Making Wide-Issue VLIW Processors Viable on FPGAs

MADHURA PURNAPRAJNA and PAOLO IENNE, Ecole Polytechnique Fédérale de Lausanne

Soft and highly-customized processors are emerging as a common way to efficiently control large amount of computing resources available on FPGAs. Yet, some processor architectures of choice for DSP and media applications, such as wide-issue VLIW processors, remain impractical: the multi-ported register file makes a very inefficient use of the resources in the FPGA fabric. This paper proposes modifications to existing FPGAs to make soft-VLIW processor viable. We introduce an embedded multi-ported RAM that can be customized to match the issue-width of VLIW processors. To ascertain the benefits of this approach, we map an extensible VLIW processor onto a standard FPGA from Xilinx. For the register file implemented in the modified FPGA, the area occupied is at least 102× smaller and the dynamic power is reduced by 41% as compared to the implementation using configurable logic blocks in existing standard FPGAs. A subset of this embedded multi-ported RAM can also be used for mapping the register file in soft-RISC processors. For the soft-RISC processor, the register file in the modified FPGA is at least 22× smaller than its equivalent that uses configurable logic blocks and 1.5× the size in comparison to the implementation using block RAMs. Reduction of routing area and the maximum net length is about 39% and 51% respectively for RISC processors. As a result, this approach works towards enhancing soft-processor density in FPGAs by orders of magnitude.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays*; C.1.1 [**Processor Architectures**]: Single Data Stream Architecture—*RISC/CISC, VLIW Architectures*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Soft-processor, resource efficiency, multi-port memory

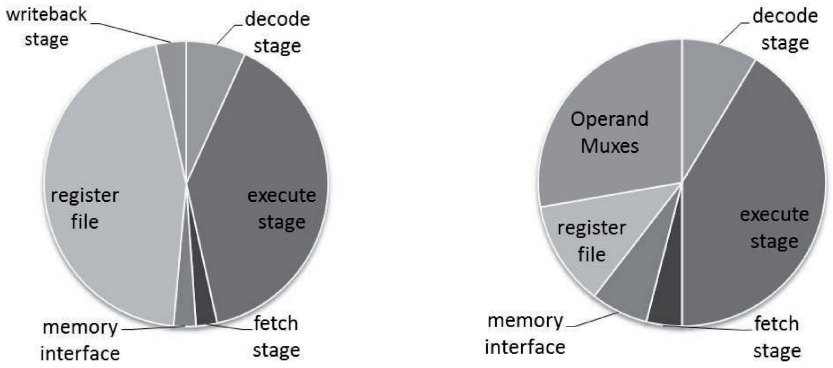
ACM Reference Format:
Purnaprajna, M. and Ienne, P. 2012. Making wide-issue VLIW processors viable on FPGAs. ACM Trans. Architec. Code Optim. 8, 4, Article 33 (January 2012), 16 pages.
DOI = 10.1145/2086696.2086712 http://doi.acm.org/10.1145/2086696.2086712

1. INTRODUCTION AND MOTIVATION

State-of-the-art FPGAs have densities that can easily accommodate multiple instances of soft-processors such as RISC-based Microblaze [Xilinx 2011a] and Nios II [Altera 2011]. The advantage with soft-processors is the ease of portability across FPGA families and the ability to software-program FPGAs, which is in contrast to the tedious *hardware description language* (HDL) based programming model. Presently, most of the existing soft-cores are RISC-based processors, typically used only for the system bring up. Nevertheless, for computationally intensive applications, specialised processors

This work was supported by the Swiss National Science Foundation, under Marie Heim-Vögtlin programme, grant application 134114.
Authors' address: M. Purnaprajna and P. Ienne, Ecole Polytechnique Fédérale de Lausanne (EPFL), School of Computer and Communication Sciences, CH-1015 Lausanne, Switzerland; email: madhura.purnaprajna@epfl.ch.

DOI 10.1145/2086696.2086712 http://doi.acm.org/10.1145/2086696.2086712



(a) VLIW (b) RISC

Fig. 1. Area distribution of building blocks within a processor.

or dedicated co-processors are employed to meet domain-specific performance requirements. VLIW processors have been quite popular in a wide range of applications [Fisher et al. 2009]. For example in the DSP domain, the key players, viz., Texas Instruments, NXP Semiconductors, Ceva, Inc., etc. have processors with a VLIW architecture, which have proven to be very successful commercially. Popular configurable processors used for DSP applications (such as Tensilica, Inc. [2011], Silicon Hive [Burns et al. 2006]) also have a VLIW architecture. The advantage of using VLIW processors is that they are purely software programmable and provide automated compiler-driven programmability. Increasingly, FPGAs are also being used for DSP applications since they are very flexible and provide extensive parallelism. In addition, FPGAs have also progressively offered support to accelerate DSP applications with dedicated blocks (such as multiply-accumulators, memory, etc.). However, the most important drawback in using FPGAs for DSP applications is the need for an HDL-based design flow in contrast to widely accepted C-based (or Matlab-based) software design flow in DSP application programming. Transforming algorithms primarily developed for sequential programming in processors to explore the extensive parallelism in FPGAs may be non-trivial. As an alternative, introducing soft-VLIW processors within the FPGA domain provides the benefit of using the commercially successful, yet easily programmable platform that bypasses the time consuming HDL-based design flow. Introducing soft-VLIW processors onto FPGAs provides the possibility to configure the issue-width to suit the parallelism exhibited by the application. This architectural possibility has higher advantages as compared to the presently supported RISC-based soft cores with limited parallelism. Additionally, application-specific hardware accelerators can be coupled to the VLIW processor, which can be accessible via instruction-set extensions. The automated compilation flow in VLIW processors manages application mapping. Whereas, with RISC processors that are presently supported by FPGA vendors such as Xilinx and Altera, requires manual application partitioning and communication management. FPGAs with VLIW support can provide application acceleration with a significant reduction in the design time.

