CDA 3201L - Thursday (3:30 - 4:45PM) Section 005

Lab #06 - Sequential Logic Circuits II

10/29/2016 - Steven Romeiro, Jennyfer Munoz

Purpose & Objectives:

This lab required us to create shift registers. Part A requires the creation of a 4-bit shift register with the use of Flip Flops. It was our choice on what flip flop to use for implementation. Part B required the implementation of IC 74LS194 which is a 4-bit bidirectional shift register with parallel and serial operating modes. The objective of this lab is to implement shift registers, which are foundational to binary multiplication and division operations. Our knowledge of memory registers from previous assignments are required in order to make this work.

Components Used:

Name	Type	Quantity
74LS08	AND IC (2-inputs)	2
74LS74	D Flip-Flop	2
74LS04	Inverter IC	3
74LS194	4-bit Bidirectional Shifter	1
470 Ω Resistor	Resistor	4
LED	Red LED	4
Power Supply	5v	1
Frequency Generator	Wavetek 4MHz	1
Wire Kit	Assorted	1

Description:

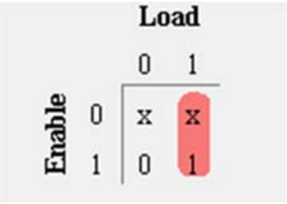
Part A:

Designing the 4-bit shift register with right shifting. Logic required the ability to parallel load asynchronously while showing parallel outputs immediately. During this load clocked inputs were to be overridden. This means that no matter what D Serial Input was placed on the first register, that input, along with the clock would be ignored until parallel load was disabled.

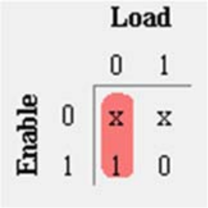
Truth Table / KMAPS:

Parallel Loading:

Input		Output	
Enable	Load	Preset	Clear
0	0	x	x
0	1	x	x
1	0	0	1
1	1	1	0



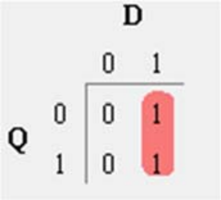
		Load	
		0	1
Enable	0	x	x
	1	0	1



		Load	
		0	1
Enable	0	x	x
	1	1	0

Serial In / D Flip-Flop:

Q	D	Q+
0	0	0
0	1	1
1	0	0
1	1	1



		D	
		0	1
Q	0	0	1
	1	0	1

Boolean Expressions:

Preset = Load

Clear = (Load)’

Q+ = D

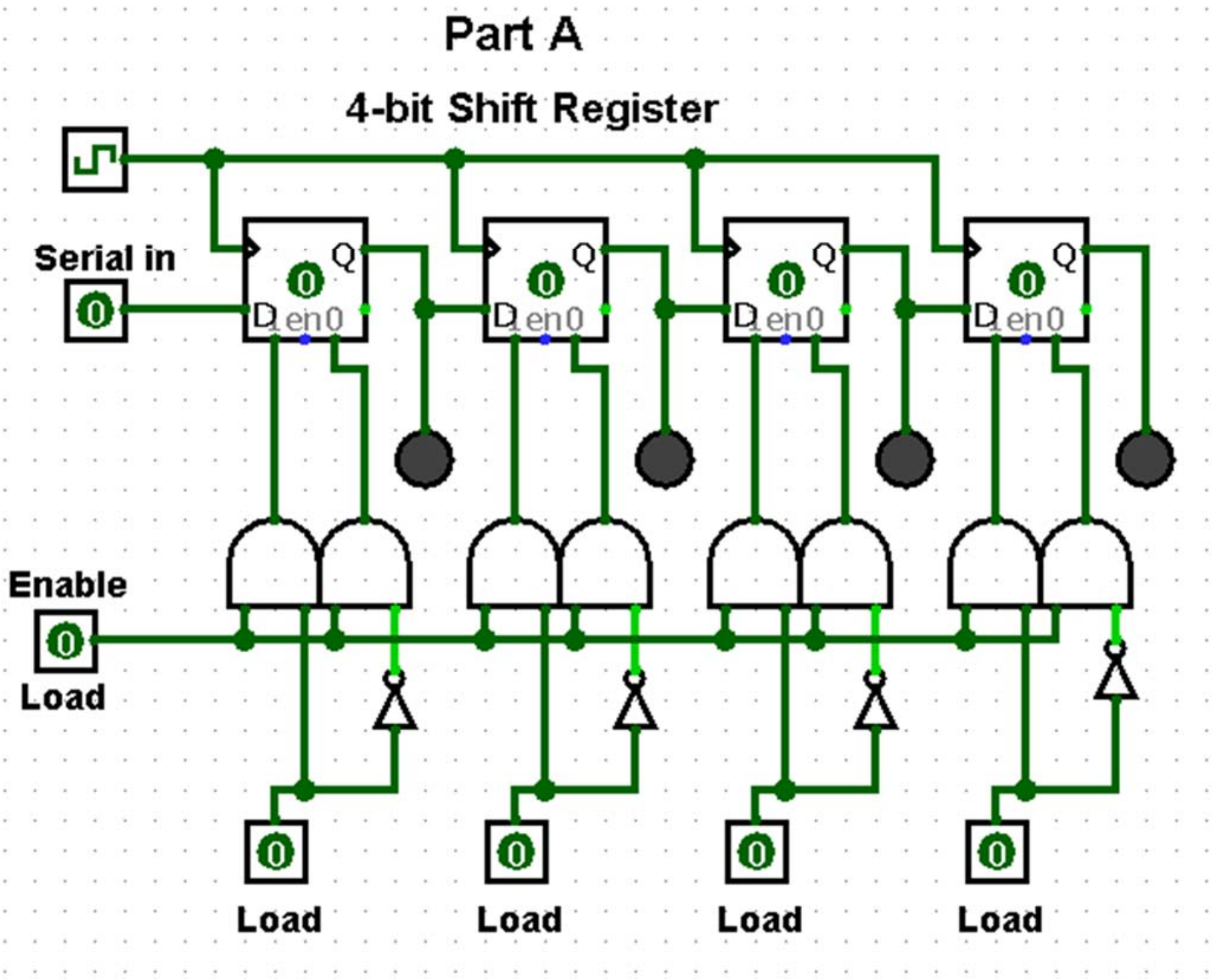
Test Case:

