

A 90k gate “CLB” for Parallel Distributed Computing

Bruce Schulman¹ and Gerald Pechanek²

¹BOPS, Inc. Palo Alto, CA bruces@bops.com

²BOPS, Inc. Chapel Hill, NC gpechanek@bops.com

Abstract. A reconfigurable architecture using distributed logic block processing elements (PEs) is presented. This distributed processor uses a low-cost interconnection network and local indirect VLIW memories to provide efficient algorithm implementations for portable battery operated products. In order to provide optimal algorithm performance, the VLIWs loaded to each PE configure that PE for processing. By reloading the local VLIW memories, each PE is reconfigured for a new algorithm. Different levels of flexibility are feasible by varying the complexity of the distributed PEs in this architecture.

1 Introduction

As the complexity of portable products has increased, along with the need to support multiple, evolving standards, processor-based solutions have become a requirement at all levels of product architecture.

While a processor provides the needed flexibility, it must do so in an energy efficient and area efficient manner. Since the type of processing required for different products includes communications, video, graphics, and audio functions, multiple data types and algorithmic computational needs must be accommodated.

Due to this wide diversity of requirements, many approaches to providing efficient processing capability in each application have been proposed. These solutions include custom designed ASICs, general-purpose processors with DSP packed-data type instruction extensions, different DSPs in each product, and reconfigurable processor designs using FPGAs. ASICs lack flexibility in the face of changing standards and changing product requirements, measured as their high cost to support changes or multiple similar instances. General-purpose processors for embedded applications are inefficient in energy and area. Reconfigurable processors using FPGAs, even with the latest process improvements, are also inefficient in implementation area and energy use. This is especially true for FPGA implementations of arithmetic units, which are still very large and slow, compared to ASIC or custom arithmetic designs [1].

Even so, there is much work being done to combine the advantages of microprocessors and FPGAs for reconfigurable co-processing units, such as DISC [2] and GARP [3]. These systems may mix general control processors, fixed function ASICs, and FPGAs in a final system, such as Pleides [4]. In addition, companies such

as Xilinx and Altera provide FPGA's and design solutions for specific reconfigurable algorithmic use [5, 6].

The difficulty with state-of-the-art FPGA designs is that the area, performance, and power cannot compete with standard cell or custom designed logic. While using FPGAs seems to hold promise, many difficult problems exist that must be solved. Two problems with FPGA designs are the programming model/tools, and consistent and efficient use of silicon area. It is important that each product has a consistent programming model and a common set of development tools across the numerous applications. It is equally important to have a programmable design that can efficiently provide high performance and low power in the intended products. Research attempting to improve the implementation efficiency of FPGA-based reconfigurable processors proposes to increase the complexity of the Configurable Logic Blocks (CLBs), to include circuitry better suited for arithmetic use [7]. These additions attempt to provide application specific improvements to the original CLB definition. The goal is still to solve the basic problem of providing processor-level flexibility in a cost and performance efficient manner.

The purpose of a reconfigurable processor is to make effective and efficient use of the available logic for a number of applications by programming the arrangement and interconnection of the logic. We propose to use standard ASIC processes for a set of flexible arithmetic units in a standard PE definition that is programmed through local VLIW memory. Programming the PE can be viewed as a method to optimize the logic make-up of the PE for different algorithms. With our scalable, parallel distributed processing configuration, the use of the available resources can be configured appropriately, cycle-by-cycle, to meet the requirements of each application. Further, these features and capabilities are provided in a single architectural definition using a consistent and standardized tool set.

In this paper we present the BOPS® ManArray™ parallel distributed computing architecture and show that by reprogramming the PEs' logic, very high-performance computing can be provided across multiple applications.

2 ManArray Parallel Distributed Computing

The BOPS® iVLIW™ PE is based upon the BOPS ManArray architecture, a parallel-distributed computer architecture targeting System-On-Chip applications. The ManArray architecture supports from 1 to 64 iVLIW PEs and a Sequence Processor (SP) for controlling the array of PEs. The SP is uniquely merged into the PE array for maximum efficiency to provide the SP controller with access to the ManArray network. The ManArray network interconnects clusters of PEs to provide contention-free, scalable, single-cycle communications. The distributed processor uses two basic building blocks, as shown in Figure 1. The PE consists of a register file, a set of execution units, a cluster switch as an interface to the ManArray network, local data memory, and local VLIW memory (VIM). The SP adds an instruction fetch unit and uses the same building block PE elements. Various core processors can be developed from these two reusable IP blocks.

