Student Name: _____ SID: _____

This
page
intentionally
left
blank

Student Name: _____ SID: _____

Question 1. *Lab/Project Short Answer Questions* (10 points)

(a) *(True/False)* If we left the Video Encoder from Checkpoint 1 unmodified to use the ITU656 standard, we can still implement the local video system so the output does not scroll. (1 point):

(Circle one) (True) or False — Camera outputs ITU601, display offset but won't scroll

(b) *(True//False)* The primary purpose of the data clipper in the video encoder is to ensure that extremely bright or dark active video data can be displayed properly on the monitor. (1 point):

(Circle one) True or (False) — Prevents active video from being interpreted as data

(c) *(True/False)* InRequestLine and InRequestPair from the video encoder are used to index into SDRAM. (1 point):

Not used at all

(Circle one) True or (False)

(d) *(Fill in the blank)* If after a request to the video VideoROM, you received the 32-bit value 0x40C1B2D4, what are the corresponding Y, Cr and Cb values for each of the pixels represented by this pixel pair? (2 points)

Left Pixel: Y 0xB2 Cr 0xC1 Cb 0xD4

Right Pixel: Y 0x40 Cr 0xC1 Cb 0xD4

(e) *(Fill in the blank)* On bidirectional communication lines, you typically use a(n) tri-state to allow each of the individual systems to drive or release the line without generating a shorted path from the supply voltage to ground. (1 point)

(f) *(True/False, Short Answer)*In the context of this semester's project, it is necessary to explicitly refresh the SDRAM. (2 points):

(Circle one) True or (False)

Briefly justify your answer: Reading generates a refresh writeback; Video encoder reads SDRAM often enough to keep the memory refreshed.

(g) *(Short Answer)* If you were designing an SDRAM controller with a 27 MHz system clock whose setup time for the DQ line was 10 ns and hold time was 20 ns, and you had inverters with no setup or hold time but a propagation delay of 3 ns, what would you do to the system clock before sending it into the SDRAM to ensure proper timing? (2 points):

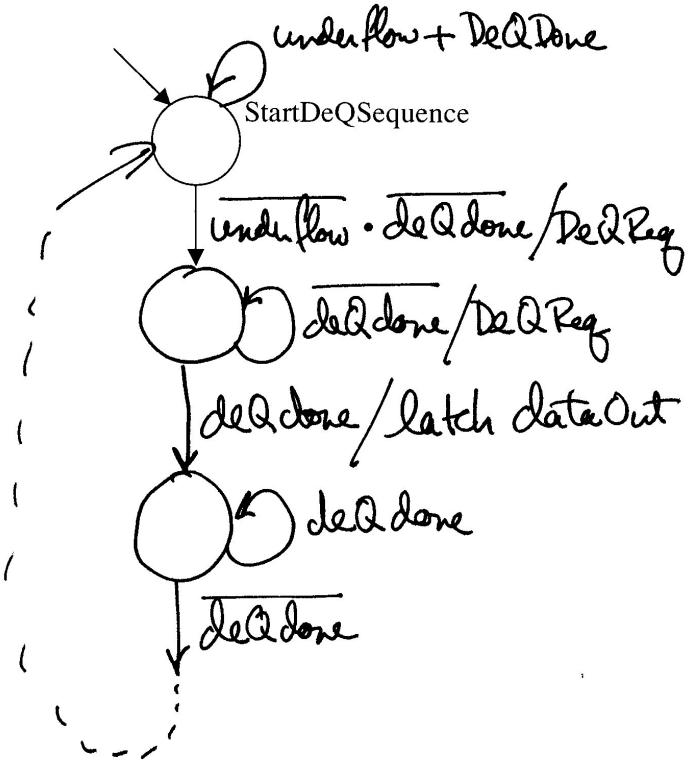
(at least) 4 inverters to delay signal by 12 ns

