The ManArray™ Embedded Processor Architecture

Gerald G. Pechanek [1] and Stamatis Vassiliadis [2]

[1] Billions of Operations Per Second, Inc. (BOPS®)
Chapel Hill, NC, USA gpechanek@bops.com

[2] Computer Engineering, Electrical Engineering Department,
Delft University of Technology
Delft, The Netherlands stamatis@ce.et.tudelft.nl

Abstract

The BOPS® ManArray™ architecture is presented as a scalable DSP platform for the embedded processor domain. In this domain, ManArray-based processors use a single architecture definition, that supports multiple configurations of processing elements (PEs) from low end single PE to large arrays of PEs, and single tool set. The ManArray (selectable) parallelism architecture mixes control oriented operations, VLIWs, packed data operations, and distributed array processing in a cohesive, independently selectable manner. In addition, scalable conditional execution and single-cycle communications across a high connectivity, low cost network are integrated in the architecture. This allows another level of selectivity that enhances the application of the parallel resources to high performance algorithms. Coupled with the array DSP is a scalable DMA engine that runs in the background and provides programmer-selectable data-distribution patterns and a high-bandwidth data-streaming interface to system peripherals and global memory. This paper introduces the embedded scalable ManArray architecture and a number of benchmarks. For example, a standard ASIC flow DSP/coprocessor core, the BOPS2040, can process a distributed 256-point complex FFT in 425 cycles and an 8x8 2D IDCT that meets IEEE standards in 34 cycles.

1. Introduction

As chip densities continue to improve, demand increases for the low cost integration of high performance, parallel processing systems. For example audio, video, and communication signal processing are but three areas that require very high performance and are in demand for low cost consumer products. For such application domains a number of proposals exist that intend to improve the over-all execution of specialized software. The proposed architectures can be divided in the following three main categories: Specialized, specialized augmented, and general-purpose. A general discussion regarding those classes follows:

Specialized processors: In this class of processors, a specific standard such as MPEG-2 may be assumed, and for such a standard, a processor is designed that uniquely performs this standard's requirements. Several available processors use this approach, for example MPEG-2 Decoders.[1,2]. Additionally, a standard may not be assumed. These processors are targeted to a specific application domain, such as video-recorders and television sets. The programmability of these processors is limited; they can only perform a predefined set of operations. Typical operations executed by these processors are image enhancement, picture-in-picture, and on-screen-display. Examples of these processors can be found in [3,4] and [5].

Specialized augmented processors: In this class of processors, programmability is assumed and no restrictions to standards are imposed. That is, the processing follows the usual general-purpose paradigm of programmability and the instructions set definition. These C-programmable processors are general program processors with extensions that make them particularly suited for media processing. For example the Philips Trimedia architecture and processors [6] use fine-grain extensions to the VLIW processor and include coarse-grain coprocessors for e.g. VLD decoding. Another example of this class of processors is the Texas Instruments Multimedia Video Processor (MVP) [7].

General-purpose processors: The family of general-purpose machines is extended with co-processing capabilities to improve the performance of multimedia applications. This approach follows the traditional extension oriented processing. That is, a general-purpose

1089-6503/00 $10.00 © 2000 IEEE

processor is extended with new architectural features in order to improve a certain application domain, which allows the design of coprocessors specialized for the considered application. Examples of such extensions include the floating point and vector extension of general purpose computing. Examples of this type of extensions with respect to multimedia include:

- Intel MMX (Multi Media eXtensions) [8],
- ALPHA MVI (Motion Video Instruction extensions) [9],
- Sun VIS (Visual Instruction Set) [10] and
- MIPS DME (Digital Media Extensions) [11].

whether an instruction is executed in parallel across the array of PEs or sequentially in the SP. In addition, the instruction set is partitioned into four groups using the high two bits of the instruction format; a control group, an arithmetic group, a load/store group, and a reserved, proprietary instruction group, Fig. 1. Control and branch instructions are executed by the Sequence Processor (SP), which can also indirectly execute local iVLIWs. To minimize the effects of branch latencies, an extensive, scalable conditional execution approach is also defined in the instruction set format. To optimize the use of the distributed register files, the ManArray network [12] is integrated into the architecture providing single-cycle

