CpE401: Advanced VLSI Design
Semester Project

Fall Semester, 2003.

For your project, you will design a simple Lempel-Ziv (LZ77) real-time encoder. The project will be designed in VHDL for implementation in a custom chip. By completion of the project, your design should be ready for shipping to be fabricated by MOSIS. Details are given below.

Background/Specification:

Lempel-Ziv is a lossless encoding scheme often used to compress data-files and more recently used for real-time compression of data for telecommunications. There are several variants of the Lempel-Ziv algorithm. We will implement one of the most simple versions, the LZ1 (also known as the LZ77) algorithm. The technique will be explained in class. For additional information see references [1], [2], [3]. Many additional resources are also available on the web.

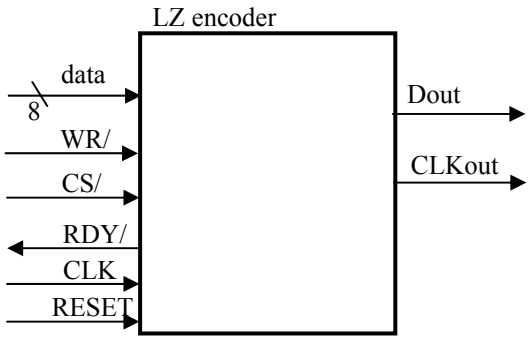

Fig. 1. Block diagram of the LZ encoder.

A block diagram of the encoder you will implement is shown in Fig. 1. The arrows show whether a pin is an input or an output. Pins on the left are used to communicate with an attached microprocessor or to receive a clock signal. The pins on the right are used to transmit compressed data.

The RESET pin is used to restore the chip to a pre-defined state.

The attached microprocessor sends the encoder information through the 8-bit data input bus. This information is buffered internal to the LZ encoder. Information in the buffer is compressed and written out 1-bit at a time through Dout. Data is sent most-significant bit first. A new value is presented on Dout for each rising edge of CLKout. CLKout is determined internally by the encoder and has a period that is an integer multiple of the input CLK's period. If the encoder does not have any information to send, then CLKout = 0.

Several parameters determine the performance of the encoder, as will be discussed in class. For your design, you will use the following:

- The overall size of the encoder's internal buffer will be 128-bytes long... that is, the encoder can store 128-bytes of information sent to it from the microprocessor.
- 32-bytes of the internal buffer make up the "codebook"... these are bytes that are either zero on reset or have already been transmitted.
- The search-window size is 16-bytes (i.e. the maximum code-word length is 16-bytes).
- The output codeword is 3-bytes long. The first byte is the value of the pointer, the second byte is the length, and the last byte is the new symbol (This code-word length obviously is not optimal, but is sufficient for our current purposes).

A communication sequence with the microprocessor looks something like this:

1. Input signals start as RESET=0, WR/=1, CS/=1.
2. If the encoder is ready to receive information, the encoder has RDY/ (ready) pulled low. The microprocessor will not send information if RDY/ is high.
3. The microprocessor places the (8-bit) data on the data lines. The address is "latched" by the transmitter when the microprocessor pulls CS/ and WR/ low (typically, CS/ is pulled low before WR/, though either is OK).
4. The data-write process ends when WR/ goes high.
5. If the written data fills the encoder's input buffer, then the encoder sets RDY/ high indicating that no more information can be received. RDY/ is set low again once the encoder's input buffer has more room available.

Your top-level entity should be declared as:

```
entity LZ_ENCODER is
        port( data: in std_logic_vector(7 downto 0);
        WR, CS, CLK, RESET: in std_logic;
        RDY: out std_logic;
        DOUT, CLKOUT: out std_logic);
end entity LZ_ENCODER;
```

You should make your design as small and as fast and to use as little power as possible. Obviously, there will be tradeoffs you will have to make (e.g. by adding extra hardware you might be able to make it faster but it will be bigger and use more power). You should explain these decisions in your report. The process of deciding among these tradeoffs is an important part of VLSI design. Some interesting suggestions for improving all three are given in [2], [3].

Preparation for MOSIS: Preparing a chip for MOSIS is not terribly different than steps we've used in CpE311 for synthesizing a design to an FPGA, though there are additional steps needed to test power usage, to specify I/O connections, and to generate the CIF file that is shipped to MOSIS. All these steps can be performed in Mentor Graphics. Additional details will follow in the near future. For information about MOSIS in general, check out http://www.mosis.org.

If we had time to fabricate your design, it would be implemented using the 0.5 micron AMIS process (see http://www.mosis.org/Technical/Processes/proc-ami-c5n.html). This

process has 3 metal layers, 2 poly layers, allows stacked contacts, and is made for 5-volt applications. For this process $\lambda = 0.3$ microns (giving a little bit of wiggle room).

