CDA 4203 Sec 001 Spring 2015
Computer System Design
Instructor: Dr. Srinivas Katkoori
Homework 3
Handed out on Monday, 30th March
DUE: 11:59PM, ***Saturday, 4th April***
NOTE: The due date is on a Saturday. No late homework will be accepted.

Your Name: John Gangemi **Your U#:** 68714612

1. **(10 pts.)** Design a sequential circuit which produces an output z =1 whenever any of the following input sequences occur: 1100, 1010, or 1001. The circuit resets to its initial state after a 1 output has been generated.
   a. (5 pts) Show the state diagram or table.

State Symbol	*Current State*			*Input*	*Next State*			*Output*
	Q_2	Q_1	Q_0	X	N_2	N_1	N_0	Z
A	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	1	0
B	0	0	1	0	0	1	1	0
	0	0	1	1	0	1	0	0
C	0	1	0	0	1	0	0	0
	0	1	0	1	0	1	0	0
D	0	1	1	0	1	1	0	0
	0	1	1	1	1	0	1	0
E	1	0	0	0	1	1	1	1
	1	0	0	1	0	0	1	0
F	1	0	1	0	1	1	1	1
	1	0	1	1	0	1	0	0
G	1	1	0	0	0	0	0	0
	1	1	0	1	1	1	1	1
H	1	1	1	0/1	0	0	0	0

b. (5 pts) Implement the state machine (i.e., perform state assignment and derive next state and output functions)

State Assignment	
000	A
001	B
010	C
011	D
100	E
101	F
110	G
111	H

Next State N_2

	$Q_2' Q_2'$	$Q_2' Q_1$	$Q_2 Q_1$	$Q_2 Q_1'$
$Q_0' X'$	0	1	0	1
$Q_0' X$	0	0	1	0
$Q_0 X$	0	1	0	0
$Q_0 X'$	0	1	0	1

$= Q_2'Q_1Q_0 + Q_2'Q_1X' + Q_2Q_1'X' + Q_2Q_1Q_0'X$

Next State N_1

	$Q_2' Q_2'$	$Q_2' Q_1$	$Q_2 Q_1$	$Q_2 Q_1'$
$Q_0' X'$	0	0	0	1
$Q_0' X$	0	1	1	0
$Q_0 X$	1	0	0	1
$Q_0 X'$	1	1	0	1

$= Q_1'Q_0 + Q_2'Q_0X' + Q_1Q_0'X + Q_2Q_1'X'$

Next State N_0

	$Q_2' Q_2'$	$Q_2' Q_1$	$Q_2 Q_1$	$Q_2 Q_1'$
$Q_0' X'$	0	0	0	1
$Q_0' X$	1	0	1	1
$Q_0 X$	0	1	0	0
$Q_0 X'$	1	0	0	1

$= Q_1'Q_0'X + Q_1'Q_0X' + Q_2'Q_1Q_0X + Q_2Q_0'X + Q_2Q_1'X'$

Output Z

	$Q_2' Q_2'$	$Q_2' Q_1$	$Q_2 Q_1$	$Q_2 Q_1'$
$Q_0' X'$	0	0	0	1
$Q_0' X$	0	0	1	1
$Q_0 X$	0	0	0	0
$Q_0 X'$	0	0	0	1

$= Q_2Q_1'X' + Q_2Q_0'X$

* Note there are two possible groupings of similar intent shown in the Karnaugh map, however only group $Q_2Q_1'X'$ was chosen. Please ignore the extra grouping.

2. **(5 pts.)** What is the difference between a Moore and a Mealy machine?

A Moore machine's outputs depend only on the current state whereas a Mealy machine's outputs depend on the current state and its input(s).

3. **(15 pts.)** Consider the following state machine:

PS	NS, z	
	X=0	X=1
A	B,1	H,1
B	F,1	D,1
C	D,0	E,1
D	C,0	F,1
E	D,1	C,1
F	C,1	C,1
G	C,1	D,1
H	C,0	A,1

a. (5 pts) Minimize the machine by finding equivalent states.

