Cost-Effective Low-Power Processor-In-Memory-based Reconfigurable Datapath for Multimedia Applications

Marco Lanuzza*
DEIS – University of Calabria
-87036- Rende (CS)
ITALY
lanuzza@deis.unical.it

Martin Margala
ECE – University of Rochester
Rochester NY 14627-0231
USA
margala@ece.rochester.edu

Pasquale Corsonello
DEIS – University of Calabria
-87036- Rende (CS)
ITALY
p.corsonello@unical.it

ABSTRACT

Multimedia applications have become a dominant computing workload for computer systems as well as for wireless-based devices. Due to their repetitive computing and memory intensive nature, they can take effective advantage from Processor-In-Memory (PIM) technology. In this paper, a new low-power PIM-based 32-bit reconfigurable datapath optimized for multimedia applications is presented. The new circuit efficiently performs parallel arithmetic operations on either 8-, 16-, or 32-bit integer data or on 32-bit single precision floating-point data. As a result, high flexibility is provided at a very low hardware cost. When implemented using the UMC 0.18 μm 1.8 V CMOS technology, the proposed datapath exhibits a 285 MHz running frequency, dissipates just 0.12 mW/MHz and occupies a silicon area of only 107,323 μm^2. When performing 2D-DCT, proposed architecture consumes 74% less power and is 28% more power efficient compared to top-of-the-line commercial TI DSP.

Categories and Subject Descriptors

B.7.1 **[Integrated Circuits]**: Types and Design Styles – *advanced technologies, algorithms implemented in hardware, VLSI (very large scale integration).*

General Terms: Performance, Design.

Keywords

Processor-In-Memory, Datapath, Reconfigurable Computing.

1. INTRODUCTION

In complex computational systems, the Central Processing Unit (CPU) and the main memory are conventionally implemented on different chips. Thus, to perform elaborations data are moved (through chip I/O pins and copper wires on the PCB) from the main memory to the processor cache, to the processor registers and functional units, and then back along the same path to the main memory. There are two main performance bottlenecks in this elaboration process.

First, moving data through the memory hierarchy a significant latency is introduced. This is exacerbated by growing processor-memory performance gap [1]. Second, the bandwidth of the external bus between processor and memory is significantly limited at both chip and board levels. Performance penalties due to the bottlenecks described above, arise especially in data-intensive elaborations [1], such as multimedia applications.

Processor-In-Memory (PIM), also known as Intelligent Ram (IRAM), is a recent design trend [2, 3, 4, 5, 6, 7] which attempts to integrate processing logic and memories into the same chip. This approach allows a significant portion of the computations (especially simple and regular operations) to be directly performed in memory, avoiding time and power consuming communications between the CPU and the main memory.

In order to actually take advantage from PIMs, logic and memory have to be more highly integrated than simply located on the same chip [3]. More precisely, to preserve the overall system's cost it is crucial to maximize the computational parallelism with minimal silicon area overhead. Recent works [5, 8] have demonstrated that using PIMs represents an opportunity for implementing algorithms in hardware in multimedia applications. In fact, the latter have inherent high degree of parallelism and frequently require reiterated computations on large streams of data (often with little temporal data reuse). Using PIMs, massive parallelism, high resource utilization, low power consumption and real time elaboration would be provided.

In this paper, a new 32-bit reconfigurable PIM-based datapath for multimedia applications is presented. The proposed architecture is composed by four low-complexity 8-bit Processing Elements (PEs) that operate in a Single Instruction Multiple Data (SIMD) fashion. Because of this property, the new circuit supports parallel arithmetic operations on multiple integer data types with variable word lengths (i.e. 8-, 16- and 32-bit) and IEEE-754 [11] compliant single precision floating point arithmetic computations. Floating-point elaboration is crucial for many media signal processing algorithms. In [8], a floating-point unit targeted for PIM systems has been presented. This circuit is associated with a SIMD integer unit. While this solution enhances the flexibility of the overall architecture and assures a very high computational capability, it results in a considerable growth of die area and power dissipation. In our datapath, floating-point operations are supported by sharing some of the major hardware resources within the SIMD integer unit. This leads to a very efficient hardware use with reduced silicon area and average power dissipation while maintaining full functionality and a very high processing capability.

