CpE311 – Semester Project

Winter Semester, 2003.

For your project, you will design a simple transmitter for the BEETNET CSMA/CD network. The project will be designed for implementation in VHDL for synthesis to an Actel FPGA (preferably the 40MX02). Details are given below.

Background/Specification:

CSMA/CD stands for Carrier-Sense Multiple Access with Collision Detection. It is a simple network protocol that's been around since the 70's. It allows several digital devices (computers) to talk over a single wire. Roughly speaking, a transmission is performed as follows:

- Begin transmission when the communication line is free. For the BEETNET, the line should be monitored for as long as it takes to send one complete packet to see if it is busy.
- Read data from memory and transmit over network
- Monitor transmission. If a collision occurs (i.e. another device tries to write to the network at the same time):
    - Jam the network by sending all 1s for several clock cycles (the BEETNET will jam the network for 64 clock cycles).
    - Wait a random period of time (the BEETNET will wait up to 255 clock cycles).
    - Start over, resending the last data packet.

For the BEETNET, data are sent in individual packets. Packets are 27-bits long and contain a start-bit, an 8-bit address, a 16-bit block of data, a parity bit, and a stop bit:

packet = [start bit][8-bit address][16-bit data][1-bit parity][stop bit]

The address is the "name" of the processor you want to send the data packet to. The parity bit is even, meaning it is set to 1 or 0 to make the total number of ones in the packet an even number (not including start and stop bits). The start bit is one. The stop bit is zero. After sending the stop bit, the transmitter continues to send a zero out until it begins the next transmission (next start bit).

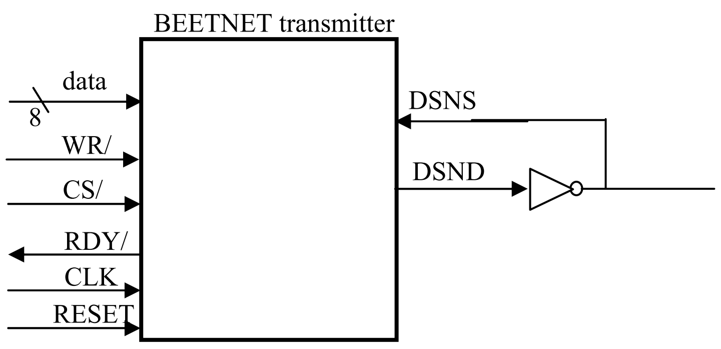

Fig. 1. Block diagram of BEETNET transmitter.

Fig. 1 shows an external view of the BEETNET transmitter. 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 communicate with the network. Information is sent to the network through DSND (data send). DSNS (data sense) is used by the transmitter to detect if the network is busy or if there is a collision. The inverter is an external component (in this case, one that pulls the line weakly high and strongly low, allowing several devices to communicate over the same line at the same time). (Question: is the network busy when DSNS sees a ‘0’ or a ‘1’?).

The RESET pin is used to restore the chip to a pre-defined state. An actual implementation may not require this pin, but it will help your simulation/synthesis significantly. The reset pin is normally low (i.e. reset on an input-high).

A transmission sequence, including communication with the microprocessor, looks something like this:

1. Transmitter has RDY/ (ready) pulled low, indicating to the microprocessor that it is ready to receive information to transmit.
2. The microprocessor places the (8-bit) packet-address on data lines. The address is “latched” by the transmitter when the microprocessor pulls CS/ and WR/ low.
3. The microprocessor places the most-significant byte of the data-packet (8-bit) on the data lines. Data is “latched” by the transmitter when the microprocessor pulls CS/ and WR/ low.
4. The microprocessor places the least-significant byte of the data-packet (8-bit) on the data lines. Data is “latched” by the transmitter when the microprocessor pulls CS/ and WR/ low.
5. The transmitter pulls RDY/ high, indicating to the microprocessor that it is busy and additional information cannot be written to it.
6. The transmitter sends the packet over the network, using the transmission scheme cited above. Transmission starts with the start bit and works toward the stop bit. The start, stop, and parity bit are supplied by the transmitter (not the microprocessor).
7. When transmission has completed successfully, the transmitter pulls RDY/ low.

Your design may be done in either schematic capture or in VHDL. For VHDL, your top-level entity should be declared as:

```
entity BEETNET_TRANS is
        port( data: in std_logic_vector(7 downto 0);
        WR, CS, CLK, RESET: in std_logic;
        RDY: out std_logic;
        DSNS: in std_logic;
        DSND: out std_logic);
end entity BEETNET_TRANS;
```

You should make your design as small and as fast 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). You should explain these decisions in your report. The process of coming up with designs with good speed and size characteristics is an important part of VLSI design.

Testing: A short testbench is provided for you on my web site. In addition to testing, this testbench should be a good source for further understanding the design specification. 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).
- An explanation of your design rationale.
- 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:
    - Number of logic modules used.
    - Maximum clock speed.
    - Maximum transmission speed (in bits per second). This calculation should account for time needed for the microprocessor

    to write data to the transmitter as well as to transmit the data… so, in affect, this is the maximum *average* transmission speed.
    - The ACTEL 40MX FPGA that must be used to hold your design (see http://www.actel.com/products/mx/index.html). Justify the increased cost of using an FPGA larger than the 40MX02, assuming cost is directly proportional to the number of logic modules. (e.g. if the MX02 with 295 modules cost $1, then the MX04 with 547 modules costs (547/295)*1 = $1.85 or 85% more).
- Hardcopy of well-commented source code or schematics, included in the appendix.

Due Dates:

- **Monday, April 28.** 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).
- **Thursday, May 1.** Functionally complete VHDL model that synthesizes (it does not have to work yet).
- **Monday, May 5.** Complete VHDL model that synthesizes and passes the testbench. This is your final design.
- **Friday, May 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 and speed, quality of explanations given in the report, and design creativity and elegance. Generally speaking, a small, fast design is worth more points than a big slow design (unless you can convince me otherwise).

Plagiarism: I would not be surprised if you find similar designs out on the web or from past CpE311 groups. You are responsible for building your own design. 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. If you have any doubts about using someone else's ideas, please ask! Each team must develop their own design. Sharing schematics or VHDL code is forbidden.