Dynamic Hardware Plugins in an FPGA with Partial Run-time Reconfiguration

Edson L. Horta,* John W. Lockwood,† David Parlour
Universidade de San Pãulo, Escola Politécnica - LSI, San Pãulo, SP, Brazil, edson-horta@ieee.org
David E. Taylor, Applied Research Lab, Washington University, Saint Louis, MO 63130, {lockwood,det3}@arl.wustl.edu
Xilinx, Inc, 2100 Logic Drive, San Jose, CA 95124, dave.parlour@xilinx.com
http://www.arl.wustl.edu/arl/projects/fpx/

ABSTRACT

Tools and a design methodology have been developed to support partial run-time reconfiguration of FPGA logic on the Field Programmable Port Extender. High-speed Internet packet processing circuits on this platform are implemented as Dynamic Hardware Plugin (DHP) modules that fit within a specific region of an FPGA device. The PARBIT tool has been developed to transform and restructure bitfiles created by standard computer aided design tools into partial bitsteams that program DHPs. The methodology allows the platform to hot-swap application-specific DHP modules without disturbing the operation of the rest of the system.

Keywords

FPGA, partial RTR, reconfiguration, hardware, modularity, network, routing, packet, Internet, IP, platform computing

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Circuits—*Design Aids*; B.7.1 [**Hardware**]: Circuits—*VLSI*; B.4.3 [**Hardware**]: Input/Output and Data Communications—*Interconnections (Subsystems)*; C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Design, Experimentation

*Research is supported by CNPq (Brazil)

†Research is supported by NSF: ANI-0096052 and Xilinx

DAC 2002 June 10-14, 2002, New Orleans, Louisiana, USA

1. INTRODUCTION

FPGAs are frequently used in networking applications, where they offer both the performance of custom hardware and the flexibility of reprogrammability [1] [2]. Systems implemented with FPGAs can make use of their reprogrammability in one of two ways: Compile-Time Reconfiguration (CTR) or Run-Time Reconfiguration (RTR) [3]. CTR systems do not change the FPGA's configuration for the lifetime of the application, e.g. SPLASH [4] and PAM [5]. RTR systems change the FPGA configuration during the course of operation, either by full reconfiguration [6] [7] or partial reconfiguration [8] [9] [10].

The present research uses partial RTR in an FPGA to provide developers of hardware packet processing applications a capability similar to the Dynamically Linked Libraries (DLLs) which are used in software applications. Just as a DLL is a software module that can be attached to or removed from a running program as an application demands, we define a Dynamic Hardware Plugin (DHP) [11] as a module which can be loaded into or removed from a running FPGA without disturbing other circuits operating in it. The ability to change the hardware feature set in a running system is particularly useful in packet processing applications such as firewalls and routers, where it is not desirable to suspend the operation of a network during reprogramming. A practical system for implementing DHPs includes:

- A suitable FPGA test platform
- A well defined DHP interface specification
- A complete DHP design methodology
- Physical implementation tools (place & route)
- Configuration bitstream management tools

These five elements are analogous to an operating system platform, Application Programming Interface (API), modular programming methodology, compiler, and linker needed to implement DLLs in the software domain.

The Field Programmable Port Extender (FPX) [12] is the FPGA-based prototyping platform used in the Washington University Gigabit Switch (WUGS) [13]. Section 2 describes the well-defined DHP interfaces, the design methodology, the CAD tools which place and route DHP modules, and the PARBIT tool that manages bitstreams. The FPX is described in Section 3, along with an application targeted to this platform.

2. PARTIAL RECONFIGURATION

Partial reconfiguration allows an FPGA to implement multiple functions and to change those functions while the system is running. The target FPGA is logically partitioned into a static infrastructure region and a number of DHP sites. The infrastructure connects each DHP site to shared resources and/or other DHP sites. In the present system the FPGA Input/Output resources are owned by the infrastructure and all DHP sites are identical, although different design choices might be appropriate in other applications.

The layout and interface specification for the DHPs are influenced by both the architecture of the FPGA and its behavior during reconfiguration. The characteristics of the Xilinx VIRTEX-E family are described below. Following this description, the tool used to generate the partial configuration bitfile is presented, along with the requirements for the interface between the static infrastructure and the DHP sites, and the methodology to generate the configuration bitfiles used by PARBIT.