The proposed datapath also exhibits a very high modularity. This allows cost-effective low-power massively parallel PIM-based

ISLPED'05, August 8–10, 2005, San Diego, California, USA.

* This research activity has been carried out during the author's summer internship at the University of Rochester in 2004.

architectures to be realized by using large arrays of memory block/processing logic pairs in either Single Instruction Multiple Data or Multiple Instruction Multiple Data (MIMD) mode.

The remainder of the paper is organized as follows. In Section 2, a description of the proposed 32-bit PIM-based reconfigurable datapath is provided. The possible operation modes of the new architecture are described in Section 3. Implementation and comparison results are presented and discussed in Section 4. Finally, examples of applications are considered in Section 5.

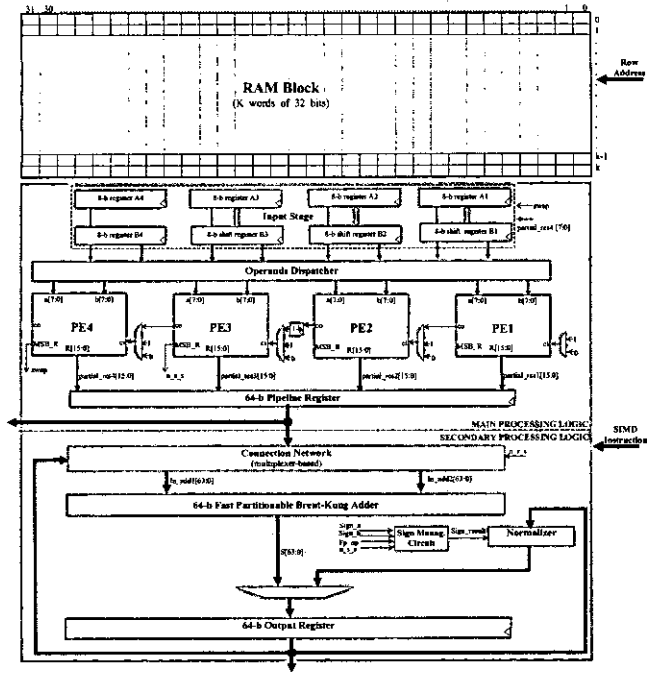

Figure 1. The proposed datapath.

2. THE RECONFIGURABLE DATAPATH

The presented PIM-based datapath has been designed to efficiently handle various integer and floating point data types with variable word lengths such as IEEE-754 compliant single precision floating-point data exploiting hardware reuse to guarantee a low-cost implementation.

The top-level architecture of the proposed 32-bit reconfigurable datapath is depicted in Figure 1. The input stage is placed directly at the memory row buffer. In this way, all the bits of a memory block row (i.e. 32-bit word) can be acquired in a single memory access. The input stage consists of eight 8-bit registers (i.e. *register A1-A4* and *register B1-B4*) plus a little logic needed for input data formatting. The proposed circuit is structured as two stages of processing logic. In the main processing logic stage the *Operands Dispatcher* selects the 8-bit subsegments of the loaded 32-bit words which have to be elaborated by each PE.

The PE is the main computational unit of the circuit. It is based on an efficient cost effective multiplier decomposition based arithmetic unit [5] that enables several different 8-bit operations to be executed by exploiting hardware reuse. Note that four PEs are linked using a multiplexer based carry management logic. The latter allows for carry propagation for higher precision (i.e. 16-, 32-bit and floating-point) SIMD addition-based operations.

The secondary processing logic stage has been specifically designed for combining partial results generated by the PEs for higher precision (i.e. 16-, 32-bit and floating-point) multiplication operations. The *Connection Network*, implementing three logic level of multiplexing stages, dispatches the correct operands to the final adder. This latter is a fast partitionable Brent-Kung-based adder that can compute one 64- or two independent 32-bit additions. The second processing logic stage includes also circuitry for normalization and sign computation used for floating-point operations.

It is important to point out that the proposed datapath has a very high scalability. In fact, as depicted in Figure 2, it allows a slice-based approach to be exploited arranging k 32-bit wide word memory block/32-bit processing logic pairs into a massively parallel PIM based architecture. This offers the flexibility for supplying the processing power and the memory bandwidth needed for advanced multimedia applications.