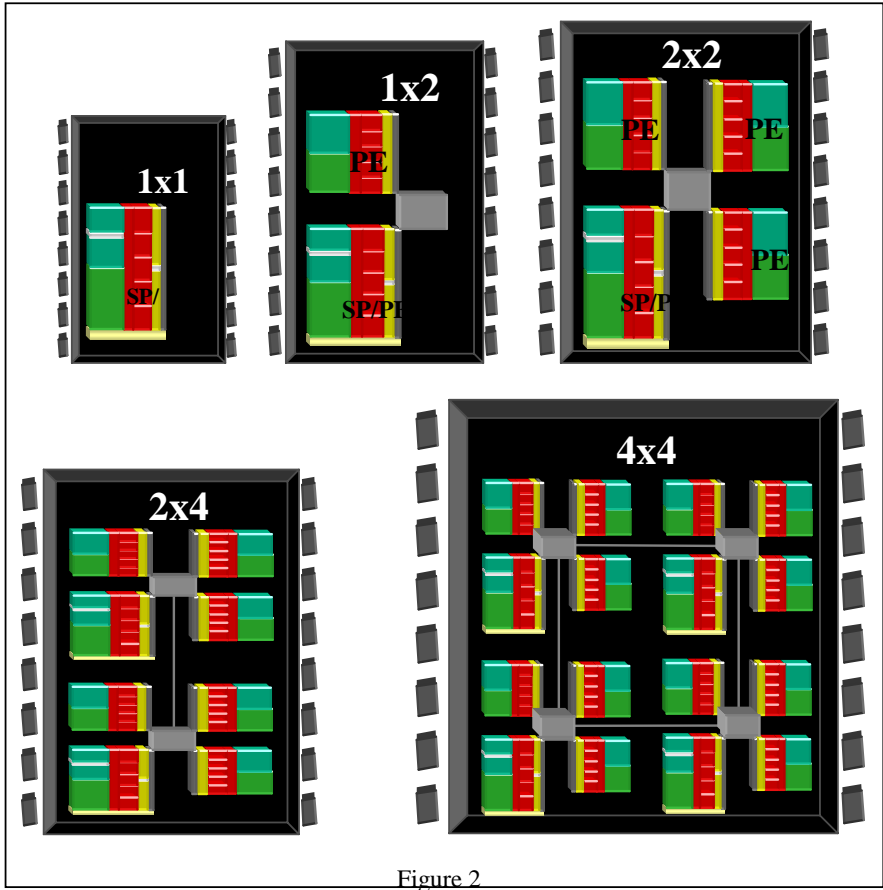
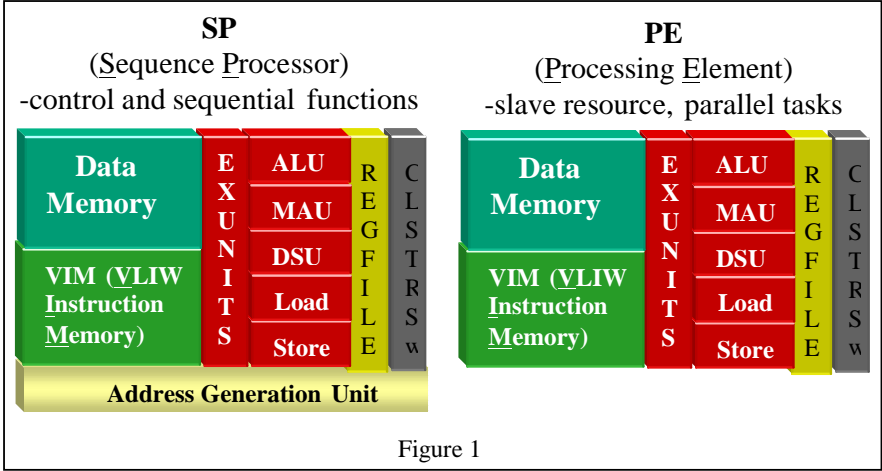


Figure 2 shows a 1x1, 1x2, 2x2, 2x4, and a 4x4 array processor. Each 2x2 cluster contains an SP control processor allowing reconfiguration of larger arrays to operate as subset array processors. For example, two 2x2 array processors can be configured in the 2x4 array processor system.

