# ANALYSIS OF THE MATRIX PROCESSING (MxP) ARCHITECTURE

*Lakshmi Girija*[1,4]

[1]*SiRF Technology*

*Andreas S. Spanias*[2]

[2]*SenSIP Center, Dept. Electrical Engineering, Arizona State University*

**ABSTRACT**

In this paper, we describe and evaluate a new DSP architecture called the matrix processing (MxP™). MxP exploits data parallelism using hardwired matrix operations and instructions. The MxP matrix computation capabilities are optimized for multidimensional transforms and filtering. A detailed analysis of the MxP performance, in terms of precision, execution time, and instruction memory requirements, is presented. We focus particularly on matrix and vector manipulations of the type embedded in video processing standards, e.g., filtering, DCT, and motion estimation. We present comparison tables of the performance of the MxP with other widely used DSPs such as the TI TMS320c55x™, and the TI TMS320c64x™.

Index Words: DSP architectures, video standards.

## 1. INTRODUCTION

Recent improvements in low-power and high-performance VLSI architectures along with the emergence of several compatibility standards created multiple new signal and multimedia coding applications [1-5]. Typical applications include smart camera phones, MPEG audio and video players, set top boxes, PC video streaming, digital TV, etc. Such signal processing applications are demanding in terms of computation, memory and power consumption. Examination of the ISO coding standards reveals that the DCT [6], the wavelet transform [7], and motion estimation/ compensation functions [1] are key in several video compression algorithms. Most processor architectures are sequential in nature and hence accommodate these algorithms by processing the data at high clock rates. Early attempts in the 1990s to address high computational requirements resulted in multicore designs [8], and architectures with multimedia extended instructions (e.g., the Intel MMX™). On the other hand, recently developed DSP chips [9-11] employ large instruction words (e.g., the TI TMS320c64x™) that enable execution of multiple instructions per cycle. In this paper, we describe and analyze an alternative architecture paradigm that is based on matrix-oriented computations. The matrix processing, MxP™[(3)] architecture [12,13] performs signal processing functions by exploiting matrix structures embedded in the algorithms. Programming the MxP architecture involves configuring and processing operations in vector or matrix form much like one would do with MATLAB™. The current version of the MxP can perform a 4x4 matrix operation in a single cycle. This single-cycle 4x4 capability may be exploited either to accommodate highly demanding (high-MIPS) algorithms or to execute algorithms with modest processing requirements at a lower power-aware clock frequency. Our comparative simulations of the MxP show that its matrix computation capability results in a significant reduction in the machine cycle count. In addition, the number of instructions and hence the program memory requirements are reduced relative to other DSP architectures. The paper organization is as follows. The MxP architecture is presented next. One particular realization examined is the embedded use of the MxP as a co-processor. The MxP implementation of certain DSP algorithms is discussed in Section 3. The MxP performance characteristics are presented in Section 4. Concluding remarks are given in Section 5.

## 2. THE MATRIX PROCESSING ARCHITECTURE

The current version of MxP was designed and tested as a coprocessor to the 32-bit ARM RISC core, Figure 1.

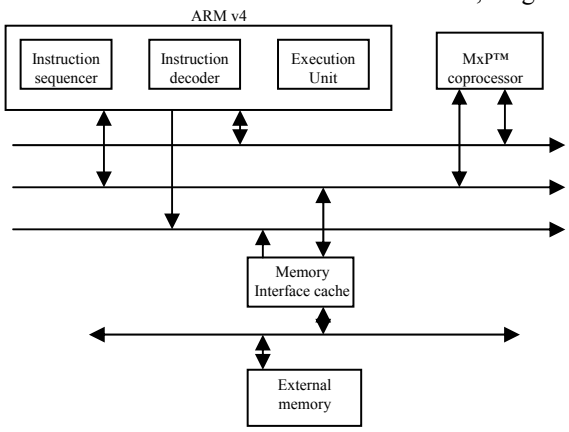


**Figure 1:** The MxP as a coprocessor to the ARM core.

The MxP is capable of complementing other CPUs, such as the ARM™, by supporting matrix instructions and matrix data types. The main CPU controls the MxP coprocessor through the co-processor bus. The instructions are visible to all processors on the coprocessor bus. In the case we examined using the ARM processor, instructions are fetched

