EECS 318 ***Due: Oct 22***

Problem 1 (25 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:

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

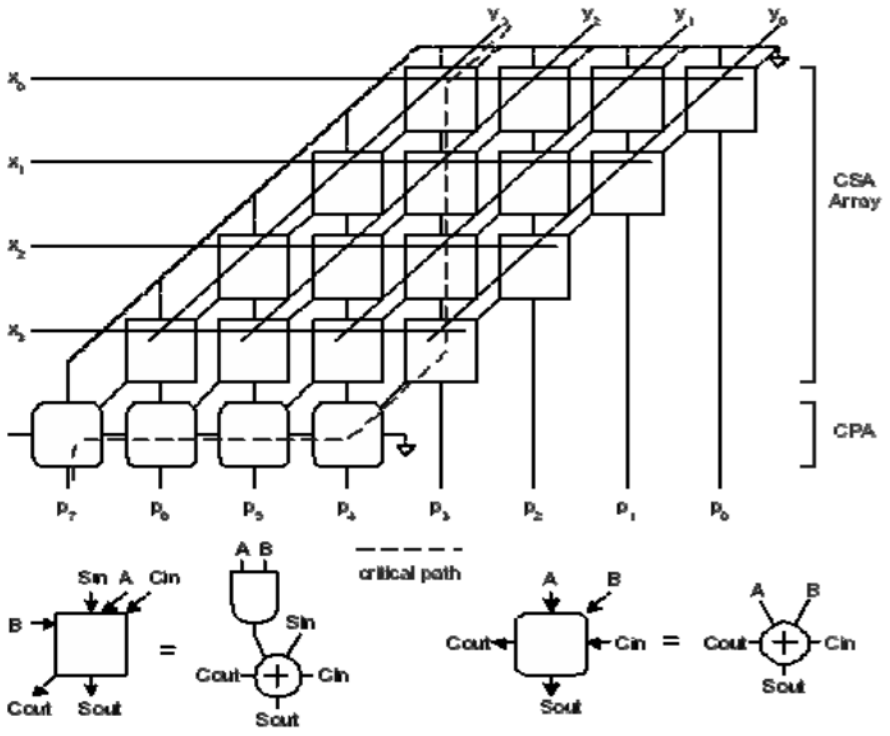
This product is found by:

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

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

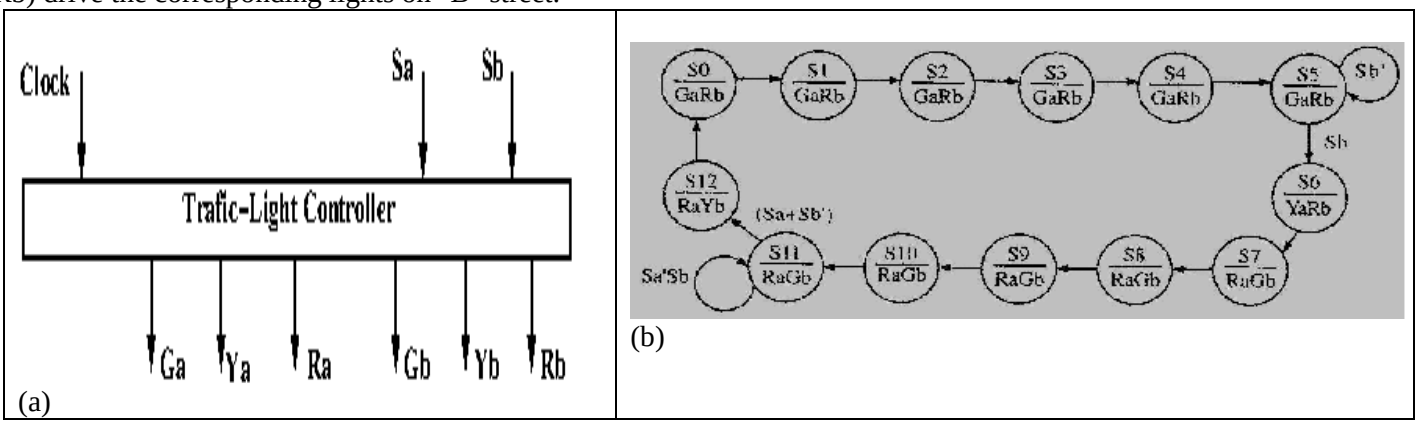
				X3	X2	X1	X0	MULTIPLICAND
				Y3	Y2	Y1	Y0	MULTIPLIER
				X3Y0	X2Y0	X1Y0	X0Y0	
			X3Y1	X2Y1	X1Y1	X0Y1		
		X3Y2	X2Y2	X1Y2	X0Y2			
	X3Y3	X2Y3	X1Y3	X0Y3				
P7	P6	P5	P4	P3	P2	P1	P0	PRODUCT