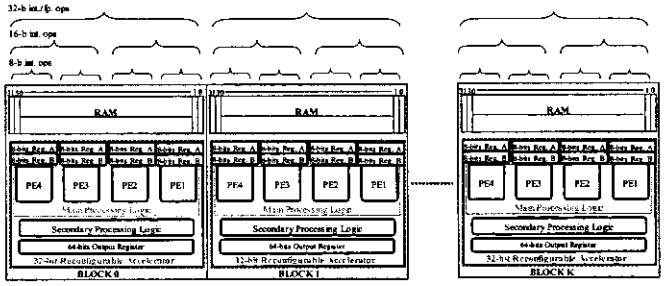

Figure 2. The overall PIM based architecture.

2.1 The Processing Element

The architecture of the proposed 8-bit PE is shown in Figure 3. It consists of two small 4x4-bit multipliers, four carry linked 4-bit simple Ripple Carry Adder (RCA) (in order to reduce the critical path of the overall circuit, two of them are linked according the carry-skip technique) and some auxiliary logic needed for data exchange between the arithmetic blocks. This simple structure allows several 8-bit operations to be performed under the control of the instruction vector *I*.

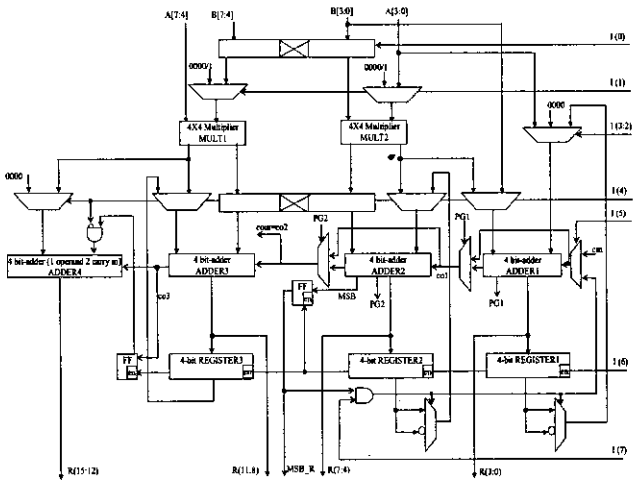

Figure 3. The Processing Element block diagram.

The 8-bit addition requires one instruction cycle to be executed using the carry linked adders *ADDER1* and *ADDER2*. The first performs the operation A[3:0] + B[3:0] + 0. The generated carry out

signal *co1* is used as carry in signal for the ADDER2 to execute the operation (A[7:4] * 0001) + (B[7:0] * 0001) + *co1*.

Subtraction is performed in a similar manner, but with one of the two inputs complemented and the carry in signal *cin* forced to the logic value 1. The result of both addition and subtraction operations generates an 8-bit result R[7:0].

The 8x8-bit multiplication requires two instruction cycles to be executed by exploiting the reusing of the two 4x4-bits multipliers [5] (i.e. *MULT1* and *MULT2*). In the first instruction execution cycle the first set of partial product A[3:0]*B[7:4] and A[7:4]*B[3:0] are computed and added by the two carry linked 4-bit *ADDER2* and *ADDER3*. The 8-bit result of the addition of these partial products is then stored in the two 4-bit register *REGISTER2* and *REGISTER3*. At the same time also the possible carry out *co3* is saved. In the next instruction execution cycle the partial products A[3:0]*B[3:0] and A[7:4]*B[7:4] are computed and accumulated with the result of the first iteration to generate the final 16-bit correct product viable in *R[15:0]*.

In order to enlarge the functional capability of the original PE [5], two new instructions (i.e. absolute value of the difference and of the summation operations) useful to speed up several media signal processing algorithms have been added. Similar to the 8x8-bit multiplication the 8-bit absolute values of the difference and of the summation require two instruction executions. In the first instruction execution the two's complement difference or summation (i.e. $D = A + \overline{B} + 1$ or $S = A + B$) is computed using the carry linked *ADDER1* and *ADDER2* and then stored in the two 4-bit registers *REGISTER1* and *REGISTER2*. In the second instruction execution two cases have to be considered. If the previously stored partial result is negative (i.e. *MSB_R='1'*) the computed value is negated and also a carry in signal is added in *ADDER1* to obtain an exact representation of the magnitude of the result otherwise the result didn't change on the subsequent instruction execution. It is worth noting that the absolute value instructions have been added at expense of just nine 2-1 multiplexers and only one AND gate without compromising the critical path of the previously proposed PE.