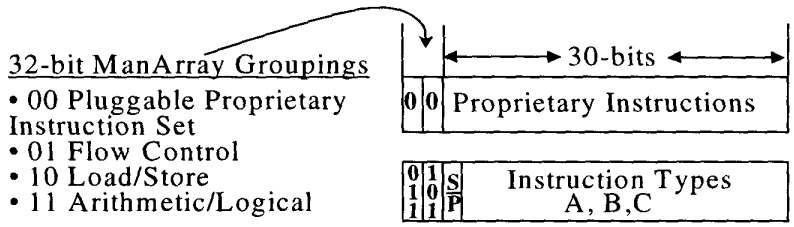

Figure 1 ManArray Instruction Groupings

Our goal is to describe a scalable family of programmable DSP cores based upon an architectural definition for application specific processors containing features of all these categories. The proposed processor family provides three basic levels of parallelism namely indirect VLIW (iVLIW), packed data, and multi-processing. This paper introduces the ManArray DSP platform as a strong contender to be a ubiquitous, high-volume signal processor in commercial applications. The presentation is as follows: The embedded ManArray processor architecture is presented in section 2. Consequently, we describe a ManArray processor design point and present some benchmark performance evaluations. We conclude the presentation with some final remarks.

The Embedded ManArray Processor Architecture consists of a control sequence processor (SP) and an array of processing elements (PEs) that use a distributed register file model with simple instructions defining operations on local register operands. All arithmetic and load/store instructions execute in one or two cycles with no hardware interlocks. Further, all arithmetic and load/store instructions can be combined in VLIW format and stored in small, distributed memories (VIMs). The VLIWs can be indirectly selected for execution from the VIMs. A dedicated bit in all instruction formats controls data transfers between orthogonal clusters of PEs. The communication instructions can also be included into VLIWs, thereby overlapping communications with computation operations, which, in effect, reduces the communications latency to zero. The ManArray network operation is independent of background DMA operations, which provide a data-streaming path between peripherals, such as a global memory. By partitioning the instruction set format into four groups the ManArray architecture reserves more than a quarter of the instruction-set space for future definition. Any of the four groups of instructions can be dynamically mixed on a cycle-by-cycle basis. The single ManArray instruction set architecture supports the entire ManArray family of cores from the single, merged SP/PE core (the 1x1) to any of the highly parallel multi-processor arrays (1x2, 2x2, 2x4, 4x4, etc..). The ManArray instruction set summary, Fig. 2, shows 32-bit simplex instructions in groupings that represent the five execution-unit slots of the first ManArray implementation plus a control group. The execution units include an arithmetic logic unit (ALU), a multiply accumulate unit (MAU), a data select unit (DSU), a load unit, and a store unit.Each PE and the array controller SP contain an iVLIW Instruction Memory (VIM) and each addressable location in the VIM consists of five simplex-instruction slots (one per execution unit) Fig. 3. Using the Load-iVLIW

Cntrl	ALU	MAU	DSU	Load	Store
Group 01	Group 11	Group 11	Group 11	Group 10	Group 10
Call	Add	MPY	Bit	Base+D	Base+D
Jump	Compare	MPYA	Divide	Direct	Direct
Loop	Absolute	MPYC	Copy	Indirect	Indirect
Return	Min/Max	ADD	SP/PEcom	Circular	Circular
Load VLIW	Logical	SUM	Shift/Rotate	Table	Table
Execute VLIW	Butterfly	Butterfly	Permute	Immediate	
			Extract	Broadcast	
				Address	

