A Mini Packet Switch Project as an Auxiliary Tool for Digital Circuit Design Learning

Bruna S. Leandro, Carlos P. V. Jr, Caroline A. Amado, Fernando S. C. Cunha, Hugo A. Henley, Paula R. F. Alves and João M. M. Silva

***Abstract*— The main approach for teaching/learning digital techniques in a classroom usually requires that theory and practice be taught together. In digital circuits laboratories classes, students can design, simulate and implement basic logical circuits as counters, coders/decoders, adders, and other circuit modules. However, those circuits are studied in a non integrated form and students do not have a greater and final objective with all those circuits. This paper points toward a project that uses the basic circuit blocks already done by students to integrate them into a totally functional data communication equipment.**

***Keywords*— Digital Circuits, Digital Techniques, Packet Switch, Teaching, learning.**

I. INTRODUCTION

The general applied approach for teaching digital techniques in under-graduation courses such as electronics, telecommunications, automation and others, is traditionally separated into two parts: theoretical and practice.

However, in a digital circuit laboratory, students are stimulated to design, simulate and implement basic logic circuits modules such as adders, counter, coders/decoders, and many others. But, usually, those students learn each circuit module and, after that, they do not use them, moving on for the next activities of a new circuit module.

Although some tools had great evolution in recent years, such as development boards and simulation tools, this latter approach was not modified too much [1].

With digital and web based technology advances, new approaches arose for digital design teaching/learning [2], [3], [4], [5].

All those tools are welcome and important for supporting digital circuit design learning/teaching, but they fail on providing students of a bigger objective such as a complete and totally functioning equipment, where all modules work together.

This paper describes a mini packet switch design, composed by basic circuit modules, which offers students:

- The opportunity to design, simulation and implement a functional data communication equipment;
- To exercise engineering team skills;
- To exercise working under time and budget requirements;
- To understand the functioning of each basic logic circuit as a part of an equipment.

The mini packet switch project, also known at Universidade Federal Fluminense as *TD-10 Project*, allow four PC-Like computers to communicate through a chat service running under a browser environment.

The authors are with Telecommunications Engineering Department of the Engineering School at Universidade Federal Fluminense, Niterói, Rio de Janeiro - Brazil. E-mail: jmarcos@telecom.uff.br.

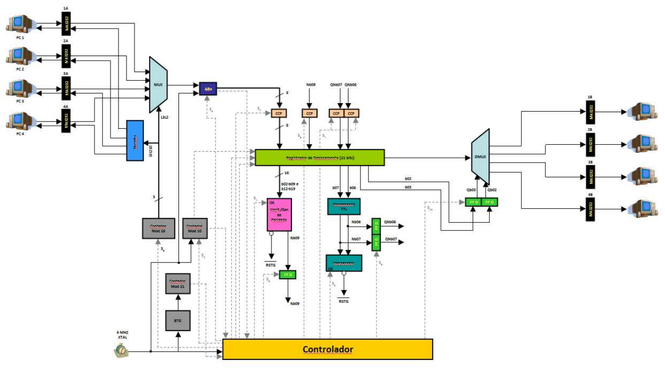

Fig. 1. Block Diagram of the Mini Packet Switch.

Each module is designed, tested and simulated by students using a VHDL developing tool, such as Max+Plus II© [6], for implementation on a Complex Programmable Logic Device (CPLD) [7], [8].

II. METHODOLOGY

The proposal of our methodology is to introduce, in first theoretical class, the complete block diagram of the mini packet switch project - Figure 1, followed by a brief description of its importance as a data communication equipment and as a practical project for laboratory studies. Requirements for the mini packet switch is also presented.

The mini packet switch uses building block circuits like adders, multiplexers/demultiplexers, controller, shift register, parity generator and parity checker, counters, asynchronous data communication, etc..., addressing the maximum number of possible theoretical concepts seen in classroom.

Each block correspond to a complete laboratory class, described on a script and involving several activities. Some of these activities are made previously by students at home or at libraries, such as understanding requirements and designing it. The remaining activities, as simulating, testing and packaging the circuit as a building block into a library, are done into the laboratory under supervision of a professor/monitor. This step is important to guarantee that requirements are being respected and, at the end of semester, that all the blocks be integrated with no problems.