12 ns > 10 ns set-up time

37 ns (27 Mhz) - 12 ns = 25 ns > 20 ns hold time

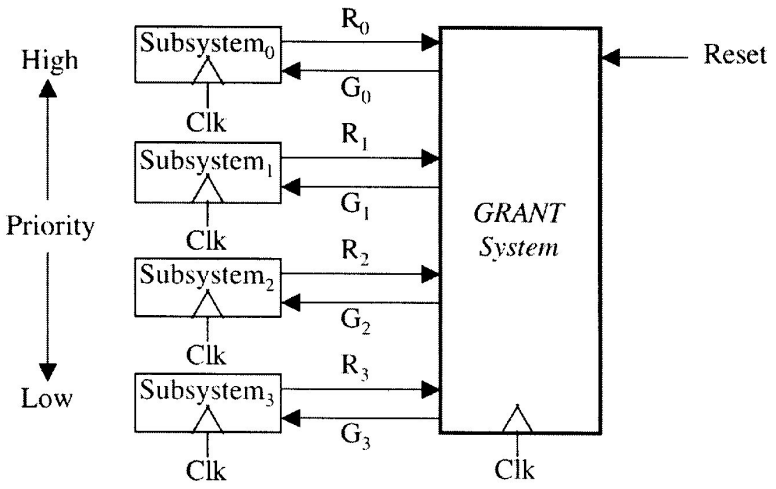
Student Name: _____ SID: _____

(c) Show a portion of a synchronous MEALY state diagram for the Consumer FSM that initiates and completes a data DeQ request through to the completion of the DeQ operation. Please use understandable labels for your states and make clear what are you outputs. (3 points)



Student Name: _____ SID: _____

Question 2. *State Machine Design* (10 Points)
You are to design a finite state machine that grants access to a shared resource to the following specification.



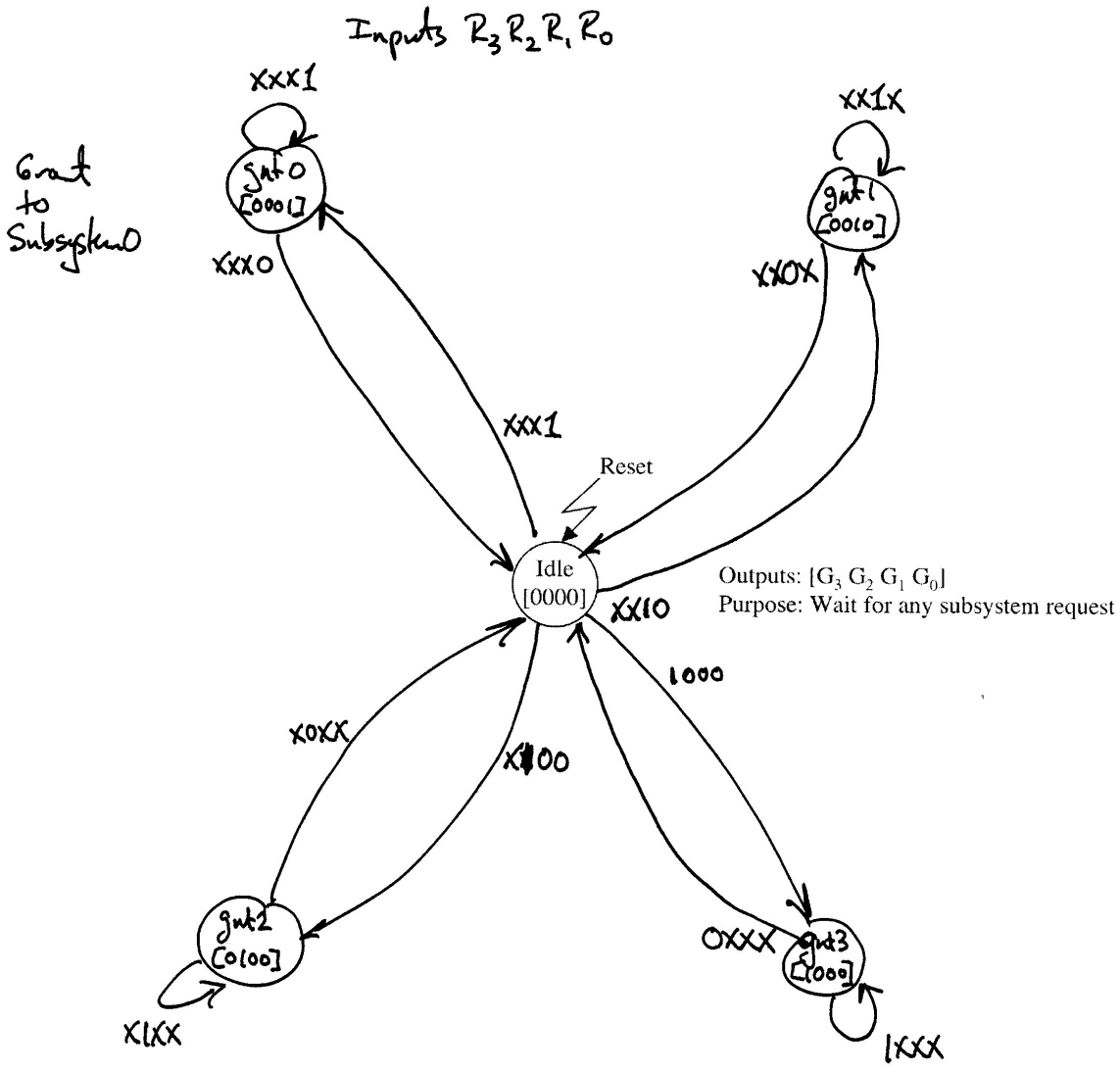
Four independent subsystems compete for exclusive access to a shared resource. The GRANT state machine you are to design gives access to the shared resource (not shown in the above figure) to the highest priority requestor in the following way.