Load = 1, D = 0, Enable = 1

*Note: Enable will toggle between allowing Serial In (0) or Parallel Load (1) to be loaded.

Preset = 1
Clear = 0
Q+ = 1

Diagram

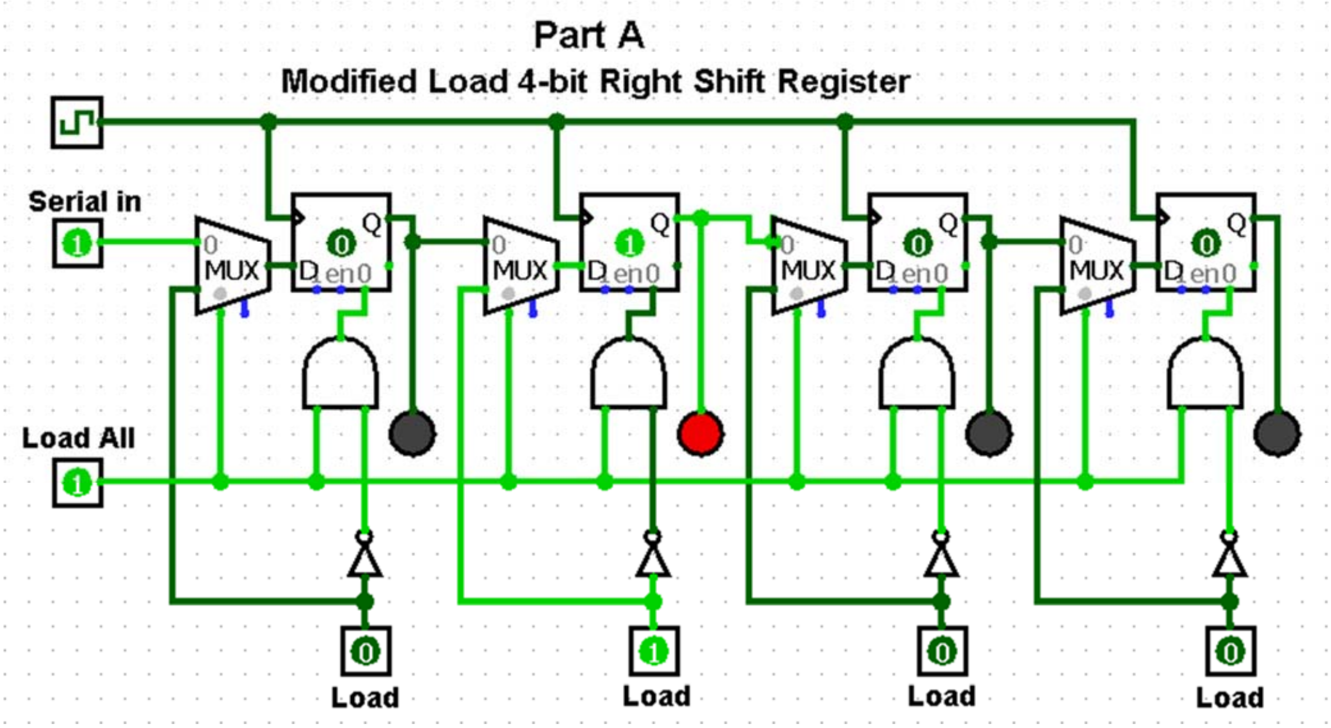


Questions:

1. How would you modify this design to provide synchronous parallel load instead of asynchronous parallel load?

Synchronous parallel load can be achieved by sending the Parallel Load inputs directly through the clock controlled D Flip-Flop. This however, needs to be somehow controlled as we have another Serial Input competing for that DFF input. This has been accomplished in our design with the use of a 2:1 MUX. The Enable toggle is sent to control the control bit on each MUX while the parallel load and serial input are used as the inputs on the MUX. The result is then fed to each DFF allowing for synchronous load.

