New Features of IPNoSys IDE

Victor Santos Batista
Universidade Federal Rural do Semi-Árido (UFERSA)
Centro de Ciências Exatas e Naturais
Mossoró - Brasil
victor.sb02@gmail.com

Sílvio Roberto Fernandes
Universidade Federal Rural do Semi-Árido (UFERSA)
Centro de Ciências Exatas e Naturais
Mossoró - Brasil
silvio@ufersa.edu.br

ABSTRACT

IPNoSys is an unconventional architecture based on networks-on-chip. It defines its own programming and computation models. To learn its concepts and to develop experiments, there are a SystemC simulator and an assembler. These tools were previously integrated into an IDE, implemented in Qt, that added a text editor and animated graphical interface for the simulation. In this paper are presented new tools to this IDE, such as autocomplete functions, support for C language and breakpoint, among others.

CCS CONCEPTS

• **Software and its engineering → Software notations and tools → Compilers**; *Computer systems organization → Architectures → Other architectures*

KEYWORDS

IPNoSys, IDE, tool, unconventional achitecture

ACM Reference format:

V. Batista, S. Fernandes. 2017. New Features of IPNoSys IDE. In *Proceedings of 17th Microelectronics Students Forum, Fortaleza, Ceará Brazil, August 2017 (SForum'17)*, 4 pages.

1 INTRODUCTION

In all science course the theory and practice are of the utmost importance to the learning process, thus computer science is not different. However, many institutions are no having laboratories with the hardware necessary to perform the practice. In this way, the importance of using simulators arises, because it provides the abstraction of the hardware operation.

The simulators represent a powerful tool for the teaching of computer architectures [10]. By providing the abstraction of the architecture, the students simulate its behavior and visualize the operational details better, favoring the learning process. The costs with the simulator are very lower than the hardware acquisition, which facilitates the teaching and manipulation of that. They are also very important in the teaching of theoretical architectures in the research phase, since the manufacture of hardware is much more expensive. With regard to unconventional architectures [1], which may have their own characteristics and paradigms, the use of tools (simulators, compilers, etc.) for learning and experiments may be indispensable.

An example of a tool with this purpose is "IPNoSys IDE" [8] that integrates a development and simulation environment for the unconventional architecture IPNoSys. Such tool aims to aid the teaching IPNoSys concepts, but also as a research tool related to the IPNoSys architecture and helps in understanding the execution of the applications since the debugging process by the previous tools to this architecture is not a trivial task.

The constant evolution of IPNoSys IDE it becomes necessary, to improve the development environment to the users, promoting the teaching and research, increasing productivity and improving abstraction to IPNoSys. Users report that the IPNoSys IDE improves the process to coding, testing and debug, however, it identified some characteristics that can improve in IPNoSys IDE and some that can be included.

Thus, this paper presents new features of IPNoSys IDE, a simulator tool dedicated to experiments with an unconventional architecture based on NoCs. The new resources added the IDE. The rest of this paper is organized into three sections. Section II describes the platform IPNoSys. Section III presents the new features of the IPNoSys IDE. Lastly, section IV discusses the conclusions.

2 IPNOSYS PLATFORM

2.1 Organization and Architecture

IPNoSys is a general-purpose processor with an unconventional organization that takes advantage of infrastructure and features from network-on-chip to favoring parallel communication and processing. Such organization is formed by a network of RPUs (Routing and Processing Unit) with a MAU (Memory Access Unit) in each corner (Fig. 1).

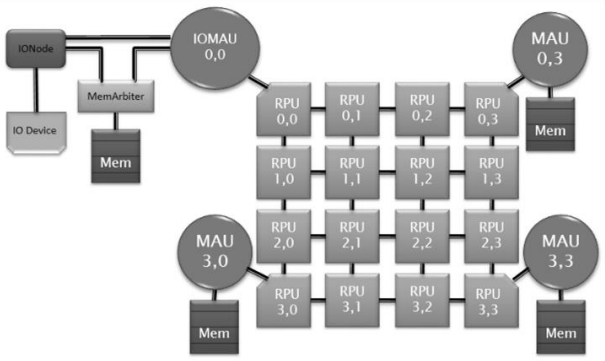

Figure 1: IPNoSys Organization.

The RPUs are responsible for routing packets and execute logic and arithmetic instructions inside them. Thus, the RPUs are formed by regular components of routers as buffers, a crossbar, and arbiters, but also includes one Arithmetic Logic Unit (ALU) and one Synchronization Unit (SU). The MAUs load and store data, as well as inject packets and execute synchronization instructions. IPNoSys also has an input and output system based on DMA (Direct Memory Access) which is controlled by one of the MAUs, called IOMAU. It has the following characteristics: topology NoC Square 2D grid; It uses at least two virtual channels; routing XY modified; switching combining VCT and wormhole; control credit-based flow; distributed arbitration; and storage at the entrance [5].