Classes, with their respective scripts, are presented in the following order:

1) Introduction to the Laboratory: The Project, Tools, Requirements and Basic Logical Gates;
2) Max+Plus II by Altera© introduction

3) Parity Check/Generator Module;
4) Decoder Module;
5) Mux/Demux Modules;
6) Decrement and Comparator Modules;
7) Parallel Load Module (CCP)
8) Shift-Register Module;
9) Counters Modules
10) Controller Module;
11) ARx Module
12) Integration - Part I;
13) Integration - Part II;
14) Integration - Part III;
15) CPLD Recording and Final Tests.

On each laboratory class, its respective circuit module is simulated, tested and packed as a building block, and inserted in an user library for future use on integration classes.

III. The Mini Packet Switch Project

A. General Description

The mini packet switch works receiving packets from a computer and sending it to one of the others computers or back to itself. Computers are numbered ranging from 0 to 3.

The project was conceived to be as simple as possible, since it must be designed, simulated, implemented and tested in a single semester under-graduation course. So, little functionalities can be implemented:

Figure 2 shows the three layers of the mini packet switch project. Layer 1 (Physical Layer) uses the RS-232 Protocol for communication between computers, and Layer 2 (Data Link Layer) uses our own data packet format.

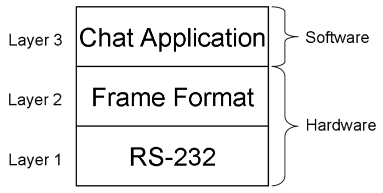

Fig. 2. The mini packet switch layers.

The packet format can be seen in Figure 3 and it is basically an array of 21 elements.

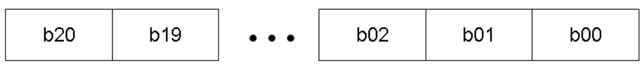

Fig. 3. Packet format containing message and header.

Each element is enumerated in an descending order:

b20	- Stop bit.
b19..b12	- Message Bits (ASCII Character).
b11	- Start bit.
b10	- Stop bit.
b09	- Even parity bit.
b08	- Type of message.
b07..b06	- Time-To-Live bits (TTL).
b05..b04	- ID of the source computer (ID Source).
b03..b02	- ID of the destination computer (ID Destination).
b01	- Start bit.
b00	- Stop bit.

A single information packet is divided on two bauds for communication and, since the project does not use a commercial UART chip for asynchronous communication, the necessary start and stop bits were placed among information and header bits.

The TTL bits and parity bit were introduced on design just to add some features found in commercial data communication equipment (routers). Those bits can be set by the user via chat application to simulate data communication fails and end of time-to-live reaching. They do not have real meaning for the mini packet switch operation.

B. Mini Packet Switch Operation - A Brief description

When the mini packet switch initiates, the decoder module enables the first computer (PC0) to send a byte. This byte is received asynchronously by the ARx module, indicating to the controller module that a byte is ready to be consumed (output signal E_0).

The controller module disables the PC0 transmission (through the decoder module) and enables the parallel load of the received byte into the shift-register module.

The controller module enables the Contador10 module to produce 10 pulses to the shift-register, shifting the received byte to its right position and freeing space to the next byte to be received.

The Contador10 module indicates to the controller (output signal E_1) the end of its execution and returns to its starting point.

The controller module resets the ARx module and re-enables PC0 transmission of the second byte.

Again, the ARx module receives and indicates to the controller that the second byte is ready to be consumed.

Thus, the controller module enables once again the parallel load of the received byte into the shift-register.

After that, the controller module enables the parity check module to verify if there is a parity error. If there is a parity error, process is re-initiated, but using the next computer.

If there is no parity error, the controller module enables the Deccomp module to decrease the TTL field, checking if this value is zero or not. If TTL is zero, process is also re-initiated, but using the next computer.

If the TTL field did not reached zero value, then the TTL field is updated into the shift-register.

The controller module enables the parity bit to be updated after the TTL bits.

The controller module locks the ID Destination into two type D flip-flops to hold the *line selection* inputs of the demux module, locking the destination PC in order to receive two bytes. The message is ready to be transmitted - See Figure 7.

Finally, the controller module enables the Contador21 module to produce 21 baud ticks in order to transmit the two bytes with the appropriate clock rate for asynchronous transmission.

The Contador21 module indicates to the controller (output signal E_2) the end of its execution and returns to its starting point.

The controller modules enables the next computer transmission and all the cycle is done again.

C. Circuit Modules Description

1) Parity Checker/Generator: The parity checker/generator module is responsible for checking if there is an error in the received message packet - Figure 4.