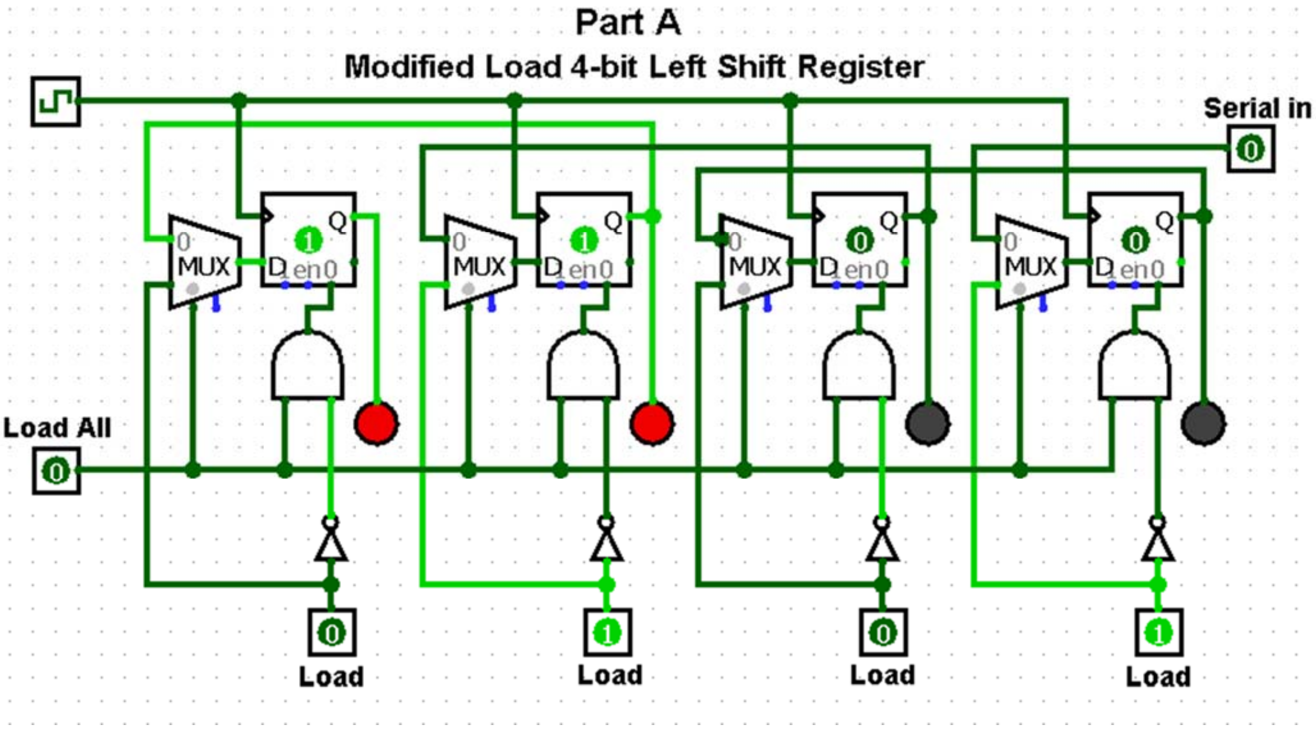
Diagram:



2. How would you modify this design to allow a synchronous right-shift along with synchronous parallel load?

The right shifting was part of our original design so I will take this question and implement the opposite of our current design. To allow synchronous *left* shifting with parallel load all that needs to be done is the reorganizing of the serial input and direction of the Q output. Serial input will be fed to the rightmost DFF and the resulting Q output is fed directly to the D input of the register directly to its left. To satisfy this question with right shifting please look at the design implemented for the previous question.