by the ARM, and instruction allocation/execution is based on a conditional code status. The MxP also interprets instructions simultaneously with the ARM and checks if the instruction is valid. In that case, a handshake occurs and the MxP accepts and executes the instruction. The MxP has 32 registers (MR0 - MR31) that are 64 bits wide. The MxP register model is shown in Figure 2.

| Matrix Config MC | MC_1 | MC_0 |
|---|---|---|
| Matrix Status MS | MS_1 | MS_0 |
| MR29 | MR29_1 | MR29_0 |
| MR28 | MR28_1 | MR28_0 |
| MR27 | MR27_1 | MR27_0 |
| …. | …. | …. |
| …. | …. | …. |
| …. | …. | …. |
| MR2 | MR2_1 | MR2_0 |
| MR1 | MR1_1 | MR1_0 |
| MR0 | MR0_1 | MR0_0 |

**Figure 2:** The MxP register model.

The Matrix Configuration (MC) Register (MR31) is a special purpose register used to configure and control the coprocessor. The Matrix Status (MS) is unaffected by data processing operations in the current version of MxP. Both MC and MS registers can be read from and written to using a special set of transfer instructions. The MxP operates with matrices as the basic data type, with each of the matrix elements identified from the packed data types. Matrix data types can be two-dimensional matrices of arbitrary size, limited only by the span of the register file. The elements of the matrices can be signed or unsigned nibble, byte, half-word, word or double-word. The packing of the data in the registers are always aligned to a power of 2, starting from bit 0 of each of the 64 bit registers. Thus, each of the 32 registers can contain 16, 8, 4, 2, or 1 of N (Nibble), B (Byte), H (Half-Word), W (Word) or D (Double) data elements respectively. Data packing is performed in a manner that avoids having data elements that would span the register boundaries.

The MxP introduces a novel methodology for handling data, that could be configured as matrix or vector thereby providing a compact and mathematically structured format. The MxP coprocessor has a load/store architecture, in the sense that all the data processing (arithmetic and logical) operations are done on the data elements stored in the internal registers. There are separate instructions for transferring data between external memory and MxP registers, and also between ARM registers and MxP registers. Note that even for the load/store data exchange the source and target are structured as matrices, thus providing more flexibility to handle (e.g. pack/unpack) complex numbers and various other data types. The MxP has three-operands Mx, My, and Md. The source matrix My operates on the source matrix Mx and the result is stored in the destination matrix, Md. Source matrices Mx and My are formed by an ordered arrangement of the packed data residing in the MxP register set.

## 3. SIGNAL PROCESSING ON THE MxP

In order to evaluate the performance of MxP, we categorize algorithms into two types, i.e., those that can be converted into 'fast' algorithms, such as, the Fourier and cosine transforms. and those that are not easily parallelizable, such as matrix multiplications used in color transformation, filtering, decimation, interpolation etc. Operations such as the DCT are highly parallelizable due to the cyclic nature of the cosine basis functions. This degree of parallelism can be exploited by processors that support either data level parallelism or instruction level parallelism. The following section describes the performance of the MxP for typical signal processing operations that occur in standardized video processing algorithms, e.g., image filtering, DCT and a full search motion estimation using minimum of absolute differences (MAD).

### 3.1. Matrix multiplication with the MxP

Matrix multiplications take place in graphics applications, color space mapping, etc. The current version of the MxP architecture supports single-cycle 4x4 multiplication. Additional cycles are needed to prepare, load, and store the data resulting in a total of 33 cycles for half word result and 43 cycles for a full word result. An 8x8 matrix half word multiplication can be performed with eight 4x4 multiplications and four additions. The data is loaded such that maximum reuse is possible without reloading.

