Evaluation of data flow experiments in the IPNoSys NoC

Jefferson Lemos[1], Jonathan Mesquita[1], Márcio Kreutz[1], Sílvio Fernandes[2], Edgard Correa[1]
brunoluno10@gmail.com, jonathan.wanderley@gmail.com, kreutz@dimap.ufrn.br, silvio@ufersa.edu.br, edgard@dimap.ufrn.br

ABSTRACT

This work is devoted to the study of a network on chip [3] model: IPNoSys (Integrated On-chip Network Processing System), having as main objective to obtain an optimal configuration of the network, comparing performance and power consumption. In this work, the platform IPNoSys shall be subject to execution of a set of synchronous dataflow generic applications. The term “generic” means that they have no real application, being in fact a set of arithmetic instructions. This approach will ensure a greater test coverage. The aim is to obtain results which provide an ideal setting for IPNoSys network components, regarding to the processing of applications which give us a known and constant data stream.

Keywords: power consumption, performance, noc, ipnosys, data flow, synchronous data flow, evaluation, soc, network

1. INTRODUCTION

As computer architectures evolve and software complexity increases, architectural models come out from a Central Processing Unit to the systems called Multi-core. IPNoSys was created by [2] which describes a network on chip capable of performing processing as routes. The allocation of multiple processing units influence the parallel execution, and the various physical interconnections between these components ensure the existence of alternative routes.

IPNoSys’ model of execution, described by [3], motivates the study of this network through the implementation of applications that have a constant known data flow. This paper begins with an introduction to the synchronous data flow model, followed by WORKFLOW section, where we present the test environment we used. RESULTS section presents graphics obtained by running applications on the network. In section DISCUSSION OF RESULTS the results are interpreted and discussed. Finally, the CONCLUSION section presents the conclusions from the studies.

2. BACKGROUND

The dataflow model of computation is not new. It exists since the 70s and is characterized by being data-driven. A program dataflow can be represented by a directed graph, whose nodes correspond to the instructions, and the edges, the flow of data. Therefore, an instruction is executed only when all its incoming edges have tokens (released by other nodes), meaning that the data necessary for their implementation are now available [5]. Figure 1 illustrates a data flow graph, where lower nodes process as soon as the top node instruction in this statement has been executed.

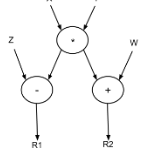

Figure 1 - Data Flow Graph.

IPNoSys has a computational model similar to the dataflow. Its programs are divided into smaller units, called packages, which encapsulate data and instructions. Packages travel over the network topology that is a 2D grid 4x4 (16 nodes). Each node corresponds to a routing and processing unit (RPU) capable of processing packages and forwarding packages to the next RPUs. Memory access is made via four memory access units (MAU) that are distributed in the four corners of the network, and are also responsible for injecting packages into the system. The IOMAU component is responsible for performing the function of an ordinary MAU, and performs input and output operations (I/O).

3. WORKFLOW

In order to obtain desired results, it was essential to create a tool able to generate a variety of dataflow programs to be executed on the IPNoSys platform. This simple tool is based on the purpose of generating synchronous dataflow programs from entries such as the amount of packages and the number of instructions per pack. As shown in Figure 3 we used the generator, the pre-existing network assembler and the simulator, each one receiving inputs and generating outputs. Finally, it is generated the log file containing the simulation results.

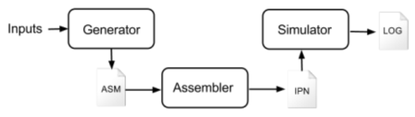

Figure 3 - Workflow for obtaining the results

4. RESULTS

The execution of an application in the simulation environment IPNoSys network generates a log file where it can be seen the number of cycles and the total energy spent to process the entire program. From these logs, graphics were plotted to obtain a better view of the samples to different test cases.

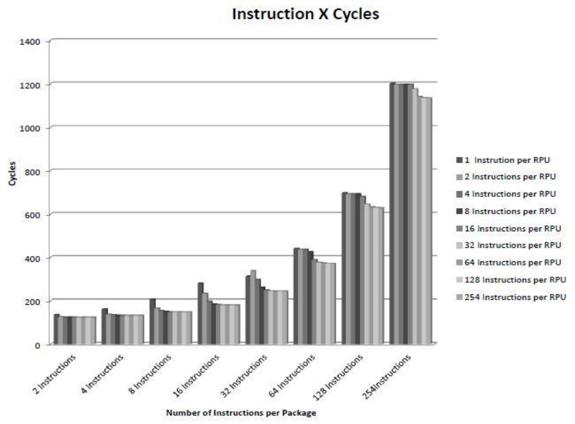

Figure 4 – Graphic Instructions x Cycles, 1 package.

The experiment conducted to obtain the graphic in Figure 4 consists of injecting only one package in the network varying the amount of instructions per packages and instructions per

This work has been developed by the first author(s) in the scope of the undergraduate studies

RPU. The graphic shows the number of cycles spent by the network for each configuration.