The architecture comprises 32 instructions [7], divided into regular (logical, arithmetic, conditional and auxiliary) and control (synchronization and memory access). The regular instructions are similar to those of any traditional processor and are executed by the RPUs. The control instructions are executed by the MAUs and consist of only: Load, Store, Send, Exec, Synexec and Sync.

2.2 Computing and Programming Models

The IPNoSys applications are described by packet format that includes a header (with information of application and packet routing), instructions and operands. An application is formed by a set of packets that are stored in memories on corners of the network. Such memories are managed by MAUs, which injects packets (ordered by EXEC or SYNEXC/SYNC instructions) and reads/writes data (by the LOAD, SEND and STORE). In each packet, instructions are queued according to the data dependencies among subsequent operations. Such dependencies establish the order that the instructions will appear in the packets. Thus, the results of previous instructions can be used as an operand in following instructions. Thus, when a packet is injected each RPU executes at least one instruction, stores temporary the result, forward the rest of the packet, inserting the result in a specific position of packet as an operand to other instruction. This computing model is represented in Fig. 2.

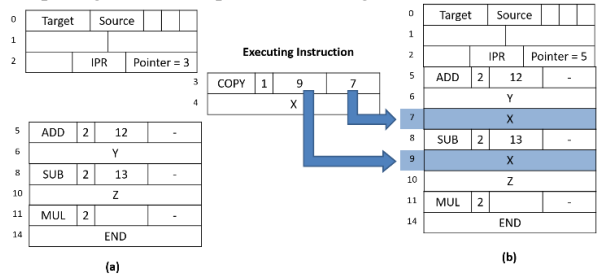

Figure 2: Computing Model: (a) before; (b) after execution

Such packets flow according to a routing algorithm that guarantees all instructions are executed and solves the conflicts and imminent deadlocks. Others purpose algorithms are presents in [2]. The packets can be injected sequentially by the same MAUs, as a pipeline, or simultaneously by four MAUs to exploit parallelism.

To develop applications to IPNoSys is used its assembly language, called PDL (Packet Description Language) that has structures equivalent to packet's fields. Such language allows the programmer parallelizes explicitly the code and indicates the join points. The parallelism with better performance in IPNoSys is in task level, mainly from unrolling loops [6].

2.3 Development Tools

The IPNoSys experimental environment relies on the following tools: an assembler, a simulator and an IDE, illustrated in Fig. 3.

The assembler, written in C++, performs a lexical and syntactic analysis from PDL code and generating an equivalent object code. The object code is read by the simulator, developed in SystemC [9] with cycle accurate, in order to execute the program and generate a simulation report with many information at the end. The simulator also allows the generation of a log file with individual information of each component, when the debug macros are enabled before the simulator is compiled. However, since the components have concurrent behavior (implemented thought SystemC threads), debug information may be chronologically difficult to track.

This has led to the development of an IDE (Integrated Development Environment) [8] in C++ and Qt [4]. Such IDE includes a text editor interconnects the original assembler and simulator, shows the simulation in a graphical interface in animated way and exhibit the simulation report.

In [8] is presented the first version of IPNoSys IDE, which is divided in three modules: editor, simulator and simulation report.

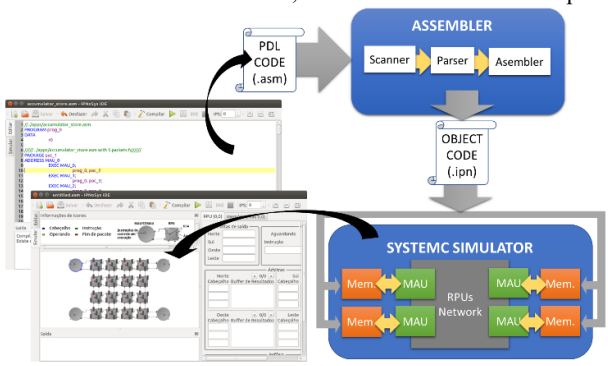

Figure 3: IPNoSys Development and Simulation Environment

The editor module highlights the reserved words of PDL and the current line, the number of lines and has tradition functions like copy, paste and undo; the editor keeps communication with assembler, providing the PDL code and displaying the successful/unsuccessful message from assembler. The simulation module keeps communication with the IPNoSys SystemC simulator, that executes in background, and provides tools for the user controls and observes the execution of the compiled code in an animated way. Through the graphical interface, the user can modify all parameters of the SystemC simulator and choose the velocity animation (in instructions per seconds) or step mode (the cycles correspond user's clicks). This module also allows observing what each component is doing at a given time, including the ALU, the input buffers and arbitrators of each RPU (by clicking on one of them), as well as the MAUs. As the packets are being processed

and routed between the RPUs the animation shows each part of the package colorfully in the components that hold it. The report module only shows the simulation report generated by the simulator.