The general organization of the BOPS iVLIW PE is shown in Figures 3 and 4, which depict the three main interfaces to each PE. These interfaces are an X-bit instruction bus, Y-bit data busses, and a single-port send/receive interface to the cluster switch that interconnects the PEs in the ManArray topology. The instruction format is typically X=32-bits but 16-bit and 24-bit formats are not excluded depending upon an application's needs.

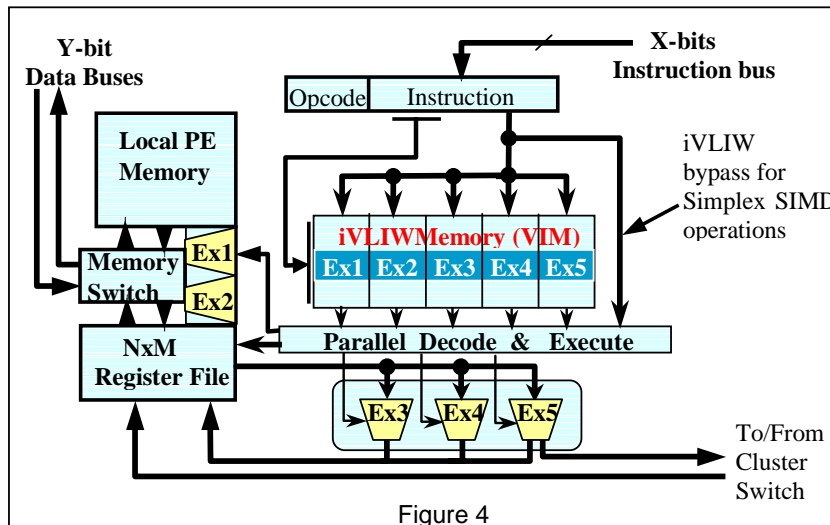
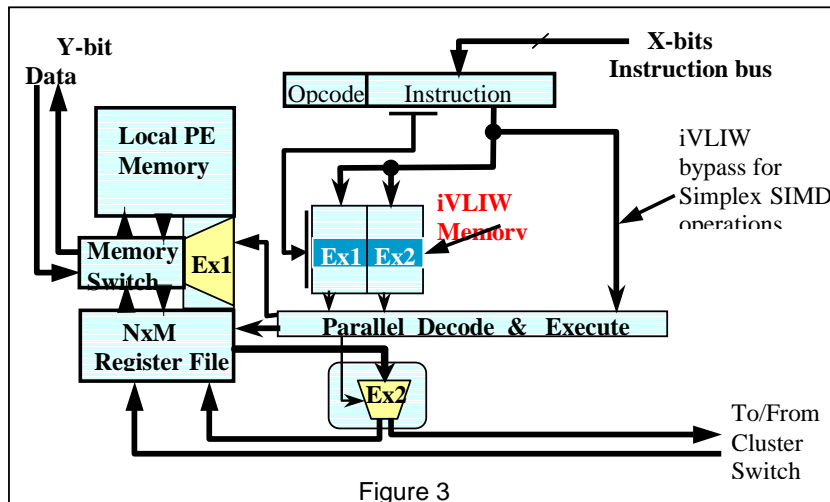
Internal to the PEs are three storage elements, the local PE data memory, the $K \times S \times X$ iVLIW Memory (VIM), and a multiported $N \times M$ -bit register file. The number of VLIW's entries, K is typically less than 128 entries, although larger iVLIW memories are not precluded. The number of instruction slots, S can vary from 1 to 8, although typically between 2 and 5 instruction slots would be used (Figures 3 and 4 respectively). Depending on the arithmetic VLIW configuration, the local PE data memory can be a one- or a two-port memory. The two-port local PE memory is configured into two Y-bit banks which support byte, halfword, word, and double word loads with Y=32-bits. With the twin banks, one memory can be loading and storing data simultaneously to/from the PE's register file while the DMA unit is loading the other bank. This effectively hides DMA delays, and supports a data streaming approach to processing on the array.