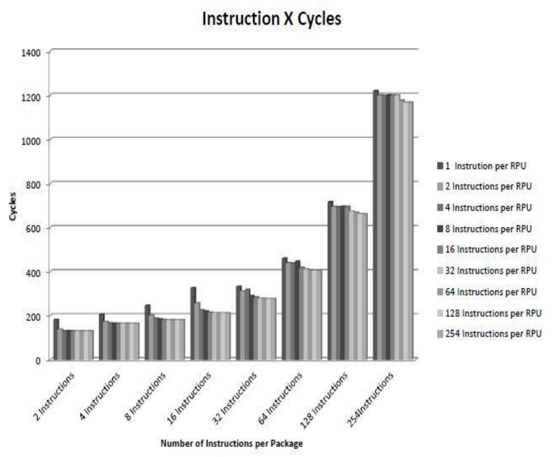

Figure 5 – Graphic Instructions x Cycles, 4 packages.

The experiment conducted to obtain the graphic in Figure 5 is the injection of 4 simultaneous packages in the network, varying the amount of instructions per packages and instructions per RPU. The graphic shows the number of cycles spent by the network for each configuration.

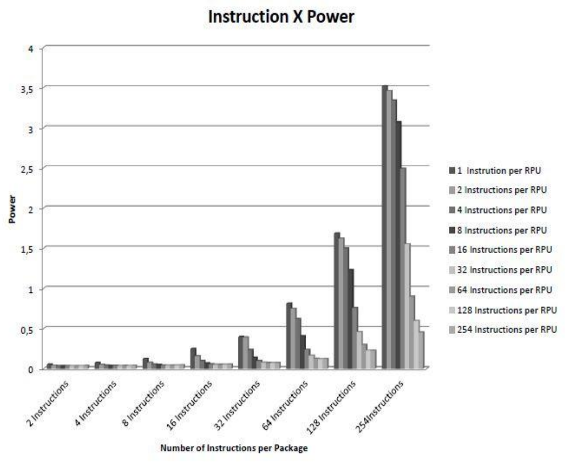

Figure 6 – Graphic Instructions x Power, 1 package.

The experiment conducted to obtain the graphic in Figure 6, consists of injecting only one packet in the network by varying the amount of instructions per packages and instructions per RPU. The graphic shows the energy spent by the network for each configuration.

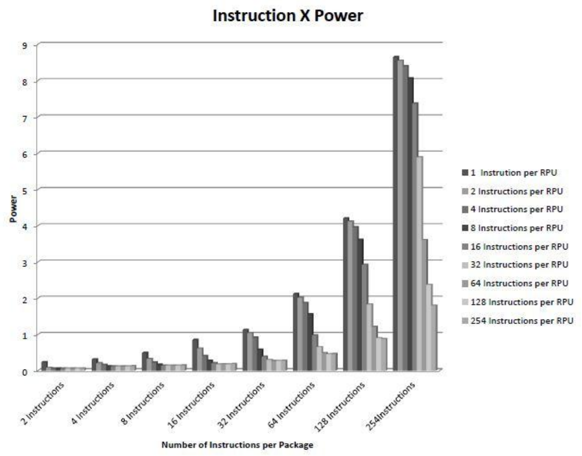

Figure 7 – Graphic Instructions x Power, 4 packages.

The experiment conducted to obtain the graphic in Figure 7 consists of injecting simultaneously 4 packages in the network, varying the amount of instructions per packages and instructions per RPU. The graphic shows the energy spent by the network for each configuration.

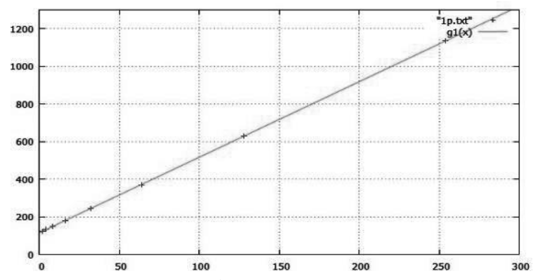

Figure 8 - Linear regression of cycles, 1 package and 128 instructions per RPU. X axis: instructions. Y axis: cycles.

Figure 8 represents the line obtained by the numerical method of linear regression for an average case of testing for the amount of cycles spent by the network, where each RPU was programmed to perform up to 128 instructions in each package, considering that only one package was injected into the network and varying the amount of instructions per package.

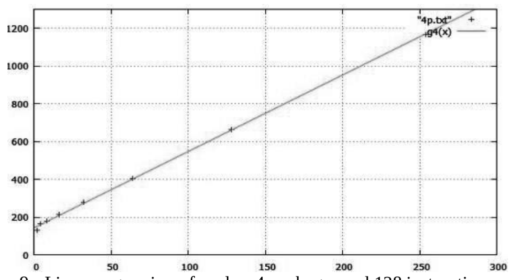

Figure 9 - Linear regression of cycles, 4 packages and 128 instructions per RPU. X axis: instructions. Y axis: cycles.

Figure 9 represents the line obtained by the numerical method of linear regression for an average case of testing for the amount of cycles spent by the network, where each RPU was programmed to perform up to 128 instructions in each package, considering the injection of 4 packages in the network and varying the amount of instructions per package.

