DATA-DRIVEN MAPPING ON A RECONFIGURABLE ARRAY

André Bigonha Toledo
Ricardo Santos Ferreira
Universidade Federal de Viçosa, Departamento de Informática
Viçosa, MG - Brasil - CEP 36570.000
(bigonha, cacau)@dpi.ufv.br

ABSTRACT

This work presents an approach for mapping a data-driven algorithm onto a reconfigurable coarse-grain array architecture. The mapping approach allows to analyze different array topologies and interconnection degrees. Moreover, an incremental input and output format is specified in XML. A N-Tap FIR filter benchmark, mapped onto on hexagonal and a grid architectures, are used to validate our approach. The results show a fast and flexible mapping to explore different array architectures.

1. INTRODUCTION

A coarse-grained array processor is a regular architecture which is based on set of interconnected processing elements (PEs). These architectures can provide a high parallelism, a low power consumption, and a high circuit reliability and scalability, which are the main drawbacks in superscalar/VLIW architectures. Moreover, in a superscalar processor, the control and connections require a large amount of circuit area. For instance, while more than 90% of Itanium 2 die is devoted to communication, control and storage overhead, only 6.5% is consumed by the ALU and the registers files [7].

Recently, the array processor architectures have been revised [1-4] by profiting from the data-driven characteristics of multimedia applications. Since many design decisions must be taken in order to implement an efficient architecture for a given set of applications, environments to efficiently experiment with different architectural features are fundamental. This work is part of an ongoing development environment [5], named as EDA (Environment for exploring Dataflow Architectures). Our contribution is to implement a mapping tool as shown in Figure 1.

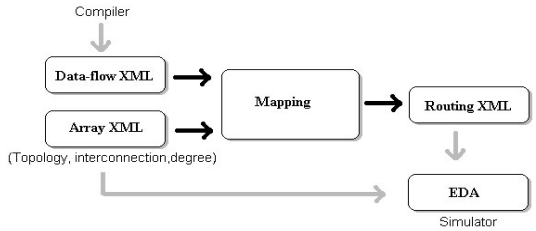

Fig. 1 Mapping flow.

Previous works [4, 6] have addressed the mapping of data-driven algorithms on regular array architectures. A place and route algorithm for a regular array architecture has been proposed in [4]. This work analyzes only the hexagonal topology with a fixed interconnection degree. On the other hand, most of array architectures are based on grid topology [2,4]. Recently, Bansal et al. [4] have explored some interconnection patterns on a grid network (See **Fig. 2a-c**), and the experimental results point out that the configuration shown in **Fig 2b** leads to good performance for the DSP considered benchmarks. Our approach differs from the previous ones in two significant ways: (a) Compare hexagonal and grid topologies; (b) Present a object-oriented mapping implementation to model different patterns.

2. A FLEXIBLE MAPPING

Our mapping algorithm takes advantages of Java and XML technology to enable a portable and flexible implementation. The object-oriented approach provides a easy way of modeling grid, hexagonal, hypercube, octal, as well as others topologies. The XML notation have been chosen as an input/output format (see **Fig. 1**) to easily integrate with other environments.

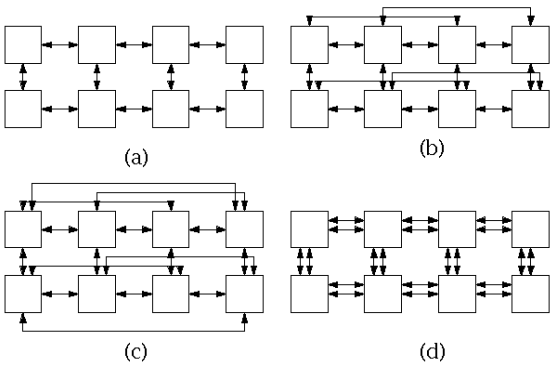

Fig. 2 Different grid connections

The implementation is based on three main classes: the array, the PE and the border. The array class implements the place and routing algorithms. The array and neighbor parameters (number, relative name and their positions) are specified by PE classes. Finally, the border class models the number and type of connections between the neighbors. A PE can be connected to 0-hop and 1-hop neighbors (see **Fig. 2b**) or can have two bi-directional connections to 0-hop neighbors (see **Fig. 2d**). Our mapping algorithm enable us to explore the connection patterns. Moreover, a comparison between different topologies can be performed.

The place and route algorithm, based on the work proposed by [6], has been implemented as an initial

solution. First, the placement is done by performing a depth first traversal on the dataflow graph. The PEs are allocated by level at the array lines. Then, each line is rearranged. For each edge in dataflow graph, the PE source node is added to the PE list . Let P1,...,Pn be the PE list. The PE list is traversed as follows: a PE is removed from the list, and its partial paths are connected to the neighbors. Then, the neighbors are added to the list. The route is done when the list becomes empty.