Table 1. Supported Instructions

Instruction	Parallelism Level	Throughput	Latency
add/sub8	4	1	1
mul8	4	2	2
abs_add/sub 8	4	2	2
add/sub16	2	1	1
abs_add/sub16	2	2	2
mul16	2	4	5
add/sub32	1	2	2
abs_add/sub32	1	4	4
mul32	1	8	9
fp_add/sub	1	5	7
fp_mul	1	6	8

3. SUPPORTED ARITHIMETIC OPERATIONS

The proposed datapath is configured through apposite SIMD instructions coming from the external control logic. Each SIMD instruction specifies an instruction opcode for the PEs enabling different operations to be executed.

The possible operation modes of the new architecture are summarized in Table 1. It is worth noting that for all supported operations the execution in the last secondary processing logic can be overlapped with the execution in the main processing logic. For example, as shown in Figure 4, it may be possible for a 16x16-bit integer multiplication to start once a floating-point addition is in its sixth cycle since the addition data is in secondary processing logic stage and not in the first. Likewise, another operation may start in the fifth cycle of a 16x16-bit integer multiplication because it is no longer accumulating partial products in the upper stage.

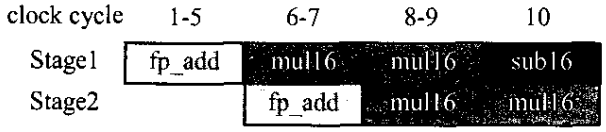

Figure 4. Operation timing diagram.

3.1 Integer Arithmetic Operations

Each PE in the first processing logic stage can be used, as described in Section 2.1, to perform independent 8-bit integer arithmetic operations thus ensuring a processing parallelism level of four. In this case, there is no carry linking between the four PEs. It is worth pointing out that all 8-bit integer arithmetic operations present a throughput corresponding to the instruction execution cycles on the first processing logic stage.

Two parallel 16-bits integer arithmetic operations are also supported by the proposed circuit. 16-bit addition/subtraction and absolute value of the summation/difference instructions are performed in the first processing logic stage using the two couples of carry linked PEs (i.e. *PE2-1* and *PE4-3*). All these instructions have the same throughput of the corresponding 8-bit instruction executions.

In general, a 16X16-bit binary multiplication can be computed by combining the 16-bit subproducts between the most significant and the least significant 8-bit operand subwords (A=A[15:8] A[7:0]; B=B[15:8] B[7:0]) [12]. This approach suggests that two PEs can be used in parallel to compute independently the 16-bit subproducts, which are recursively added and accumulated in the secondary processing logic stage by a half portion of the *64-bit Partitionable Brent-Kung Adder*. This scheme allows larger multiplies to be computed without the need for bigger, dedicated multipliers. The penalty for this area savings is increased latency. However, the increased cycle count is balanced by the improvement in data parallelism of the proposed design. In fact, two 16*16-bit multiplication operations can be computed in parallel using simultaneously the two halves of the datapath.

The 32-bit addition/subtraction and the absolute value of the summation/difference operations are computed by the first processing logic stage using the four carry linked PEs. With respect to the corresponding 8- and 16-bit operations, these instructions present an additional instruction cycle of latency due to the flip-flop inserted in the carry chain between the *PE3* and the *PE2*.

The 32X32-bit multiplication instruction can be executed using the above mentioned merged hardware technique. This operation

execution requires that sixteen 16-bit subproducts are computed by the four PEs and then recursively added and accumulated in the secondary processing logic.

3.2 Floating Point Arithmetic Operations

Floating-point elaboration capability is provided by exploiting SIMD integer execution unit cooperatively with support circuitry needed to deal with the signs, alignment preshift and normalization postshift. Although floating point instructions cannot proceed in parallel with the execution of other SIMD integer operations, this approach enables to reduce significantly processing logic and power dissipation thus guarantying a very high processing capability.