Figure 2 ManArray Instruction Set Summary

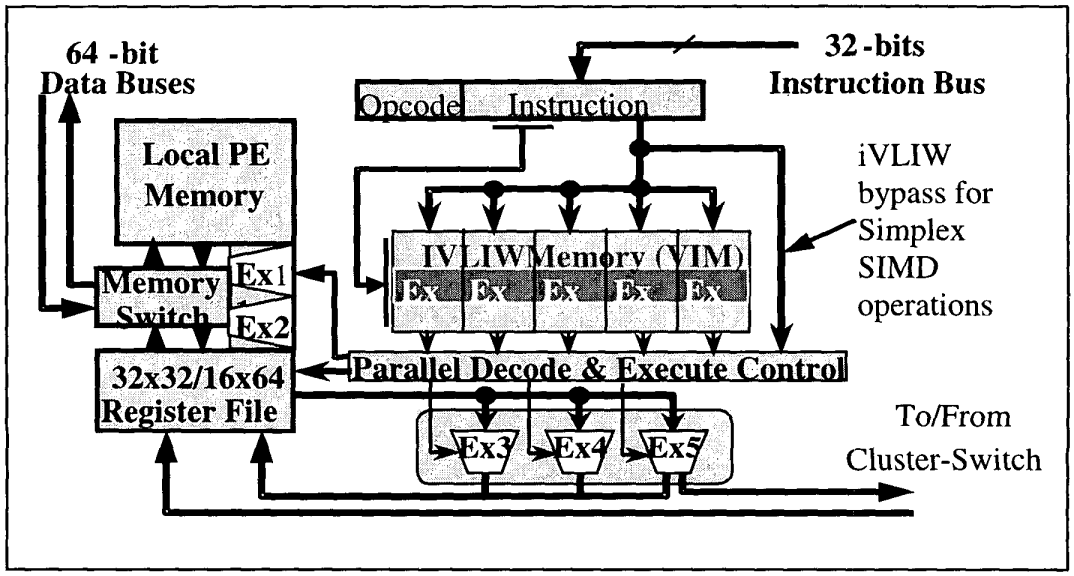

Figure 3 indirect VLIW Architecture in the Spand each PE

instruction (LV), the programmer or compiler loads individual instruction slots with the 32-bit simplex instructions optimized for the algorithm. A single 32-bit Execute iVLIW instruction (XV) triggers the execution of an iVLIW at a specific VIM address in the SP, or in all PEs. Instructions in the five slots execute in parallel, each within its respective execution unit. Consequently, the VIM contains application-specific iVLIW s that are optimized for a specific task and that can be easily changed for subsequent tasks.

Next, two variations of a PE communication instruction and a Multiply Complex instruction are reviewed. Fig. 4 shows a PE exchange instruction (PEXCHG Rt, Rs, Transpose) in operation. The figure shows a 2x4 cluster-switch with two levels of multiplexing connecting the PEs and a symbolic depiction of the operation in the upper left hand corner. For the transpose operation on a 2x2 with row-major PE ordering as shown, a transfer of data between PEs 1 and 2 is to occur. This is a single cycle DSU operation which can be specified in an iVLIW and uses the receive model of communications. In the receive model each PE makes a specified source operand available to its output port and provides the proper control signals to its path control portion of the cluster switch. In the example shown, PE 1 provides Rs1 to its output port and PE2 provides Rs2 to its output port. PE 1 receives the Rs2 value through its cluster switch multiplexors and PE 2 receives Rs1. The received values

are loaded into the specified target register in each PE. Fig. 5 shows a PE exchange (PEXCHG Rt, Rs, 2x4PE6) operation on a 2x4, demonstrating the broadcast capability of the ManArray network and receive model of communications used. The symbolic depiction in the upper left hand corner shows that PE6 is the source of the 2x4 array. It should be noted that all inter-cluster paths are primarily "point-to-point" supporting the high performance interface.

Fig. 6 depicts the pipelined operation sequence for a Multiply Complex Instruction. For ManArray high performance PEs, Multiply Complex provides superior