If there is an error, than the module will reset all circuits (the exception is the Contador16 module) and a new cycle begins. Input S_4 is responsible for enabling the $\overline{RST1}$ output to reset modules.

If there is no error, the other operations will take place and, after the TTL field have been finally updated, a new parity bit will be generated (NB9) and the parity bit (bit 9 of the shift register module) will be updated.

There is no automatic retransmission of any packets neither a messaging warning about the event.

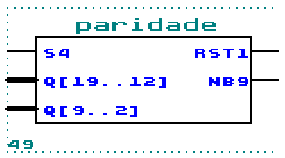

Fig. 4. The Parity Checker and Generator Module.

The parity bit is evaluated by the chat application, but users can toggle it to force a parity error and test this function in hardware.

2) Mux/Demux Modules - The PC Computer Selection: One of the most interesting schema designed for the packet switch is PC computer source and destination selection.

Figures 5 and 6 show how a multiplexer, a demultiplexer and a decoder works together to receive and send bytes from and to PC computers.

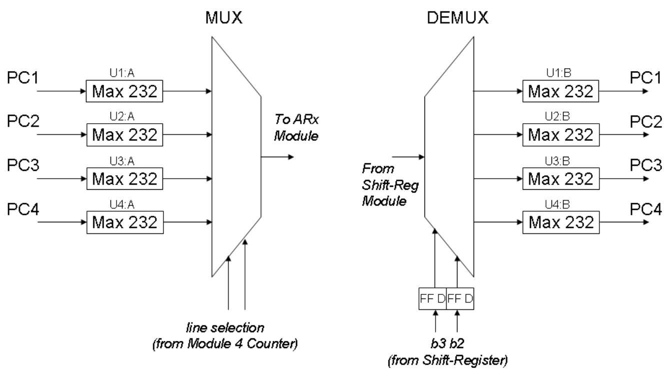

Fig. 5. Multiplexer and Demultiplexer for computer source and destination selection.

A module four counter is enabled/disabled by the controller module. This counter, together with de decoder module, is responsible for enabling one computer at a time, to transmit two bytes of information.

To keep design as simple as possible, our main goal, whenever a computer has no information to transmit, it will transmit

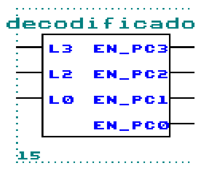

Fig. 6. Decoder for computer source enabling.

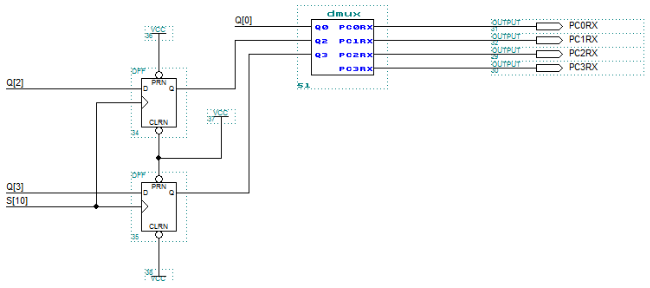

Fig. 7. Demultiplexer with some additional components.

zeros as information to PC1. This feature is implemented by the chat application, relieving hardware from more complexity.

Figure 7 shows the demultiplexer with two type D flip-flops, responsible for holding the *ID Destination* PC computer bits.

3) Deccomp Module: The Time-To-Live bits are used in Internet Protocol Datagram to prevent packets from circulating indefinitely on network.

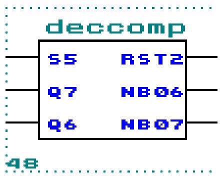

Fig. 8. TTL processing module.

The Deccomp module decreases the TTL number until it reaches zero value. When it happens, this module resets the other modules (exception is the Contador16 module) and a new cycle of communication begins. Figure 8 shows details about TTL Decrement and Comparator modules. Q6 and Q7 inputs come from bits 6 and 7 of the shift register (TTL bits). Input S_5 is responsible for enabling $\overline{RST2}$ output. Outputs NB06 and NB07 are the new TTL bits, to be updated into the shift register.

4) Parallel Load Module - CCP: With two inputs and two outputs, the Parallel Load Module (CCP) is a combinational logic circuit that allows someone set or reset a flip-flop. Figure 9 shows the logical gates with a type D flip-flop for the parity bit into the shift-register module. Table I is the truth-table for this circuit.

As long as the enable input is on low level (logic state 0), the flip-flop works normally. When enable input line is on high level (logic state 1), the flip-flop state follows the second input Nb09 directly.