Addition of two 32-bit single precision floating point numbers A and B can be executed by the proposed datapath using the conventional algorithm [12]. When a floating point operation is required, the two operands are loaded and unpacked in the *Input Stage* of the datapath. More precisely, the 8-bit operand exponents are loaded in 8-bit register A4 and B4 respectively, whereas the fractional parts with the leading 1 made explicit (i.e. the significand 1.fraction) are loaded into the 24-bit concatenated registers, *register A3-A1* and *register B3-B1* respectively. Instead, the operand signs (i.e. *sign_a* and *sign_b*) are read from the *Sign Manag. Circuit* that will use their logic values to evaluate the correct sign of the result.

The absolute difference of the two exponents (i.e. $abs(\exp A - \exp B)$) computed by the *PE4* is used to determine the amount of alignment right shift and the operand to which it should be applied.

In order to economize on hardware, preshifting capability is provided only for operand B. For this reason, if it is the operand A that needs to be shifted, the two operands are previously swapped according to the logic value of the *swp* signal. After the preshifting, the two significands are added using linked PE3-1. Note that if the operands present a different sign, the significand of the operand B is 2'complemented at the same time of the addition in the PEs (i.e. in the PE3-1 is performed the operation $significandA + \overline{significandB} + 1$). The resulting significand is available in 2's complement representation in *partial_res3[7:0] link partial_res2[7:0] link partial_res1[7:0]*, whereas the resulting exponent (i.e. the larger exponent of the two operands) is available in *partial_res4[15:8]*. Moreover, a single bit signal *n_s_s* indicates the most significative bit of the significands adding result. If *n_s_s* is '1' and the input operands have a different sign, the result is negated and a carry in signal is added in the secondary processing logic adder to restore the sign magnitude format otherwise the result is stored unchanged in the *Output Register*. After the significands are added, a normalization step is performed by the *Normalization Unit* using an additional clock cycle.

The floating point multiplication is executed using the PE-4 for adding of the exponents and the PE3-1 to compute the 16-bits subproducts of the 24X24-bit multiplication of the significands. The latter are recursively added and accumulated in the second processing logic as described for the case of integer multiplications and the resulting 48-bits are truncated down to 24 bits. Then a normalization step is performed to restore the IEEE-754 compliant format.

4. IMPLEMENTATION RESULTS

In order to ensure a process independent design and to enable rapid technology and library migration, an appropriate HDL code has been developed for the new 32-bit reconfigurable accelerator in memory. All the implemented operations have been functionally verified using ModelSim VHDL Simulator.

Therefore, the circuit has been synthesized by Synopsys Design Compiler using the UMC 0.18μm 1.8V CMOS standard cells library. Finally, using the Silicon Ensemble from the Cadence tool, the final layout design has been carried out (see Figure 5). In this phase, an appropriate clock tree also has been designed to minimize the clock-skew effects.

PrimeTime and PowerCompiler from Synopsys have then been used for post-layout measurement of the worst-case delay and the power consumption, respectively. For the power measurement, the uniform distribution of the operands has been assumed.

Post-layout characterization results show that the new datapath can efficiently support both variable precision integer and floating point computations. It is capable of running up to 285 MHz at 1.8 V with an average energy consumption of only 0.12 mW/MHz and with a silicon area of just 107,323 μm^2.

In Table 2, the proposed circuit has been compared with architectures described in [8] and in [13]. Moreover, also the presented 32-bit SIMD datapath without floating point elaboration capability has been considered. It can be observed that the choice of sharing functional units for both SIMD integer and floating point operations involves a minimal increase of the silicon area and power dissipation of about 20% without sacrificing the critical path of the circuit. However, the minimum increase in circuit complexity is balanced from the overall improved versatility.

Even though, both of the circuits presented in [8] and in [13] assure higher computational capability, they appear more area and power-hungry with respect to the presented architecture. More precisely, the reconfigurable datapath for multimedia applications proposed in [13] sacrifices silicon area and power dissipation by using full dimension arithmetic circuits to increase the throughput in elaborating highest precision data (i.e 32-bit data). Note that the PIM oriented arithmetic unit presented in [8] is also referenced, even though comparison with it is not effective, because it provides support only for floating-point operations.