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: (4*4, 5*12)

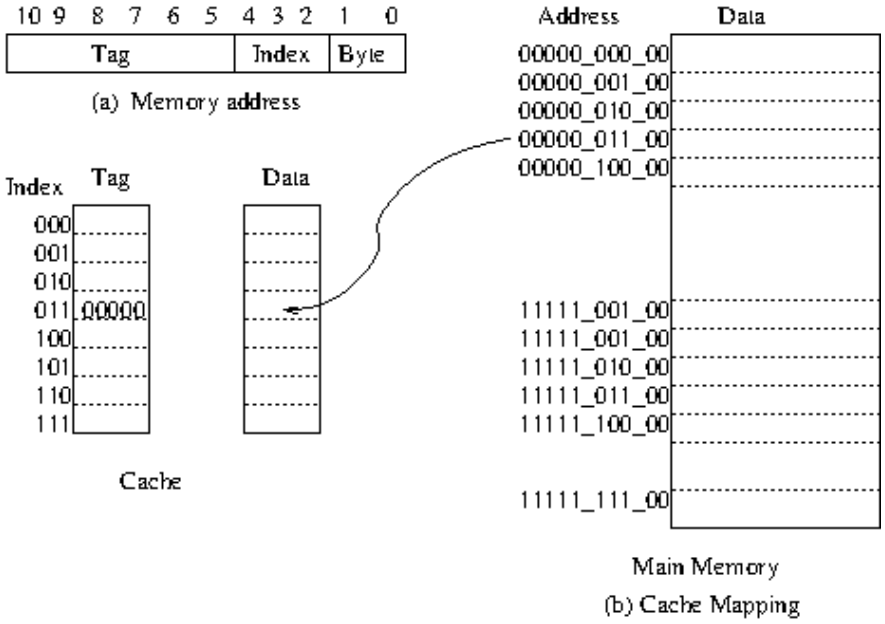
Problem 2(25points): The objective of this lab is to design a sequential traffic-light controller for the intersection of "A" street and "B" street. Each street has traffic sensors, which detect the presence of vehicles approaching or stopped at the intersection. Sa = 1 means a vehicle is approaching 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.



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.

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

Problem 3(30 points): To illustrate the concept of cache memory, we assume a very small cache of eight 32-bit words and a small main memory with 1 KB (256 words), as shown in **Figure P3a**. Both of these are too small to be realistic, but their size makes illustration of the concepts easier. The cache address contains 3 bits, the memory address 10. Out of the 256 words in main memory, only 8 at a time may lie in the cache.



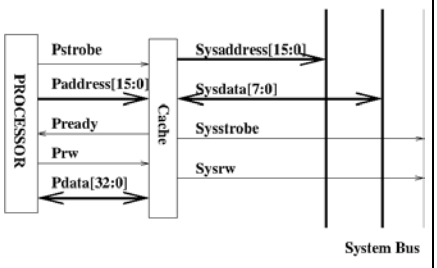
P3a

In order for the CPU to address a word in the cache, there must be information in the cache to identify the address of the word in main memory. If we consider the example of a loop, clearly, we find it desirable to contain the entire loop within the cache, so that all of the instructions can be fetched from the cache while the program is executing most of the passes through the loop. The instructions in the loop lie in consecutive word addresses. Thus, it is desirable for the cache to have words from consecutive addresses in main memory present simultaneously. A simple way to facilitate this feature is to make bits 2 through 4 of the main memory address be the cache address. We refer to these bits as the index, as shown in Figure P3a. Note that the data from address 0000001100 in main memory must be stored in cache address 011. The upper 5 bits of the main memory address, called the tag, are stored in the cache along with the data. Continuing the example, we find that for main memory address 0000001100, the tag is 00000. The tag combined with the index (or cache address) and byte field identify an address in main memory.