### 3.2. Filtering

This section gives the MxP cycle count estimate for a 16-tap FIR filter with 40 output samples. The input data is assumed to be real valued. An FIR filter output samples *y(n)* can be expressed as

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

where *x(n*) are the input samples, *N* is the number of coefficients and *h(n)* are the filter parameters. For a 16 tap filter with 40 output samples, the above equation can be expressed in matrix form as:

$$\begin{bmatrix} y(0) \\ y(1) \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ y(39) \end{bmatrix} = \begin{bmatrix} x(0) & 0 & \dots & \dots & \dots & 0 \\ x(1) & x(0) & 0 & \dots & \dots & .. \\ x(2) & x(1) & x(0) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x(15) & x(14) & \dots & \dots & \dots & x(0) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x(39) & x(38) & x(37) & \dots & \dots & x(24) \end{bmatrix} \times \begin{bmatrix} h(0) \\ h(1) \\ \dots \\ \dots \\ \dots \\ h(15) \end{bmatrix}$$

For halfword data size, this filtering operation involves 40 cycles for data load, ten cycles for matrix configuration, 50 multiplications, and 23 cycles for storing the result back to memory. The total number of cycles for this operation is 127 cycles. Note that sequential realization even on a chip with a single-cycle MAC instruction requires 600 cycles for this example excluding the loading, etc.

### 3.3. An 8x8 DCT using the MxP

The DCT is widely used in many image and video compression algorithms due to its high energy compaction capability. For example in JPEG and MPEG it can be shown that most of the pixel energy is concentrated on the lower spatial frequency components. DCT coefficients are also highly de-correlated making it possible to encode and reconstruct the original signal efficiently. Several fast algorithms and architectures were developed for DCT implementation. We investigated some of the fast DCT algorithms for efficient implementation on the MxP. Due to the 4x4 matrix multiplication capability of MxP, we chose the decimation-in-frequency (DIF) DCT algorithm. Using the DIF-DCT algorithm, an *N* point 1-D DCT can be calculated using two half-size (*N/2-point)* DCTs. Similarly, for the 2D case, one *NxN* point DCT can be performed using four (*N/2 x N/2*) point DCTs. Figure 3 shows the flow graph of a 1-D 8 point DCT. The 8x8 DCT used in the MPEG standard can be realized efficiently using the DIF algorithm with four 4x4 DCTs. The DCT implementation on MxP involves decimation up to the first stage and then transformation using the single-cycle 4x4 matrix multiplication. The implementation of this algorithm on the MxP involves twelve 4x4 matrix multiplies, five 4x4 matrix adds, and the rest of the cycles are for data load/ store.

### 3.4. Motion estimation on the MxP

Correlation among adjacent frames in a video sequence is exploited using motion estimation which estimates the movement of the current block relative to a reference block.

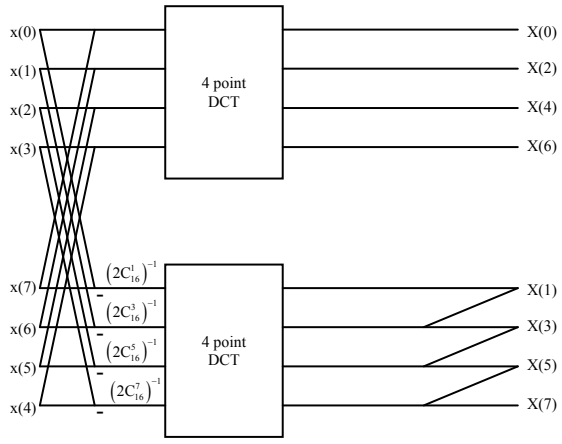


**Figure 3:** Flow graph for 8-point DCT.

The current block is compared with the reference blocks within a search window and the best match is chosen as the prediction based on certain matching criterion such as the sum of absolute differences (SAD). The motion estimation process is shown in Figure 4.

