Volume-4,Issue-2, Aug - 2015
ISSN (O) :- 2349-3585

HARVESTING 2D PROCESSING IN SIGNAL PROCESSING APPLICATIONS

A Chempak Kumar[1],V R Sudheer[2]
[1]Department of electronics and communication,College of engineering Thalassery
[2]Department of electronics and communication,College of engineering Perumon

Abstract— Many advanced features and methods have been proposed and implemented as a result of the immense research activities going on to quench the thirst for the ever increasing demand for signal processing capabilities. But all the algorithms and ideas, how much intelligent and unique they are, do face the severe limitation posed by the available processors. Further in the industrial and commercial arena cost, power consumption, processing time, performance competitiveness, reliability, feasibility, adaptability, compactness etc pose the major challenges. The concept of two dimensional processing in matrix form, offering every option and flexibility to utilize the matrix properties, do promise immense options not only in signal processing but also in other related fields. The Matrix Processor, (MxP™) developed at G4Matrix Tech. Inc. Corp. based on the above concept stands unique to revolutionize the signal processing arena. The processor, developed as a coprocessor to work with almost all the existing processors, handle data by treating it as a data set that could be configured as matrices and vectors of very flexible but mathematically structured format to operate on matrices as operands. This provides efficient signal processing algorithms at much reduced machine cycles, power and cost. Here we took an auto-correlation block of G.729 speech standard as a benchmark for MxP™, as a coprocessor for PowerPC440.

Introduction

Recent developments 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]. Such signal processing applications are demanding in terms of computation, memory and power consumption. Examination of the ISO coding standards reveals that the auto-correlation [6], the wavelet transforms [7] and motion estimation/compensation functions [1] are key in several video and audio compression algorithms. Most processor architectures are sequential in nature and hence accommodate these algorithms by processing the data at high clock rates. Attempts to address high computational requirements resulted in multicore designs [8], and architectures with multimedia extended instructions (e.g., the Intel MMXTM). On the other hand, recently developed DSP chips [9-11] employ large instruction words (e.g., the TI TMS320c64xTM) 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 processor, MxP™ [12-13], a coprocessor for most conventional RISC processors developed by G4Matrix Tech. Inc. Corp., 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 16 element (4X4, 1X16,..) matrix operation in a single cycle. This single-cycle 16 element 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, the number of instructions and hence the program memory requirements relative to other DSP architectures. An auto-correlation block of G.729 speech standard has been taken as a benchmark for evaluating the performance of MxP™.

2.THE MATRIX PROCESSING ARCHITECTURE

The current version of MxP was designed and tested as a coprocessor to the 32-bit PowerPC440 core, Figure 1

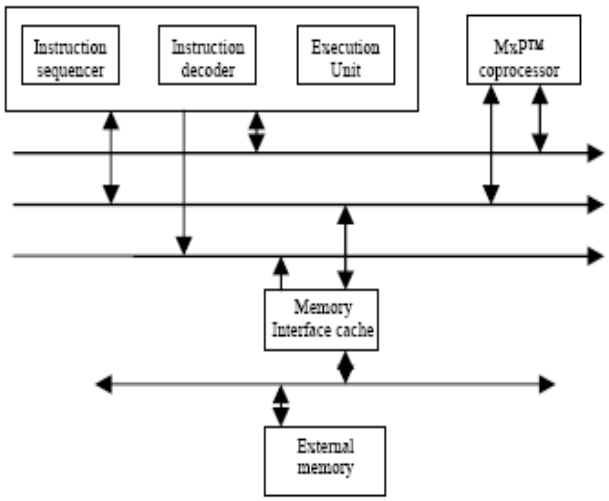

Figure 1: The MxP as a coprocessor to the PowerPC core.