Testing: A simple testbench has been provided for you on my web site. At a minimum, your design should pass this testbench; however, you are also responsible for more complete testing of your project. Your project will be tested by another much more complete testbench of my own after it is turned in.

Teams: Projects will be done in teams of 3 of your own choosing. Team member contribution will be evaluated in the final report and through a confidential evaluation the last day of the semester.

Report: Your completed project will be documented in a final written report. The report itself (not including source code, schematics, or simulation results) should not exceed 12 pages unless you get special permission from the instructor. *At a minimum*, your report should include:

- Title and team members
- Summary. Summarize entire project in 200 words or less. Be as specific as possible (similar to cliff notes).
- A clear explanation of your design rationale. Include a block-diagram of your intended design. Explain how you ensured implementation of this design through your VHDL code (i.e. it is sometimes possible to write VHDL code that synthesizes to something very different than you intended).
- An explanation of your test rationale, including testbenches, results, a logical explanation of why your test results/methods convince you your design REALLY works (think of me as your boss, getting ready to spend $100k getting your design fab'd. I wouldn't be happy if I spent money to fab it and it didn't work. Convince me to spend the $$). Testing with my testbench is the minimum requirement.
- If your design did not work, give me an explanation of why it didn't work and what would be needed to fix it.
- If you tried multiple design options, summarize options you tried and discarded with an explanation why they were discarded (for example, if you were trying to get the smallest, fastest design possible).
- Work effort distribution. List each person in your group. Tell what their job was and the total percentage effort they contributed to the completion of the project.
- Design "datasheet". The datasheet will summarize the design's features and performance specifications. A good example of a word-based datasheet is given at http://www.umr.edu/~daryl/classes/cpe311/ds1822-datasheet_v2.doc. (Yours does not have to be so complete). Your datasheet MUST include:
    - Maximum clock speed.
    - Maximum output transmission speed (in bits per second). I just want the rate that bits are coming out -- you do not have to account for the compression of the input bits (which should give you a higher overall throughput).

    - The size of the chip. Specify which MOSIS chip size you would need to implement your design (e.g. the 1.5mm by 1.5mm AMIS tinychip).
    - The average power used by the chip.
- Hardcopy of well-commented source code or schematics, included in the appendix.

Due Dates:

- **Thur., Nov. 20.** Functionally complete VHDL model that passes vcom (it does not have to synthesize or work yet).
    - Send me the model via email. Your entire project should be tar'd and zipped into a single file, including the testbench where appropriate. To understand how to do this, do a "man" on tar and gzip on a Unix machine… I believe the following command should work: "tar –cf – *directory_name* | gzip -9 > *yourname*.tar.gz", where *directory_name* is the name of the directory containing your design and *yourname* is YOUR name (so I can easily identify which files belong to whom). This command should be issued from the directory containing your design directory (the parent).
- **Tues., Nov. 25.** Functionally complete VHDL model that synthesizes (it does not have to work yet).
- **Fri., Dec. 5.** Complete VHDL model that synthesizes and passes the testbench. This is your final design. Email me your complete code as well as the .CIF file.
- **Tues., Dec. 9**. Report

Grading: Grading criteria include the information given in the report, the form and style of your report, performance of your design - especially with my own testbench, timeliness in meeting deadlines, design features, project size, speed, and power, quality of explanations given in the report, and design creativity and elegance. Generally speaking, a small, fast, low-power design is worth more points than a big, slow, power-hungry design (unless you can convince me otherwise).

Plagiarism: I would not be surprised if you find similar designs out on the web. You are welcome to use *ideas* from other designs, but are responsible for building the design yourself. If you use ideas from someone else's design, **you must clearly reference their design in both your report and in your VHDL code/schematic**. Failure to do so will bear very unpleasant consequences (failure of the course or worse!). If you have any doubts about using someone else's ideas, please ask! Each team must develop their own project. Sharing schematics or VHDL code is forbidden.

References

[1] *Ziv, J.; Lempel, A. "*A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, Volume: 23(3), pp. 337 -343, May 1977.

[2] R. Ranganathan and S. Henriques, "High-speed VLSI design for Lempel–Ziv-based data compression," *IEEE Trans. Circuits Syst.,* vol. 40, pp. 96–106, Feb. 1993.

[3] *Bongjin Jung; Burleson, W.P.;* "Efficient VLSI for Lempel-Ziv compression in wireless data communication networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6(3), pp. 475 -483, Sept. 1998.