When reset, the GRANT system enters an IDLE state (see Page 5). Each subsystem outputs a single Request line that is input to the GRANT system, R_i, where i = 0 to 3. If more than one subsystem issues a request at the same time, the GRANT system will grant access to the resource to the requesting subsystem with highest priority. Subsystem 0 has a higher priority than Subsystem 1, which in turn has a higher priority than Subsystem 2. Subsystem 3 has the lowest priority. The ability to access the resource by $Subsystem_i$ is indicated by the GRANT system by asserting the signal G_i, i = 0 to 3, which is input to $Subsystem_i$. *At any given time, at most one Gi will be asserted.* Once granted access, the subsystem must hold the R_i line high until it is done with the resource. By unasserting R_i, the subsystem signals the GRANT system to unassert G_i. This allows the GRANT machine to return to its initial IDLE state. Here it awaits the next request or proceeds to grant access to the next highest priority subsystem should there be a pending request.

You may assume that the requesting subsystems and the GRANT system all operate on a common clock.

The correct solution requires the fewest possible number of states and transitions.

On Page 5, complete the MOORE state machine for the GRANT System, naming your states and labeling your transitions so that their purpose is rendered clear to the grader.

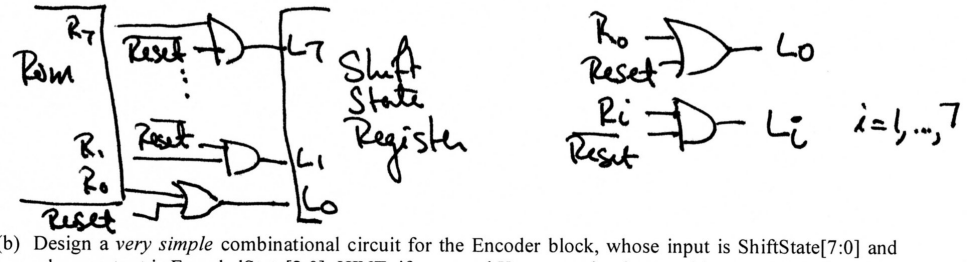

Inputs $R_3 R_2 R_1 R_0$
Grnt to Subsystem 0
XXX1
XXX0
XX1X
XX0X
XXX1
Reset
Idle
[0000]
XX10
Outputs: [G_3 G_2 G_1 G_0]
Purpose: Wait for any subsystem request
1000
XOXX
X100
OXXX
XIXX
IXXX

Student Name: _____ SID: _____

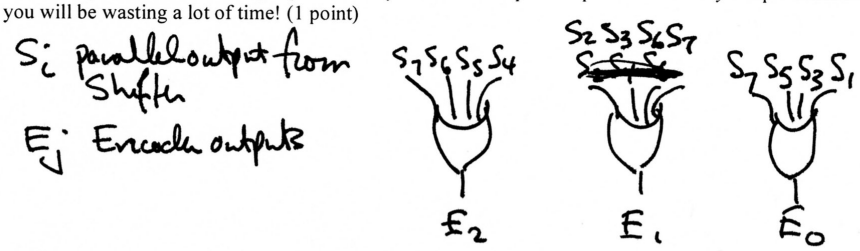
Question 3. *Shift State Controller* (10 Points)
The jump counter makes use of a counter to implement the state register of a state machine. In this problem, you will use a *shift register* for the state register rather than a counter.