The next section presents the new features purpose by this paper for editor and simulator modules.

3 NEW FEATURES OF IPNOSYS IDE

3.1 Editor Module

In order to increase the abstraction and explore parallelism automatically, [3] developed C2PDL, a compiler of language C to PDL. This compiler, with the objective of to explore parallelism, has three levels to optimization: O (no optimization); O1 (reducing LOAD-STORE and COPY instructions); O2 (unrolling loops executing by parallel threads).

This compiler is being included to IPNoSys IDE, thus this paper already includes in the editor the highlighting the C code syntax (Fig. 4). The highlight of C as PDL, is achieved by Qt resources for regular expressions and syntax highlighting classes [4].

The new features purpose in this paper for editor module are: find and replace functions, autocomplete to C and PDL code, memorization of working directory and the construction of an infrastructure for editor communication with the C2PDL compiler and the assembler.

The autocomplete resource has been implemented by monitoring events in the editor text area. As the user types, a search is initiated for reserved words (no case sensitive) of the language (C or PDL) depending on how the file was saved (.c or .asm). Identifiers created by users like program names, packages, variables, and result labels are included in another list to be searched for auto-completion. All results of the search are listed in a pop-up at the current text cursor location and refreshed while the user is typing. The user can select any word of pop-up for choose that it will appear in the text area (Fig. 4).

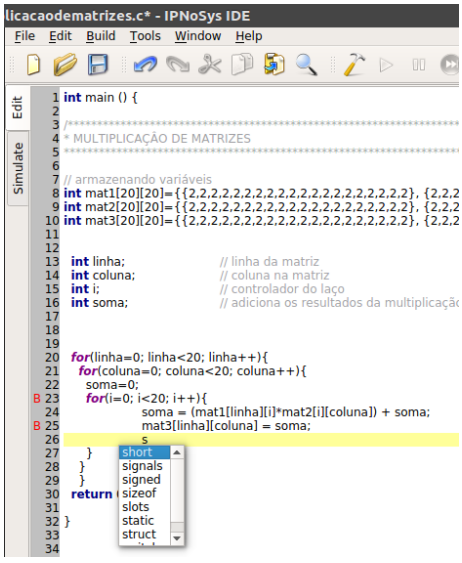

Figure 4: Example of new features in IPNoSys IDE

3.2 Simulator Module

In the simulation module, it was added the functionality of open the object code file without the need to open PDL code, nor compile it. It is thus possible to start the simulation of a previously compiled code. And the more relevant feature proposed in this paper is the inclusion of breakpoint function, which makes it possible to perform a code debugging even it is parallel.

The breakpoint mechanism affects almost all components in IPNoSys platform, since editor until SystemC simulator, aided by the breakpoint table. In order to insert a breakpoint, a programmer clicks on left side of a line, which causes the appearance of the letter B next to the line number (lines 23 and 25 of Fig. 4).

This causes the editor to insert the source code line into the breakpoint table. During assembler work, the generated object code statements corresponding to the lines marked in the source code and the MAU identifier that will inject the packet of those instructions are also included in this table. Finally, while loading the packets to the memories of each MAU, the memory addresses of the breakpoint instructions are marked in the table. Fig. 5 summarizes this process.

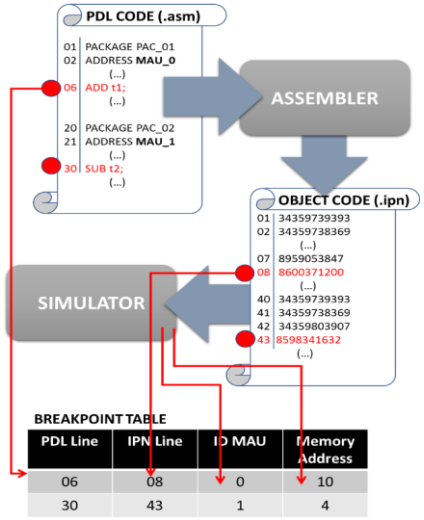

Figure 5: Breakpoint table

During the simulation, the packets are injected by the MAUs, which send the address of each packet word to the memory and receive the corresponding value. In this process, when an address is found in the breakpoint table, it causes the simulation of the program to stop. Fig. 6 shows a simulation stopped by a breakpoint, that is reported in output box as "Breakpoint found". This allows the programmer to perform debugging more easily, thereby increasing productivity and improving the validation of simulations. The user can clicks on any RPU to see (at right side) the current information about its buffers, operation and results. Clicking on a MAU is shown the data in the correspondent memory.