2.1 VIRTEX-E Architecture

The application presented in this paper targets a device from the Xilinx Virtex-E family [14]. Programmable Input/Output Blocks (IOBs) around the edge of the array are used to interface to off-chip resources. The interior consists of a matrix of Configurable Logic Blocks (CLBs) containing: lookup tables, flip-flops and programmable interconnect. The lookup tables in the CLB can be used as function generators, small distributed RAMs or programmable-length shift registers. A number of columns in the CLB matrix are replaced with Block SelectRAMs, which are dedicated dual-ported memories. A column of clock drivers used for global clock distribution run vertically though the center of the chip.

Virtex-E configuration bits are organized in columns corresponding to a column of the FPGA's logic resources [15]. The Center Column controls the global clock pins. The IOB Columns control the configuration for the left and right side IOBs. Each CLB Column controls one column of CLBs and two IOBs above and bellow these CLBs. Each column has n rows, with one CLB per row. The Block SelectRAM Interconnect Columns define the interconnection of each RAM column. The Block SelectRAM Content Columns define the contents of each RAM column.

To configure the VIRTEX-E FPGA, a series of bits, divided into fields of commands and data, are loaded into the device. Each one of the configuration columns are divided in smaller slices, called frames. A frame is the smallest part of the configuration memory that can be written or read.

2.2 PARBIT

In order to partially reconfigure a FPGA, it is necessary to isolate a specific area inside the device and download the configuration bits related to that area. A tool called PARBIT [16] has been developed to easily transform and restructure bitfiles to implement dynamically loadable hardware modules.

To restructure the configuration bitfile, the tool utilizes the original bitfile, a target bitfile, and parameters given by the user. These parameters include the block coordinates of the logic implemented on a source FPGA, the coordinates of the area for a partially programmed target FPGA, and the programming options. The tool reads the original configuration bitfile and copies to the partial bitfile only the configuration bits related to the area defined by the user.

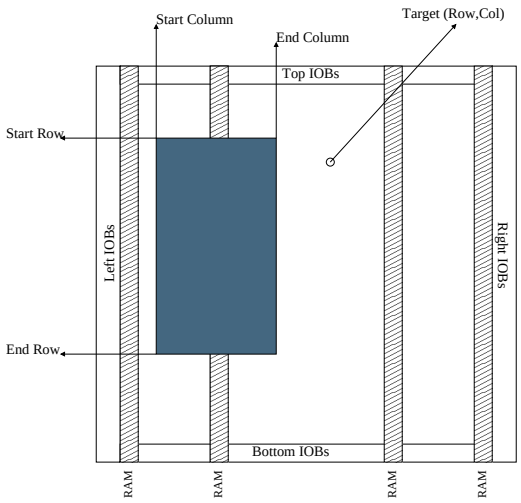

Figure 1: Block of logic selected from original bitfile

The target bitfile is used by PARBIT to copy the configuration bits that are inside a column specified by the user, but outside the partial reconfigurable area. This happens due to the fact that one frame takes all the rows of a column and the partial reconfigurable area is smaller than a whole column.

PARBIT allows arbitrary block regions of a compiled design to be re-targeted into any similar size region of an FPGA. It is possible to define an area inside the CLB columns of the chip, without the top and bottom IOB configuration bits. The tool generates the partial bitfile containing the area selected by the user (from the original bitfile) and this file will be used to reconfigure the FPGA.

To relocate blocks from the original bitfile, a user defines the start and end columns and rows for the block in the original design. Then, the user defines where to put this block into a target bitfile of the same device type. The tool then generates the partial bitfile containing the area selected by the user (from the original bitfile). This data is used to reconfigure the target device. A sample of a block within a VirtexE FPGA is shown in Figure 1.

It is important to note that the configuration bits for the top and bottom IOBs from the target device do not change after the partial bitfile is loaded. The configuration bits for the columns from the original and target bitfile are merged according to the rows defined by the user.

2.3 Gasket Interface

DHP modules that are downloaded into the FPX need fixed interconnection points to communicate with the infrastructure logic on the FPGA. These points are connected by special wires, called antennas.

Gaskets allow DHP modules to be built with only a minor modification to the standard FPGA design flow. When building a DHP, PAR (Place And Route) is made to limit routing to only the *included routes*. It may not use the *excluded routes*. By placing antennas between the DHP and the infrastructure region, the routing of signals through the gasket is fixed. Figure 2 shows how antennas cross out of the *DHP free routing zone*.

