1. The figure below is a block diagram of an 8 bit pipelined multiplier. This circuit produces the product of two 8 bit numbers after a delay of several clock ticks. The use of registers between the ranks of adders makes it possible to input a new multiplier and multiplicand on every clock tick and get a product out on every clock tick. This kind of pipelined operation is commonly used in high performance systems.
Assuming that we start with no multiplication operations in progress, how many multiplies can be completed in 30 ns, assuming that the clock period is 4 ns? How many multiplies can be completed in 100 ns?

Assuming that the adders are implemented using the look-ahead circuit illustrated below, what is the maximum gate delay we can tolerate and still have the circuit operate correctly with a clock period of 4 ns? Assume that the flip-flop setup time is 1 ns, that the maximum propagation delay is 1.5 ns and that the maximum clock skew is 300 ps.
2. Draw a schematic for a 4 bit grey code counter that has a **load** input, a **count** input and a **reset** input. If **reset** is high, the circuit should reset to 0000, otherwise if **load** is high, it should load a new value, otherwise if **count** is high it should advance by 1, otherwise, it should retain its current value. When counting, your counter should produce the sequence 0000, 0001, 0011, 0010, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000 and should then wrap around to 0000. Show the Karnaugh maps and logic equations used to produce the increment logic.

3. In the simple processor, the controller determines when each component in the system is permitted to use the bus. In other types of systems, there may be several independent subsystems that share a common bus. In such situations a *bus arbiter* is used to determine which subsystem gets to use the bus. In this problem you are to design a bus arbiter that can support three bus *users*. For each user, there is a **request** input and a **grant** output. The arbiter is a sequential circuit, which keeps track of the state of the bus. If the bus is free and one or more of the request lines is high, the arbiter selects one of the users and raises the corresponding grant signal. When the user is done with the bus, it is required to drop its grant signal for at least one clock tick. Design your arbiter so that if more than one user needs to use the bus multiple times, they take turns. Start by producing a state diagram for the arbiter, then design a VHDL module that implements the state diagram. Include an asynchronous reset input.

4. The figure below is a block diagram of a special-purpose processor that can perform any of several operations on the data in its two registers, $A$ and $B$. In particular, if op=000, it loads the $A$ register from the input, if op=001, it loads the $B$ register, if op=010, it sends $A$ to the output, if op=011, it sends $B$ to the output, if op=100, it replaces $A$ with the sum of $A$ and $B$, if op=101 it replaces $B$ with the negation of $B$, if op = 110, it replaces $A$ with the bitwise AND of $A$ and $B$ and if op=111, it replaces $B$ with the bitwise OR of $A$ and $B$. Write VHDL modules to implement the registers, the ALU, the controller and a top level module that combines them using structural VHDL. Assume that the clock period is 10 ns, and that the ALU requires 12 ns to do an AND or OR operation and 22 ns to perform an add or negate operation. Make the registers 16 bits wide.
5. Write a program for the simple processor from section 6 of the notes that checks to see if a given ASCII character string is a palindrome. The inputs to your program are stored at locations 30 and 31 (hex). The value at location 30 is a pointer to the first character in the character string. The value in location 31 is the number of characters in the string. Your program should write 1 in location 32 if the string is a palindrome and 0, if it is not. Try loading your program in the memory for the simple processor and running a simulation that executes your program. Does your program work correctly?

6. Show how to modify the VHDL for the simple processor to implement a new instruction that performs a logical AND operation. Specifically, extend it so that the instruction code B\texttt{xxxx} causes the value in the accumulator to be replaced with the bitwise AND of its original value with the value in location \texttt{xxx}. The VHDL for the simple processor is on the web site.