The structure of the shift state controller is shown on the next page. The state register is an 8-bit right (SHR)/shift left (SHL) shifter. The states are represented in a "one-hot" encoding, i.e., $State_0 = 00000001_2$, $State_1 = 00000010_2$, $State_2 = 00000100_2$, …, $State_7 = 10000000_2$. When the Reset signal is asserted, the register must be loaded with the value 00000001_2. This is handled by the Reset Block shown on the next page. The one hot encoding is converted to a binary encoding by the encoder block. The encoder output forms the address of the Load ROM and the selection inputs of the LD, SHR, and SHL multiplexers.

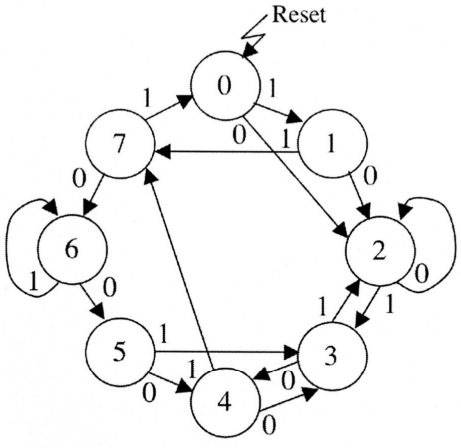
(a) Design a *very simple* combinational circuit for the Reset block that generates the correct data inputs to the Shift State register when Reset is asserted, otherwise passing through the ROM outputs. (1 point)



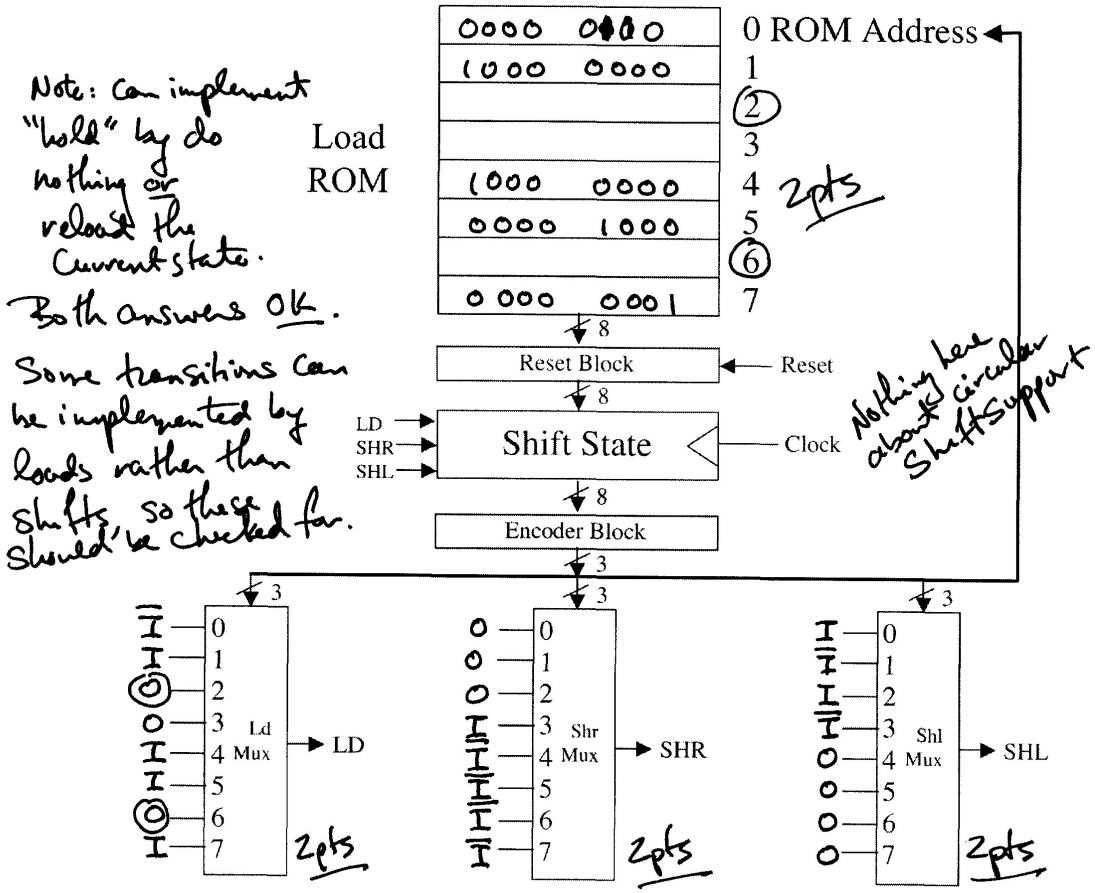
(b) Design a *very simple* combinational circuit for the Encoder block, whose input is ShiftState[7:0] and whose output is EncodedState[2:0]. HINT: if you need K-maps to implement this *very* simple function, you will be wasting a lot of time! (1 point)