P_1 : (ABEFG)(CDH)

P_2 : (AB)(EFG)(CDH)

P_3 : (AB)(EFG)(CD)(H)

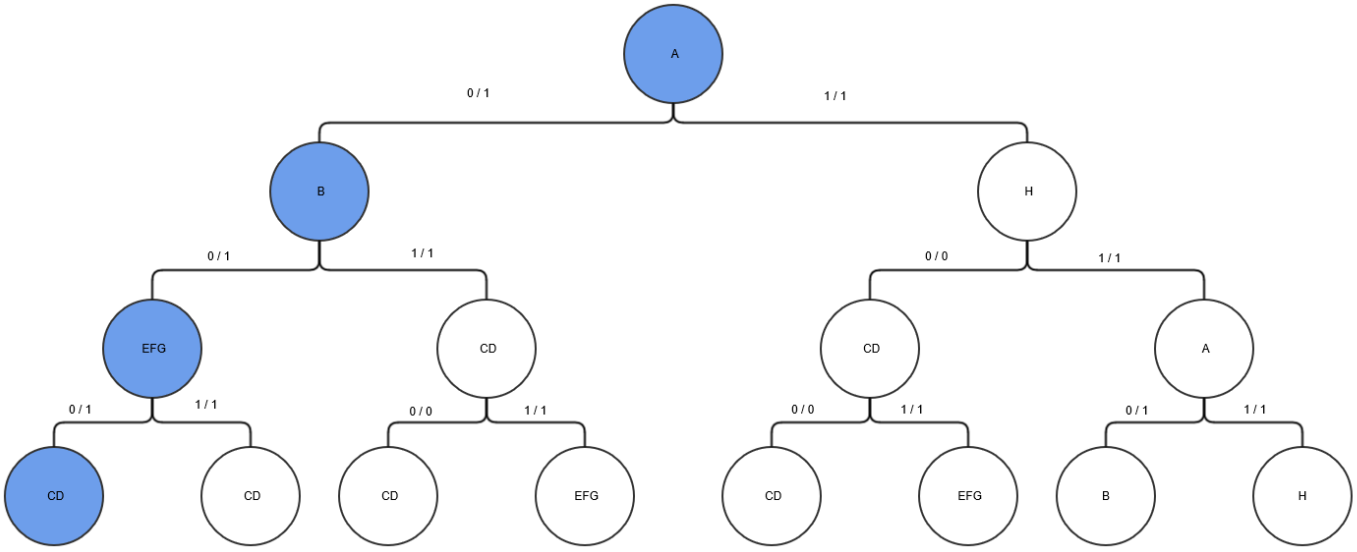
P_4 : (A)(B)(EFG)(CD)(H)

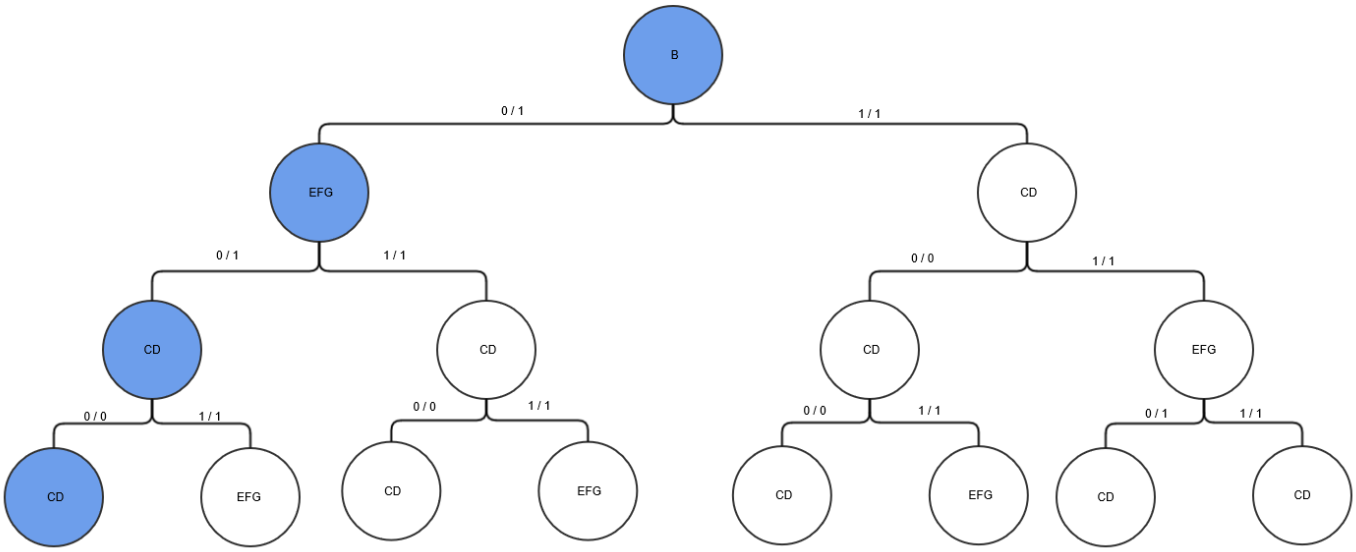
b. (5 pts) Show a ***standard form*** of the corresponding reduced machine.

