COARSE-GRAINED RECONFIGURABLE ARCHITECTURES

Alisson Gonçalves Damasceno Garcia
Ricardo Santos Ferreira
Universidade Federal de Viçosa, Departamento de Ciência da Computação
Viçosa, MG Brasil CEP 36570-000
alissongarcia@gmail.com - cacau@dpi.ufv.br

ABSTRACT

This work presents a set of tools to explore a reconfigurable system based on scalar processors and Coarse-Grained Functional Units Accelerators. These tools have been added to an EDA environment [1], which has been developed by using an incremental approach. Our environment allows the designer to model and to simulate several processor/array architecture characteristics in order to evaluate implementation tradeoffs. The tools are implemented in Java/XML to provide portability and extensibility. Our main contributions are: web interface, array routing algorithm, import/export features, a flexible Java processor simulation, a flexible DSP kernel simulation based on array architecture, and a XML array specification.

1. INTRODUCTION

Recently, array processor architectures have been proposed as extensions of microprocessor-based systems. These architectures have been used to accelerate streaming applications and to save energy. An array is a regular structure, which is scalable and presents a high degree of parallelism.

A previous work, named EDA (Environment for Exploration of Data-Driven Array Architecture) [1] has been presented to support coarse-grained array exploration. Our goal is to improve EDA by adding more tools to model and simulate new architectures.

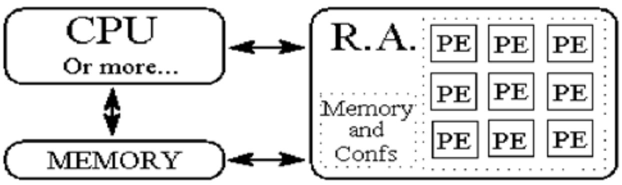

Fig. 1 CPU, Memory and Reconfigurable Array Architecture

Moreover, scalar processors connected to Coarse-Grained architectures shows a very interesting solution (see Fig. 1).

We have been using a Java-based editor/simulator tool called Hades [2] and XML, to provide more portability and extensibility.

2. EXTERNAL INTERFACE

This work aims to support multiprocessor. A set of Low power Java processors has been present in [3]. This set is composed by a multicycle version, a 5-stage pipeline, a VLIW and an array of functional units attach to the pipeline version. All processors have been evaluated by using a power simulator, named CACO-PS [4]. We have implemented a java parser to import a CACO-PS description into EDA. To validate our approach, we have mapped a low power pipeline FemtoJAVA processor, which has 338 components and 352 signal wires. In addition, we have added a graphic interface (see Fig. 2), and web applet [5].

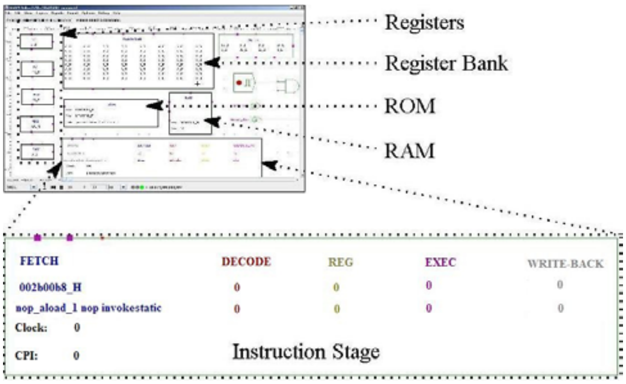

Fig. 2 FemtoJAVA Pipeline on Hades.

The graphical interface provides detailed characteristics about the functioning like dynamic behavior, instant power consumption, etc.

3. ARRAY FRAMEWORK

A coarse-grained array processor is a regular architecture which is based on set of interconnected processing elements (PEs). To simulate a generic array structure, we propose a frame component. This frame is a reconfigurable Java class. One or more functional units can be instanced inside a frame. A XML file specifies how the set of frames are interconnected. The interconnect pattern could be a bus, a crossbar, a grid, a hypercube, a hierarchical clusters.

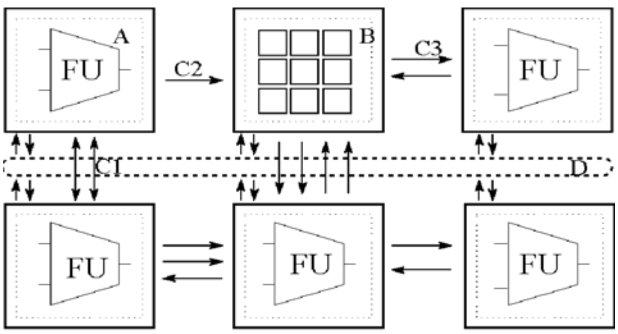

Fig. 3 Frames.

Figure 3 shows a heterogeneous array. For example, a frame can have a FU Unit (A) or a cluster of frames (B), the interconnection can be heterogeneous (C1, C2, C3) and the array can also have a bus (D).