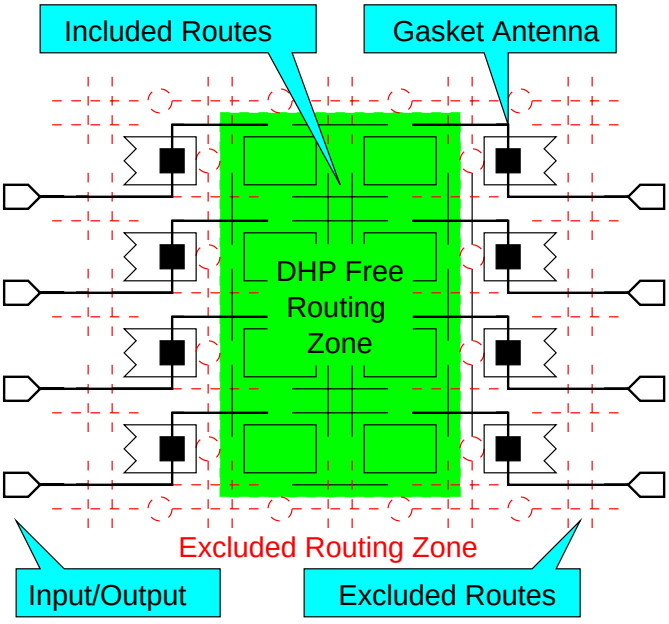

Figure 2: Gasket Antennas

2.4 Generation of Bitfiles

In order to generate bitfiles that can be read by PARBIT to create the partial bitfile, it is necessary to follow a few high-level rules regarding the logical names of the entities that make up the logic within the FPGA. These rules are applied during the synthesis, routing, and placement of the FPGAs that hold the infrastructure circuit and the DHP modules.

2.4.1 Synthesis

The bitfiles that are manipulated by PARBIT are generated from VHDL files containing the infrastructure and the user module descriptions (plugins), along with constraints commands used to define the regions of the FPGA that will be reconfigured.

The infrastructure bitfile contains the fixed logic area in the FPGA, encompassing all of the I/O pads, signals and flops that interface to the module, and logic that make up the on-chip system. In the VHDL file, there is one entity, called "INFRA", that contains the infrastructure, and one or more entities, called "GASKET", that contains the flops used to interface with each user module. Additional VHDL files are used to generate the physical constraints necessary to lock the gasket flops in fixed positions inside the FPGA. These constraints are also used to lock the infrastructure logic into a specific area, and to reserve space for the user modules inside the FPGA.

The bitfile for the DHP contains the description of a module and is generated in a similar way. The difference is that there is one user module entity, called "DHP", connected to one GASKET entity, as shown in Figure 3. For the GASKET entity in the design, the constraints are set so that this GASKET is placed in the same position as the first GASKET of the infrastructure design.

2.4.2 Placement

The placement of the infrastructure and DHPs can be accomplished with conventional FPGA placer tools, that have the ability to constrain logic to specific regions of the array. Infrastructure logic is kept out of the DHP sites, and DHP logic is confined to the appropriate rectangular area. These constraints can be expressed in the source VHDL as attributes or kept in a separate physical constraint file read by the placer.

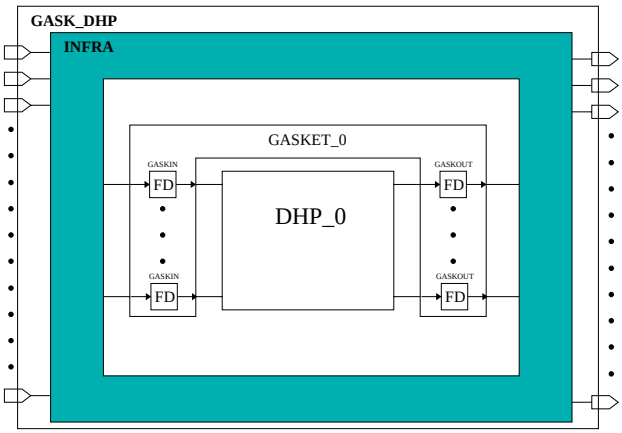

Figure 3: Logical Design Entities

2.4.3 Routing

The routing problem is more complex than placement, and requires a greater degree of control over the use of routing resources than is currently available in off-the-shelf FPGA CAD tools. To guarantee that there is no interference between DHPs and infrastructure as new DHPs are being configured, the nets in a design are sorted into one of three categories and then routed with special constraints as follows:

- All nets which are internal to the DHPs are routed on interconnect resources that do not cross the boundary of the DHP site. The Virtex-E architecture includes long lines which traverse the entire array, and hex lines which can reach six CLBs in any direction. These wires may not be used in a DHP since they could possibly cause interference when the DHP is loaded
- All nets which are completely contained in the infrastructure are routed using any wiring resource that is not used for use in routing DHPs.
- Nets which cross the the boundaries between the infrastructure and the DHP site are forced to follow the same path for each DHP. These are the gasket antennas which are identical for every DHP, since the DHP interface conforms to a fixed specification. These nets have identical configurations in every possible DHP implementation so the act of reconfiguration is guaranteed to cause no interference in the configuration or operation of the infrastructure or adjacent DHP sites

These routing constraints are satisfied using a modified version of the router in which the use of any individual Programmable Interconnect Point (PIP) in the array can be enabled or disabled. Individual nets can also have wire segments assigned to force the route solution to follow a predictable path as needed.

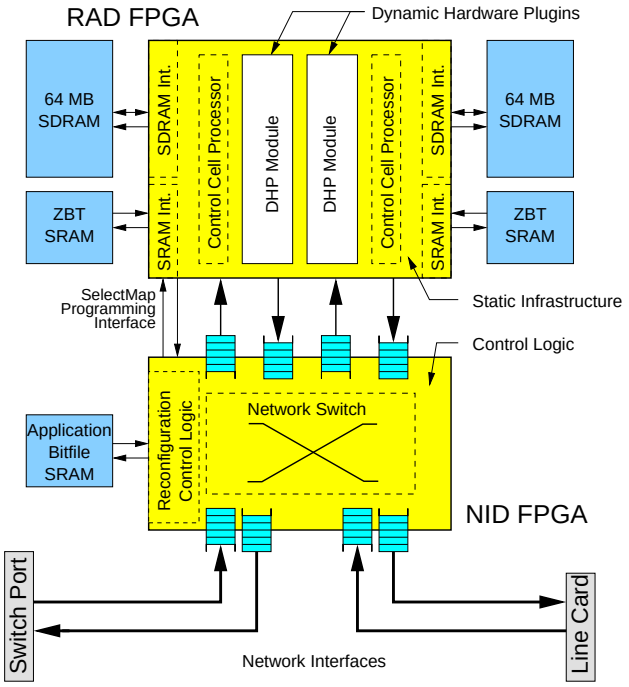

Figure 4: FPX System

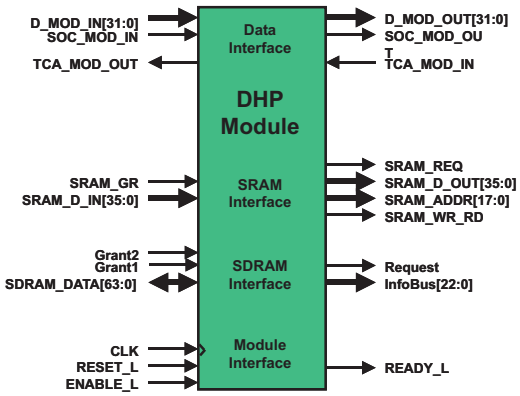

Figure 5: Modular Component of FPX

3. THE FPX PLATFORM

The Field-programmable Port Extender enables the rapid prototype and deployment of hardware components for modern routers and firewalls [12]. The system allows new packet processing functions to be quickly prototyped as DHP modules in hardware, then downloaded into reconfigurable logic over the network [17]. New features are added by loading DHPs into well-defined DHP module interfaces.

All functions on the FPX are implemented with FPGAs. The core functionality of the FPX is implemented on the Networking Interface Device (NID) and on part of the Reprogrammable Application Device (RAD). The NID is a Xilinx XCV600E FPGA that contains the control logic to reconfigure regions of the RAD. The RAD is a Xilinx XCV-2000E FPGA that holds the DHP modules.

In order to reprogram a RAD module, the NID implements a reliable protocol to fill the contents of the an SRAM with configuration data that is sent over the network. A final control cell is sent to NID to initiate the reprogramming of RAD using the contents of the reprogram memory [18].

The NID also contains a network switch that forwards individual traffic flows between network interfaces and DHP modules on the RAD. This combination of partial reconfiguration control logic and per-flow routing circuits allow the FPX install new DHP modules without affecting the operation of the rest of the system.