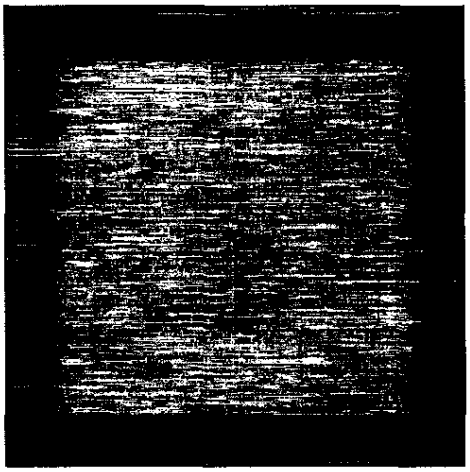

Figure 5 Layout of the proposed datapath.

Table 2. Comparison Results

	Tech.	Gate Count	Speed (Mhz)	Area (μm^2)	Energy (mW / Mhz)	SIMD integer instr.	Floating-Point support	Throughput/Latency for 32*32-bit int. multiplication (# clock cycles)	Throughput/Latency for floating-point addition (# clock cycles)
proposed with floating point support	**0.18 µm CMOS**	**3,856**	**285**	**107,32**	**0.12**	**YES**	**YES**	**8/9**	**5/7**
prop. without floating point support	**0.18 µm CMOS**	**3,076**	**285**	**85,87**	**0.10**	**YES**	**NO**	**8/9**	**N.A.**
[13]	**0.25 µm CMOS**	**N.A.**	**200 ****	**N.A.**	**N.A.**	**YES**	**YES**	**1/2**	**1/2**
[8]	**0.18 µm CMOS**	**8,446***	**300**	**278,21 ***	**0.6**	**NO**	**YES**	**N.A.**	**1/5**

*It has been obtained from data given in [8] ignoring the logic for the divide instruction support; ** post-synthesis results.

5. EXAMPLES OF APPLICATIONS

In order to demonstrate the efficiency of the proposed circuit a frequently used multimedia application has been chosen as a frame of reference: image compression.

In image compression standards such as JPEG and JPEG2000, the discrete image transforms are a very important step of the coding task. Due to their computationally and bandwidth intensive nature, discrete image transforms are well suited for implementation on the proposed architecture.

In JPEG standard, the 2D-Discrete Cosine Transform (2-D DCT) is performed on an 8x8 block of pixels. Thanks to the separability property, the computational complexity of the 2-D DCT can be reduced by decomposing it into separate 1-D DCT processes [14]. A 1-D DCT is computed on each row (8 rows), and again on each column (8 columns). It results in 16 8-point 1-D DCTs for an 8x8 2-D DCT. Several algorithms such as the Chen DCT [15] have been proposed in the literature for fast 1-D DCT implementation. However, these algorithms often involve high-complexity and irregular hardware architectures. Instead, the straightforward 1-D DCT here referred [16] possesses a very regular computational flow. Moreover, it presents an inherent high degree of parallelism that can be efficiently exploited from the proposed datapath.

In [16] has been evaluated that the computation of an 8-point 1-D DCT requires 8 8-bit integer additions, 24 16-bit fixedpoint additions, and 32 8-bit fixed-point multiplications. Assuming that a single 32-bit reconfigurable datapath is responsible for an 8x8 pixel space, 52 clock cycles are needed to execute a single 8-point 1-D DCT. This translates to 832 clock cycles to compute an entire 8x8 2-D DCT. At a 3.5 ns clock, the entire 8x8 2-D DCT computation requires 2.9 µs.

The proposed architecture has been characterized also for the 2D-Discrete Wavelet Transform (2-D DWT) execution. The 2-D DWT computation requires an apposite control unit used to determine the operations to be performed and the storage of intermediate results.

The 2-D DWT can be performed on an entire image or on independent rectangular blocks of the image known as tiles. It is usually computed by first applying 1-D DWT to the rows of the image, and then repeating the same filtering on its columns. Moreover, 2-D DWT has to be computed many times on the same image when multiple levels of decomposition are needed. Both Daubechies (5,3) and (9,7) filters [17] have been considered for 2-D DWT computations.