The MxP is capable of complementing other CPUs, such as the PowerPC™ , 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 PowerPC processor, instructions are fetched by the PowerPC, and instruction allocation/execution is based on a conditional code status. The MxP also interprets instructions simultaneously with the PowerPC and checks if the instruction is valid. In that case, a handshake occurs and the MxP accepts and executes the instruction. The MxP has 32registers (MR0 - MR31) that are 64 bit wide. The MxP register model is shown in Figure 2

Matrix Config	MC	
MR30	MR30_U	MR30_L
MR29	MR29_U	MR29_L
MR28	MR28_U	MR28_L
	……..	……..
	……..	……..
	……..	……..
MR2	MR2_U	MR2_L
MR1	MR1_U	MR1_L
MR0	MR0_U	MR0_L

Figure 2: The MxP register model.

The Matrix Configuration (MC) Register (MR31) is a special purpose register used to configure and control the coprocessor. MC register 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 byte, half-word, 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 8, 4, or 2 of B (Byte), H (Half-Word), or W (Word) 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.

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 multiplications. 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 Autocorrelation

A signal correlated with itself. It is useful because the Fourier transform of the autocorrelation is the power spectrum of the original signal. This is an optimal way to detect a known waveform in a signal.

Autocorrelation requires that several initial assumptions be made about the set or sequence of speech samples, {yn }, in the current segment. First, it requires that {yn } be stationary and second, it requires that the {yn } sequence is zero outside of the current segment. In autocorrelation, each E[yn-i yn-j] is converted into an autocorrelation function of the form Ryy (| i-j |). The estimation of an autocorrelation function Ryy (k) can be expressed as:

$$R_{yyy}(k) = \sum_{n=n_0+1+k}^{n_0+N} y_n \, y_{n-k}$$

Using Ryy (k), the M equations that were acquired from taking the derivative of the mean Squared error can be written in matrix form RA = P where A contains the filter coefficients.

$$R = \begin{bmatrix} R_{yy}(0) & R_{yy}(1) & R_{yy}(2) & \cdots & R_{yy}(M-2) & R_{yy}(M-1) \\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & \cdots & R_{yy}(M-3) & R_{yy}(M-2) \\ R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(M-4) & R_{yy}(M-3) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ R_{yy}(M-1) & R_{yy}(M-2) & R_{yy}(M-3) & \cdots & R_{yy}(1) & R_{yy}(0) \end{bmatrix}$$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_M \end{bmatrix} \qquad P = \begin{bmatrix} R_{yy}(1) \\ R_{yy}(2) \\ R_{yy}(3) \\ \vdots \\ R_{yy}(M) \end{bmatrix}$$

3.2.1 Autocorrelation algorithm in MxP

The autocorrelation implementation on MxP involves only 4x4 matrix multiplication and element matrix multiplications. The rest of the cycles are for data load/ store.

- Initialization of base register.
- Configure MC for matrix loading.
- Load 16x1 input matrix from memory.
- Matrix element multiplication of the above matrix with
- the same is done.
- Configure MC for shifter matrix.
- Load 16x1 shifter matrix memory.
- Matrix element multiplication of the resultant of step4 and the above matrix is done.

- Configure MC for matrix loading.
- Load 1x16 unity matrix from memory.
- Matrix multiplication of resultant matrix of step 7 and the above matrix is done.
- Store the output matrix in a 1x1 matrix.
-

3.3 Radix-4 Fast Fourier Transform (FFT)

When the number of data points N in the DFT is a power of 4 (i.e., N = 4v), we can, of course, always use a radix-2 algorithm for the computation. However, for this case, it is more efficient computationally to employ a radix-r FFT algorithm. Let us begin by describing a radix-4 decimation-in-time FFT algorithm briefly. We split or decimate the N-point input sequence into four subsequences, x(4n), x(4n+1), x(4n+2), x(4n+3), n = 0, 1, ... , N/4-1.

