Problem 1 **(20 points):** The structure of an unsigned parallel multiplier is based on the observation that partial products in the multiplication process can be computed in parallel. For example we can consider the following unsigned binary integers:

$$X = \sum_{i=0}^{m-1} x_i 2^i \quad ; MULTIPLICAND$$
$$Y = \sum_{j=0}^{n-1} y_j 2^i \quad ; MULTIPLIER$$

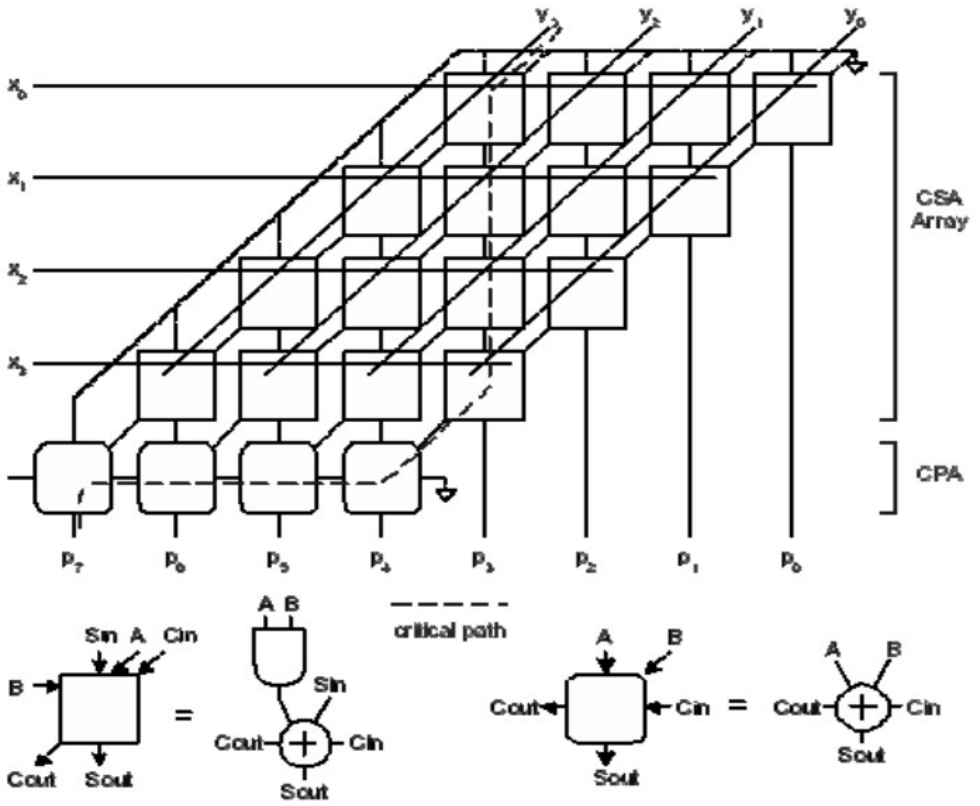
This product is found by:

$$P_r = X_r Y_r = (\sum_{i=0}^{m-1} x_i 2^i)(\sum_{j=0}^{n-1} y_j 2^j) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (x_i y_j) 2^{i+j} = \sum_{k=0}^{m+n-1} P_k 2^k$$

For a 4 by 4 multiplier the expression can be expanded as follows:

