<u>CpE 318, Section A: Design Project #1</u>

(due Thursday, April 8)

* This project can be done in groups of up to five. Groups must be approved by me no later than Tuesday, March 9. No collaboration between groups is permitted. Identical or nearly identical solutions will receive NO CREDIT for either group (at my discretion).

Problem: Design a Huffman Encoder Chip:

The chip is initially reset to the idle state by asserting the synchronous *reset* signal. The encoding process begins when *start* is asserted, and ends when start is deasserted. Starting with the first character of the file to be encoded, a new character is read every rising edge of clk_i when *Crqst* is asserted and *start* is asserted. During the encoding process, the number of occurrences of each character is noted, and the characters are stored for later encoding. An 'eot' character (ASCII code 04_{16}) should be internally added as an input with an occurrence of 1, to be used to designate the end of the encoded file. This 'eot' character is not part of the input file, but will be part of the Huffman tree and will be the last encoded character output. After start is deasserted, a Huffman tree is to be constructed, such that each character is assigned a Huffman code. After this, the machine should wait until *ready* is asserted, and then assert *done* and begin outputting the bit stream on *S*. A new bit is output every rising edge of clk_o. First each character is output, starting from its MSB, followed by its corresponding Huffman code. Since Huffman codes are various lengths, the *EOE* signal is asserted when the LSB of a character's code is being output, signifying the end of the character's encoding. *EOE* is again asserted when the first bit of the encoded file is being output (i.e. two consecutive asserted *EOE* signals denote the start of the encoded file being output). After the character encoding is output, the encoded file is output (the characters in this file were stored when read in), starting with the MSB of the encoding of the file's first character. *done* is deasserted when the LSB of the encoded 'eot' character is output, signifying that the encoding process is complete. The maximum file length to be encoded for this project is 65,000 characters.

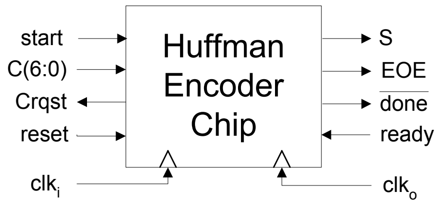

Figure 1: Block Diagram of Huffman Encoder Chip.

• see the following webpage for a discussion of Huffman encoding: www.cs.duke.edu/csed/poop/huff/info

Requirements:

You must design your system with a datapath controlled by an ASM. Each component in the datapath can be designed as either a behavioral, structural, or dataflow model. Be especially careful that the entity name, input/output names, and input/output order are exactly the same as what is shown below for your chip, otherwise your design will not run with my testbench. The name of your main entity should be: *Huffman_Chip*. The input order should be: *start*, *C*, *reset*, clk_i, clk_o, *ready*. The output order should be: *Crqst*, *S*, *EOE*, *done*.

Turn in a report containing the following:

1) a block diagram and description of your chip (similar to above)
2) a diagram and explanation of your datapath
3) the ASM(s) used to control the datapath in 2), along with explanation
4) the TPC diagram. what is the TPC for your design? explain
5) the VHDL code for your chip (also email to: smithsco@umr.edu)
6) the VHDL code for your testbench (also email to: smithsco@umr.edu)
7) your simulation macro (also email to: smithsco@umr.edu)
8) your pre-synthesis simulation along with a description of your simulation
9) your post-synthesis VHDL code (make sure there are no errors or warnings during synthesis)
10) your post-synthesis simulation along with a description of your simulation (this should be the same as your pre-synthesis simulation)
11) calculate area as the total number of "gates" reported by Leonardo + 1000 gates for each memory module + 1 gate for each bit of memory

- use a clock period of 50 ns for clk_i and 10 ns for clk_o
- change inputs on the falling edge of the clock
- **DO NOT** synthesize the memory modules

Extra Credit:

1) 5 points extra credit for the design with the least area
2) 5 points extra credit for the design with the minimum TPC