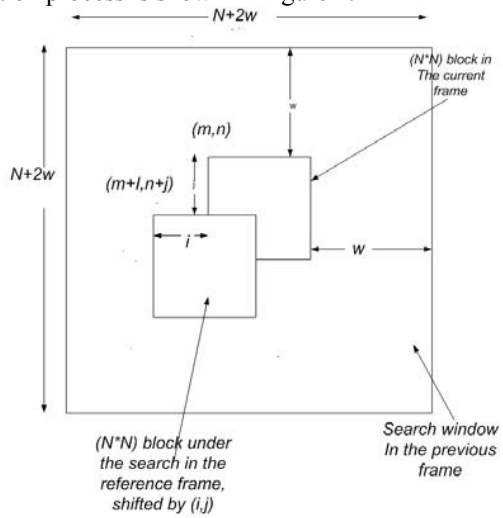


**Figure 4:** Motion Estimation using full search (after [1]).

The prediction block selected is the one that has the minimum SAD value (minimum absolute difference). The MxP instruction set has a hardwired SAD instruction that calculates the SAD values between two 4x4 blocks of data. This process was implemented on the MxP. The size of the current block was chosen to be 8x8 and the search window size was chosen to be 12x12. This choice was based on an efficient implementation using all the registers of MxP with no intermediate data load/ store. A current block of size 8x8 and the search window of size 12x12 would require 5 SAD operations in the horizontal direction and 5 SAD operations in the vertical direction, a total of 25 SAD operations. Since the maximum size of matrix operation performed by MxP is 4x4, the SAD of an 8x8 block is calculated as four partial SADs. Thus the operation is repeated for both horizontal and vertical directions and the best match is selected as the motion vector for the current frame. The implementation of this algorithm on the MxP involves 100 SAD operations, 18 sorting operations, and the remaining cycles are for data load/ store and matrix configuration.

## 4. PERFORMANCE EVALUATION OF MxP

In this section, we provide the results of performance analysis of MxP for filtering operation, DCT and maximum amplitude difference (MAD) computation in terms of machine cycles. We also provide of MxP with other widely used DSPs from Texas Instruments comparisons in Figs. 5 and 6. Table 1 gives the MxP cycle count of matrix multiplication, 16-tap FIR filter, 8x8 DCT and MAD computation for an 8x8 block over a 12x12 search window. The table also provides a comparison of MxP cycle count

with other widely used DSPs such as the Texas Instruments (TI) TMS320c55x and TMS320c64x.

**Table 1:** Machine cycles

| | MxP | TMS320c64 | TMS320c55 |
|---|---|---|---|
| 8x8 Multiply | 146 | 283 | 464 |
| 16-tap FIR | 127 | 285 | 400 |
| 8x8 DCT | 125 | 126 | 238 |
| MAD | 203 | 194 | 711 |

From these examples, it is evident that in terms of machine cycles the MxP performs quite well relative to the TMS320c55x. For matrix multiplication and filtering, the MxP cycle count is lower than that of TMS320c64x. On the other hand for the DIF DCT, both the MxP and the TMS320c64x have comparable performance. Table 2 and Figure 6 shows code size comparisons for the DCT and the MAD computations. It can be seen that the MxP performs these tasks with reduced code size. Reduced code size results in significant reduction of onchip program memory.

## 5. CONCLUSIONS

In this paper, we presented and evaluated the MxP architecture. Comparative results were given for select signal and video processing applications. In particular, the MxP performance was compared against the widely used TI DSPs, namely the TMS320c64x and the TMS320c55x. Comparison results in terms of machine cycles and code size favored the MxP™. The code size for the MxP proved to be more compact than the other DSPs resulting in reduced instruction fetches. The reduction in instruction fetches and the lower overall clock can be exploited for low power realizations on the MxP.