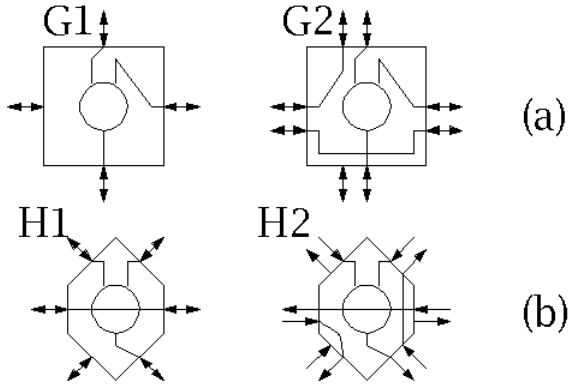

Fig.3 Different topologies and connections

3. TARGET ARCHITECTURES

An array architecture is scalable thanks to the regular design and the local connections. A grid topology can have different configurations. **Fig. 3a** shows that a grid with two bi-directional connections (G2) can perform a computation and two parallel routing. However, the grid with only one connection (G1) can either perform a computation or two routing. **Fig. 3b** shows two hexagonal PE named as H1 and H2, respectively. H1 can perform a computation and a routing. H2 can perform a computation and three routing. In this example, H2 have an input and an output connection, which is more constrained than two bi-directional connections.

4. EXPERIMENTAL RESULTS

The N-tap FIR filters have been chosen to explore different configurations by using our mapping approach. **Table 1a** shows the mapping results for a range of N-Tap FIR (see **Column 1**). Column N,E specifies the number of nodes and edges for each dataflow FIR graph. The average path length of mapped FIR onto three topologies are shown in Column 3-5, for 0-hop hexagonal, 0-hop grid (see **Fig 2a**), and 1-hop grid (see **Fig. 2b**). All PE have two bi-directional connections. From the results in Table 1, we see that the hexagonal achieves the shortest path.

Table 1a. N-Tap FIR mapping

FIR	*N,E*	*HEX*	*GRID*	*1-HOP GRID*
2	7,7	**1**	1.28	1.28
4	13,15	**1.2**	1.86	1.46
8	25,31	**1.32**	1.58	1.45
16	49,63	**1.52**	1.83	1.57

Our mapping approach has been developed to easily explore different configurations. The second experiment was done to evaluate the number and type of interconnections (see **Table 1b**). Three configurations have been tested: (c1) 1 directional, (c2) 1 input and 1 output, (c3) 2 bi-directional. The 0-hop hexagonal (H) and the 1-hop grid (G) have been chosen. Again, the bi-directional hexagonal achieves the best results. Previous work [4] have shown that 1-hop was the best configuration. However, no direct comparison can be done, since the benchmarks are different and the hexagonal topology was not evaluated.

Table 1b. N-Tap FIR mapping

	4-TAP		*8-TAP*		*16-TAP*	
	H	**G**	**H**	**G**	**H**	**G**
C1	1.4	1.8	1.9	1.7	2	1.87
C2	1.27	1.53	1.35	1.48	1.57	1.6
C3	**1.2**	1.46	**1.32**	1.45	**1.52**	1.57

5. CONCLUSION

This work presented a fast and flexible mapping algorithm for data-driven array architecture. The CPU time was less than 200 milliseconds. Moreover, this approach is flexible and the topology as well as the node interconnection degrees can be easily added, in reason of the Java and XML technologies. Future works will include other topologies and routing algorithms.

6. ACKNOWLEDGMENTS

The student André Bigonha Toledo receives his scholarship from PIBIC/CNPq.

7. REFERENCES

1. Arthur H. Veen, “Dataflow machine architecture,” in *ACM Computing Surveys*, Vol. 18, Issue 4, Dec. 1986, pp. 365-396.
2. R. Hartenstein, “A Decade of Reconfigurable Computing: a Visionary Retrospective,” In *Int’l Conf. on Design, Automation and Test in Europe (DATE’01)*, Munich, Germany, March 12-15, 2001, pp. 642-649.
3. PACT XPP Technologies, Inc., “The XPP White Paper,” Release 2.1.1, March 2002, http://www.pactxpp.com.
4. N. Bansal, S. Gupta, N. Dutt, and R. Gupta, “Network topology exploration of mesh-based coarse-grain reconfigurable architectures,” DATE, 2004.
5. Ricardo Ferreira, João M. P. Cardoso, and Horácio C. Neto, "An Environment for Exploring Data-Driven Architectures" in 14th International Conference on Field Programmable Logic and Applications (FPL'04), September, 2004, LNCS Springer-Verlag.
6. Israel Koren, Bilha Mendelsom, Amherst Irit Peled, Gabriel M. Silberman, “A Data-Driven VLSI Array for Arbitrary Algorithms”, IEEE Computer, 21 (10), 1988, pp: 30 – 43
7. Ujval J. Kapasi, Scott Rixner, William J. Dally, Brucek Khailany, Jung Ho Ahn, Peter Mattson, and John D. Owens. Programmable Stream Processors. IEEE Computer, pages 54-62, August 2003.