As shown in [18], the computational flows for Daubechies (5,3) and (9,7) can be made very regular and parallelizable. However, while two wavelet values can be elaborated at the same time by the proposed datapath (i.e. similarly to the DCT computation exploiting the 16-bit fixed-point computation capability of the circuit) for Daubechies (5,3) filtering, only one wavelet coefficient can be processed at a time for Daubechies (9,7) filtering. This is because the Daubechies (9,7) filtering requires 32-bit floating-point elaboration to prevent inaccurate results.

For a typical tile size of 32x32 pixels, the first level of wavelet decomposition requires 32 1-D DWTs applied horizontally and then 32 applied vertically. With the (5,3) filter, this amounts to 1024 16-bit additions (512 for each dimension) and 512 16-bit multiplications (256 for each dimension) [18]. The (9,7) is almost twice as large with 2048 floating-point additions and 1536 floating-point multiplications (1024 and 768 for each dimension, respectively) [18]. After each level x of decomposition, the amount of additions and multiplications exponentially decreases by a factor of 2^x.Considering three levels of decomposition and assigning two 32-bit reconfigurable datapath for each 32x32 tile, approximately 2240 and 19824 clock cycles are needed for Daubechies (5,3) and (9,7) filtering, respectively. Thus, three levels of 2-D DWT can be executed on a 32x32 tile using two 32-bit reconfigurable datapath at a 3.5 ns clock in about 7.84 µs and 69.38 µs for Daubechies (5,3) and (9,7) filtering, respectively.

Table 3 shows the complete characterization results for 2-D DCT and 2-D DWT implementations. Note that the Power-Delay-Product (PDP) for a discrete image transform has been estimated by the relation (1),

$$PDP\ (discrete\ image\ transform) = Pavg\ (architecture) * td \quad (1)$$

In Table 3, the proposed circuit is also compared to the Texas Instruments (TI) TMS320VC5509 DSP [19] for the 2-D DCT computation. The TI 5509 is a new generation commercial DSP targeted for high-speed, power-efficient applications. It operates at a maximum frequency of 144 MHz with a core voltage supply of 1.5V dissipating an average power of 131.4 mW [20].

In the best case, the referred DSP requires 151 cycles to complete an 8x8 2-D DCT [21]. At 144 MHz, this translates to a 1.049 µs to compute an 8x8 2-D DCT. Thus, the proposed circuit is about 2.7 times slower than the TI 5509 DSP for the 2-D DCT computation. However, due to its very low power consumption (74% reduction), it demonstrates a 28% lower PDP with respect to the TI 5509 DSP.

Table 3. Characteristics of the proposed datapath in discrete image transform and a comparison to Texas Instruments DSP.

	Delay (μs)	Average Power (mW)	PDP (mW*μs)
8X8 2-D DCT (proposed datapath)	**2.9**	**34.2**	**99.18**
8x8 2-D DCT TI TMS320VC5509	**1.049**	**131.4**	**137.8**
32X32 3-level Daubechies (5,3) (2 datapaths)	**7.84**	**68.4**	**536.26**
32X32 3-level Daubechies (9,7) (2 datapaths)	**69.38**	**68.4**	**4745.6**

6. CONCLUSION

In this paper, the design of a new cost-effective low-power PIM-based 32-bit reconfigurable datapath has been presented. The new architecture is extremely flexible. In fact, it can adapt itself at run-time to the needs of different multimedia applications.

For the proposed circuit a proper HDL code has been developed. This offers a fully synthesizable and process independent design that can be easily migrated to different standard-cells libraries and technologies.

When implemented using standard-cells in UMC 0.18 μm CMOS technology, the proposed datapath is operating at up to 285 MHz with an average energy consumption of only 0.12 mW/MHz and a silicon area occupation of just 107,323 μm^2.

Due its high-modularity, low-complexity and low-power consumption the proposed circuit is very suitable for block processing and other bandwidth intensive algorithms.