(c) Given the following state diagram, with Reset, input I, their complements, the constants 0 and 1, and logical combinations of these, on Page 7 finish the wiring to complete the state machine implementation. (8 points):



Student Name: _____ SID: _____



Note: Can implement "hold" by do nothing or reload the current state.

Both answers OK.

Some transitions can be implemented by loads rather than shifts, so these should be checked for.

2pts

Nothing here about circular shift support

NOTE: By default, signals with the same label are implicitly connected. You need not explicitly connect the Shift State Register control inputs and the mux outputs.

To correctly handle Reset, need to insert the following logic Reset, LD → LD' ; Reset, SHR → Shr' ; Reset, SHL → SHL'

and wire these to the State Shift Reg.

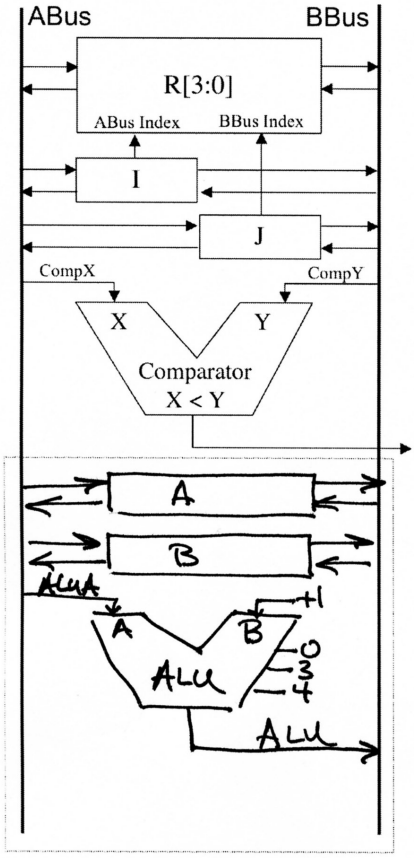
OR have the inputs to LD mux be Reset + X and SHR/SHL Mux $\overline{Reset} \cdot X$ ...

Student Name: _____ SID: _____

Question 4. *Vertical Microcode* (10 Points)
Your task is to implement a controller to sort the contents of four registers in ascending order. The following is pseudocode for the function you are to implement:

```
for i = 0 to 2 do
   A = R[ i]
   for j = i + 1 to 3 do
      B = R[ j]
      if B < A then
         R[ i]  = B
         R[ j]  = A
         A = R[ i]
      endif
   endfor
endfor
```

A partial datapath for this controller, along with the microoperations it supports is shown below. I and J are registers that hold indices to select one of the four elements of the register file R to read or write to the ABus and the BBus respectively.



R[I] → ABus	R[J] → BBus
ABus → R[I]	BBus → R[J]
I → ABus	ABus → I
I → BBus	BBus → I
J → ABus	ABus → J
J → BBus	BBus → J
ABus → CompX	BBus → CompY
X < Y true/false	

Note: I is for R/W Ri to/from ABus
J is for R/W Rj to/from BBus
to swap registers, you need some way to get from ABus to BBus!

ABUS → A A → BBUS
ABUS → B B → BBUS
ABUS → ALU A ALU → BBUS
Pass 0 ALU A + 1
Pass 3
Pass 4

Student Name: _____ SID: _____

(a) To implement the sorter controller, you need registers and functional units (FU) in addition to those shown. Augment the datapath figure by: (1) drawing in the missing datapath elements and how they interconnect to the existing components of the datapath *within the dashed box*, and (2) add to the list the new microoperations necessary for the controller to use the added elements. (4 points)

(b) Assume that you have available a vertical microcode encoding with a register transfer format and a conditional branch jump format similar to the one shown in class.