In this paper, our objective is to benefit from the ease of programmability in processors along with the extensive parallelism available in FPGAs. To achieve this objective, our technique is to map applications onto soft-VLIW processors instantiated on FPGAs, as an alternative to the time-consuming HDL-based flow. The biggest barrier for introducing VLIW processors onto current FPGAs is the inefficiency in its implementation using the configurable logic blocks. Figure 1 shows the area distribution of the

individual building blocks of a soft-VLIW processor [van As 2008] and a soft-RISC [OpenCores] processor implemented on an FPGA from Xilinx. As can be seen, the implementation of the VLIW processor is expensive in terms of area. Figure 1(a) shows that the register file of the VLIW processor requires nearly 50% of the total slices occupied by the processor. This is in contrast to RISC processors, where the register file only contributes to about 12% of the total area, shown in Figure 1(b). In this paper our objective is to suggest modifications to existing commercial FPGA fabrics to make VLIW processors area and power efficient. Modifications to the FPGA are introduced primarily with VLIW processors in mind, but prove to be applicable for use even in RISC-based processors.

Our contributions in this paper are the following.

(1) Mitigate the performance bottleneck in soft-VLIW processors by introducing an embedded multi-ported RAM for register file implementation.
(2) Use a subset of the embedded multi-ported RAM as register files for VLIW processors with smaller issue width and single-issue RISC processors.

The rest of the paper is organized as follows. Section 2 discusses related work in this area of mapping VLIW processors onto FPGAs and techniques used for optimizing the implementation of multi-ported register files. Section 3 presents the main idea of this paper, which is to design processor-centric FPGAs. Area and I/O bandwidth comparisons for existing memory blocks in current day FPGAs and the proposed Embedded Multi-ported RAM are provided in the same section. Based on these comparative figures a new modified FPGA layout for processor-centric FPGAs is presented. Section 4 discusses the experimental setup and the advantages of introducing the proposed Embedded Multi-ported RAM for use in a soft-VLIW processor and its applicability in soft-RISC processor. Finally, Section 5 presents the conclusions and future work in this direction.

2. RELATED WORK

FPGAs provide extensive parallelism. The parallelism can be explored to enhance application performance by the use of dedicated hardware accelerators, instantiating multiple soft-processors as can be accommodated by the FPGA, or by modifying the soft-processor to support higher level of parallelism. Designing dedicated hardware accelerators requires a HDL-based design flow, which is time consuming and delays the time-to-market. In addition, to use applications that have an entirely software-based design flow requires translating a software-based design flow into HDL-flow for mapping them onto FPGAs. Instead, instantiating multiple soft-processors avoids the HDL-based flow entirely. However, only RISC-based soft processors (such as Microblaze and NIOS) are presently supported by the FPGA vendors. Using a RISC-based multiprocessor system has the tedious overhead of application partitioning and managing inter-processor communication and synchronization. An alternative to existing single-issue soft RISC processors is to consider soft processors that support a higher degree of parallelism, such as multi-issue VLIW [van As 2008] or vector processors [Yu et al. 2008]. In soft-VLIW processors the number of functional units can be chosen to match instruction-level parallelism in the application. The logic density in FPGAs makes it possible to accommodate a large number of functional units and/or multiple instances of the multi-issue VLIW processor. Using an FPGA-based VLIW processor [Jones et al. 2005], performance speedups of up to 30× and an average of 12× have been observed for benchmarks from MediaBench. However, the single most important performance bottleneck in the scalability of VLIW processors lies in implementation of the multi-port register file.

Register files can be mapped onto block RAMs or the distributed memory present in the *configurable logic blocks* (CLBs). In current FPGAs, the maximum number of memory ports is limited to two read/write ports. Factors such as word length, size of register file, the number of read/write ports together make an impact on the size and latency of the register file [Jones et al. 2005]. Saghir and Naous [2007] present a modified multi-port register file design using block RAMs to implement register files with higher number of read/write ports than available. The implementation splits a given memory into multiple banks and replicates memory according to the number of read ports required. The organization of memory as banks with several write ports requires explicit measures for register allocation and instruction scheduling. Consequently, instructions cannot be executed in parallel if they modify registers from the same bank. This scheme of register file organization enforces a restriction on the VLIW compilation and execution schemes, which is undesirable. LaForest and Steffan [2010] present three other possibilities of implementing multi-ported RAMs in FPGAs. First is the replication method, where memory contents are replicated for every additional read port. The drawbacks of memory replication is the high memory usage that is proportional to the number of read ports and the limitation of only one write port. The second scheme is the banking method, where memory locations are divided into banks and each bank supports an additional read/write port. In this case, simultaneous access for the read/write ports is limited to its bank, which is not a true multi-ported RAM implementation. The third scheme, called multipumping, relies on clocking memories at higher rates to compensate for the higher number ports. The drawbacks of this scheme is that a higher clocking frequency that reduces the overall operating frequency and that a large amount of multiplexers are required to hold addresses and data for upcoming read and write operations. In the same paper LaForest and Steffan [2010], present an efficient scheme for implementing multi-ported memories in FPGAs. This scheme uses block RAMs and replicates memory for every read/write port. It also uses a *live value table* (LVT) to identify the location of the most recently used copy among the replicated memories. Using this scheme, the area and delay ratios are much improved as compared to the register-based implementation in FPGAs. However, the LVT scheme still has a performance difference in comparison to dedicated multi-ported RAMs. The performance difference for a 8-read/4-write (8R4W) ports for the delay ratio and area densities is about 12× and 24×, respectively. To further improve performance, our objective in this paper is to study the feasibility of introducing multi-ported RAMs in FPGAs as a first step towards to designing processor-centric FPGAs.

High performance processors are very inefficient on FPGAs. Wong et al. [2011] compare the CMOS and FPGA versions of general purpose processors and observe a performance gap of about 17-27× in area ratios and 18-28× in delay ratios. The significant drawback for implementing high performance and resource efficient processors is mainly because of the generality of the FPGA fabric and the absence of any processor-specific architectural feature in FPGAs. Techniques of accelerating soft-processors such as Warping [Lysecky and Vahid 2005] are parallel approaches to this paper. Alternative FPGA fabrics, such as Amorphous [Lin 2008] and Triptych [Borriello et al. 1995] have been proposed earlier. Amorphous [Lin 2008] attempts to allocate logic and routing resources on a per-mapping basis. Architectures that modify the granularity of the configurable logic blocks have also be explored [Hartenstein 2001]. These coarse-grained reconfigurable architectures still attempt to retain the generality and universal applicability of FPGA architecture, which translate to low application-specific performance. Our objective is to customise the FPGA architecture for processor-based design and thus enhance the performance of processors mapped onto FPGAs. By making soft-processor efficient, the usability of FPGAs is extended to a purely software-based design flow that can be widely used.