This logic circuit is useful for executing parallel load on flip-flops and update values inside the shift-register (e.g., TTL and parity bits).

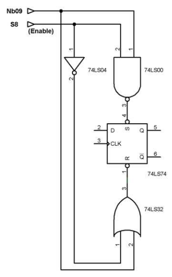

Fig. 9. The loader module is a combinational circuit composed by a NAND, OR and INVERTER logical gates.

TABELA I
TRUTH TABLE FOR THE CCP MODULE COMBINATIONAL CIRCUIT.

S_8 (enable)	Nb09	Set	Reset
0	0	1	1
0	1	1	1
1	0	1	0
1	1	0	1

The mini packet switch has eleven CCP modules, eight for the received byte, in order to do the parallel load of the shift-register, one for updating the parity bit, and two for updating the TTL field (two bits).

5) The Shift-Register: One of the most important circuit modules is the shift-register (21 bits) with parallel load (b19 to b12 bits, b09, b07-b06 bits) and its input controls, shown in Figure 10. It receives the incoming bits from one of the PCs through de ARx Module.

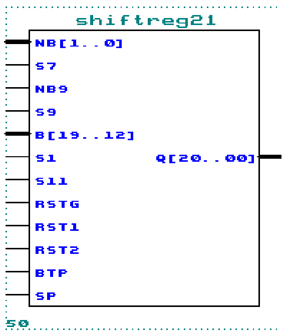

Fig. 10. The Shift-Register Module.

The first baud received contains information about the source ID computer, destination ID computer, TTL and parity bit.

6) Counter Modules: The mini packet switch project uses four counters:

- Module 10 counter (Contador10): Used to shift the first received byte to its correct position into the shift-register module;
- Module 16 counter (Contador16): Used to enable/disable PC computers to transmit data;
- Module 21 (Contador21): Used to transmit all bytes into the shift-register (including start and stop bits) to the destination PC computer;
- BTG Module: Used to generate baud ticks.

The BTG (Baud Tick Generator) module is a counter with two inputs: Clock and RSTG (General Reset), and one output (BT - Baud Tick) - Figure 11.

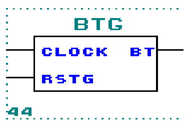

Fig. 11. The Baud Tick Generator Module.

This module is responsible for clock division, lowering the original clock frequency (4 MHz) to 9600 KHz, necessary to the asynchronous serial communication with PC computers.

7) Controller Module: All modules are under a controller operation. It has three inputs, thirteen outputs and twenty-two states - Figure 12.

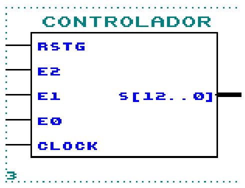

Fig. 12. Controller module.

The controller outputs are:

S_0	Increases the Contador16 module for PC computer selection;
S_1	Load the received byte into the shift-register (b19-b12) when high;
S_2	This line, when on high logic level, enables Contador10 module in order to shift bits b19-b12 to b09-b02 location;
S_3	Rearm ARx module for receiving the second byte from PC when high;
S_4	Enables $RST1$ output for the Parity Checker/Generator module;
S_5	Enables $RST2$ output for the Deccomp module;
S_6	Enables the latch of the new TTL bits (Nb07 and Nb06) for posterior updating on shift-register module;
S_7	Enables the TTL CCP module to update bits b07 and b06 for its new value (Nb07 and Nb06 bits from latch) on shift-register module;
S_8	Load the new parity bit into the type D flip-flop;
S_9	Enable the parallel load of the parity bit into the shift-register module;
S_{10}	Enable the latch of the ID of the destination PC (b03 and b02 bits), which will drive the demultiplexer;
S_{11}	This state sets the start and stop bits into the shift-register module;
S_{12}	Enable Contador21 module in order to transmit the message to the destination PC computer. After that, returns to S_0 state.

8) ARx Module: For the physical layer, asynchronous communications were chosen for the sake of simplicity. Figure 13 show the block diagram representation of this module.

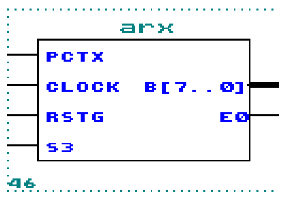

Fig. 13. ARx - The Asynchronous Receiver Module.

The ARx module receives bauds from a PC computer (input PCTx), through a dual driver/receiver (MAX 232) e presents a data byte on its output (B[7..0]).

This module has a second output (E_0) to indicate to controller that a byte is ready to be used.