RT format: `RT ABus Src, ABusDst, BBusSrc, BBusDst, Func unit ops`

BJ format: `BJ Condition, value, jump address if match`

For example, to branch to location X if R0 < R1, you would write the following microcode:

`RT R[ I] , CompX, R[ J] , CompY, 0` (NOTE: write zero for fields you don't use or need).

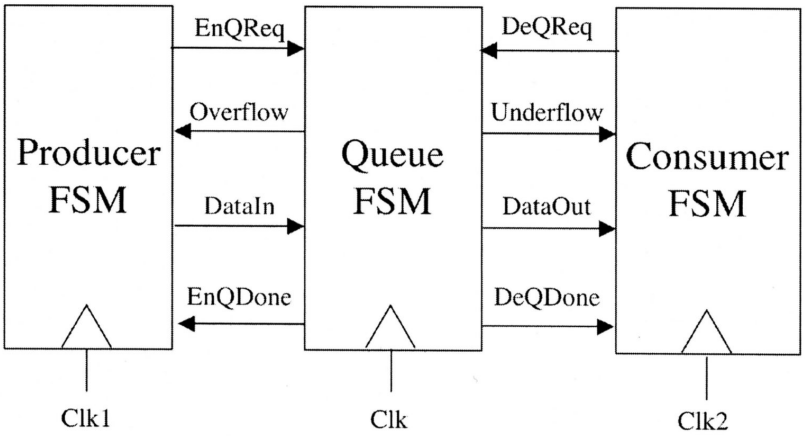
`BJ X<Y, True, LabelZ` (NOTE: use symbolic labels just as in assembler language).

Write microcode in this form in the lines below to implement the sorter controller. (6 points)

Label	RT/BJ	Microoperations
	RT	0,0 ALU, I, Pass 0
FORI	RT	I, CompX, ALU, CompY, Pass 3
	BJ	X<Y, False, Done
	RT	R[I], A, 0, 0, 0
	RT	I, ALUA, ALU, J, ALUA+1
FORJ	RT	J, CompX, ALU, CompY, Pass 4
	BJ	X<Y, False, JEND
	RT	0,0, R[J], B, 0
	RT	B, CompX, A, CompY, 0
	BJ	X<Y, False, Skip
	RT	B, R[I], 0, 0, 0
	RT	0, 0, A, R[J], 0
	RT	R[I], A, 0, 0, 0
SKIP	RT	J, ALUA, ALU, J, ALUA+1
	BJ	X<Y, TRUE, FORJ
	BJ	X<Y, FALSE, FORJ
JEND	RT	I, ALUA, ALU, I, ALUA+1
	BJ	X<Y, TRUE, FORI
	BJ	X<Y, FALSE, FORI
DONE		

Student Name: _____ SID: _____

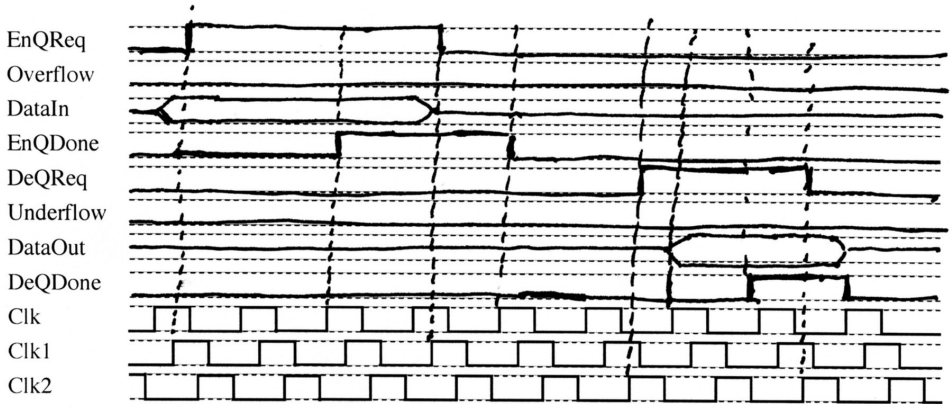
Question 5. *Design Problem* (10 Points)
You are to design parts of a system for managing a queue that couples together independently clocked producer and consumer subsystems as shown below:



The communication between producer, queue, and consumer state machines work as follows. If the Overflow signal is not asserted, it means that the Queue FSM can accept data. The Producer can commence a new enqueue operation if EnQDone is unasserted and Overflow is false. The producer places its data on the DataIn output and asserts EnQReq. It must continue to drive the DataIn lines and EnQReq until the QueueFSM asserts EnQDone. This is how the Queue FSM indicates that it has saved the DataIn in the queue.