3. PROCESSOR-CENTRIC FPGA

Soft-processors are increasingly being mapped on to FPGAs on account of their flexibility, portability across FPGA families, and the ease-of-use programming model. As FPGAs provide large amount of resources, applications can benefit from concurrency available by packing multiple soft-processors onto the fabric. Specifically for DSP applications, the parallelism available in the application can be best explored by using processors that provide the same kind of concurrency, such as configurable VLIW processors. In addition, FPGAs also provide DSP-friendly blocks such as multiple-accumulate units, high-speed memories, etc. In addition to dedicated resources, hardware accelerators can be mapped onto the FPGA fabric to further improve performance. The generality of the FPGA's configurable logic fabric itself is very inefficient the mapping a VLIW processor.

Soft-processors are optimized to make efficient use of the FPGA fabric. The ALU can benefit with the presence of hard multipliers, barrel shifters, and carry chain logic. Depending on the size and port configurations, the local memories, caches, and register files can be mapped on the block RAMs or distributed RAMs. Despite the fact that certain parts of the processor structure is generic across architectures, the rest of the processor is mapped onto the configurable logic blocks which do not provide any processor-specific feature in current FPGAs. Consider the instruction decoding unit; the structure translates to a wide demultiplexer with the location of the operands specific to the instruction-set architecture. In the absence of any wide multiplexing or demultiplexing unit, the operation is performed by merging multiple look-up tables and multiplexers present in the configurable logic blocks. It has to be noted that even in latest Virtex 7 series of FPGAs from Xilinx [2011c], the largest multiplexer is a 4:1 multiplexer that is used to combine the outputs of the individual look-up tables within a configurable logic block. Similarly, the implementation of the shared register files in VLIW processors is the most expensive unit in terms of area utilization. Current state-of-the-art FPGAs support a maximum of two read/write ports in the dedicated memory blocks, which is only sufficient for a maximum of two address read/write operation. Even for RISC processors, which typically need three addresses (two for operand read and one for operand write), requires a two-read/one-write port configuration. This configuration requires replicating memory and using two independent block RAMs for simultaneous read/write access. For any configuration that requires a higher number of read/write ports, the register file has to be mapped onto the configurable logic, which has a lower operating frequency as compared to dedicated block RAMs. Consequently, the register file in a multi-issue VLIW processor has to be implemented using the configurable logic fabric, which results in large amounts of resource usage and lowered performance. The use of distributed configurable resources also translates sub-optimal resource utilization since fewer processors or fewer functional units can be mapped on to the device. With fewer processors that can be mapped, the possibility of building systems with a large number of processing units or processors becomes infeasible. This drawback works against the benefit that could have been achieved by merging the extensive parallelism in FPGAs with concurrent execution possible with VLIWs.

In current FPGAs, the granularity of logic resources provides fine-grained control over the architecture for mapping a wide variety of applications. This fine granularity requires elaborate synthesis and place & route tools to map a design to the FPGA. In contrast to this approach, the architecture of our processor-centric FPGA is fine-tuned to support soft-core based designs. Applications to be mapped onto the FPGA are compiled onto the VLIW processor, with the issue-width being adapted according to the application's parallelism. The FPGA fabric itself is customized to support processor-based designs. Processor-specific building blocks such as register files introduced in addition

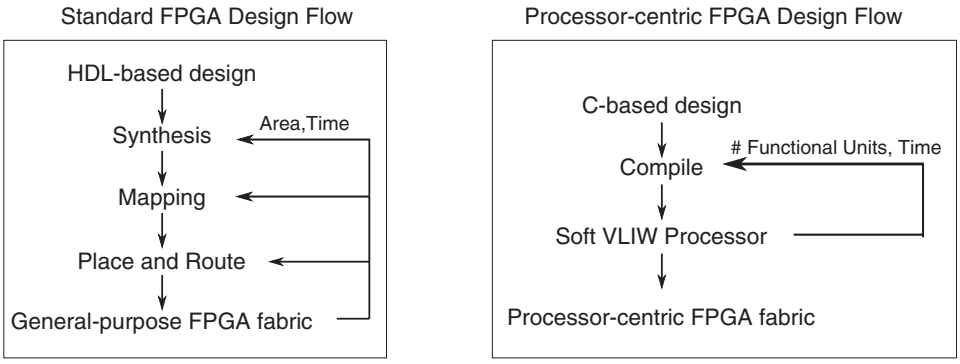


Fig. 2. Comparison of existing standard FPGA design flow and our approach of processor-centric FPGA designs.

to conventional logic block components result in significant area reduction. The savings in area can be used for further customization, such as addition of application-specific instruction for application acceleration. Figure 2 compares the two approaches. The figure on the left shows the generic HDL-based design flow that requires time consuming design entry, synthesis, mapping and place route. This design flow is the standard flow in current FPGAs. The figure on the right shows our technique for processor-centric FPGAs. In contrast to the tedious HDL-based design flow, a standard C-based application can be directly mapped on the configurable soft-VLIW processor. Since the FPGA architecture itself is customised for VLIW processors, the implementation is very efficient. Further, with the availability of configurable logic blocks additional functional units can be added to the VLIW processor to meet the concurrency requirements of the application mapped. Within the VLIW processing framework, additional application-specific instructions can be introduced for acceleration.

3.1. Embedded Multi-Ported RAM

In order to tackle the performance gap between FPGAs and custom CMOS implementation of multi-ported register files, we have introduced *embedded multi-ported RAMs* (MPRAM) as hard memory blocks within the FPGA fabric. Our objective is to accommodate these special memories in existing standard FPGA fabrics. The appropriate choice of the register file configuration, like the number of read/write ports, register file depth and width, are made using the area and I/O bandwidth estimation of the embedded multi-ported RAM with those of existing configurable slices and embedded block RAMs. The register file characteristics are derived using the 40nm technology library to match the 40nm technology in Virtex VI family of FPGAs from Xilinx. For the I/O bandwidth comparison, the port details were obtained from the Virtex VI user guide [Xilinx 2011b]. The area ratio of one block RAM to the number of slices was obtained from the layout information using the tools from Xilinx (PlanAhead and Floorplan Editor). In Virtex VI, the height of one block RAM corresponds to the height of 5 slices. It has to be noted that in Virtex VI, a configurable logic block has two slices, with each slice having four LUTs and eight flip-flops. The layout organization in Xilinxs FPGAs is column-based, the relative difference in the size of individual building blocks in an FPGA is measured in terms of the relative height of the slice and the width of the resource usually spans one column.