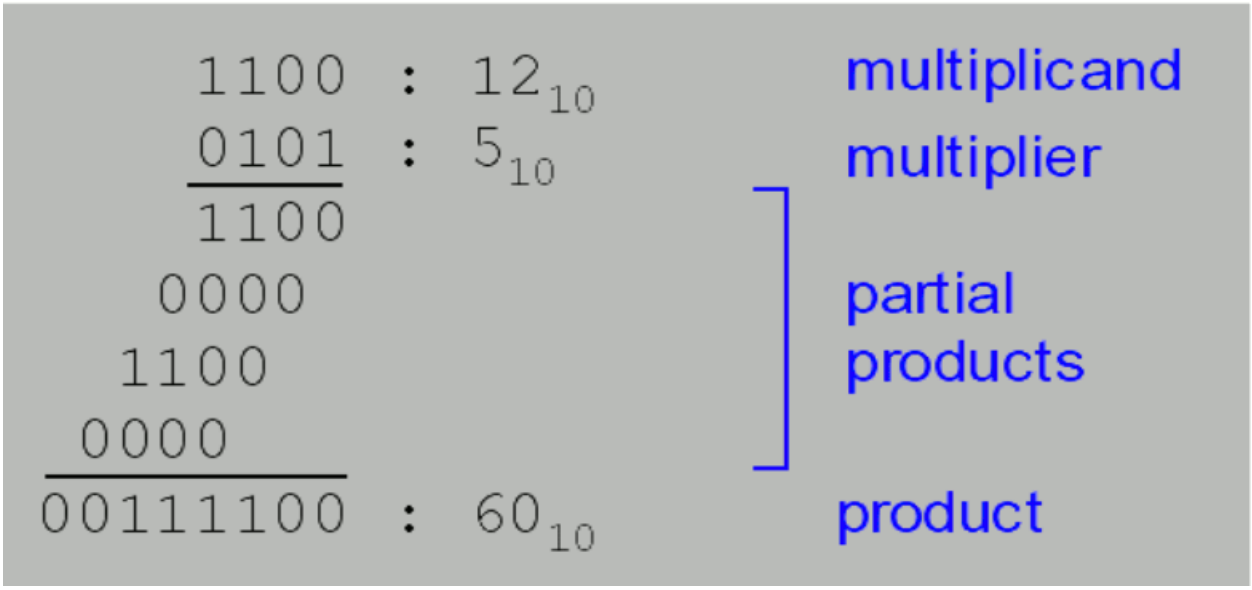
				X3 Y3	X2 Y2	X1 Y1	X0 Y0	MULTIPLICAND MULTIPLIER
				X3Y0	X2Y0	X1Y0	X0Y0	
			X3Y1	X2Y1	X1Y1	X0Y1		
		X3Y2	X2Y2	X1Y2	X0Y2			
	X3Y3	X2Y3	X1Y3	X0Y3				
P7	P6	P5	P4	P3	P2	P1	P0	PRODUCT

The basic structure is presented in the diagram below:

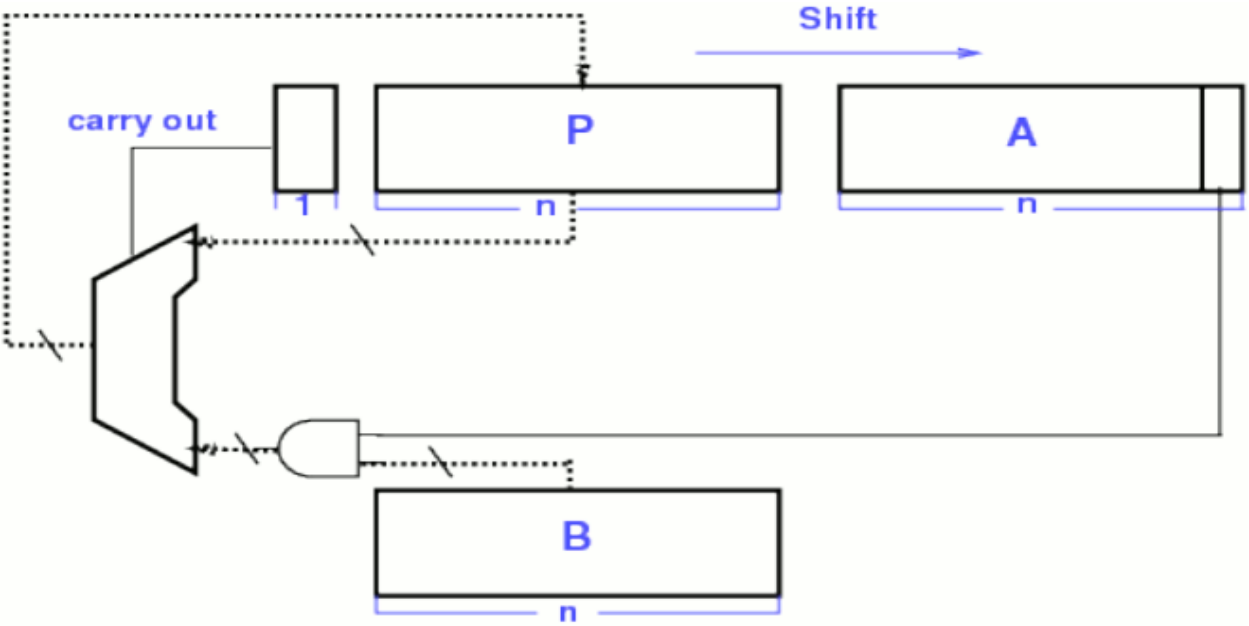


a) Write a verilog model to describe this basic structure.
b) Use Modelsim to compile your multiplier model.
c) Use Modelsim to simulate your multiplier with the following cases: (2*4, 15*3)

Problem 2(20 points): An N by N-bits multiplication produces N m-bits partial products. The sum of N partial products produces an (M+N)-bits product shown in the example below:



This can be implemented using the shift add structure shown below.



In this design, P, A, and B are n bits registers. Carry-out is a one bit flip-flop. Initially, P and carry out are set to 0. {carryout, P, A} is a shift register which performs one shift every clock cycle. Every bit of B is ANDED with the lowest bit of A to form and new set of n-bits. The new n-bits are added the P and the result are loaded into P and the carry-out is loaded into the flip-flop in the next clock cycle. At the end of n-clock cycles, the results of the multiplication is stored in {P,A} register.

a) Write a verilog model that implement the multiplier.
b) Use Modelsim to compile your verilog model.
c) Use Modelsim to simulate your multiplier with the following cases: (2*4, 15*3)

Problem 3(20 points): Multiplication of signed operands, which generate a double-length product in the 2's-complement number system. The general strategy is the accumulated partial products by adding versions of the multiplicand as selected by the multiplier bits.

First, consider the case of a positive multiplier and a negative multiplicand, when we add a negative multiplicand to a partial product, we must extend the sign-bit value of the multiplicand to the left as far as the product will extend (as shown in the figure below). The sign extension of the multiplicand is hand-written.

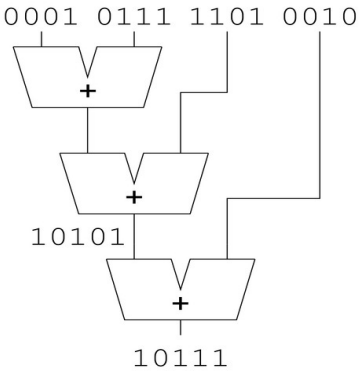
					1 0	0 1	0 0	1 1	1 1	MULTIPLICAND (-13) MULTIPLIER (+11)
	1 **1** **0** **1** 0	**1** **1** **0** 1 0	**1** **1** 0 0 0	**1** 1 0 0 0	1 0 0 1 0	0 0 0 1	0 1 0	1 1	1	
1	1	0	1	1	1	0	0	0	1	PRODUCT

For a negative multiplier, a straightforward solution is to form the 2's complement of both multiplier and multiplicand and proceed as in the first case.

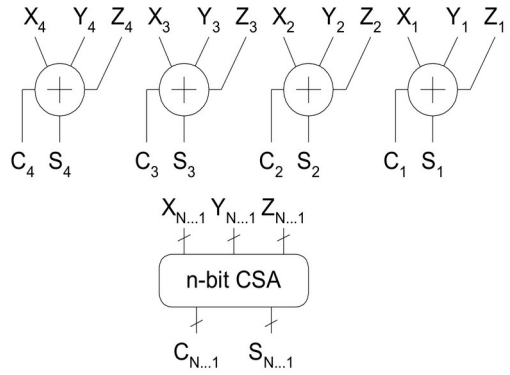
a) Write a verilog file to model the above behavior for 5 bits multiplier and 5 bits multiplicand.
b) Use Modelsim to compile your verilog file.
c) Use Modelsim to simulate your multiplier with the following cases: (-10*4, 11*-3, -10*-11)

Problem 4(20 points): To add k N-bit words, you need k-1 N-bit adder. For example to add 0001 + 0111 + 1101 + 0010 = 10111 you need a structure similar the one shown below (a). Classical full adder sums 3 inputs to produce 2 outputs Carry (C) and Sum (S) where the C has twice the weight of the sum output. If you design N full adder in parallel shown in (b), then this produces N Sums and N Carry outs. This is called Carry Save Adder (CSA).

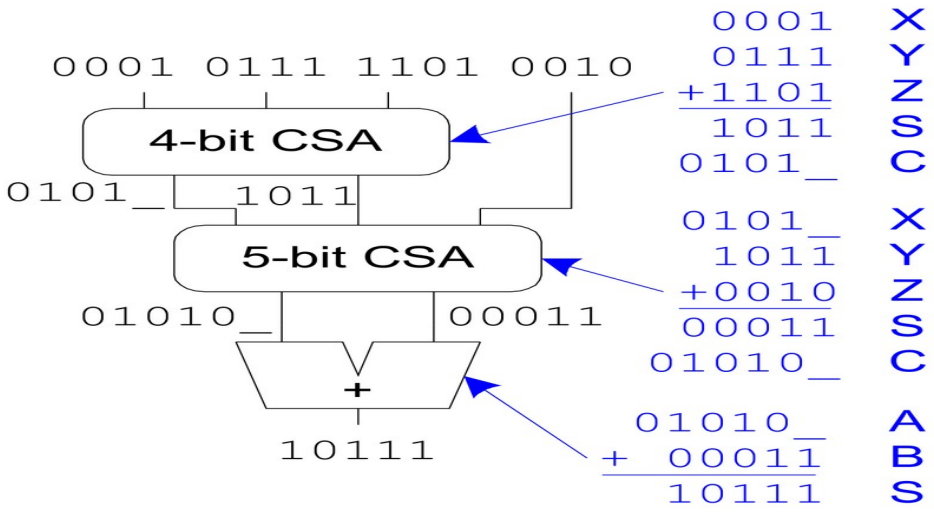
To add N-bit words, a 2-stage of CSA followed by a regular adder as shown. In this structure, the carry bits are logically shifted by one bit after each stage to reflect the weight of the carry. The size of the CSA is increased by a single bit at the next stage. At the last stage, the carry propagation is performed.



(a)



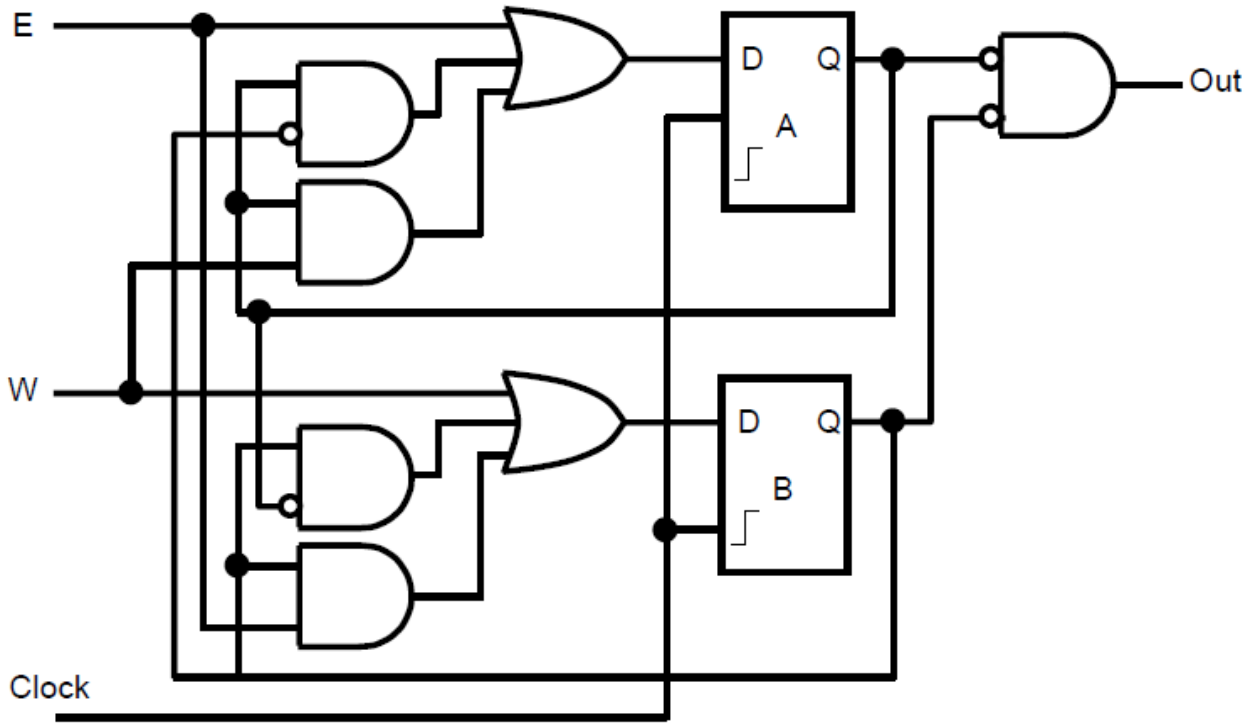
(b)



(c)

a) Design a CSA to perform add a sequence of 10 8-bit binary numbers. How many CSA stages are needed (explain). Write a Verilog model for your design.
b) Compile you Verilog using ModelSim.
c) Use Modelsim to simulate your design with the following sequence: (1,2,3,4,5,6,7,8,9,10) and (3,4,5,6,7,8,9,10).

Problem 5(20 points): Analyze the logic circuit given below.



a) Derive a state diagram for the circuit.
b) Write a structral Verilog Model for this design and simulate your circuit to verify correctness. (Explain your approach)
c) Write a behavioral Verilog model based on the derived state-diagram and simulate to verify correctness.
d) Using ModelSim, can you verify that the model in a) and c) are equivalent (Explain).