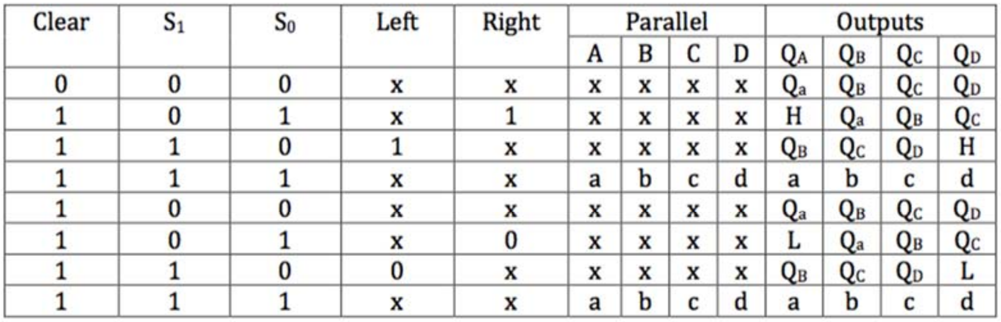
Diagram:



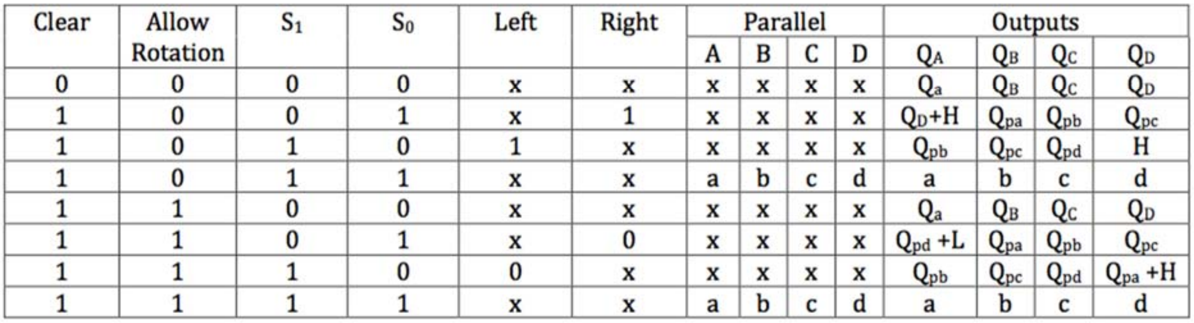
Part B:

In part B the implementation of the IC 74LS194 was required. This implementation tests our ability to understand the data sheet and implement the IC with its proper usage. The chip comes with selector bits that allow left or right shifting. Circular shifting can be easily implemented with the use of an AND gate tied to the last output bit and an enabling control bit that either allows or prevents circular shifting. The design shown in the diagram below is not representative of the current IC design. The simulation only allows for one way shifting. Actual implementation of the IC requires use of the selector control bits for directional shifting and has been demonstrated in a lab environment.