In order to make a fair, target-independent comparison of area between the block RAM and embedded multi-ported RAM, CACTI 6.5 [Tarjan et al. 2006] was chosen. It provides time, area and power estimations for pure SRAM memories with varying port

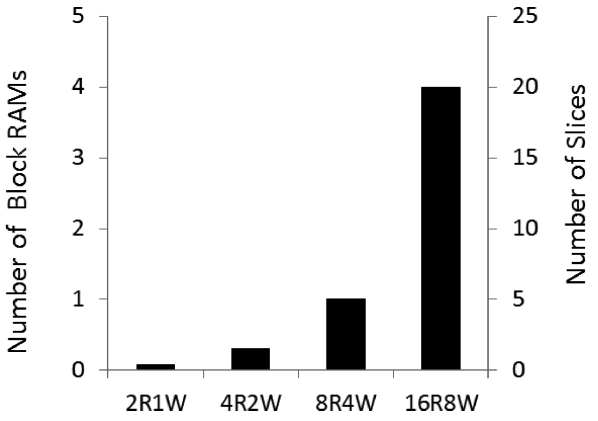


Fig. 3. Area of the 32×32bit embedded multi-ported RAM in comparison to the area occupied by block RAMs and slices.

configurations, dimensions, and sizes. For the Virtex VI family from Xilinx, each block RAM is a 36Kbit true dual-port memory. To represent a 36Kbit block RAM, a 5KByte memory was chosen with true dual port memory configuration (to be conservative). In accordance to the register file dimensions in standard processors, a 32×32bit memory (128 bytes) was chosen. For the multi-port register file, a 2-read/1-write port (denoted by 2R1W), a 4-read/2-write (4R2W) memory, a 8-read/4-write port (8R4W) memory, and a 16-read/8-write port (16R8W) memory was chosen to suit the configuration of a register file in RISC processors, 2-issue, 4-issue, and 8-issue VLIW processors.

For the area and power estimation in CACTI, process specific parameters required for calculating the transistor and wire parasitics were obtained from ITRS [Semiconductor Industry Association 2005]. The high performance library in CACTI (ITRS-HP) was chosen to make area comparison for memory. Figure 3 compares the area of a 32×32bit embedded multi-ported RAM with varying port configurations to the area of a dual-port 5 Kbyte block RAM and the relative area occupied by each slice in Xilinx Virtex VI. As can be seen in the plot, one unit of block RAM corresponds to nearly $12\times$ of the size of the 32×32bit memory with 2-read/1-write port (2R1W), nearly $4\times$ for a 4-read/2-write (4R2W) memory and is nearly the same size as a 8-read/4-write port (8R4W) memory. A 16-read/8-write port (16R8W) memory is $4\times$ the size of a block RAM memory. In comparison to slices, the relative sizes are $0.4\times$, $1.5\times$, $5\times$ and $20\times$ that of a slice for 2R1W, 4R2W, 8R4W and 16R8W port memories respectively.

In order to ensure routability of the embedded multi-ported RAM within the FPGA layout organization, the I/O bandwidth of a slice and a single 36 Kbit block RAM are compared to that of the multi-ported RAM with varying port configurations. The I/O port information for each slice and the block RAM memories were obtained from the Xilinx Virtex VI User guide [Xilinx 2011b]. Figure 4 compares the ratio of I/O bandwidth of a slice and a block RAM to that of the embedded multi-port RAM with varying port configurations. Although the area of the 2R1W memory is smaller than a block RAM, in terms of I/O bandwidth it is almost 62% of the size of a block RAM. In comparison to a slice, the I/O bandwidth is about $2.5\times$ that of a slice. Similarly, a 4R2W memory is $1.3\times$ that of a block RAM and $5\times$ that of a slice. A 8R4W port memory is $2.5\times$ that of a block RAM and $10\times$ that of a slice. The largest multi-ported RAM with 16R8W ports is $5\times$ that of block RAMs and $20\times$ that of a slice.

The comparison of area and I/O bandwidth of the embedded multi-ported RAM to a block RAM shows that the I/O bandwidth limitation far exceeds the area limitations of a block RAM. For layout and routability, the I/O bandwidth dominates the area

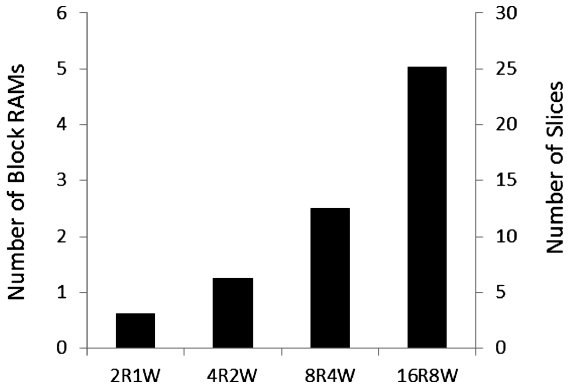


Fig. 4. I/O bandwidth of the 32×32 bit embedded multi-ported RAM in comparison to the I/O bandwidth of block RAMs and slices.

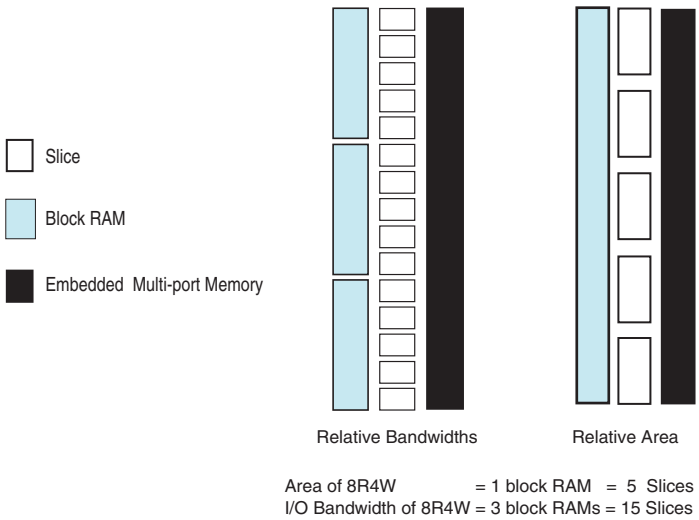


Fig. 5. Layout representation of I/O bandwidth and area of the embedded multi-ported RAM, block RAM and slices.

requirements, both for the block RAM and the slices. In order to adhere to the layout and routing infrastructure in the Virtex VI architecture, the number of embedded memory blocks considers the I/O bandwidth ratios. As an example consider the 8R4W port embedded memory, one such memory replaces 13 slices, even though the area corresponds to that of 5 slices. Similarly, a 16R8W port configuration replaces 5 block RAMs, despite the fact that the area equivalent corresponds to 4 block RAMs.