3.1 Modular Logic

Dynamic Hardware Plugins (DHPs) are used to implement application-specific functionality on the FPX. Multiple DHPs can be loaded into the RAD and run in parallel on the single FPGA device. Data flows may pass through multiple hardware plugins. In order to support a broad spectrum of applications, DHPs can access off-chip memory resources.

Hardware plugin modules on the RAD consist of a region of FPGA gates and internal memory, bounded by a well-defined interface to the network and external memory [19]. The modular interface of an FPX component is shown in Figure 5. Data arrives at and departs from a module over a 32-bit wide, Utopia interface. Data passes through modules in a cell. Large IP datagrams pass through the interface in multiple cells.

The module provides two interfaces to off-chip memory. The SRAM interface supports transfer of 36-bit wide data to and from off-chip SRAM. The Synchronous Dynamic RAM (SDRAM) interface provides a 64-bit wide interface to off-chip memory. In the implementation of the IP lookup module, the off-chip SRAM is used to store the data structures of an Internet route table [12].

3.2 DHP Implementation on the FPX

Figure 6 shows the infrastructure of the RAD when viewed by the Xilinx FPGA Editor. The logic for the CCP (Control Cell Processor), the SDRAM controller, and the SRAM interface is placed and routed in the left and the right sides of the chip. The center portion of the chip is reserved area for two hardware plugins modules.

The floorplan for the FPGA circuit that implements a DHP is shown on Figure 7. Here, the logic for the module is confined to a region the size of a DHP. Further, the I/O signals around the DHP include Flip/Flops that lock the location of the signals and provide timing isolation between DHP circuits. Inputs and outputs from the DHP are routed to the external I/O pins of the FPGA so that standard synthesis tools can be used for their implementation.

After generating these two bitfiles, PARBIT is run to generate a partial bitfile suitable for run-time reconfiguration. The parameters passed to PARBIT include the coordinates of the DHP module and the coordinate of the new position of this module inside the FPGA. The partial bitfile generated for a module has a size of 187 KBytes as compared to the complete XCV2000E configuration size of 1,270 KBytes.

Current efforts focus on the implementation of an enhanced on-chip gasket interfaces for additional types of signals. Some types of on-chip shared resources require tri-state busses that span the width of the device. By enhancing the gasket interface to support the use of horizontal long lines, it is possible to efficiently implement this type of interconnect.

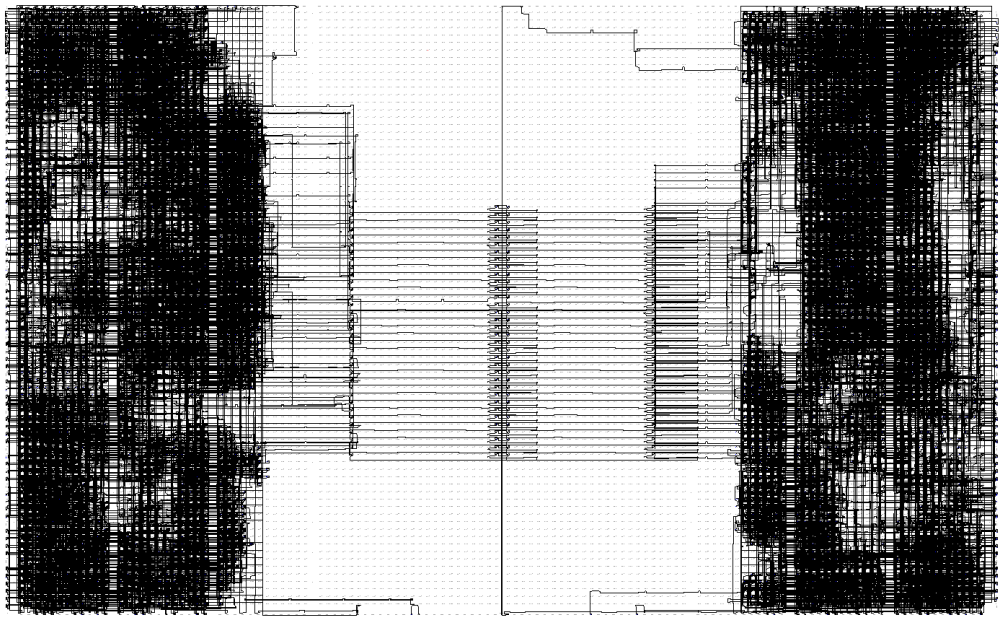

Figure 6: Infrastructure Floorplan of the RAD (A Xilinx Virtex 2000E FPGA). Note that Infrastructure logic, including the pair of SRAM controllers, the pair of SDRAM controllers, and the Control Cell Processor (CCP) are routed and placed on the left and right sides of the FPGA. Two regions have been reserved for DHP modules and are left empty when synthesizing the infrastructure.