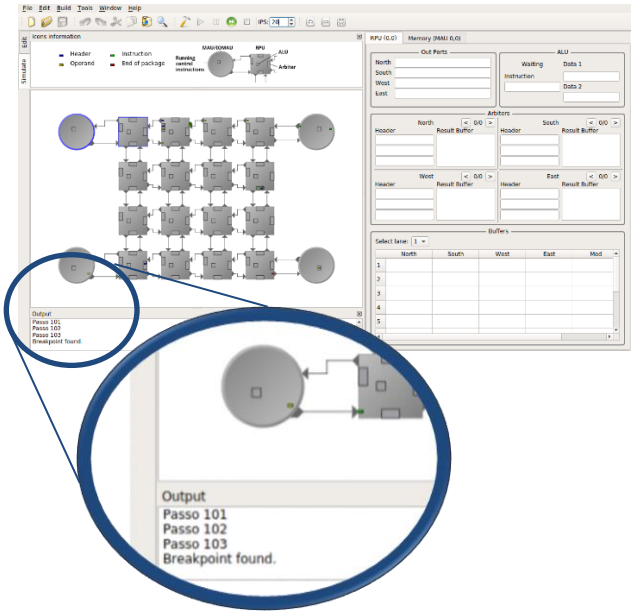

Figure 6: Simulation stopped by breakpoint

4 CONCLUSIONS

The paper presents new features of IPNoSys IDE, a development and simulation environment to the unconventional architecture IPNoSys, that provides to students, programmers and researchers, a greater abstraction of the architecture and tools to support the learning its paradigm, development of programs and simulation process.

Aiming to improve the abstraction, productivity and accuracy we added new resources in the editor and simulator modules. In this version, IPNoSys IDE supports C language to integration to C2PDL compiler; includes "Find and Replace" tool; has autocomplete function to PDL and C source codes; opens object code (.ipn) and simulates directly; and provide breakpoint mechanism. The constant evolution and update of the IDE besides favoring programmers and researchers, allowing focus on applications and the organization and architecture. This IDE has been used in many in undergraduated and post-graduate researches. As the IPNoSys IDE was developed in a modular way, if changes are made to base tool code (assembler and SystemC simulator), and preserve the communication interfaces with the GUI, new architecture models may arise and to benefit from this integrated environment.

Future works include: to improve the information shown by RPUs during the simulation, also show the internal state of MAUs; allow to change, in runtime, the GUI idiom (Portuguese or English), currently is done in compiler-time; show automatically the source code that cause the breakpoint during simulation; include all IPNoSys versions (developed by other researchers) and allow the user choose one to simulation and summarize a simulation report comparing them; evaluate the IDE's usability by the users feedback and their options about what can improve.

ACKNOWLEDGMENTS

This work was supported by UFERSA from PICI project "Implementação de Novas Funcionalidades na IPNoSys IDE".

REFERENCES

[1] A. Adamatzky. Unconventional computing. International Journal of General Systems, 2014 Vol. 43, No. 7, 671–672

[2] Raimundo Valter Costa Filho, Silvio Fernandes, Dênis Nunes, I. A. Alves, and W. Dantas. 2014. Exploração de espaço de projeto do roteamento na arquitetura IPNoSys. *HOLOS* 4: 175–184. https://doi.org/http://dx.doi.org/10.15628/holos.2014.1909

[3] Juliene Vieira Couto. 2016. Geração de Código Otimizado para Exploração de Paralelismo em uma Arquitetura não Convencional. UFERSA, Mossoró.

[4] Digia. Qt Project. Retrieved from http://qt-project.org/

[5] Sílvio Fernandes, Bruno C. Oliveira, and Ivan Saraiva Silva. 2009. Using NoC Routers As Processing Elements. In *Proceedings of the 22Nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes* (SBCCI '09), 24:1–24:6. https://doi.org/10.1145/1601896.1601927

[6] Silvio Roberto Fernandes, Bruno Cruz de Oliveira, Miklécio Costa, and Ivan Saraiva Silva. 2009. Processing while routing: a network-on-chip-based parallel system. *IET Computers & Digital Techniques* 3, 5: 525–538.

[7] Silvio Roberto Fernandes, Ivan Saraiva Silva, and Márcio Kreutz. 2010. Packet-driven General Purpose Instruction Execution on Communication-based Architectures. *Journal of Integrated Circuits and Systems (JICS)* 5: 53–66.

[8] Lucas Oliveira and Silvio Fernandes. 2015. IPNoSys IDE Ambiente de Desenvolvimento e Simulação Integrado para uma Arquitetura não Convencional. *International Journal of Computer Architecture Education (IJCAE)* 4, 1: 21–24.

[9] ACELLERA. SystemC. Retrieved from: http://accellera.org/downloads/standarts/systemC

[10] G. Wolffe, W. Yurcik, H. Osborne and M. Holliday, Teaching Computer Organization/Architecture With Limited Resources Using Simulators, ACM SIGCSE Bulletin 34, (1), 176-180, 2002.