7. REFERENCES

[1] J.Suh, E.-G. Kim, S. P. Crago, L. Srinivasan, M. C. French, "A performance analysis of PIM, stream processing, and tiled processing on memory-intensive signal processing kernels", Proc. 30th Annual International Symposium on Computer Architecture, 2003, pp. 410–419, 9-11 June 2003.

[2] D.G. Elliott, M. Stumm, W.M. Snelgrove, C. Cojocaru, R. Mckenzie, "Computational RAM: Implementing Processors in Memory", IEEE Design & Test of Computers, Vol. 16, Issue 1 , pp. 32 – 41, January-March 1999.

[3] J.T. Zawodny, P.M. Kogge, "Cache-In-Memory", Innovative Architecture for Future Generation High-Performance Processors and Systems, 2001, pp. 3 – 11, 18-19 January 2001.

[4] J. Draper, J. Sondeen, S. Mediratta, I. Kim, "Implementation of a 32-bit RISC processor for the data-intensive architecture processing-in-memory chip", Proc. of the IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2002, pp.163 – 172, 17-19 July 2002.

[5] M. Margala, R. Lin, "Highly efficient digital CMOS accelerator for image and graphics processing", 15th Annual IEEE International ASIC/SOC Conference 2002, pp.127 – 132, 25-28 September 2002.

[6] J. Suh, C. Li, S.P. Crago, R. Parker, "A PIM-based multiprocessor system", Proc. 15th International Parallel and Distributed Processing Symposium, IPDPS 2001, pp. 6, 23-27 April 2001.

[7] B. S.-H. Kwan, B. F. Cockburn, D. G. Elliott, "Implementation of DSP-RAM: an architecture for parallel digital signal processing in memory", Canadian Conference on Electrical and Computer Engineering, 2001, Vol. 1, pp. 341 – 345, 13-16 May 2001.

[8] J.-S. Moon, T.-J. Kwon, J. Sondeen, J.Draper ,"An area-efficient standard-cell floating-point unit design for a processing-in-memory system", Proc. of the Conference on European Solid-State Circuits, 2003, ESSCIRC '03, pp. 57-60, 16-18 September 2003.

[9] J. Fritts, W. Wolf, B. Liu, "Understanding multimedia application characteristics for designing programmable media processors, "SPIE Photonics West, Media Processors '99, San Jose, CA, pp. 2-13, January 1999.

[10] S. Nazareth, R. Asokan. "Processor Architectures for Multimedia", academics paper, November 2001. http://www.cs.dartmouth.edu/~nazareth/academic/CS107.pdf

[11] The institute of Electrical and Electronics Engineers, Inc., "IEEE Standard for Binary Floating-Point Arithmetic", ANSI/IEEE Standard No. 754-1985, New York, Aug. 1985.

[12] J. J. F. Cavanaugh, "Digital Computer Arithmetic- Design and Implementation" McGraw-Hill Book Company, 1984.

[13] A. Farooqui, V. G. Oklobdzija, "A Programmable Data-Path for MPEG-4 and Natural Hybrid Video Coding", *34th Annual Asilomar Conference on signals, Systems and Computers*, Pacific Grove, California, October 29 - November 1, 2000.

[14] Y. Wang, J. Osternmann, Y-Q Zhang, "Video Processing And Communications", Prentice Hall, 2002.

[15] W.H. Chen, C.H. Smith, and S.C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Communications*, vol. 25, pp. 1004–1009, 1977.

[16] B. J. Jasionowski, "A Processor-in-memory for Image and Video Compression", MS Thesis, University of Rochester, Rochester, New York, 2004.

[17] I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets." *Comm. Pure Appl. Math.*,Vol 41, pp. 909-996, 1988.

[18] M. K.-F. Lay, "A Processor-in-Memory for Image Compression using the Discrete Wavelet Transform", MS Thesis, University of Rochester, Rochester, New York, 2004.

[19] TI TMS320VC5509 Fixed-Point DSP, http://focus.ti.com/docs/prod/folders/tms320vc5509.html

[20] TI TMS320VC5509 Revision D Power Consumption Measurements, January 8, 2002.

[21] TI TMS320C55x Hardware Extensions for Image/Video Applications Programmer's Reference, February 2002, http://focus.ti.com/lit/ug/spru098/spru098.pdf