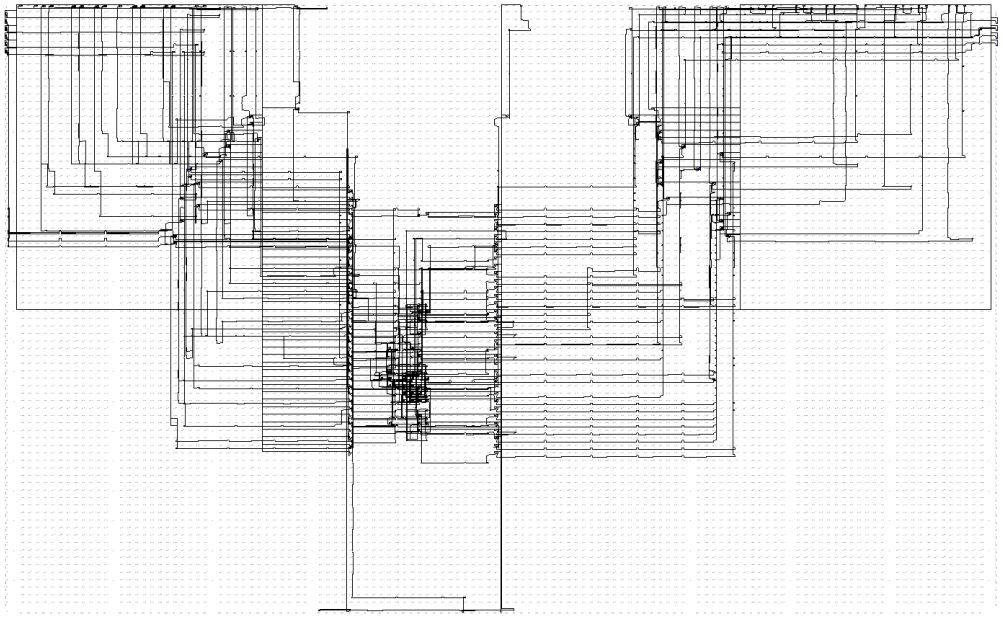

Figure 7: DHP Floorplan of the RAD FPGA. A single module has been routed and placed to fit within the area of one DHP module. Using PARBIT, this DHP can be placed into either location of the Infrastructure shown in the floorplan above. Note that input/output signals are routed to I/O pins so that standard design flows for synthesis and simulation of the FPGA circuit can be followed.

4. CONCLUSIONS

A technique has been demonstrated for designing partial RTR systems in a VIRTEX-E FPGA. The methodology uses PARBIT to generate the partial bitfiles. This tool can extract and relocate regions of logic in a compiled FPGA.

The interface between the DHP plugins and the infrastructure, called a gasket, is able to lock fixed interconnection points between infrastructure logic and dynamically reconfigured regions of an FPGA.

The approach shown in this paper reduces the time needed to download the new hardware module, due to the fact that these modules are pre-compiled and that the reconfiguration of DHP modules is performed in hardware (not software). This feature enables the implementation of network modules that perform a variety of networking functions and can be deployed at run time in a networking router, switch, or firewall.

5. FUTURE PLANS

Going forward, we are collaborating with application developers and network equipment vendors to design reconfigurable hardware into several routing, switching, and firewall platforms. As the installed base of reconfigurable platforms grows, it becomes easier to rapidly deploy new features in DHP modules. By following a standard and open modular interface, DHPs modules are fully interoperable.

6. ACKNOWLEDGMENTS

The authors would like to thank Richard Y. Sun and Robert Wells, from Xilinx, Inc., for their assistance in solving the routing problems associated with DHP implementation.

7. REFERENCES

[1] S. Hauck, "The roles of FPGAs in reprogrammable systems," *Proceedings of the IEEE*, vol. 86, pp. 615–638, Apr. 1998.

