

Fabric-Based Systems: Model, Tools, Applications

Christophe Wolinski
Los Alamos National Laboratory
Los Alamos, NM, U.S.A.
IRISA, IFSIC France

Maya Gokhale
Kevin McCabe
Los Alamos National Laboratory
Los Alamos, NM, U.S.A.

Abstract—A Fabric Based System is a parameterized cellular architecture in which an array of computing cells communicates with an embedded processor through a global memory. This architecture is customizable to different classes of applications by functional unit, interconnect, and memory parameters, and can be instantiated efficiently on platform FPGAs. In previous work [1], we have demonstrated the advantage of reconfigurable fabrics for image and signal processing applications. Recently, we have build a Fabric Generator FG, a Java-based toolset that greatly accelerates construction of the fabrics. A module-generation library is used to define, instantiate, and interconnect cells' datapaths. FG also generates customized sequencers for individual cells or collections of cells. We describe the Fabric-Based System model, the FG toolset, and concrete realizations of fabric architectures generated by FG on the Altera Excalibur ARM that can deliver 4.5 GigaMACs/s (8/16 bit data, Multiply-Accumulate).

I. INTRODUCTION AND RELATED WORK

A computational fabric is a form of cellular array composed of application-specific, interconnected compute cells. A Fabric-Based System (FBS) combines a computational fabric with a conventional control processor such that the processor views the fabric as a large intelligent memory [2], [1]. Fabric-Based Systems combine the compute power of a fabric with the flexibility, visibility, and control of traditional processing.

In this paper, we describe the Fabric-Based System model that is a **generalized version of the fabric model** presented in [2], [1]; a toolset, the Fabric Generator [3], to create fabrics; and applications that demonstrate the performance of FBS.

Fabric-based architectures have been popular since the invention of cellular automata. They are attractive because small, localized cells with small degree interconnect are efficiently implemented in VLSI or Programmable Logic Devices (PLDs), and for some application classes, exceed conventional processors' or DSPs' performance by orders of magnitude. Recent fabric-based architecture proposals include [4], [5], [6], [7], [8], and [9].

Our architecture design has many similarities to these proposals. The novel aspects of our approach are that in our parameterized cellular architecture, the cell, the interconnect, and the memory architecture all can be customized to the application. In addition, an embedded processor is an integral part of our model. The processor loads data, synchronizes

the cells, and can inspect and retrieve data from the cells' memories.

The Fabric Generator (FG) is similar to JHDL[10], LDG[11], PamDC[12], and other CAD tools embedded in high level programming languages. With these tools, it is possible to write a high level language program to describe, instantiate, and interconnect hardware modules. JHDL and PamDC, by using the overloading features of Java and C++, also provide simulation capability, which our system does not yet include.

In contrast to these tools, FG uses a built-in programming model of the Fabric-Based System, and automatically generates control signals associated with a cell's datapath. When given a microcode program for a specific datapath, FG generates a cell sequencer to control one or more cells. It also generates a complete fabric with interconnected cells and associated controllers. Unlike JHDL, FG generates Register-Transfer-Level VHDL. The present implementation is targeted to the Altera Excalibur part with Apex PLD.

II. FABRIC-BASED SYSTEM

A Fabric-Based System consists of a standard processor closely coupled to a computational fabric through a memory interface.

The computational fabric contains simple, inter-connected datapath cells, each with an optional local memory. The collection of local memories forms a (dual-ported) global memory that can be loaded and examined from the attached processor. The cells composing the fabric need not all have the same datapath. A fabric may contain groups of homogeneous cells and heterogenous cells as illustrated in Figure 1, in which different cell types are distinguished by different names ("Send," "Rec," "P," and "Ele").

Each datapath cell may have its own controller, or alternatively, a group of identical cells may share a controller. A controller can run many different programs so the same Fabric can be used to execute different applications using dynamic reconfiguration of the Fabric at the functional level. This can be accomplished in the single clock cycle.

Many different sorts of communications patterns may be realized within a single fabric.

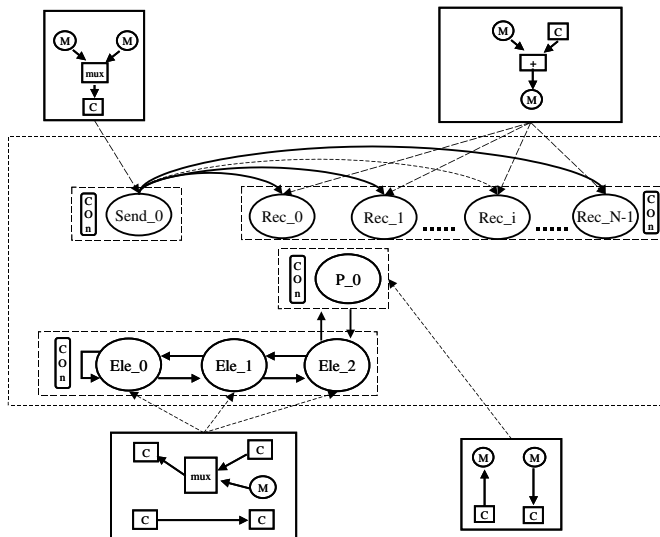


Fig. 1. Example Fabrics

III. FABRIC GENERATOR

In order to facilitate the generation of fabrics in a FBS, we have created a Java-based fabric generator library (Figure 2). The FG library contains classes to: define a module, create a datapath of interconnected modules, instantiate cells consisting of datapaths with associated sequencers and create a fabric of interconnected cells.