Let state(s) A = Ω, B = α, EFG = μ, CD = θ, and H = ω

PS	**NS, Z**	
	x = 0	**x = 1**
Ω	α, 1	ω, 1
α	μ, 1	θ, 1
μ	θ, 1	θ, 1
θ	θ, 0	μ, 1
ω	θ, 0	Ω, 1

c. (5 pts) Find a minimum-length sequence that distinguishes state A from state B.

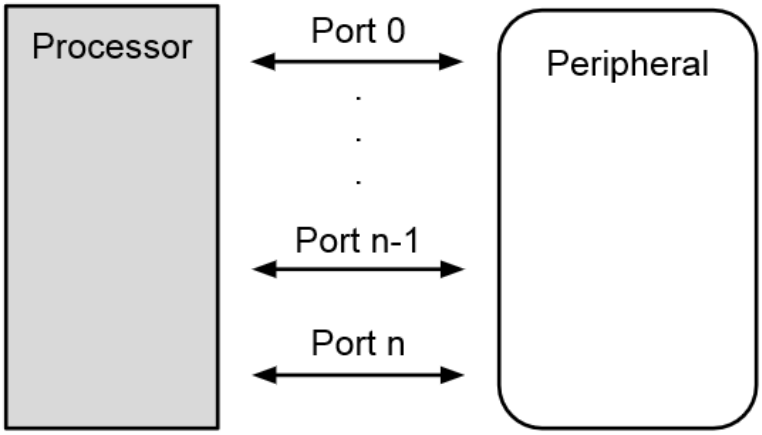




*The circles highlighted in blue are part of the minimum distinguished sequence given by inputs 0-0-0

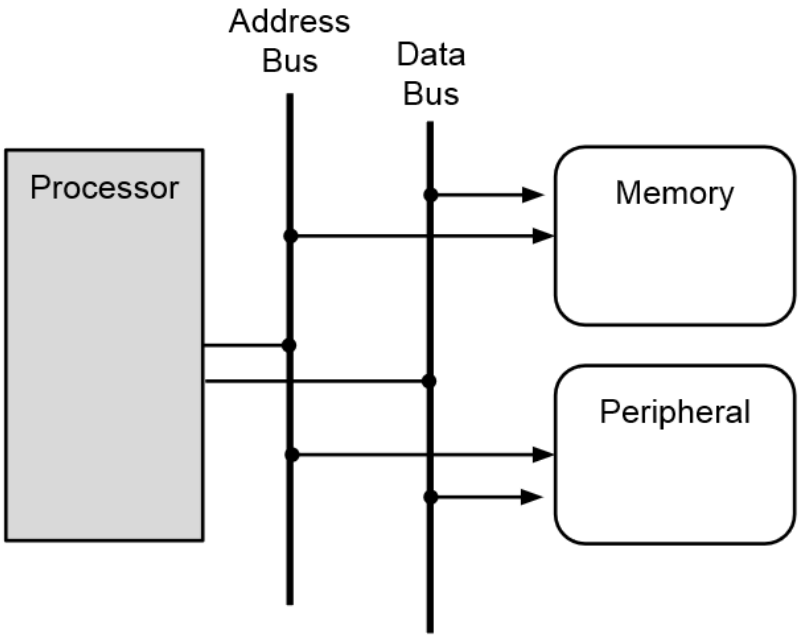
4. **(10 pts.)** Briefly explain the following with the help of a diagram
   a. Port based IO

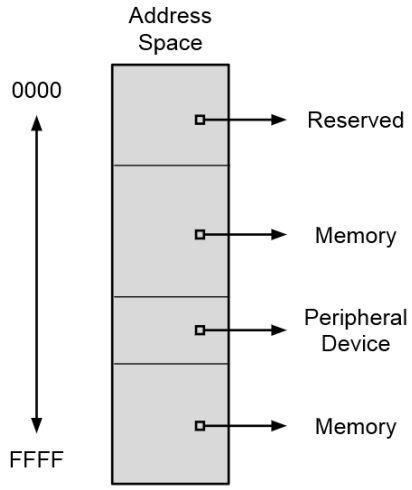
Also referred to as "parallel IO". A processor has one or more n-bit ports in which the processor reads/writes port(s) similar to a register using software.