Figure 5 shows the relative area and I/O bandwidth of the new embedded multi-ported RAM in 8R4W port configuration as a ratio of slices and block RAMs. As can be seen, the area measurements using the high performance ITRS-HP library are much lower than the I/O bandwidth requirements for the embedded multi-ported RAM. In our analysis we assume the relative sizes for the embedded memories with respect to its I/O bandwidth requirements.

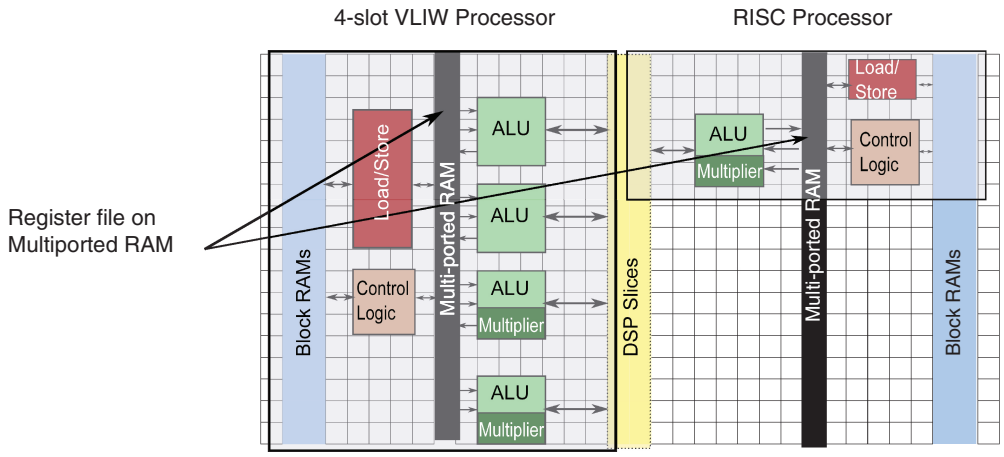


Fig. 6. Using multi-ported RAMs to map register files to increase the number of processors in an FPGA.

3.2. Resource Structuring for Processor-Centric FPGAs

The area and I/O bandwidth comparison leads to two possible layouts, viz., to introduce a single, huge multi-ported RAM (say 16R8W) that occupies a significant part of an FPGA column or introduce multiple blocks with fewer memory ports (say 8R4W). There are pros and cons in both the layout styles. Introducing a single huge register file, say 16R8W, accelerates and enhances the density of soft-VLIW processors with a maximum of 8 functional units, whereas the number of such processors is limited by the number of multi-ported RAM columns that can be accommodated in a single FPGA device. Additionally, the configurable logic blocks available for implementing the 8-slot VLIW processor are also a consideration. In the absence of one wide VLIW processor the benefits of using this configuration for RISC processors is minimal, since the over-sized multi-ported RAM can only be used as a 2R1W port register file for a single RISC processor, leading to under-utilization of these huge multi-ported register files.

On the other hand, when multi-port memories with a lesser number ports (say 8R4W) are chosen, many such blocks may be embedded in a dedicated column. This configuration provides the possibility of instantiating multiple, homogeneous VLIW processors with maximum of four functional units per processor. This configuration also enables using additional slices in conjunction with the dedicated multi-ported memory to build clustered VLIW processors with a higher number of issue-slots. For mapping the RISC processors, the loss in performance and utilization in this configuration is lower as compared to a single 16R8W port configuration.

In order to analyze the ratio of logic blocks and multi-port memories for processor-centric FPGAs, we analyze the percentage distribution of resources in VLIW processors and RISC processor mapped onto Xilinx FPGAs. The ρ-VEX VLIW processor in the default configuration with four functional units (or ALUs) and two multipliers was chosen. The register file in this case has eight-read/four-write ports (8R4W). Nearly 50% of the total logic blocks is occupied by the register file. In contrast to the VLIW processor, the register file in a RISC processor only contributes to about 12% of the total logic when implemented entirely on logic resources.

Based on the diversity in resource distribution, the two possible use for the organization of resources using multiple embedded memory blocks in a column is shown in Figure 6. In the first option shown in the left-half of the figure, a 4-issue VLIW processor uses the entire embedded memory. On the left-half of the figure, the 8R4W

memory is used by a RISC processor, in place of a 2R1W port memory. Since one unit of 8R4W port embedded memory occupies, at the most, an equivalent of 15 slices (or 3 block RAMs), multiple such copies of memories can be placed in a single column of the FPGA. Since, the objective is to increase the density of processors in an FPGA, replacing 15 slices with embedded memories is a more efficient solution than replacing block RAMs.

