytes are received by uart\_rx FSM and written to RAM memory in testbench

Primary port list for **uart\_tx\_control** module:

* **input** logic clk, rstn : posedge clock, synchronous active low reset
* **input** logic[7:0] mem\_read\_data : input data bytes from memory
* **output** logic [3:0] mem\_read\_addr : address to memory to read input data bytes
* **output** logic mem\_read\_enable : if set to '0', read data bytes from memory
* **output** logic transmission\_done : set to '1' by FSM when all data bytes are transmitted to receiver
* **input** logic uart\_tx\_done : comes from uart\_tx FSM as indication that data byte requested by tx control FSM has been
* transmissted to uart receiver
* **output** logic [7:0] uart\_tx\_data : data byte sent to uart\_tx FSM to transmit serially data to uart\_rx
* **output** logic uart\_tx\_start : tx control FSM instructs uart\_tx FSM to start data byte transmission to uart\_rx

UART Control System Simulation snapshot for reference purpose

A screen shot of a computer

Description automatically generated

# Submission Requirements

Submit a report on Gradescope in PDF format which includes the following :

**For homework-6a:**

* SystemVerilog design code for both Moore and Mealy FSM.
* Synthesis resource usage and schematic generated from RTL netlist viewer for both Moore and Mealy FSM.
* Simulation waveform snapshots and explain the simulation results for both Moore and Mealy FSM
* FSM state transition diagram for both Moore and Mealy implementations.
* Explanation of FPGA resource usage in the report is not required
* State transition diagram needs to be submitted in report and it can be either hand drawn with picture taken and pasted in report or it could be drawn in word or powerpoint or auto-generated state machine diagram from Quartus prime is also acceptable. If you’re submitting an auto-generated Quartus state machine diagram then also attach a state transition table generated from Quartus prime.
* For Moore FSM, diagram from the discussion slides can also be used.
* Simulation transcript is **not** required in report since there are no prints from testbench

**For homework-6b:**

* Synthesis resource usage snapshot and RTL netlist viewer schematic generated from Quartus for sync\_fifo top level module.
* Simulation snapshot (including the transcript) and explain simulation result of sync\_fifo :
  + Describe how data is sent to sync fifo and read from sync\_fifo, Explain full and empty flags are generated, explain how sync fifo works.
  + Explain role of dual\_port\_ram in sync\_fifo design and how read and write operation is performed to dual\_port\_ram.
* Code snippet not required.
* Explanation of FPGA resource usage in the report is not required.

**For Homework-6c:**

* SystemVerilog code snapshot for UART Receiver, UART Top, and UART Control System.
* Synthesis resource usage for all three parts.
* Simulation snapshot and a brief explanation for simulation snapshot.
* Modelsim transcripts with test passed message for all three parts of UART.
* FSM State Transition diagram and state transition table, either generated from Quartus Prime or hand-drawn.