   b. Memory Mapped IO

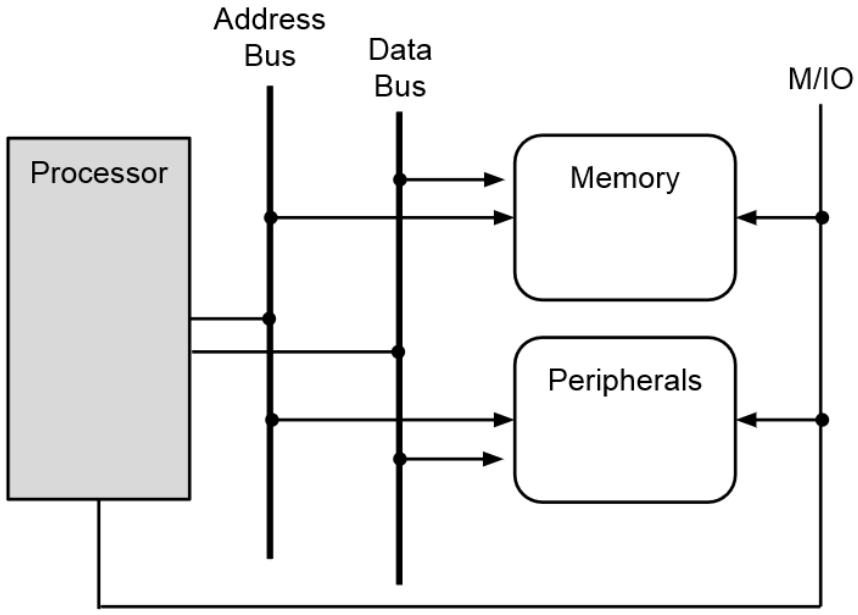
Peripheral devices share the same addressing bus with system memory as controlled by the processor. Essentially the registers in a peripheral device are mapped to addresses in the same address space used by memory. This sharing of address space allows the processor to communicate with peripheral register(s) using assembly instructions associated with memory.

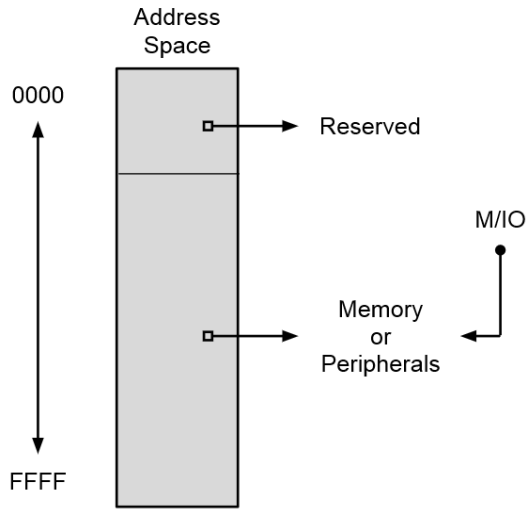




c. Standard Bus based IO

Peripheral devices share the same addressing bus with system memory however a special signal (M/IO) exists allowing the processor to specify a memory or peripheral access. The addition of the M/IO signal eliminates the need to reserve address space to peripheral devices, therefore available address space can be delegated to either memory or peripherals when necessary.





5. **(5 pts.)** What is the difference between fixed and vectored interrupts? What are their advantages?

With fixed interrupts the address of the ISR (interrupt service routine) is hard-built into the microprocessor and thus not configurable whereas with vectored interrupts the address of the ISR is not hard set in the microprocessor but provided by peripheral devices.

One advantage to fixed interrupts is a faster interrupt handling time when compared to vectored interrupts. However, vectored interrupts are more capable of handling interrupt requests from a large number of peripherals.

6. **(5 pts.)** Compare and contrast priority arbitration and daisy chain arbitration schemes.