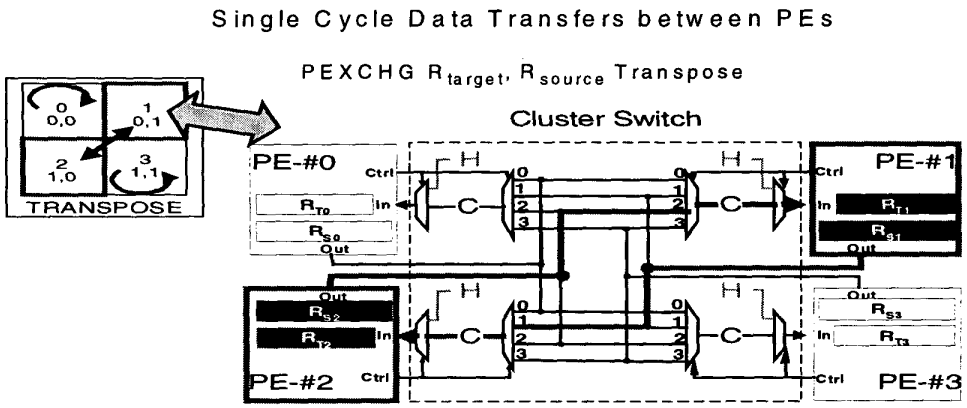

Figure 4 Single-Cycle Data Transfers Between PEs

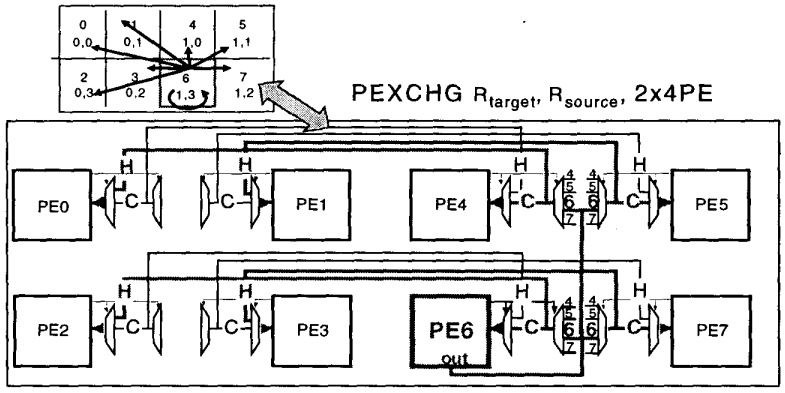

Figure 5 PE Broadcast Between PEs

data to be broadcast to all active PEs in the 2x4. This single-cycle DSU operation can also be specified in a iVLIW. In this case PE6 makes its source operand available and each of the other active PEs in the array specify the proper path control to receive PE6's data. This is a cross cluster broadcast of data to all PEs in the performance support for algorithms that require complex multiply operations. Complex numbers are stored in two halfwords where H1 represents the real component and H0 the imaginary component. The complex result returned contains the real part in H1 and the imaginary part in H0. This is a two cycle operation with rounding

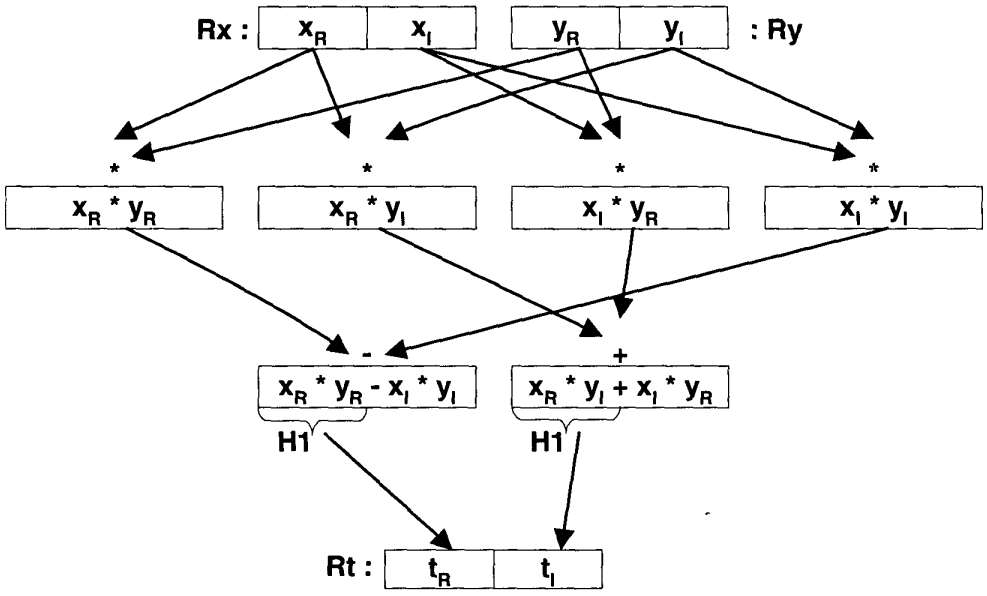

Figure 6 Multiply Complex Instruction Pipeline

occurring in the second cycle. By pipelining, 2 cycles latency and single-cycle throughput are achieved.