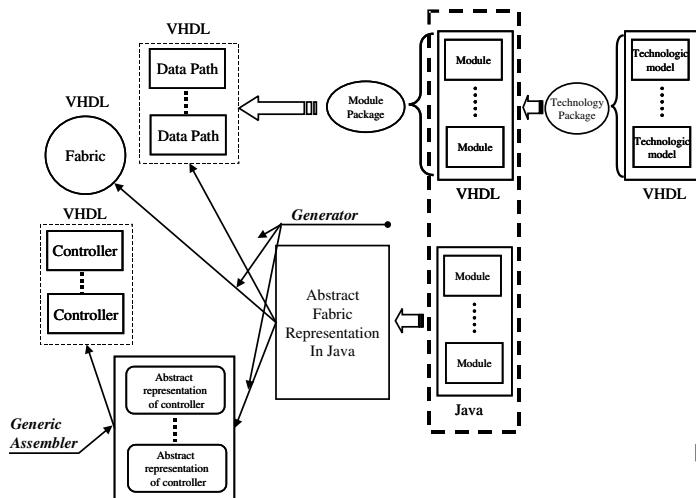


Fig. 2. Fabric Generator

As cells are instantiated, information is automatically collected about the control signals associated with each datapath and a separate file is created. The designer fills into this file the microcode sequence for a datapath and then runs an assembler, which generates the state machine to control the datapath.

The instructions in a microcode program specify the cell's control flow and control signals to the datapath. Control

flow specifications include branching (conditional or unconditional), loop instructions, and synchronization with the processor. Loops can be nested to arbitrary depth. A final field of the instruction defines communication between a cell and its neighbor(s).

The combination of datapath cells and sequencers is then synthesized by standard logic synthesis tools. The result is a component with a standard memory interface which can be connected to any processor. In our case the Fabric was connected to ARM processor inside an Altera Excalibur ARM.

IV. APPLICATIONS

For this paper, performance results are given of two examples of applications mapped to an FBS using the FG toolset: an unsupervised clustering algorithm [13], and a matched filter. Results are for an Excalibur ARM at 33 MHz. The clustering algorithm uses 95% of the ESB blocks and 60% of the logic elements. This fabric can compute 150 classes, and operates at 4.5 Gops/s, where the operation is defined as before (8-bit abs/accumulate). The matched filter uses 88% of the ESB blocks, 73% of the logic elements and can compute 140 Matched Filters, at a rate of 4.5 GMACs/s.

REFERENCES

- [1] C. Wolinski, M. Gokhale, and K. McCabe, "A new polymorphous computing fabric," *IEEE Micro*, Sept. 2002.
- [2] —, "A reconfigurable computing fabric," *ERSA 2002*, June 2002.
- [3] —, "Rapid construction of reconfigurable computing fabrics for systems on a programmable chip," *HPCA-9/SSRS*, 2003.
- [4] J. Vuillemin, P. Bertin, *et al.*, "Programmable active memories: Reconfigurable systems come of age," *IEEE Transactions on VLSI Systems*, vol. 4, no. 1, pp. 56–69, Mar. 1996.
- [5] T. Miyamori, "A quantitative analysis of reconfigurable coprocessors for multimedia applications," *IEEE Symposium on FPGAs for Custom Computing Machines*, Apr. 1998.
- [6] C. E. D. C. Green and P. Franklin, "RaPiD – reconfigurable pipelined datapath," in *Field-Programmable Logic: Smart Applications, New Paradigms, and Compilers. 6th International Workshop on Field-Programmable Logic and Applications*, R. W. Hartenstein and M. Glesner, Eds. Darmstadt, Germany: Springer-Verlag, Sept. 1996, pp. 126–135.
- [7] J. R. Hauser and J. Wawrzynek, "GARP: A MIPS processor with a reconfigurable coprocessor," in *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, J. Arnold and K. L. Pocek, Eds., Napa, CA, Apr. 1997.
- [8] E. Waingold, M. Taylor, *et al.*, "Baring it all to software: raw machines," *IEEE Computer*, pp. 86–93, sep 1997.
- [9] V. Baumgarte, F. May, *et al.*, "Pact xpp - a self-reconfigurable data processing architecture," *International Conference on Engineering of Reconfigurable Systems and Algorithms*, June 2001.
- [10] B. Hutchins *et al.*, "Jhdl-an hdl for reconfigurable systems," *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 1998.
- [11] M. Gokhale, A. Kopser, S. Lucas, and R. Minnich, "The logic description generator," *International Conference on Application Specific Array Processors*, Sept. 1990.
- [12] Compaq, "Pamdc," research.compaq.com/SRC/pamette/PamDC.pdf, 1999.
- [13] M. Gokhale, J. Frigo, K. McCabe, J. Theiler, C. Wolinski, and L. Dominique, "Experience with a hybrid processor: K-means clustering," *Journal of Supercomputing*, Vol 24, No. 4, 2003.