EE365 – Advanced Digital Circuit Design
Fall 2003
Design Project 3 – 4-bit Binary Counter

Project Overview:

Your task is to implement a 4-bit binary counter in VHDL at the gate level. As part of this task, you will be required to determine the maximum frequency at which you clock a certain set of parts, as well as determine the optimal set of parts needed to reach a certain frequency.

Project Background and Design:

A N-bit counter uses N bits to count from 0 to N-1 utilizing modulo arithmetic. This means that when the counter output reaches N-1, the next output in sequence will be 0 (note that this is the operation for an "up" counter, which is all we will be dealing with in this project; however, "down" counters also exist which count in reverse). The counter uses a clock signal to determine when to change, incrementing to the next value on the rising edge of the clock.

In this project, we will be implementing a 74x163 counter (Wakerly presents this counter starting on p. 696). This counter includes (besides the basic counting and enable function) clear and load functions. The clear function sets all of the outputs to 0 when the CLR_L input is set low. The load function sets QA-QD to be equal to the inputs A-D when the LD_L input is low. Note that the CLR_L input has priority over the LD_L input. The 74x163 also includes two separate enables as well as a ripple carry-out output. The second enable is used to enable or disable the ripply carry-out, which is used to cascade to another counter (for instance, using two 4-bit counters to create an 8-bit counter). We will not be implementing the ripple carry-out in this project, so we only need one enable. A block diagram for this counter (similar to the one on Wakerly p. 697 but with the ripple carry-out and second enable removed) is shown below.