4. EXPERIMENTS AND RESULTS

For area and timing analysis, two processors with differing register file architectures were chosen. The ρ-VEX processor with 8R4W ports is representative of soft-VLIW processors and the OpenRISC processor with 2R1W port register file represents most commercially available soft-RISC processors, such as Microblaze and Nios. To evaluate the quality of the proposed solution, the special embedded multi-ported RAMS need to be introduced by replacing a section of the slices. In our design flow, the embedded multi-ported RAMs were introduced as dedicated blocks with inputs/outputs to these blocks obtained through other existing logic blocks within the FPGA. For such architectural experiments, the academic place and route tool VTR (Verilog to Routing) [Luu et al. 2011] provides a language-based support for defining custom FPGA architectures with dedicated blocks defined as black boxes. The present version of the tool is limited to area-driven optimization. The earlier version of the tool VPR 5.0 [Luu et al. 2009], which is timing driven does not support timing-driven optimization of black boxes. Another scheme that could be applied is the methodology that uses a concept called Virtual Embedded Blocks [Ho et al. 2006], where the performance gain of introducing dedicated blocks can be measured used existing commercial tools. This scheme assumes that existing logic resources can be used to match the position, area, and delay of a specific circuit implemented as an embedded block. We use this design flow to introduce dedicated memories for the register files by them onto equivalent number of slices. The floorplan of the design with the dedicated memory blocks mapped onto its equivalent slices is obtained using Xilinx PlanAhead. Equivalent area and timing reports are obtained after using this floorplan in normal Xilinx design flow for place and route. For all the experiments, Xilinx design tools and architecture has been used. In our experiments, we compare area and power reports for the FPGA-based implementation of the register file using commercial tools and those of the equivalent embedded multi-port RAM. The reports for the two processors were obtained after mapping them on to Xilinx XC6VLX760-2FF11760 using Xilinx-ISE 11.1. CACTI 6.5 was used for obtaining area and timing values for the embedded multi-port RAM used in the implementation of the embedded multi-ported RAMs. VTR [Luu et al. 2011] was used to compare the routing area for the modified RISC architecture. Our experiments are limited to area and power analysis since modifications to the original processor are only in the register file implementation, which does not appear in the critical path during timing analysis. Although timing improvements are significant for the register file, it does not affect the overall performance of the processor.

4.1. Advantage of VLIW Processing

The proof of performance advantage in a multi-issue VLIW processor as compared to a single-issue RISC processor targeting DSP applications is shown in Figure 7. For the analysis, the default configuration of the ρ-VEX processor with four ALUs, two multipliers and two load/store units was compared with the same processor with a single-issue, representing RISC processors. The operating frequency remains unchanged in two the processors, the 4-slot VLIW and the 1-slot RISC processor [van As 2008]. Applications from the DSPStone fixed-point benchmarks [Zivojnovic et al. 1994] were chosen for comparison. The figure shows that a speedup in the range of $2\times$-$3\times$

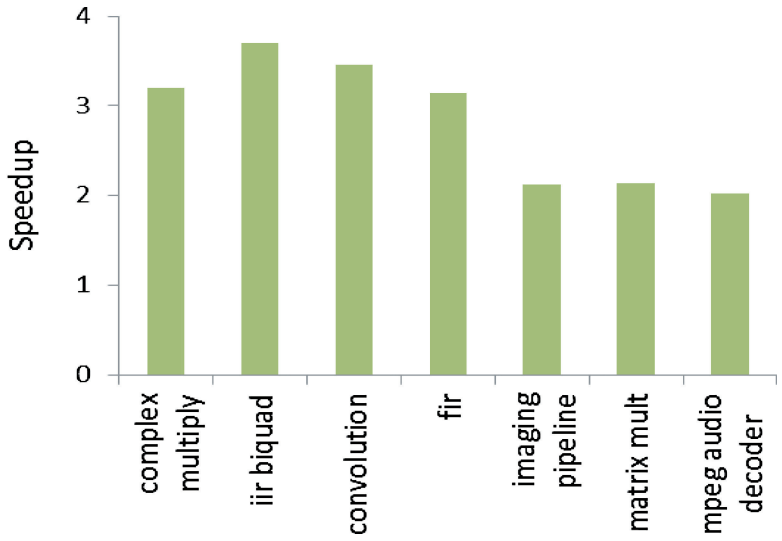


Fig. 7. Speedup achieved: 4-slot VLIW in comparison to a 1-slot RISC.

is achieved in comparison to a single-slot machine, which confirms the advantage of VLIW processors for DSP applications.

The thesis by van As [2008] provides additional application analysis of the speed improvement for the VLIW VEX processor. In the thesis the ρ-VEX processor has been compared to other single-issue processors such as PowerPC 405 and Intel Pentium II using the SPECINT95 benchmarks. The analysis show a performance gain of up to 4$\times$ with the ρ-VEX processor in comparison to Intel Pentium II. A bigger gain in performance has been reported with an increase in the issue-width from 2 to 4 for the ρ-VEX processor than with an increase in issue-slot from 4 to 8. Performance gains of up to 27$\times$ are seen with the ρ-VEX processor with caches in comparison to the Power PC 405 with caches.

The compiler for the ρ-VEX processor is a descendant from the commercially available Lx compiler from HP. Additional performance reports to validate the benefits of parallelism using the VLIW configuration can also be obtained from the paper by Faraboschi et al. [2000].

In addition to these reports, the performance benefits of augmenting a FPGA-based VLIW processor with custom hardware accelerators has been presented by Jones et al. [2005]. Custom hardware accelerators in addition to the VLIW processor provide an overall speedup of 30$\times$ and a speedup of 12$\times$ on an average for signal processing benchmarks from MediaBench. The authors correctly point out the overhead of power and area on account of the shared register file, which is the price paid for the speedup. Scaling the number of read/write ports to the register file has the greatest impact on performance and area. This paper presents results on minimizing area and power of the FPGA-based of the multi-ported register file.

4.2. Area Comparisons using Embedded Multi-Port RAM

To analyze the area savings achieved by introducing the *multi-port RAM* (MPRAM) in the FPGA fabric, both the ρ-VEX and the OpenRISC processor were mapped onto Xilinx Virtex VI. The area occupied by the two processors and the size of the embedded multi-port RAM obtained in terms of slices are listed in Table I. For the VLIW processor, the new register mapped onto the dedicated embedded multi-port RAMs is nearly

Table I. Reduction in Area of the Processor's Register File

	ρ-VEX (slices)	OpenRISC (CLBs) (slices)	OpenRISC (block RAM) (mm^2)
Reg. file area in existing FPGA	1535	335	0.152
MPRAM area in equivalent slices	15	1	0.006
Change in Area	102× smaller	335× smaller	25× smaller

Ratio of area required for the register file implementation using existing FPGA resources to the area required using the Embedded Multi-ported RAM.

102× smaller using the ITRS-HP library; as compared to the original register file implemented using logic slices. For the OpenRISC, the new register file is nearly 335× smaller when using the ITRS-HP library for the RISC processor, in comparison to the implementation using slices.