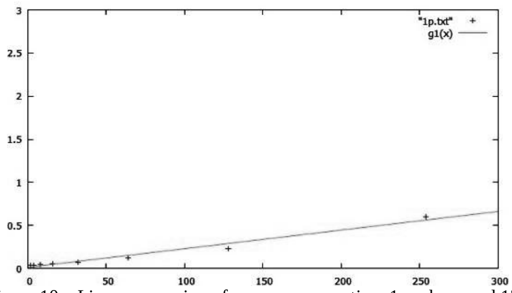

Figure 10 – Linear regression of power consumption, 1 package and 128 instructions per RPU. X axis: instructions. Y axis: mW.

Figure 10 represents the line obtained by the numerical method of linear regression for an average case of tests for the energy spent by the network, where each RPU was programmed to perform up to 128 instructions in each package, considering the injection of 4 packages in the network and varying the amount of instructions per package.

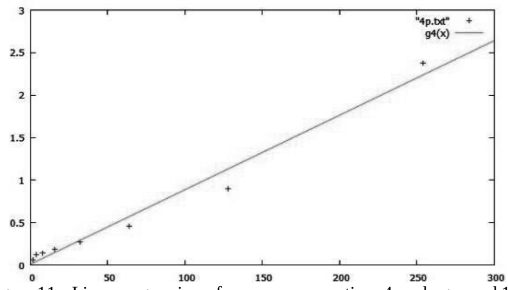

Figure 11 - Linear regression of power consumption, 4 packages and 128 instructions per RPU. X axis: instructions. Y axis: mW.

Figure 11 represents the line obtained by the numerical method of linear regression for an average case of tests for to the energy spent by the network, where each RPU was programmed to perform up to 128 instructions in each package, considering the injection of 4 packets in the network and varying the amount of instructions per package.

5. DISCUSSION OF RESULTS

5.1 PERFORMANCE

Interpretation of the results of performance tests (in cycles) shows that for each configuration (assuming that in this case, the term configuration refers to the number of instructions executed for each RPU and the amount of instruction per package), as the number of packages in the network increases, the number of cycles necessary for executing the program remained almost constant, having an average increase of 2.5% for each package added.

The nearly linear behavior is due to the fact that all packages injected into the network present instruction independence. A package doesn’t the need to wait for the other’s processing, ensuring a fully parallel implementation. Part of the cycles spent to process the application can be justified by the implementation of verification algorithms and treatment of deadlocks.

From the regression was observed that the time spent in cycles to process different programs with an equal number of packages, varying only the instructions in each package, presents a linear behavior, being easily estimable by the function described in the figures 7 and 8.

5.2 POWER CONSUMPTION

From the graphics, it is concluded that by increasing the number of packages in the network, the energy consumed does not increase linearly with the instruction package, with a behavior describable by a polynomial function. It was also observed that the amount of energy required to process a program showed a linear increase for the same configuration.

Furthermore, according to Figures 6 and 7, the power expended decreases as the amount of instructions per RPU increases. By increasing the number of instructions per RPU, there is less communication between nodes, consequently, less switching RPU ports and a lower buffer allocation for the transfer of packages. When the number of instructions per RPU is greater than or equal to the number of instructions per package, data processing always happens on one single node, causing under-utilization of network resources.

6. CONCLUSION

In this paper we evaluate the IPNoSys platform by implementing a set of generic synchronous dataflow applications, presenting the results in graphics, where it was possible to observe extreme and medium cases of execution. From the interpretation of results, we extracted an optimal configuration for the IPNoSys the synchronous data flow applications scenario.

IPNoSys by its computational model proved to be an ideal architecture for synchronous dataflow applications. An optimal configuration for the network will depend on its use for a particular purpose. For large data processing it is ideal to use it with their RPUs programmed to perform a large number of instructions, because the closer this number is the amount of instructions in the package, the higher the performance (less cycle spending), and lower energy cost energy, by not occurring a high transmission of packages among RPUs. However, this practice under-utilize network resources, while maintain a higher traffic through only four corner nodes.

7. REFERENCES

[1] FERNANDES, S. et al. Processing while routing: a network-on-chip-based parallel system. In: IET Computers & Digital Techniques, v. 3, n. 5, p. 525-538, 2009a. Available at: < http://link.aip.org/link/?CDT/3/525/1 >

[2] FERNANDES, S.; OLIVEIRA, B. C.; SILVA, I. S. Using NoC routers as processing elements. In: SBCCI, 2009b. Natal, Brazil. ACM.

[3] FERNANDES, S. et al. IPNoSys: uma nova arquitetura paralela baseada em redes em chip. In: IX Simpósio em Sistemas Computacionais - WSCAD SSC 2008, 2008. Campo Grande. SBC.

[4] ZEFERINO, C. A. Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho.
2003. 242 Doutorado (Doutorado). Instituto de Informática, Universidade Federal do Rio Grande do
Sul, Porto Alegre.

[5] LEE, E. A., MESSERSCHMITT, D. G., Synchronous Data Flow, Proceedings of the IEEE, Vol 25, no., 9, Sept 1987.