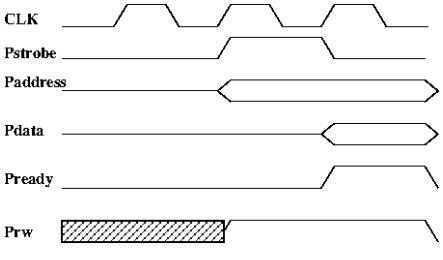
Suppose that the CPU is to fetch an instruction from location 000001100 in main memory. This instruction may actually come from either the cache or main memory. The cache separates the tag 00000 from the cache address 011, internally fetches the tag and the stored word from location 011 in the cache memory, and compares the tag fetched with the tag portion of the address from the CPU. If the tag fetched is 00000, then the tags match, and the stored word fetched from cache memory is the desired instruction. Thus, the cache control places this word on the bus to the CPU, completing the fetch operation. This case in which the memory word is fetched from cache is called a cache hit. If the tag fetched from cache memory is not 00000, then there is a tag mismatch, and the cache control notifies main memory that it must provide the memory word, which is not available in the cache. This situation is called a cache miss. For a cache to be effective, the slower fetches from main memory must be avoided as much as possible, making considerably more cache hits than cache misses necessary.

When a cache miss occurs on a fetch, the word from main memory is not placed just on the bus for the CPU. The cache also captures the word and its tag and stores them for future access. In our example, the tag 00000 and the word from memory will be written in cache location 011 in anticipation of future accesses to the same memory address.

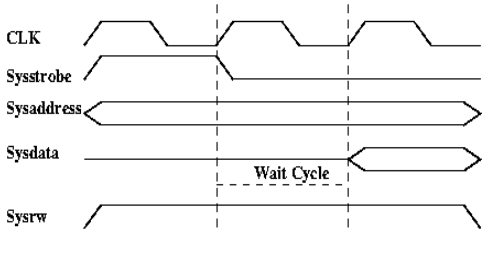
Interfaces: A cache system typically lies between the processor and the main system bus. The following block diagram Figure **P3b** describes the signals and buses that the cache needs to communicate with the processor and the system. Note that in our system the system bus is synonymous with main memory.



P3b



P3c



P3d

Processor Interface: The processor interface consists of the processor address bus, Paddresst[15:0], the processor data bus, Pdata[32:0], and control signals Pstrobe and Pready. The Pstrobe is asserted when the processor is starting a bus transaction and a valid address is on the Paddress bus. Pready is used to signal to the processor that the bus transaction is completed. The timing diagram in **Figure P3c** demonstrates a simple read cycle. The Prw signal is high for a read and low for a write.

System Bus Interface: For our cache model we assume a simple bus model. For a read operation, the Sysaddress is first presented to the bus along with the Sysstrobe signal and the Sysrw. The Sysrw signal is high for read operations and low for write operations. After a set number of wait states (in our case it is four clock cycles), the data is returned. A write operation is similar, but the data is driven onto the Pdata bus immediately and then waits for the set number of wait states (in our case it is four clock cycles) before issuing another write operation. Figure P3d shows a system read with one wait state.

Cache Architecture: The direct-mapped cache is the simplest of all cache architectures. A direct mapped cache consists of a single tag RAM, cache RAM, and a simple controller. You have to model each of these parts separately and then bring them together in the final model.