Based upon present application evaluations, two banks of 512x32-bits are typically proposed, although there is no architecture limit. In a 5-issue iVLIW, the VIM typically consists of up to 64x160 bits of iVLIW memory with 160-bit read out capability. The VIM is loaded sequentially, one X-bit instruction at a time, after being primed by a LoadV delimiter instruction. The 160-bit field is made up of 5 Y-bit instruction slots, with each slot associated with an execution unit. In addition, a $N \times M$ -bit 8-read-port 4-write-port register file is available. This register file is split into two banks of 16x32-bits allowing the architecture to support 64-bit data flows as well as 32-bit data flows. One bank is associated with the even register addresses and the other bank is associated with the odd register addresses. The split register-file design takes full advantage of the instruction set architecture and reduces the number of ports required per register bank.

Finally, the ManArray architecture supports up to eight execution units in each PE. The first release uses five execution units: one Load Unit (LU), one Multiply Accumulate Unit (MAU), one Arithmetic Logic Unit (ALU), one Data Select Unit (DSU), and one Store Unit (SU). The execution units support 1-bit, 8-bit, 16-bit, and 32-bit fixed point data types, and 32-bit IEEE floating point data to meet the requirements of a large number of applications. For high-performance applications, each PE supports 32-bit and 64-bit packed data operations that are interchangeable on a cycle-by-cycle basis. Specifically, the MAU supports quad 16x16 Multiply and Accumulate operations per cycle, and the ALU performs standard adds, subtracts, four 16-bit absolute-value-of-difference operations, and other DSP functions. The DSU performs bit operations, shifts, rotates, permutations, and ManArray network communication operations. Supporting the computational elements are 64-bit load and store units. It should be noted that there is a bypass path around the VIM

allowing single 32-bit instructions to be executed separately in classical SIMD mode in each PE and consequently on the array.

We use a linearly scalable switch fabric to connect the PEs with an interconnect maximum length of 2 for large embedded arrays, and length 1 for orthogonal interconnected PEs [8]. This ManArray network is integrated in the architecture of the PEs such that data movement between PEs can be programmed and overlapped with other arithmetic operations and load/store operations. This interconnect is programmable per cycle to allow many different interconnect patterns to match the current processing task.



The programmer controls an array of PEs by writing a program for the SP, which includes the personalization of the PEs VLIW memory for the intended algorithm or algorithms to be executed. In addition, the SP controls the DMA unit to move data through the PEs, while controlling the program flow to perform the desired computation. Depending upon the size of the VLIW memory and the number of VLIWs needed for each algorithm in an application, the optimized set of VLIWs for multiple tasks can be resident in the VIM, allowing instantaneous reconfiguration as tasks change. Even with small VIMs that must be reloaded for each task, the loading of a five issue VLIW entry into all the PEs' VIMs @100MHz takes only 60nsec. The loading steps are - a Load VLIW instruction followed by the five instructions to be loaded into each VLIW in each PE in parallel. To load 32 VLIWs in all the PEs, sufficient for many tasks, the total load time is 1.92 μ sec. A state of the art, reconfigurable computer takes approximately 100 μ s to reconfigure, a relative factor of 50x. [9].

3 Evaluation

In TSMC .25u ASIC flow process, the 5-issue iVLIW PEs has a worst-case clock rate of 100MHz. Higher speeds are available utilizing more custom design methodologies and/or synthesizing the Verilog soft macro cores to different processes. With full capabilities in the PEs, including both fixed and floating point MAU, ALU, and DSU which also includes a state of the art single-precision floating-point divide/square root unit, a 2x2 processor array with DMA, 1 Mbit of SRAM, and system interfaces including PCI 2.0 (32-bit/33 MHz), SDRAM (PC-100, 64-bit), and host processor interface (MIPS SYSAD bus compatible with QED5231) is 90 sq mm. *A fully featured fixed-point 5-issue iVLIW PE requires 90K gates* excluding the register file and memory elements. Depending upon performance needs, the PEs can be reduced in size as indicated in Figure 3 by reducing the number of iVLIW issue slots, which reduces the number of register file access ports, subsetting the instruction set appropriately, and reducing the local PE memory requirements. This scalability provides great flexibility while still maintaining the same instruction set architecture.