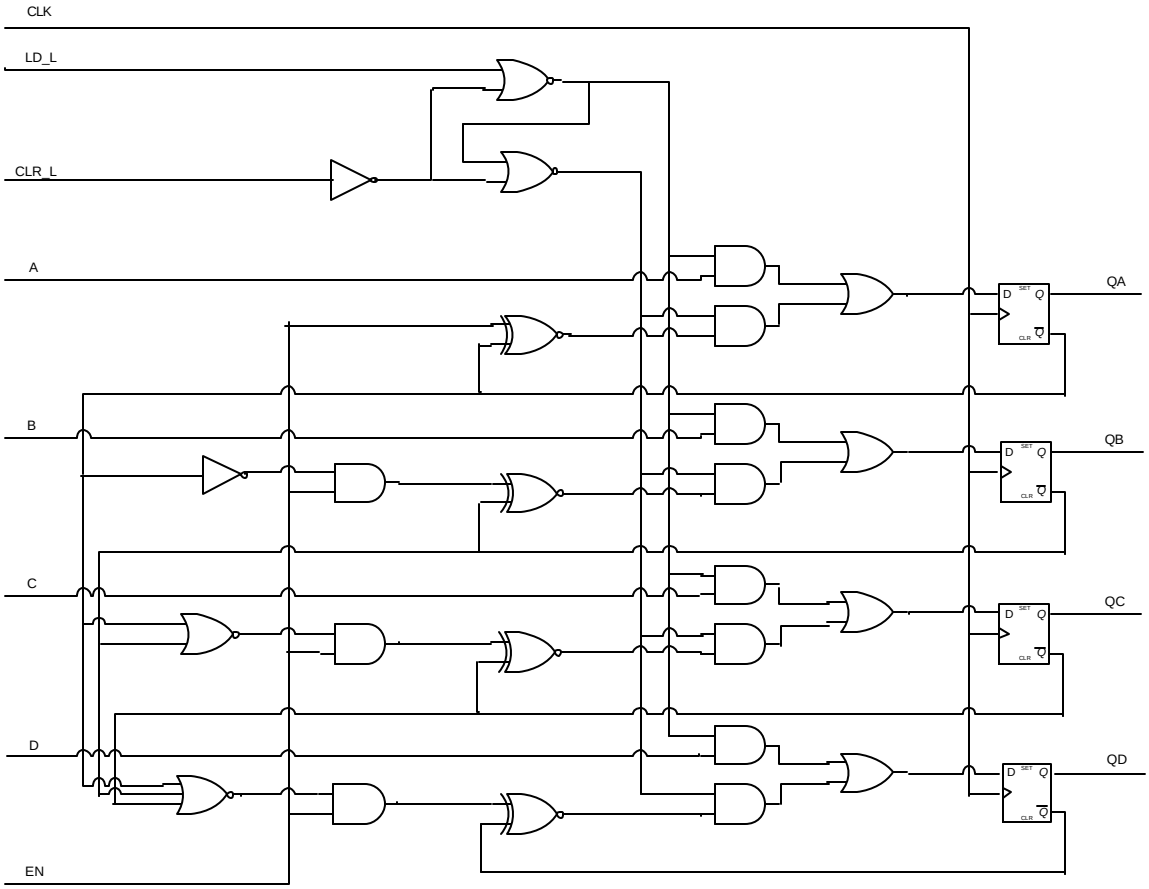

Figure 1: Logic diagram for 74x163-style 4-bit counter (modified from Wakerly p. 697).

Project Specifications:

Your task will be to implement this 74x163 counter in VHDL, and perform a series of timing simulations on this counter. The specific requirements are as follows:

1) The top level module for the counter will be implemented using structural techniques; you should create all of the gates that you need (an inverter, two-input NOR, two-input AND, two-input OR, three-input NOR, two-input XNOR and a D flip-flop) within their own modules and instantiate the gates in your top level.
2) For your first timing simulation, you should implement each part using 74LS family timing. Within this implementation, you should determine the maximum frequency at which you can run your counter ($f_{max\text{-}LS}$). In implementing each gate, please use the separate low-to-high and high-to-low timing for each gate.
3) For your second timing simulation, you should modify your design by changing the part families of the combinatorial logic gates (everything but the D flip-flops) in the design to reduce the minimum clock period of the circuit to the smallest possible. The restriction placed on you is that your circuit cannot cost more than **15 units** (please refer below for further explanation). Please use 74LS74 parts for the D flip-flops for this simulation. In order to achieve a cost minimization, you should determine where your "critical delay paths" are (the longest delay paths in your circuit), and attempt to lessen the delay for those.

Make sure you report the minimum clock period that your circuit can handle.

For the second timing simulation, please use the following data:

74LS family – 0 units/gate
74ALS family – 1 unit/gate
74F family – 2 units/gate
74AS family – 3 units/gate

Note that these "costs" do not correspond to actual monetary costs for these ICs. Please calculate and report your circuit cost for the second timing simulation (you should use a block diagram to show the family used for each gate).

The VHDL module for the DFF is provided on the course website. Note the use of the generic parameters in the entity statement; you will need to assign those with your timing information for the family that you are using when you instantiate the flip-flops in your top level module.

When you turn your project in, your report should include the following:

1) A brief description of the project, including the high-level functionality and interface.
2) A description of your test plan. For this project, it is important to put together a test suite with reasonable coverage in order to test each part of the circuit. In your write-up, you should discuss how your test cases provide this coverage.
3) Timing diagrams for both simulations. With each of these, you should include the necessary data to interpret each waveform (clock period, part families for each part). Also, if you did any calculations to determine the critical path delays, you should include those as well.
4) A copy of the VHDL code for each module used in the implementation and testing of this project.
5) Answers to the following questions:
   a) Note that there is no guarantee that the input signals A-D will occur in time to avoid setup delays at the output flip-flops. How would you restrict use of this counter to avoid problems such as these? Please use specific timing numbers. An example of one such restriction would be "the CLR_L falling edge must occur at least 15 ns after the LD_L falling edge" (note that this is not the answer to this question; it is just an example of the form that your answer should be in).
   b) One use for binary counters such as this one is to implement a divide-by-2^n function within a digital logic circuit. For instance, a free-running 74x163 counter (one that is always enabled with no clears or loads) will result in a QD output that has a 50% duty cycle and toggles every 8 clock cycles, giving it a period equal to 16 (2^4) times greater than the original input clock period. If you were given a clock signal of *f* and wished to clock a flip-flop within the same design at a rate of *f/16*, you might try to use the QD output of the 74x163 to clock the flip-flop; however, this is not a very good design practice! Why would you not want to do this? Is there a better way to accomplish the task? (Hint: many flip-flops have an ENABLE input that allows an external signal to control whether data is allowed to be clocked through or not).