EECS 318 ***Due: Oct 22***

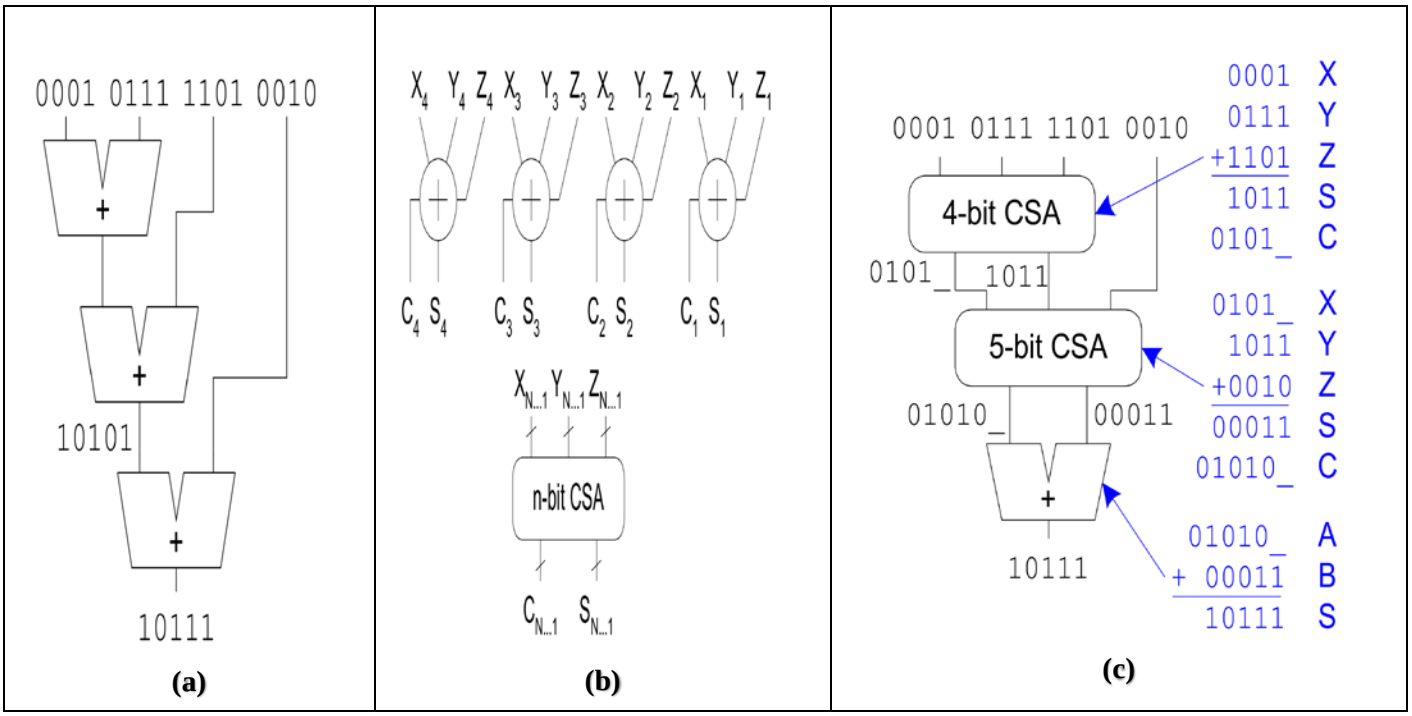
Assume that the processor has a 16-bit address composed of the following fields: The byte field (bits [1:0]), the index field (bits [9:2]), and the tag filed (bits [10:15]).

To simplify the cache, all writes from the processor update both the cache and main memory. This is called write-through mode operation. In write-through mode, the cache memory is always kept coherent with main memory.

Assume the following that a bus should be tristated at the end of a bus transaction, memory write and cache write can be performed in parallel, a signal assignment incurs one unit delay, and the clock period is 100 unit.

a) Write a VHDL model for the above system.
b) To test your system, simulate a processor request of the following sequence of Hex addresses: 75, 65, 75, 59, 75.

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) Design a CSA to perform add a sequence of 10 8-bit binary numbers. How many CSA stages are needed (explain). Write a VHDL model for your design.
b) Use Modelsim to simulate your design with the following sequence: (11,2,13,4,5,6,7,8,9,10) and (3,14,5,6,7,8,19,10).