[2] W. Marcus, I. Hadzic, A. McAuley, and J. Smith, "Protocol boosters: Applying programmability to network infrastructures," *IEEE Communications Magazine*, vol. 36, no. 10, pp. 79–83, 1998.

[3] B. L. Hutchings and M. J. Wirthlin, "Implementation approaches for reconfigurable logic applications," in *Field-Programmable Logic and Applications (FPL'1995)* (W. Moore and W. Luk, eds.), (Oxford, England), pp. 419–428, Springer-Verlag, Berlin, Aug. 1995.

[4] D. T. Hoang, "Searching genetic databases on splash 2," in *IEEE Workshop on FPGAs for Custom Computing Machines* (D. A. Buell and K. L. Pocek, eds.), (Los Alamitos, CA), pp. 185–191, IEEE Computer Society Press, 1993.

[5] P. Bertin, H. Touati, and E. Lagnese, "PAM programming environments: Practice and experience," in *IEEE Workshop on FPGAs for Custom Computing Machines* (D. A. Buell and K. L. Pocek, eds.), (Los Alamitos, CA), pp. 133–138, IEEE Computer Society Press, 1994.

[6] J. M. Ditmar, "A Dynamically Reconfigurable FPGA-based Content Addressable Memory for IP Characterization," Master's thesis, KTH- Royal Institute of Technology, Stockholm, Sweden, 2000.

[7] D. Ross, O. Vellacott, and M. Turner, "An FPGA-based Hardware Accelerator for Image Processing," in *More FPGAs: Proceedings of the 1993 International workshop on field-programmable logic and applications* (W. Moore and W. Luk, eds.), (Oxford, England), pp. 299–306, 1993.

[8] J. D. Hadley and B. L. Hutchings, "Designing a partially reconfigured system," in *Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing, Proc. SPIE 2607* (J. Schewel, ed.), (Bellingham, WA), pp. 210–220, SPIE – The International Society for Optical Engineering, 1995.

[9] S. McMillan and S. Guccione, "Partial run-time reconfiguration using JRTR," in *Field-Programmable Logic and Applications / The Roadmap to Reconfigurable Computing (FPL'2000)*, (Villach, Austria), pp. 352–360, Aug. 2000.

[10] E. L. Horta and S. T. Kofuji, "The architecture of a reconfigurable ATM switch (RECATS)," in *Workshop de Computação Reconfigurável*, (Marília, SP, Brazil), Aug. 2000.

[11] D. E. Taylor, J. S. Turner, and J. W. Lockwood, "Dynamic Hardware Plugins (DHP): Exploiting reconfigurable hardware for high-performance programmable routers," in *IEEE OPENARCH 2001: 4th IEEE Conference on Open Architectures and Network Programming*, (Anchorage, AK), Apr. 2001.

[12] J. W. Lockwood, J. S. Turner, and D. E. Taylor, "Field programmable port extender (FPX) for distributed routing and queuing," in *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2000)*, (Monterey, CA, USA), pp. 137–144, Feb. 2000.

[13] S. Choi, J. Dehart, R. Keller, J. W. Lockwood, J. Turner, and T. Wolf, "Design of a flexible open platform for high performance active networks," in *Allerton Conference*, (Champaign, IL), 1999.

[14] Xilinx Inc., "Virtex-E 1.8V Field Programmable Gate Arrays." Xilinx DS022, 2001.

[15] S. Kelem, "Virtex configuration architecture advanced user's guide." Xilinx XAPP151, Sept. 1999.

[16] E. Horta and J. W. Lockwood, "PARBIT: a tool to transform bitfiles to implement partial reconfiguration of field programmable gate arrays (FPGAs)," Tech. Rep. WUCS-01-13, Washington University in Saint Louis, Department of Computer Science, July 6, 2001.

[17] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor, "Reprogrammable Network Packet Processing on the Field Programmable Port Extender (FPX)," in *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2001)*, (Monterey, CA, USA), pp. 87–93, Feb. 2001.

[18] J. W. Lockwood, "Evolvable internet hardware platforms," in *NASA/DoD Workshop on Evolvable Hardware (EH'2001)*, pp. 271–279, July 2001.

[19] D. E. Taylor, J. W. Lockwood, and N. Naufel, "Generalized RAD Module Interface Specification of the Field-programmable Port eXtender (FPX)," tech. rep., WUCS-01-15, Washington University, Department of Computer Science, July 2001.