For a 2x2 array of 5-issue iVLIW PEs with an SP, the computations on 16-bit data per cycle include: 16 multiplies, eight 32-bit sums, eight 40-bit accumulates, 16 absolute differences, 16 rotates, 16 loads and 16 stores. At 100 MHz, this equates to ~10 bops. The FFT is one application that uses the performance and data flow capability of the ManArray. For Discrete MultiTone (DMT) based ADSL or VDSL, a 256-point FFT is needed. OFDM-based digital terrestrial television will use larger FFTs. The BOPS 2x2 can continuously process 256-point 16-bit complex FFTs in less than 5 μ s [10]. This includes all the data movement and address reordering (bit reversed or digit reversed). Xilinx LogiCore Data Sheet "256-Point Complex FFT/iFFT V1.0.3" [5] said it takes 1643 logic slices to do a 256-point 16-bit complex FFT in 40 μ s. To get to 5 μ s per block, 8 such chips would be needed (8x less compute density). It is generous to NOT account for off-chip interconnect delays and forgive the fixed scaling in this estimate.

The basis of the most popular image compression algorithms (JPEG, MPEG) is the 8x8 iDCT. The algorithm takes in 8-bit data, but needs higher dynamic range to meet the S/N IEEE STD 1190. A 2x2 array can continuously process 8x8 blocks at a rate of 128 MBytes/second [11]. The Altera Discrete Cosine Transform AMPP Datasheet [6] shows an 8x8 iDCT processing rate of 17.5 MBytes/second, so you would need 8 of them to keep up.

4 Conclusions

BOPS offers the highest performance DSP IP in the industry and targets mass-market applications in 3D graphics, multimedia, Internet, wireless communications, VoIP and digital imaging. With this new level of performance and cost/performance, Embedded High Performance Computing can become a reality in consumer products from 3G cell phones with streaming video, to broadband Internet, to higher performance 3D Graphics in Set-top, games, and PCs.

The ManArray Architecture, including PEs, SP, DMA and Cluster Switch, delivers the highest performance, scalable, reusable, reconfigurable DSP IP in the industry. Compared to FPGAs, BOPS delivers more than 8x improvement in performance @100MHz in standard ASIC flow parts. Depending upon the array size, BOPS solutions cover the range from 1 to over 100 billion integer math operations/second.

References

1. O. T. Albaharna, P. Y. K. Cheung, and T. J. Clarke, "Area & Time Limitations of FPGA-based Virtual Hardware," Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp. 184-189, Cambridge, Mass., October 10-12, 1994, IEEE Computer Society Press.
2. M. J. Wirthlin and B. L. Hutchings, "DISC: The Dynamic Instruction Set computer," Proceedings of the SPIE, Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing, Vol. 2607, pp. 92-102, 1995.
3. J. R. Hauser and J. Wawrzynek, "Garp: A MIPS Processor with a Reconfigurable Coprocessor," Proceedings of IEEE Workshop on FPGAs for custom Computing Machines (FCCM), Napa, CA, April 1997.
4. Marlene Wan, Hui Zhang, Varghese George, Martin Benes, Arthur Abnous, Vandana Prabhu, Jan Rabaey, "Design Methodology of a Low-Energy Reconfigurable Single-Chip DSP System", *Journal of VLSI Signal Processing*, 2000.
5. Xilinx website: <http://www.xilinx.com/>
6. Altera website: <http://www.altera.com/>

7. Nelson, Brent, "Reconfigurable Computing," HPEC 98 proceedings, September 1998
8. G. G. Pechanek, S. Vassiliadis, and N. Pitsianis, "ManArray Interconnection Network: An Introduction," EuroPar'99, Toulouse, France, Aug. 31-Sept. 3, 1999.
9. National Semiconductor NAPA 1000 – DARPA ITO Sponsored Research 1988.
www.darpa.mil/ito/psum1998/e257-0.html
10. N. P. Pitsianis and G. G. Pechanek, "High-Performance FFT Implementation on the BOPS ManArray Parallel DSP," International Symposium on Optical Science, Engineering, and Instrumentation, Denver, Colorado, July 18-23, 1999.
11. G. G. Pechanek, B. Schulman, and C. Kurak, "Design of MPEG-2 Function with Embedded ManArray Cores," Proceedings DesignCon 2000 IP World Forum section, Jan. 31-Feb. 3, 2000.