2. ManArray-based DSP Cores

Generally speaking, ManArray defines building blocks that can be combined into high-performance DSP Cores with different processor-array configurations. The basic building blocks are

- The SP/PE0 - the array controller SP merged with PE0 of the array
- Individual PEs
- Configuration-specific cluster-switches to connect the PEs
- A powerful scalable DMA engine

These basic building blocks can be combined into array building blocks. For example an SP/PE0 with three additional PEs, and cluster-switch creates a 2x2-array building block. Such a 2x2 building block can be combined with other building blocks (additional PEs or additional arrays), or it can be a used by itself as the basis for a DSP core, like in the BOPS2040 DSP core in Fig. 7 (shown without DMA). Combining multiple 2x2 building blocks into larger arrays, such as 2x4s and 4x4s, also enables software controlled reconfiguration options. For example a 4x4 array can be programmed to act as 4 independent 2x2s.

The SP, as part of the merged SP/PE building block, provides program control, contains the instruction and data address generation units, and controls the dispatch of instructions to the processor array. Merging the controller SP into the array of PEs allows the SP access to the PE-to-PE interconnection network and consequently easy access to each PEs internal resources. A rich set of addressing modes is provided in the SP and each PE, and the SP also supports high-performance interrupts and unique event-point looping.

Each SP and PE uses a 64-entry register space that is configured as a 32x32 Reconfigurable Compute Register File (RCRF), 8x32 Address Register File (ARF), and 24x32 Miscellaneous Registers (MRF). In ManArray, the address registers are separated from the compute register file. This approach maximizes the number of registers available for compute operations and guarantees a minimum number of dedicated address registers. This does not require any additional address generation ports from the compute register file, and it allows independent PE memory addressing for such functions as local, data-dependent table lookups. The RCRF is configurable to support both 32-bit data types including quad byte, dual halfword, and word, and 64-bit data types including octal byte, quad halfword, dual word, and double word. The balanced architectural approach taken for the compute register file provides the high performance features needed by many applications. It supports octal byte and

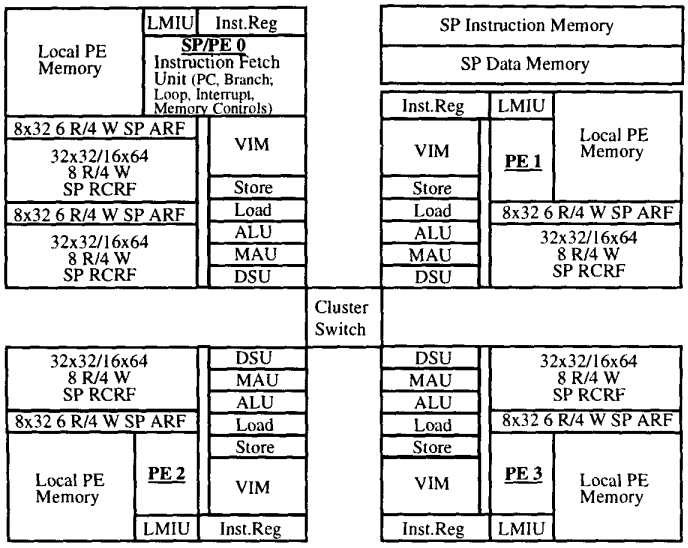

Figure 7 BOPS2040 2x2 ManArray DSP Core

quad halfword operations in a logical 16x64-bit register file space without sacrificing the 32-bit data type support in the logical 32x32-bit register file. Providing both allows optimum usage of register file space and minimum overhead in manipulating packed data.

In Fig. 7, each PE contains five execution units, identified in the instruction set groupings of Fig. 2. Each execution unit supports various 8/16/32/64-bit packed data types, a 32x32-bit reconfigurable compute register file, a VLIW-Instruction-Memory unit (VIM), and local data memory. The DSU supports shifts, rotates, and single-cycle PE-to-PE communications across the ManArray network. With the indirect VLIW (iVLIW) architecture, the communications operations can be overlapped with the compute operations, thereby providing *zero-latency* data transfers between PEs. The 32x32-bit reconfigurable register file is multi-ported, which allows the parallel iVLIW execution. The load and store units provide independent data paths between the local memories in each PE in the array. The local PE data memory is a two-port memory that is configured into two 32-bit banks, that support byte, halfword, word, and double word loads. 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/storing to/from the other bank. This effectively hides DMA delays and supports a data streaming approach to processing on the array, which allows very high memory bandwidth support for compute-intensive algorithms.