Truth Table

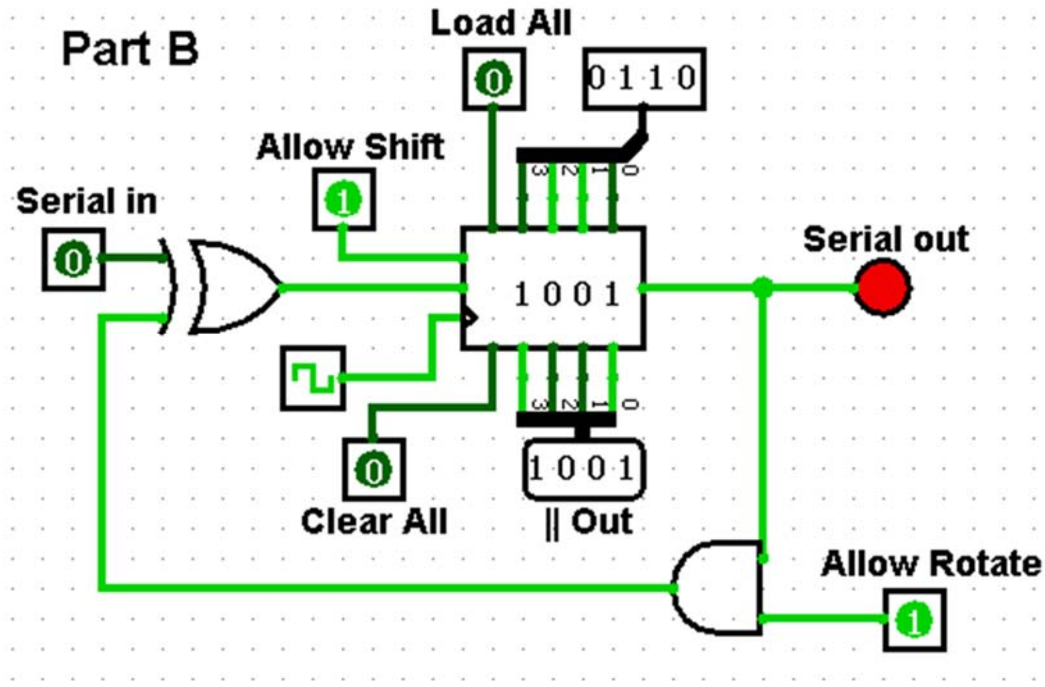


Clear	S_1	S_0	Left	Right	Parallel				Outputs			
					A	B	C	D	Q_A	Q_B	Q_C	Q_D
0	0	0	x	x	x	x	x	x	Q_a	Q_B	Q_C	Q_D
1	0	1	x	1	x	x	x	x	H	Q_a	Q_B	Q_C
1	1	0	1	x	x	x	x	x	Q_B	Q_C	Q_D	H
1	1	1	x	x	a	b	c	d	a	b	c	d
1	0	0	x	x	x	x	x	x	Q_a	Q_B	Q_C	Q_D
1	0	1	x	0	x	x	x	x	L	Q_a	Q_B	Q_C
1	1	0	0	x	x	x	x	x	Q_B	Q_C	Q_D	L
1	1	1	x	x	a	b	c	d	a	b	c	d



Clear	Allow Rotation	S_1	S_0	Left	Right	Parallel				Outputs			
						A	B	C	D	Q_A	Q_B	Q_C	Q_D
0	0	0	0	x	x	x	x	x	x	Q_a	Q_B	Q_C	Q_D
1	0	0	1	x	1	x	x	x	x	Q_D+H	Q_{pa}	Q_{pb}	Q_{pc}
1	0	1	0	1	x	x	x	x	x	Q_{pb}	Q_{pc}	Q_{pd}	H
1	0	1	1	x	x	a	b	c	d	a	b	c	d
1	1	0	0	x	x	x	x	x	x	Q_a	Q_B	Q_C	Q_D
1	1	0	1	x	0	x	x	x	x	Q_{pd} +L	Q_{pa}	Q_{pb}	Q_{pc}
1	1	1	0	0	x	x	x	x	x	Q_{pb}	Q_{pc}	Q_{pd}	Q_{pa} +H
1	1	1	1	x	x	a	b	c	d	a	b	c	d

Diagram



Test Cases:

S1 = 0, S0 = 1, right = 0 , allow rotation = 0, → Q_a = 0, Q_b = Q_{pa} ,Q_c = Q_{pb} , Q_d = Q_{pc}

S1 = 1, S0 = 1, allow rotation = 0, a = 1, b = 0, c = 1, d = 0 → Q_a = 1, Q_b = 0 ,Q_c = 1 , Q_d = 0

S1 = 1, S0 = 0, left = 1 , allow rotation = 1, → Q_a = Q_{pb} , Q_b = Q_{pc} ,Q_c = Q_{pd} , Q_d = 1

Discussion & Conclusion:

The designs implemented in Part A of this lab have shown how sequential circuits can be used with a clock to allow for storage of data in a sequential manner. This time property allows for the idea of “shifting” data in the direction one chooses. The ability to control such shifting or insertion of data at any point the user requests is extremely useful logic. The basis of multiplication and division, as previously discussed, in a binary number system involves such shifting. Introducing the ability to control that flow in either direction or parallel loading of information allows a great deal of versatility. These observations allowed us to gain a better insight on how shift registers work. The design implementation of Part A gave the necessary insight on how IC 74LS194 functions. Some simple tweaks to Part A will allow the use of circular shifting and even directional specification. The logic for sequential and asynchronous data flow can be implemented whether the user has access to such pre-built IC’s or has to create one himself.