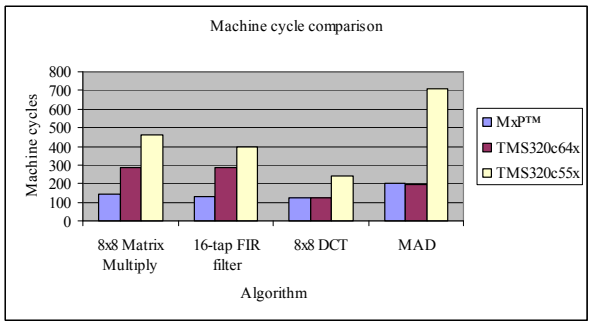


**Figure 5. Machine cycle comparison.**

**Table 2:** Code size in bytes

| | MxP | TMS320c64 | TMS320c55 |
|---|---|---|---|
| Matrix Multiply | 84 | 416 | 215 |
| 16 tap FIR | 140 | 544 | 107 |
| 8x8 DCT | 88 | 976 | 480 |
| MAD | 432 | 788 | 1080 |

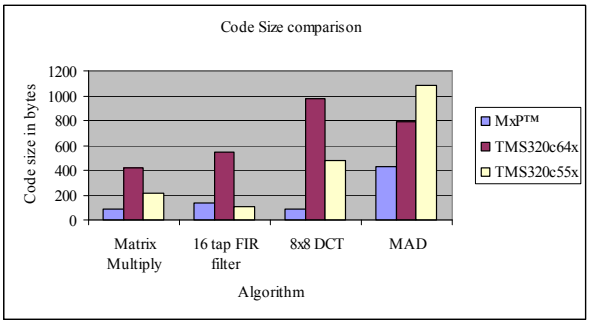


**Figure 6. Code size comparison.**

## 6. REFERENCES


[1] M. Ghanbari, *Standard codecs: Image compression to advanced video coding*, The IEE Press, London, 2003.

[2] M. Vrhel, E. Saber, H. Trussell, "Color Image Generation & Display Technologies", IEEE SP Mag., Jan 2005.

[3] A.S. Spanias, "Speech Coding: A Tutorial Review," *Proc. IEEE*, Vol. 82, No. 10, pp. 1441-1582, Oct. 1994.

[4] T. Painter and A. Spanias, "Perceptual Coding of Digital Audio," *Proc. IEEE*, pp.451-513, Vol. 88, No.4, Apr 2000.

[5] A. Spanias, T. Painter, V. Atti, Audio S*ignal Processing and Coding*, Wiley, ISBN:0-471-79147-4, March 2007.

[6] P. Yip, and K. R. Rao, "The decimation-in-frequency algorithms for computing the discrete cosine transform," *Circuits, Systems and Signal processing*, pp. 4-19, 1988.

[7] R.M. Rao and A.S. Bopardikar, Wavelet Transforms: Introduction to Theory and Applications, Addison-Wesley Longman, Reading, MA, 1998.

[8] A. Spanias, M. Deisher, P. Loizou[+], G. Lim[+], B. Mears, "A New Highly Integrated Architecture for Speech Processing and Communication Applications," *ITJ*, pp. 41-56, Spring 1994.

[9] SPRU422H, TMS320c55x DSP Programmer's Reference, October 2004.

[10] SPRU565H, TMS320c64x DSP Programmer's Reference, October 2003.

[11] SPRU037C, TMS320c55x Image, Video processing Programmer's Reference, January 2004.

[12] MxP Architecture, Rev 0.1.0, Doc. No. GS\ EN\PR\ DE\2004\020, Aug. 2004, G4 Matrix Tech. Inc. (formerly GemTech Solutions (P)), Thiruvananthapuram, India.

[13] Apparatus and method for Matrix Data Processing, G. Nair, US Patent No. 6,944,747, Sept. 13, 2005.


---

3 - TM (trade mark) of MxP belongs to G4 Matrix Tech. Inc. TM of TI TMSxxxx.. belongs to Texan Instruments Inc., TM on MMX belongs to Intel Corp., and TM on MATLAB belongs to The Mathworks.

4. L. Girija performed this work as a graduate research assistant, as part of her Masters degree at ASU. She is currently with SiRF Technology.