Each of the five execution units on each PE on the array supports up to 64-bit packed data types, which makes ManArray's data parallelism very flexible. Adding PEs to an array increases the overall available data parallelism, such that a 1x2 array (1 SP/PE + 1PE) in effect provides 128-bit packed data support, a 2x2 (1 SP/PE + 3PEs) 256-bit packed data support, etc. Yet, on the same array the programmer can also choose to process different streams of data in each PE. So the appropriate level of data parallelism can be used, without sacrificing any processing bandwidth.

The VIM can contain an unlimited number of iVLIWs, though products for video compression, graphics, signal processing, or communications typically require no more than a 64-entry VIM. Local PE data memory is not limited by the architecture but is also sized according to the intended applications.

Conditional execution, also referred to as predicated execution, allows the programmer to have a given instruction execute or not, based on the machine state generated previously. This feature improves performance by avoiding the use of branches that incur a branch delay penalty on pipelined processors. ManArray extends the concept of conditional execution to an array of processing elements by specifying the type of conditional execute action desired in a hierarchical

manner, with a 1-bit, 2-bit, and 3-bit specifier depending upon the instruction type. In addition, ManArray supports conditional execution within the iVLIWs. The ManArray PE independent conditional execution architecture provides a comprehensive level of data-dependent parallelism that none of the currently known and available SIMD machines provide. The standard conditional branch capability is also provided in the SP, as well as SP-local, data-dependent conditional execution in all its arithmetic operations. Initially, four basic types of operations are supported in ManArray cores, namely unconditional execution, execute on true-condition, execute on false-condition, and specialized conditional execution actions specified on an instruction basis.

3. Benchmarks

The ManArray architecture contains application specific instructions for video, graphics, communications, and generalized DSP applications. The first silicon is a 2x2 core that includes both fixed and floating point ALU, MAU, and DSU and all the architectural features discussed in this paper for use on an evaluation board. Subsets of the architecture can be prepared to optimize a core for a specific task, if it is seen as necessary. Consequently, BOPS is developing application libraries of algorithms for use by product developers. Many of these are reported on the BOPS web site [14]. Two example algorithms are briefly discussed next.

In this first algorithm, discussed in detail in [15], an 8x8 two-dimensional inverse discrete cosine transform (8x8 2D IDCT) is executed on the BOPS2040 core in 34-cycles. The algorithm is developed from the 2D 8x8 IDCT algorithm by separating it into 1D 1x8 IDCTs on all eight rows and then on all eight columns. The 8x8 data is distributed among the four PEs, according to the 1x8 IDCT signal flow graph. The strategy for processing the IDCT algorithm makes use of a number of features of the ManArray architecture to provide a unique software pipeline using indirect VLIWs operating in multiple PEs. For example, the indirect eXecute VLIW (XV) instruction, with the unique enable feature, allows a software pipeline to be built up and torn down without creating new VLIWs. Quad 16x16 multiplications are used on each PE in the Sum-of-Two-Products with and without accumulate instructions to produce two 32-bit results on each PE. The use of common instructions that can execute in either the ALU or MAU or both, for example the butterfly-with-saturate instruction improves performance. In addition, the DSU supports permute instructions to aid in organizing the data prior to processing and PEXCHG instructions for communicating between PEs. Tests on the 34-cycle algorithm confirm that the 2D 8x8 IDCT meets the IEEE standard 1180-1990 for precision of the IDCT.

In this next algorithm, discussed in detail in [16], a 256 point complex FFT is executed on the BOPS2040 core in 425-cycles. Our implementation utilizes the multiple levels of parallelism that are available on the ManArray such as the special multiply complex instruction, that calculates the product of two complex 32-bit fixed-point numbers in 2 cycles pipelineable for single-cycle throughput. Instruction level parallelism is exploited via the indirect VLIW. With the iVLIW, in the same cycle

- a complex number is read from memory,
- another complex number is written to memory,
- a complex multiplication starts and another finishes,
- two complex additions or subtractions are done, and
- a complex number is exchanged with another processing element.

