Homework #3 Computer Organization Due: February 22, 2008 (F)

1) In class we consider implementing a 32-bit adder by "rippling" together smaller adders, we got the following gate delays:

Type of Adder	Sum-Of-Products (SOP) Boolean formula for the $Carry_{out}$	# AND gates used	Gate Delay per Adder	Number of Adders Needed	Total Delays for a 32-bit Adder
1-bit	$c_{i+1} = x_i\ y_i + x_i\ c_i + y_i\ c_i$	3	2	32	64
2-bit	$c_{i+2} = x_{i+1}\ y_{i+1} + x_{i+1}\ (x_i\ y_i + x_i\ c_i + y_i\ c_i) + y_{i+1}\ (x_i\ y_i + x_i\ c_i + y_i\ c_i) = x_{i+1}\ y_{i+1} + x_{i+1}\ x_i\ y_i + x_{i+1}\ x_i\ c_i + x_{i+1}\ y_i\ c_i + y_{i+1}\ x_i\ y_i + y_{i+1}\ x_i\ c_i + y_{i+1}\ y_i\ c_i$	7	2	16	32
3-bit	$c_{i+3} = x_{i+2}\ y_{i+2} + x_{i+2}$ (the c_{i+2} SOP formula above with 3 products) + y_{i+2} (the c_{i+2} SOP formula above with 3 products) = $x_{i+2}\ y_{i+2} + x_{i+2}\ x_{i+1}\ y_{i+1} + x_{i+2}\ x_{i+1}\ x_i\ y_i + x_{i+2}\ x_{i+1}\ x_i\ c_i + x_{i+2}\ x_{i+1}\ y_i\ c_i + x_{i+2}\ y_{i+1}\ x_i\ y_i + x_{i+2}\ y_{i+1}\ x_i\ c_i + x_{i+2}\ y_{i+1}\ y_i\ c_i + y_{i+2}\ x_{i+1}\ y_{i+1} + y_{i+2}\ x_{i+1}\ x_i\ y_i + y_{i+2}\ x_{i+1}\ x_i\ c_i + y_{i+2}\ x_{i+1}\ y_i\ c_i + y_{i+2}\ y_{i+1}\ x_i\ y_i + y_{i+2}\ y_{i+1}\ x_i\ c_i + y_{i+2}\ y_{i+1}\ y_i\ c_i$	15	3	10 (and a two-bit adder)	32
4-bit	$c_{i+4} = x_{i+3}\ y_{i+3} + x_{i+3}$ (the c_{i+3} SOP formula above with 15 products) + y_{i+3} (the c_{i+3} SOP formula above with 15 products) = $x_{i+2}\ y_{i+2} + x_{i+3}\ x_{i+2}\ y_{i+2} + x_{i+3}\ x_{i+2}\ x_{i+1}\ y_{i+1} + \ldots + y_{i+3}\ y_{i+2}\ y_{i+1}\ y_i\ c_i$	31	3	8	24

Explain why the gate delay of a 3-bit adder is 3 while the gate delay for a 2-bit adder is only 2.

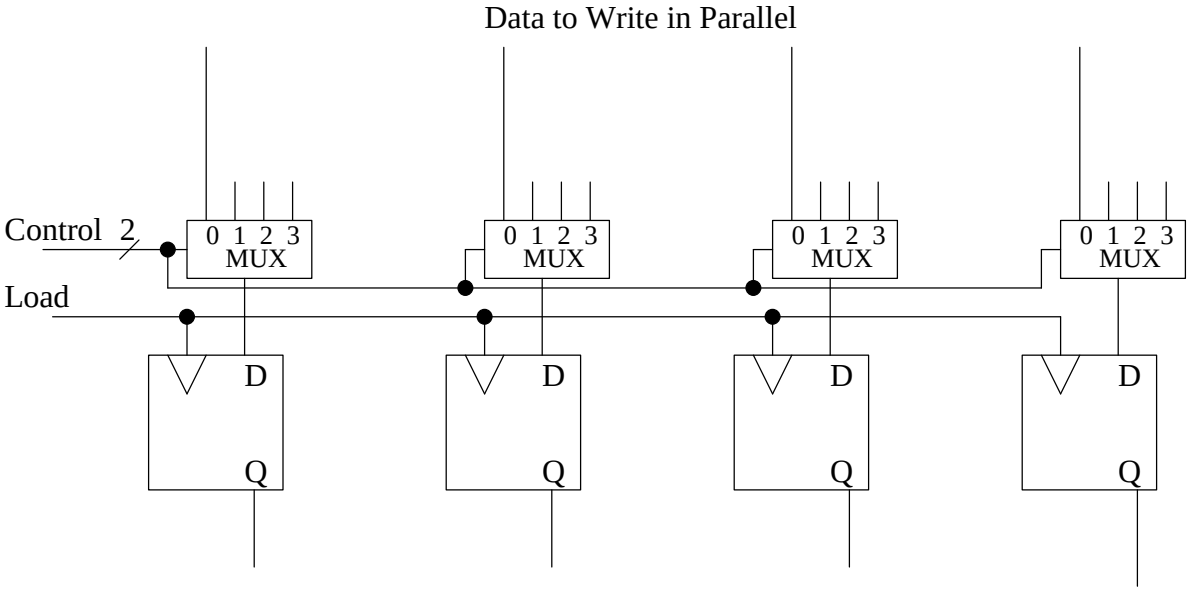
2) Extend the above table by completing the following table (remember the 9-input limit on both OR and AND gates)

Type of Adder	# AND gates used	Most # of inputs into any of the AND gates	# gate delays due to AND gates	# gate delays due to OR gates	Gate Delay per Adder	Number of Adders Needed	Total Delays for a 32-bit Adder
4-bit	31	5	1	2	3	8	24
5-bit						6 5-bit & a 2-bit	
6-bit						5 6-bit & a 2-bit	
7-bit						4 7-bit & a 4-bit	
8-bit						4	
9-bit						3 9-bit & 5-bit	

3) Complete the below diagram of a 4-bit shift register so that it is able to perform the following operations:

- parallel read/output of all bits (just look at the Q values)
- parallel write/input of all bits
- logical shift right one bit position (value shifted out of least-significant bit is lost and a "0" is shifted into the most-significant bit)
- arithmetic shift right (sign-extend the most-significant bit when shifting)
- circular shift left one bit position (value shifted out of most-significant bit is shifted around into the least-significant bit)

Note: For each D-flip flop, the output of a MUX is used as the D-input.



4) Using the discussion in the book and the register file handout (http://www.cs.uni.edu/~fienup/cs041s08/lectures/lec13_reg_file.pdf), draw a complete (not just a one-bit slice) register file that has:

- 4 registers
- **3-bits per register**
- one write-port
- two read-ports

You can draw block-diagrams for flip-flops, decoders, and MUXs without showing their gate implementations, but you should show all the flip-flops, decoders, MUXs, and connecting wires.

5) How well does this register-file design scale? Suppose that we are implementing a 16 M x 8 (16M registers, each with 8 bits) register file with one write-port and one read-port.

a) How many and what type of decoder(s) would be needed?

b) How many total gates (assume 9-input limit on AND & OR gates) would be needed to implement this (these) decoder(s)?

c) How many and what type of MUX(s) would be needed?

d) How many total gates (assume 9-input limit on AND & OR gates) would be needed to implement this (these) MUX(s)?

e) Assuming D flip-flops to store each bit (5 gates/flip-flop). What % of the total gates is used to implement the D flip-flops?

6) Redo the previous question using the 16 M x 8 square-memory implementation similar to the "Implementation of Large Memory Chips" class handout.