The Consumer FSM operates in a similar fashion. The Underflow signal indicates that the Queue is empty, and there is no data to consume. If Underflow is false, and DeQDone is unasserted, then the Consumer FSM can assert DeQReq. The Queue FSM detects the DeQReq, it drives the DataOut lines and asserts DeQDone. When the Consumer FSM has saved the DataOut, it unasserts the DeQReq. The Queue FSM acknowledges this by deasserting DataOut, removing the data from its queue, and reseting DeQDone.

(a) To show that you understand the behavior of the system, use the timing diagram below to show one enqueue operation followed by one dequeue operation. You may assume that the Queue FSM has available space for new data and currently contains some data waiting for the consumer. (4 points)

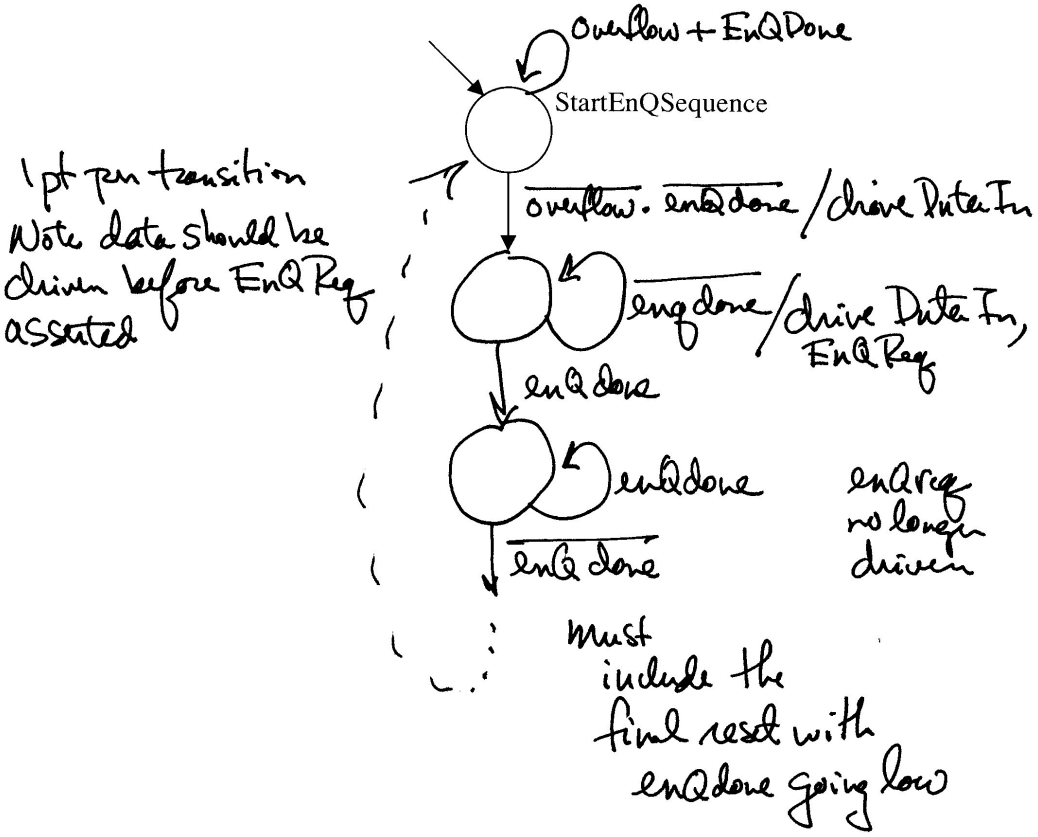


Key concept: 4-cycle handshake Req/Done 2pts

Plus proper timing of placing Data on its lines 2pts

Student Name: _____ SID: _____

(b) Show a portion of a synchronous MEALY state diagram for the Producer FSM that initiates and completes a data EnQ request through to the completion of the EnQ operation. Please use understandable labels for your states and make clear what are you outputs. (3 points)



1 pt per transition
Note data should be driven before EnQReq asserted