3.1. N-Hop Arrays

Recently, the N-Hop topologies with interconnections between adjacent and non-adjacent frames (see Fig.4), have been presented as a suitable alternative to coarse grained arrays, where N represents the number of frames the longest interconnection from a frame can leap.

We have implemented a Hybrid Non Symmetrical N-Hop Pattern, where each PE could have non-symmetrical interconnection. For instance (see Fig. 4C) PE_0 is connected to PE_2 by using a 1-hop and PE_1 is connected to PE_6 by using a 4-hop. Initially it has demonstrated good trade-offs, reducing significantly the distance between all the PEs if compared to other grid topologies.

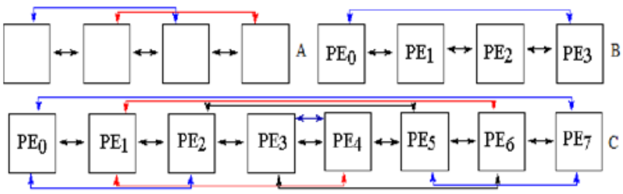

Fig. 4 1-Hop(A), 0_2Hop(B) and Non Symmetrical N-Hop(C) example.

We proposed a tool that builds a Java-based array architecture from a XML description. See below a part of 0_2Hop example (see Fig. 4B). This extract of the XML file describes an array of 4 elements. Then, the PE_0 is defined. The *borders* number defines the PE interconnection. In these case, PE_0 is connected to PE_3 and PE_0 is connected to PE_1.

```
<ARRAY rows="1" columns="4">
   <PE row="0" column="0" >
        <BORDERS number="2">
                ...
                <BORDER1 outs="1" ins="1">
                   <OUT0 cRow="0" cColumn="3">
                </BODER1>
                <BORDER2 outs="2" ins="2">
                   <OUT0 cRow="0" cColumn="1">
                </BODER2>
                ...
        </BORDERS>
   </PE>
   ..
</ARRAY>.
```

This tool was validated by the running dataflow array implementation of typical DSP kernel algorithms. Let us consider the FDCT kernel, which is a fast discrete cosine transform implementation mapped on 10x10 array. The resulting simulation structure has around 9,000 objects (wires, Functional units, frames, configuration inputs). A 250 sample stream took less than 1 minute to be simulated at a register transfer level for FDCT.

3.2. Routing

We have also been working on routing algorithms to improve the current version of EDA. The previous work [1] presents a greedy routing algorithm. However, this greedy approach [1,6] is not generic, and it is constrained to grid topologies. We have implemented a greedy approach based on Dijkstra's Shortest Path. We have found a generic approach that can route any topologies and shows better and more efficient results than previous work [1].

Our tool was validated with successful routing of major algorithms into the array like Fir128 (with 385 PEs and 511 connections) in less than 1 second.

4. CONCLUSION

This paper presents a set of tools, based on Java and XML, to provide early evaluation of data-driven, reconfigurable, array architectures. This approach includes a flexible processors/array simulation and routing scheme developed to easily evaluate different array topologies. Further work is also needed to generate an automatic VHDL prototype of a certain array or data-driven solution in an FPGA. Long-term plans include a front-end compiler to continue studies of some data-driven array features with complex benchmarks.

5. REFERENCES

[1] Ricardo Ferreira, João M. P. Cardoso, Andre Toledo, and Horácio C. Neto, "Data-driven Regular Reconfigurable Arrays: Design Space Exploration and Mapping," in *Embedded Computer Systems: Architectures, Modeling, and Simulation 5th International Workshop* (SAMOS'05), LNCS 3553 Springer, pp. 41-50, Samos, Greece, July 18-20, 2005.

[2] N. Hendrich, "A Java-based Framework for Simulation and Teaching," in *3rd European Workshop on Microelectronics Education* (EWME'00), Aix en Provence, France, 18-19, May 2000, Kluwer Academic Publishers, pp.285-288.

[3] Antonio C. S. Beck, Luigi Carro. "Dynamic reconfiguration with binary translation: breaking the ILP barrier with software compatibility", in *ACM IEEE Design Automation Conference*, San Diego, California, USA, Pages: 732-737, 2005

[4] Antonio C. S. Beck, Julio C. B. Mattos, Flavio R. Wagner, and Luigi Carro (2003). "Cacops: A general purpose cycle-accurate configurable power simulator". *In 16th Symposium on Integrated Circuits and Systems Design (SBCCI'03)*, page 349, São Paulo, Brazil.

[5] http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/96-femtojava/caco-pipeline/quicksort.html

[6] Israel Koren, Bilha Mendelsom, Amherst Irit Peled, Gabriel M. Silberman, "A Data-Driven VLSI Array for Arbitrary Algorithms", *IEEE Computer*, pp: 30 – 43, 21 (10), 1988