Similarities:

- Both schemes attempt to resolve issues with multiple peripheral interrupts especially simultaneous interrupts

Differences:

- Priority arbitration requires the implementation of a single-purpose processor to handle peripheral requests whereas Daisy Chain arbitration requires the arbitration to be done by the peripherals.
- Priority arbitration connects all peripherals in parallel to a single module whereas Daisy Chain arbitration connects all peripherals in serial.
- With Priority arbitration there are two types (Fixed and Rotating) of mechanisms to handle peripheral requests priority.
- With Daisy Chain arbitration there is a single hard set method of handling peripheral requests.

- Peripheral priority can change with Priority arbitration whereas peripheral priority remains constant with Daisy Chain arbitration as the peripheral connected directly to resource receives highest priority.
- If one peripheral fails in Daisy Chain arbitration then all subsequent peripherals will cease to function as well, however Priority arbitration is not concerned with peripheral failure due to the intrinsic parallel connections.

7. **(10 pts.)** Briefly explain the operation of the following memory types of memory and compare them with respect to the storage permanence and writeability.
   a. OTP ROM

      Can only be programmed once as the physical connections between bit cells are done so with fuses and the ROM programmer blows those fuses in which no connection should exist.

   b. EPROM

      Programmable MOS transistor with a floating gate is configured to store either a logical 1 or logical 0. Given a high voltage at the gate the MOS transistor will be configured to store a logical 0 and when shining UV light on the surface of the chip for some time the MOS transistors will reset to their default state of logic 1. These devices can be reprogrammed a few thousand times.

   c. EEPROM

      Similar to EPROM in that this ROM is erasable (reprogrammable) and implements floating gate MOS transistors, however the convention to write and erase data differs from EPROM in that high voltage is used to write and erase data from the chip. These devices can be reprogrammed tens of thousands of times and are far more convenient to do so than EPROM.

   d. Flash Memory

      An extension of EEPROM, similar in functionality, write ability, and storage permanence. Although there are options to erase and write whole blocks of memory at once rather than words.

   e. SRAM

      Volatile memory that uses a 6 transistor flip-flop to hold data. A number of primary signals must be asserted to access each individual cell (Bit-line, Word line, Precharge, optional Write).

      Provides fast access time to data stored by the back-to-back coupled inverters. Relatively expensive per bit compared to other memory technologies. Area per bit (6 transistors) is large and therefore it's available in smaller sizes when compared to other memory types.

   f. DRAM

      Volatile memory that uses a single MOS transistor and capacitor to hold data.

Provides slower access time than SRAM but offers far smaller area per bit and thus it's available in larger sizes. Downside to the simplistic design is the capacitor will leak current and therefore it all bits must be constantly refreshed in order to retain the appropriate value.

Memory Type	Write Ability	Storage Permanence
OTP ROM	few writes	product life-time
EPROM	thousands of writes	tens of years
EEPROM	tens of thousands of writes	tens of years
Flash	tens of thousands of writes (block-capable)	tens of years
SRAM	unlimited writes	dependent upon power
DRAM	unlimited writes	dependent upon power

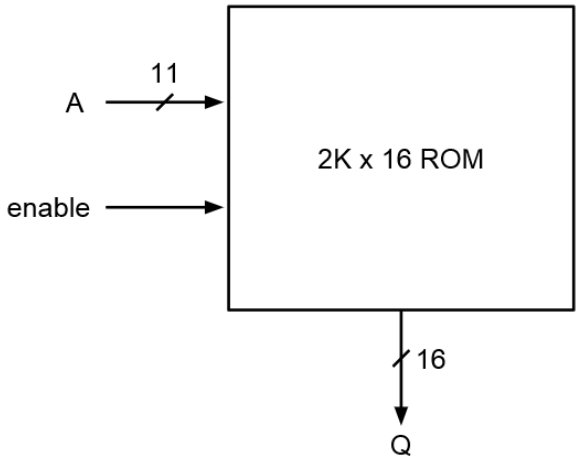
8. **(10 pts.)** Compose 1K x 8 ROMs into a 2K x 16 ROM.

To compose the 1K x 8 ROMs into 2K x 16 ROM it was necessary to find the total number of bits needed to address all 2K words of 16-bits in width.

- $2K = 2^1 * 2^{10} = 2^{11}$
- 11-bits are needed to address 2K words

The MSB will be used to delineate between "levels" of 1K ROMs, therefore a decoder must be used with this design to select the create level of 1K ROMs.

Block diagram of 2K x 16 ROM:



2K x 16 ROM Design: