EECS 318 ***Due: Oct 26***

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:

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

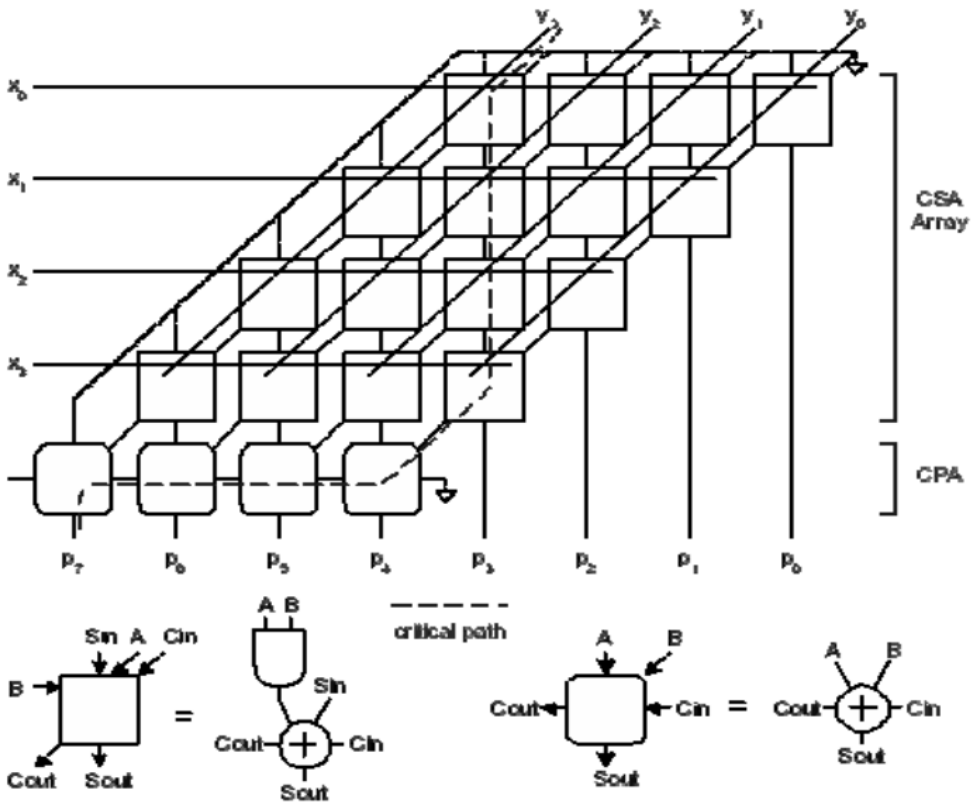
This product is found by:

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

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

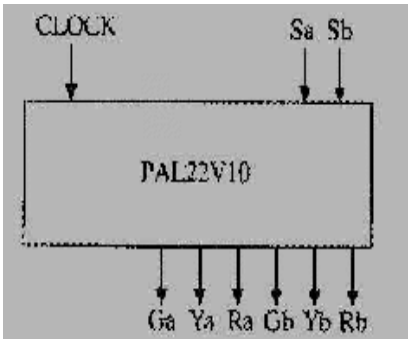
The basic structure is presented in the diagram below:



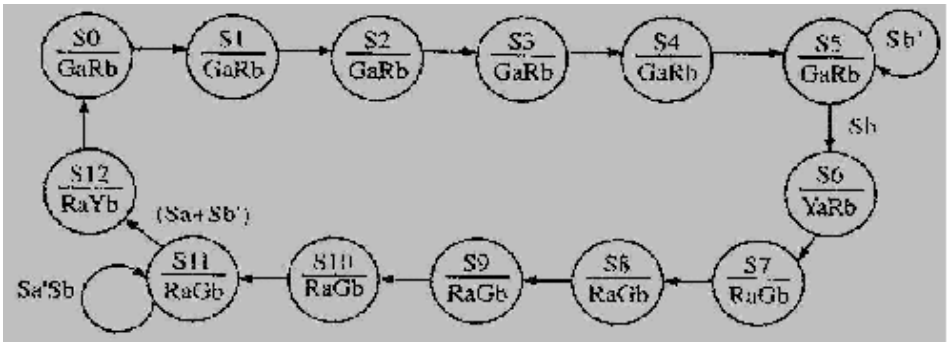
a) Write a VHDL model to describe this basic structure.
b) Use Modelsim to compile your multiplier model.
c) Use Modelsim to simulate your multiplier with the following cases: (2*4, 15*3)

EECS 318 ***Due: Oct 26***

Problem 2(30points): The objective of this lab is to design a sequential traffic-light controller for the intersection of "A" street and "B" street. Each street has traffic sensors, which detect the presence of vehicles approaching or stopped at the intersection. Sa = 1 means a vehicle is approaching on "A" street, and Sb = 1 means a vehicle is approaching on "B" street. "A" street is a main street and has a green light until a car approaches on "B". Then the light changes, and "B" has a green light. At the end of 50 seconds, the lights change back unless there is a car on "B" street and none on "A", in which case the "B" cycle is extended 10 more seconds. When "A" is green, it remains green at least 60 seconds, and then the lights change only when a car approaches on "B". The figure below shows the external connections to the controller. Three of the outputs (Ga, Ya, and Ra) drive the green, yellow, and red lights on "A" street. The other three (Gb, Yb, and Rb) drive the corresponding lights on "B" street.



(a)



(b)

The figure shows a Moore state graph for the controller. For timing purposes, the sequential network is driven by a clock with a 10-second period. Thus, a state change can occur at most once every 10 seconds. The following notation is used: GaRb in a state means that Ga = Rb = 1 and all the other output variables are 0. Sa'Sb on an arc implies that Sa = 0 and Sb = 1 will cause a transition along that arc. An arc without a label implies that a state transition will occur when the clock occurs, independent of the input variables. Thus, the green "A" light will stay on for 6 clock cycles (60 seconds) and then change to yellow if a car is waiting on "B" street.

Write a VHDL model for the traffic light controller. Simulate the file in ModelSim to verify correctness.

Problem 3 (30 points): You are asked to implement a processor with ten instructions**: load, store, add, complement, xor, shift, rotate, nop, halt, and branch**. The instruction set and format for this processor are shown below.

1. Instructions: **ADD R1,R2** means R1 = R1 + R2 and **CMP R1, R2** means R1 = ~(R2)

Name	Mnemonic	Opcode	format(Instdst,Src)	
NOP	NOP	0	NOP	
LOAD	LD	1	LD reg, mem1	set PSR,PSR[0]=0
STORE	STR	2	STR mem, src	clear
BRANCH	BRA	3	BRA mem, CC	
XOR	XOR	4	XOR reg, src	set PSR,PSR[0]=0
ADD	ADD	5	ADD reg, src	set PSR
ROTATE	ROT	6	ROT reg, cnt	set PSR
SHIFT	SHF	7	SHF reg, cnt	set PSR
HALT	HLT	8	HLT	
COMPLEMEN	CMP	9	CMP reg, src	set PSR,PSR[0]=0

2. Condition Code (CC)

Name	Means	code
A	Always	0
P	Parity	1
E	Even	2
C	carry	3
N	Negative	4
Z	Zero	5
NC	No carry	6
PO	Positive	7

4. Operand Addressing

Mem	Memory address
mcm1	Memory address or immediate value
Reg	Any register index
Src	Any register index, or immediate value
CC	condition code
Cnt	Shift/Rotate cnt, cnt > 0 means right. < 0 means left.+/-16

6. Instruction Register (IR) Format

IR[31:28]	Opcode
IR[27:24]	CC
IR[27]	source type O=reg(mem),1=imm
IR[26]	destination type O=reg, 1=imm
IR[23:12]	Source address
IR[23:12]	shift/rotate count
IR[11:0]	destination address

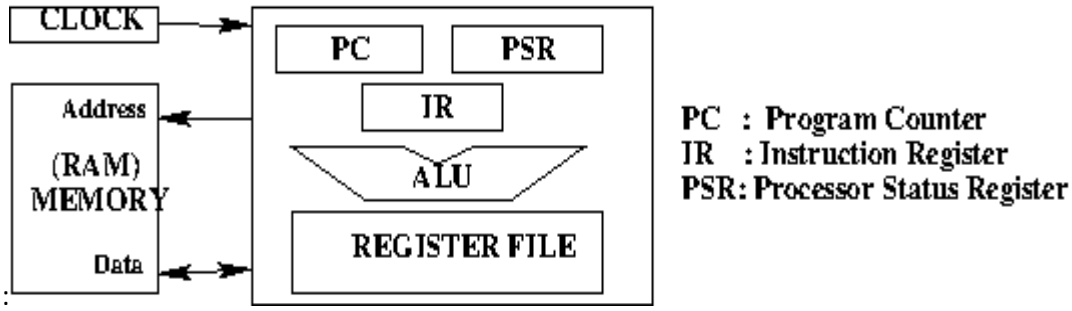
7. Processor Status Register (PSR)

PSR[0]	Carry
PSR[1]	Parity
PSR[2]	Even
PSR[3]	Negative
PSR[4]	Zero

Implementation of the corresponding hardware to execute these instructions is not the issue. For example, an add instruction may simply be modeled as:

```
assign {carry,sum} = A + B;
```

Without going into detail of whether the addition be performed using a ripple-carry adder or a carry-look-ahead adder. Similarly, we treat memory as a large set of registers directly visible to the processor. The processor structure is shown in the figure below



Write a Verilog model for this processor (Assume that the width of the data-path is 32 bits, the size of the address field is 12 bits, and the register file contains 16 registers).

To verify your model write a program to read a positive number 'N' stored at memory location zero and compute the two's complement representation of '-N'. The number -N in two's complement is to be stored in memory location one. Test your code for N=6.

Problem 4 (10points): To verify your model write a program to count the number of 1's in memory location zero. The number of ones is to be stored in memory location one.

Problem 5 (10points): Write a code to multiply two 4 bit numbers A and B and store the results in C. A, B, and C are stored in memory location 0, 1 and 2 respectively.