The area improvements are even higher (nearly 150× smaller for the register file in ρ-VEX) when considering the equivalent area estimations for embedded multi-ported RAM. The area requirements for the register file are lower than the I/O bandwidth requirements. In our experiments, the equivalent size of the memories is as obtained by the I/O bandwidth calculations to retain the original internal routing architecture in Xilinx FPGAs.

For the RISC processor, the register file can also be mapped onto the block RAMs. For mapping a 2R1W port register file, two block RAM units are required, which are both only partially utilised. In this case, the relative size of the block RAM version compared to a 2R1W register file is nearly 25× smaller using the ITRS-HP library. It has to be noted that the Xilinx's architecture only allows ρ-VEX processor's register file to be mapped onto slices, since the block RAM memories are limited to two read/write ports.

The savings in area aids in increasing the density of processors in a single FPGA device. The slices that are rendered unused on including the embedded multi-ported RAM can be further used to map additional functional units, processors and peripherals. For the ρ-VEX processor, the register file contributes to nearly 50% of slices occupied by the entire processor. Hence, introducing the embedded multi-ported RAM almost doubles the number of processors that can be packed in the same area. In the OpenRISC core, the register file contributes to 12% of the total area of the processor. Thus, introducing a dedicated multi-ported RAM that replaces the register file allows adding an extra processor for every 8 processors previously possible in the same area.

To measure the savings in routing area, the OpenRISC processor was synthesized, placed and routed using the VTR design flow [Luu et al. 2011]. The processor was entirely mapped onto configurable logic blocks and then compared with the version where the register file was introduced as a block in a dedicated column. In this design flow, the black box represents the embedded multi-ported RAMs. The savings in routing area is nearly 39% after introducing the register file as a dedicated block. Introducing the embedded multi-ported RAM also resulted in reduction in the maximum net length by nearly 51%. Since the present version of the tool only supports designs defined in Verilog, the same flow could not be used for the ρ-VEX processor. Even higher advantages can be expected in the VLIW processor, on account of higher usage of configurable resources by the multi-ported register file. For applying the VTR design flow for the ρ-VEX processor, we are also limited by the size of the black box that can be introduced.

4.3. Reusing the Embedded Multi-Port RAMs for RISC Processors

Introducing multi-ported RAM improves FPGA area utilization while using soft-VLIW processors. In the absence of VLIW processors, the same embedded memory can be used

Table II. Reduction in Area of the RISC Processor's Register File

	8R4W	16R8W
2R1W Register file area in slices	335	335
MPRAM area in equivalent slices	15	25
Change in Area	22× smaller	13× smaller

Ratio of the register file implementation using slices to the implementation using the VLIW processor's register file.

Table III. Increase in Area of the RISC Processor's Register File

	8R4W	16R8W
2R1W Register file area on block RAM (in equivalent slices)	10	10
MPRAM area in equivalent slices	15	25
Change in Area	+1.5× larger	+2.5× larger

Ratio of area required for the register file implementation using block RAMs to the area required using the VLIW processor's register file.

for RISC processors. The register file in the RISC processors occupies 335 slices using the original FPGA architecture. The embedded multi-ported RAM is much smaller and can be used for the register file in the RISC processor even when all the read/write ports are not entirely used. Area savings are observed even when the ports register file is underutilized. The relative savings in area in comparison to the implementation using slices and block RAMs is listed in Table II and Table III, respectively.

The 8R4W port memory is almost of the same size as the block RAM implementation, but with respect to the I/O bandwidth it corresponds to the size of 3 block RAMs, hence occupies 15 slices. Similarly, the 16R8W port memory occupies the size of 4 block RAMs but corresponds to 5 block RAMs in terms of I/O bandwidth, hence occupies 25 slices. In comparison to the slice-based implementation of 2R1W port register file, which occupies 335 slices, the 8R4W port memory is 22× smaller and the 16R8W port memory is 16× smaller.

The block RAM version of the register file implementation requires 2 block RAMs. In comparison register file mapped onto block RAMs the 8R4W port memory is 1.5× larger and the 16R8W port memory is 2× larger. Comparatively, the 8R4W port memory has higher advantages in area utilization. It is comparable in size to three block RAM and results in an implementation that is smaller than the one using slices. As can be seen, introducing multiple blocks 8R4W port memory is only slightly larger than the version implemented with block RAMs, hence beneficial for both VLIW processors and RISC processors. It has to be noted that the area savings are even higher when considering the equivalent area estimations for embedded multi-ported RAM. The present calculations are based entirely on the equivalent I/O bandwidth considerations.

4.4. Area and Power Analysis for OpenRISC and ρ-VEX Processors

To analyze the overall impact of introducing multi-ported RAMs in the FPGA architecture, the two processors ρ-VEX and OpenRISC were compared for area and power utilization on the existing FPGA and its modified version.

Table IV shows the area savings of the entire processor for the ρ-VEX implemented using slices and the OpenRISC mapped onto slices and block RAMs. With the processor-centric FPGA fabric the area occupied by the ρ-VEX processor is 1.8× smaller than the one using the original FPGA fabric. For the OpenRISC, comparing the version implemented with CLBs, the processor-centric FPGA implementation has an area that is 1.5× smaller. In comparison to the block RAM version, the slice utilisation is almost the same in both cases.

Table IV. Reduction in Processor Area

	ρ-VEX (using CLBs)	OpenRISC (using CLBs)	OpenRISC (using block RAM)
Existing FPGA architecture	3672	1178	785
Eqv. slices with Embedded MPRAM	2039	790	790
Change in Area	1.8× smaller	1.5× smaller	Almost the same size

Ratio of the area required for the processor implementation in the existing FPGA architecture to the area required in modified FPGA with Embedded Multi-port RAM (in terms of equivalent slices).

Table V. Reduction in Processor Power

	ρ-VEX (using CLBs)	OpenRISC (using CLBs)	OpenRISC (using block RAM)
Existing FPGA architecture	68 mW	24 mW	2 mW
With Embedded MPRAM	40 mW	21 mW	2 mW
Change in Power	41% lower	12% lower	No change

Percentage difference in power required for the processor implementation in the existing FPGA architecture to the power requirements for the implementation in the modified FPGA with Embedded Multi-RAM.

Table V shows the power savings achieved after introducing the embedded multi-ported RAMs. The power savings for the ρ-VEX processors is nearly 41% on account of the reduction in the number of slices used. For the OpenRISC implemented using distributed RAM, the power savings is nearly 12%. For the block RAM version, the difference in power is negligible.