Controller module rearms the ARx module for the next byte through line S_3 through a high logic level.

Clock input comes from the BTG (*Baud Tick Generator*) module.

9) Clock Generator: The clock generator was implemented with an oscillator, model XO53CTDNA4M by Vishay Dale, reducing complexity. Figure 14 shows the oscillator[1].

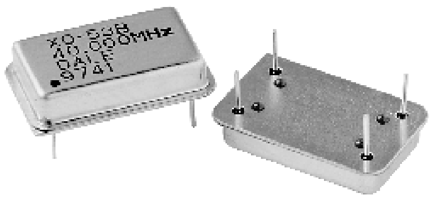

Fig. 14. The oscillator used as a clock generator.

D. Chat Software Application

As an application, a chat software was conceived for allowing users to communicate with a web browser.

Python was the chosen language for some reasons such as modules and packages already included, modules developed by various programmers around the world, easy of learning.

Thanks to its facilities, it was quite easy to find a library that was already had the necessary treatment for the serial port, called *pyserial*.

This library encapsulates modules to access the serial port, which served for both Windows and Linux. Once imported into a Python program, we could configure a serial communication port by creating an object that would carry parameters to choose: 9600 bps, parity bit none, two stop bits, etc.

User may configure a nickname and send it by broadcast to the other PC computers. The chat application receives the messages and verifies the *Type of Message* bit (b08).

If the Type of Message bit is 1, then the message is about the nickname from sender. If it is 0, then the message is to be displayed on chat area. With this procedure, users know who is in chat application.

[1]Picture extracted from XO-53 datasheet by Vishay Dale.

IV. Conclusion

Although we had not used any kind of technique to measure improvements on our new teaching methodology yet, it is quite easy to see that a new enthusiasm arose among students of digital techniques discipline, showing more interest on learning.

The mini packet switch helps students to get a better understanding of the functions of each basic logic circuit learned and how they work together to form a complete functional equipment.

With this approach, students are stimulated to learn technical skills as Hardware Description Language, minimization techniques, debug tools. It also helps them to develop the social abilities necessary as future engineers: Team working, sharing responsibilities, time schedules, budgets, etc.

Actually, the team is working on a new PCB layout design that will provide students just to program the CPLD and plug it directly into the PLCC-84 socket and start communications through a browser based chat.

This project begun in 2011 on telecommunications engineering undergraduate courses, and it is still evolving trough contributions from all people interested on using the project to teach or learn digital techniques.

All the modules are fitted into one single EPM7128SLC84-15 CPLD unit by Altera.

All the sources of the project can be accessed on our provisory website [9].

Referências

[1] A. Sudnitson, D. Mihhailov, M. Kruus, "Advanced topics of FSM design using FPGA educational boards and web-based tools", *East-West Design and Test Symposium*, pp. 514–517, 2010;

[2] A. Jutman, J. Raik, R. Ubar and V. Vislogubov, "An educational environment for digital testing: hardware, tools, and Web-based runtime platform", *Proceeding of the* 8^{th} *Euromicro Conference on Digital Systems Design*, pp. 412-419, 2005;

[3] F. Machado, S. Borromeo, N. Malpica, "Project based learning experience in VHDL digital electronic circuit design", *IEEE International Conference on Microeletronic System Education*, pp. 49-52, 2009;

[4] H.-D. Wuttke, K. Henke, "Teaching digital design with tool-oriented learning modules living pictures", 32^{nd} Annual Frontiers in Education, Vol. 3, S4G-25 - S4G-30, 2002;

[5] F. Morgan, S. Cawley, F. Callaly, S. Agnew, P. Rocke, M. O'Halloran, N. Drozd, K. Kepa and B. McGinley, "Remote FPGA Lab with Interactive Control and Visualisation Interface", *International Conference on Field Programmable Logic and Applications*, pp. 496-499, 2011;

[6] www.altera.com

[7] R. Duek, "Digital Design with CPLD Applications and VHDL", *Delmar Cengage Learning* - 2^{nd} edition, 2004.

[8] I. Grout, "Digital Systems Design with FPGAs and CPLDs", *Newnes* - 1^{st} edition, 2008.

[9] B. S. Leandro, C. A. Amado, C. P. V. Júnior, F. S. C Cunha, H. A. Henley, P. R. F. Alves e J. M. M. Silva, *http://www.professores.uff.br/jmarcos/index.php ?option=com_content&view=article&id=18&Itemid=34*