Both PE-local FFTs and distributed FFTs are executed in this algorithm which is described by use of Kronecker product mathematics. Not only are the total cycles for the calculation of the 256-point complex FFT very small but the program code is also very compact.

4. Conclusions

In this paper we have presented the ManArray architecture, an innovative architecture for DSP core processing. The proposed architecture is especially suited for emerging applications such as broadband communications, digital video, imaging and graphics. Based on this architecture BOPS is currently developing a hardware implementation platform, software and a programming environment [17]. The ManArray provides three basic levels of parallelism (indirect VLIW, packed-data, and multi-processing), all independent of each other and available to the compiler or programmer on an as-needed basis. The combination of these features enables a 2x2 ManArray-based DSP like the BOPS2040 to produce

- a distributed 256-point FFT in 425 cycles, using complex numbers of 32 bits (16 bits for real and imaginary parts), and
- an 8x8 IDCT that meets IEEE standards for precision in 34 cycles.

And finally, since these emerging markets are primarily System-On-Chip markets, BOPS is providing the ManArray as licensable IP in the form of Cores, SW, and Programming Tools.

REFERENCES

[1] SGS Thomson Microelectronics, MPEG audio/MPEG-2 Video intergrated decoder Sti3520. *http://www.st.com*, October 1996.

[2] SGS Thomson Microelectronics, MPEG 2.5 Layer III audio decoder STA013. *http://www.st.com*, September 1999.

[3] A. Van Roermond, P. Snijder, H. Dijkstra, C. Hemeryck, C. Huizer, J.Schimtz, and R. Sluijter. A Genera;-Purpose Programmable Video Signal Processor. *IEEE Transactions on Consumer electronics*, 1989.

[4] J. Goto, K. Ando, T. Inoue, M. Yamashina, H. Yamada, and T. Enomoto. 250-Mhz BiCMOS Super-High-Speed Video Signal Processor (S-VSP) ULSI. *IEEE Journal of Solid-State Circuits*, 1991.

[5] P.A. Ruetz, P. Tong, D. Bailey, D.A. Luthi, and P.H. Ang. A High-Performance Full-Motion Video Compression Chip Set. *IEEE Transactions on Circuits and Systems for Video Technology*, 1992.

[6] S.Rathnam and G. Slavenburg. An Architectural Overview of the Programmable Multimedia Processor, TM-1. *In Proceedings of COMPCON '96*, pages 319-326. IEEE, 1996.

[7] K. Guttag, R. J. Gove, and J.R. van Aken. A single-chip multiprocessor for multimedia: The MVP, *IEEE Computer Graphics and Applications*, November 1992.

[8] A. Peleg and U. Weiser. MMX Technology Extension to the Intel Architecture. *IEEE Micro*, 16(4):42-50, August 1996.

[9] P. Rubinfeld, B. Rose, and M. McCallig. Motion Video Instruction Extensions for Alpha, *http://www.digital.com/semiconductor/papers/pmvi-abstract.htm*, October 1996.

[10] M. Tremblay, J. M. O'Conner, V. Narayanan, and L. He. Vis speeds new media processing. *IEEE Micro*, 16(4):10-20, August 1996

[11] MIPS digital media extension. *http://www.sgi.com/MIPS/products/*.

[12] G.G. Pechanek, S. Vassiliadis, N. Pitsianis. ManArray Processor Interconnection Network: An Introduction. In *Proceedings of 5th Int'l EURO-PAR Conference* Toulouse, France, August/September 1999

[13] BOPS, Inc., BOPS 2040 core data sheet, http://bopsnet.com/cores_bops2040.shtml

[14] BOPS, Inc., BOPS benchmarks, http://bopsnet.com/benchmarks.shtml

[15] G. G. Pechanek, B. Schulman, and C. Kurak. Design of MPEG-2 Function with Embedded ManArray Cores. In Proceedings of DesignCon 2000 IP World Forum Santa Clara, California, January/February 2000.

[16] N. P. Pitsianis and G. Pechanek. High-Performance FFT Implementation on the BOPS ManArray Parallel DSP. In proceedings International Symposium on Optical Science, Engineering, and Instrumentation, Denver, Colorado, July, 1999.

[17] BOPS, Inc., BOPS Software Development Kit (SDK), http://bopsnet.com/tools.shtml