5. CONCLUSIONS AND FUTURE WORK

Customizing a generic FPGA to support efficient implementation of processors by introducing processor-centric features in the FPGA has been the focus of this paper. Currently, most FPGA vendors only support soft-RISC processors. In this paper we highlight the performance advantages of using soft-VLIW processors, especially in DSP applications. FPGAs are well suited for concurrent data processing, whereas soft-RISC processors offer limited possibilities for exploring parallelism, both at data and instruction-level. However, with the current FPGA architectures, implementing a VLIW processor is very inefficient in terms of area and power. To mitigate the problem of resource efficiency, a hard multi-ported RAM has been introduced to ease mapping soft-VLIW processors on FPGAs. In the absence of such memory support, the multi-ported register file implementation requires significant amount of FPGA resources and results in lowered performance. In addition to using this multi-ported RAM for VLIW processors, a subset of the embedded multi-ported RAMs can also be used for the register file in RISC processors. Vector processors can also benefit from the modifications, which enable a higher density of vector lanes to be packed into the same device.

Increasingly FPGAs are being targeted for DSP applications, in this direction, our approach helps in achieving high performance with reduced design efforts. The design modification eases mapping of VLIW processors on FPGAs, which can outperform present day soft-RISC processors for DSP applications. In this technique, the design flow uses a purely software-oriented programming model for processors that are targeted on an FPGA. Hence, high performance DSP applications can be executed on FPGAs with almost no hardware design overhead.

Long term research goal in this direction is to gain insight on building resource-efficient multi-core systems on FPGAs. The goal is to build a multi-core-centric FPGA fabric to merge the benefits of parallel processing in multi-core processors and large

amounts of concurrency available in FPGAs. Consequently, the next architectural challenge will be to adapt the FPGA fabric to enable packing a large number of homogeneous and heterogeneous cores and introduce modifications to ease the interaction between the cores.

REFERENCES

ALTERA. 2011. Nios II processor reference. http://www.altera.com.

BORRIELLO, G., EBELING, C., HAUCK, S. A., AND BURNS, S. 1995. The triptych FPGA architecture. *IEEE Trans. VLSI Syst. 3*, 491–501.

BURNS, G., JACOBS, M., LINDWER, M., AND VANDEWIELE, B. 2006. *Silicon Hive's Scalable and Modular Architecture Template for High-Performance Multi-Core Systems*. Silicon Hive Inc.

FARABOSCHI, P., BROWN, G., FISHER, J. A., DESOLI, G., AND HOMEWOOD, F. 2000. LX: A technology platform for customizable vliw embedded processing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA'00)*. 203–213.

FISHER, J., FARABOSCHI, P., AND YOUNG, C. 2009. VLIW processors: Once blue sky, now commonplace. *IEEE Solid-State Circuits Mag. 1,* 2, 10–17.

HARTENSTEIN, R. 2001. Coarse grain reconfigurable architecture (embedded tutorial). In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'01)*. ACM, New York, 564–570.

HO, C., LEONG, R., LUK, W., WILTON, S., AND LOPEZ-BUEDO, S. 2006. Virtual embedded blocks: A methodology for evaluating embedded elements in FPGAs. In *Proceedings of the 14th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06)*. 35–44.

JONES, A. K., HOARE, R., KUSIC, D., FAZEKAS, J., AND FOSTER, J. 2005. An FPGA-based VLIW processor with custom hardware execution. In *Proceedings of the ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays (FPGA'05)*. 107–117.

LAFOREST, C. E. AND STEFFAN, J. G. 2010. Efficient multi-ported memories for FPGAs. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, New York, 41–50.

LIN, M. 2008. The amorphous FPGA architecture. In *Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays*. ACM, New York, 191–200.

LUU, J., ANDERSON, J. H., AND ROSE, J. S. 2011. Architecture description and packing for logic blocks with hierarchy, modes and complex interconnect. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM. New York, 227–236.

LUU, J., KUON, I., JAMIESON, P., CAMPBELL, T., YE, A., FANG, W. M., AND ROSE, J. 2009. VPR 5.0: FPGA cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling. In *Proceeding of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, New York, 133–142.

LYSECKY, R. AND VAHID, F. 2005. A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning. In *Proceedings of the Symposium on Design, Automation and Test in Europe*. 18–23.

OPENCORES. OR1200 OpenRISC processor. http://www.opencores.org.

SAGHIR, M. A. R. AND NAOUS, R. 2007. A configurable multi-ported register file architecture for soft processor cores. In *Proceedings of the 3rd International Conference on Reconfigurable Computing: Architectures, Tools and Applications*. 14–25.

Semiconductor Industry Association 2005. *International Technology Roadmap for Semiconductors*. Semiconductor Industry Association. http://public.itrs.net/Links/2005ITRS/Home2005.htm.

TARJAN, D., THOZIYOOR, S., AND JOUPPI, N. 2006. CACTI. Tech. rep., HP Labs.

Tensilica Inc. 2011. ConnX BBE64 (baseband engine) 64-128 MAC/cycle high performance DSP. Tensilica Inc.

VAN AS, T. 2008. ρ-VEX: A reconfigurable and extensible VLIW processor. M. S. thesis, Department of Electrical Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology.

WONG, H., BETZ, V., AND ROSE, J. 2011. Comparing FPGA vs. custom CMOS and the impact on processor microarchitecture. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. 5–14.

XILINX. 2011a. *Microblaze Processor Reference Guide*. http://www.xilinx.com.

Xilinx 2011b. *Virtex-6 FPGA User Guide*. Xilinx. http://www.xilinx.com.

Xilinx 2011c. *Virtex-7 FPGA User Guide*. Xilinx. http://www.xilinx.com.

Yu, J., Lemieux, G., and Eagleston, C. 2008. Vector processing as a soft-core CPU accelerator. In *Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays (FPGA'08)*. ACM, New York, 222–232.

Zivojnovic, V., Martinez, J., Schlaeger, C., and Meyr, H. 1994. DSPstone: A DSP-oriented benchmarking methodology. In *Proceedings of the International Conference on Signal Processing Applications and Technology (ICSPAT'94)*.

Received July 2011; revised October 2011; accepted November 2011