Thus the four N/4-point DFTs F(l, q)obtained from the above equation are combined to yield the N-point DFT.

$$X(p,q) = \sum_{l=0}^{3} [W_N^{lq} F(l,q)] W_4^{lp}$$

$$F(l,q) = \sum_{m=0}^{(N/4)-1} x(l,m) W_{N/4}^{mq}$$

$$p = 0,1,2,3; \quad l = 0,1,2,3; \quad q = 0,1,2,\ldots,\frac{N}{4}-1$$

and

$$x(l,m) = x(4m+1)$$

$$X(p,q) = X(\frac{N}{4}p + q)$$

The expression for combining the N/4-point DFTs defines a radix-4 decimation-in-time butterfly, which can be expressed in matrix form as

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F(0,q) \\ W_N^q F(1,q) \\ W_N^{2q} F(2,q) \\ W_N^{3q} F(3,q) \end{bmatrix}$$

The radix-4 butterfly is depicted in Figure 3 and in a more compact form in Figure (b). Note that each butterfly involves three complex multiplications, since WN0 = 1, and 12 complex additions.

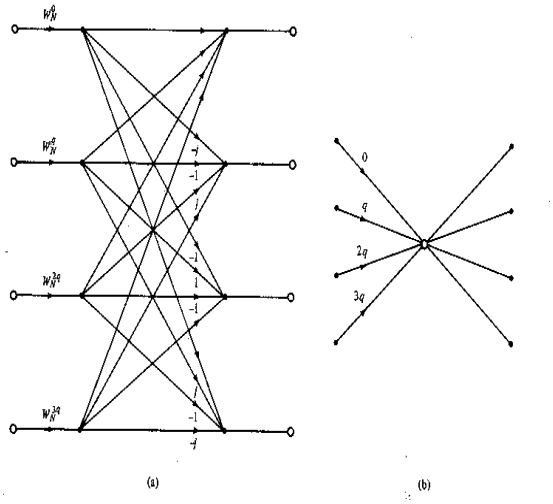

Figure 3: Radix-4 butterfly

By performing the additions in two steps, it is possible to reduce the number of additions per butterfly from 12 to 8. This can be accomplished by expressing the matrix of the linear transformation mentioned previously as a product of two matrices as follows

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} W_N^0 F(0,q) \\ W_N^1 F(1,q) \\ W_N^2 F(2,q) \\ W_N^3 F(3,q) \end{bmatrix}$$

3.3.1 256-point Radix-4 (FFT) algorithm in MxP

Due to the 4x4 matrix multiplication capability of MxP, we chose the decimation-in-frequency (DIF) FFT algorithm. The FFT implementation on MxP involves one 4x4 matrix multiplication and one complex multiplication in first three stages and a single 4x4 matrix multiplication in the fourth stage butterfly. The rest of the cycles are for data load/ store.

Once per FFT:

Load 4x4 transformatiom matrix coefficients (16 hwords)
Load four matrix configuration values (4 dwords)

Same for first three stages: once per Butterfly:

Load four complex inputs - real and imaginary part (8 hwords)

Perform complex transformation (4x4 * 4x2)

Load 1x4 complex twiddle factors (8 hwords) and 1x4 negative

of the imaginary part (4 hwords)

Perform multiplication with twiddle factors

Store four complex outputs - real and imaginary (8 hwords)

Once per Butterfly:

Load four complex inputs - real and imaginary part (8 hwords)

Perform complex transformation (4x4 * 4x2)

Store four complex outputs - real and imaginary (8 hwords)

4. PERFORMANCE EVALUATION OF MxP

In this section, we provide the results of performance analysis of MxP for 8x8 DCT matrix multiplications in terms of machine cycles. We also provide performance comparison results of MxP with other widely used DSPs from Texas Instruments comparisons in Figures. 4 and 5. Table 1 gives the MxP cycle count with other widely used DSPs such as the Texas Instruments (TI) TMS320c55x and TMS320c64x

Process	MxP	TMS320c 64	TMS320c 55
8x8 Multiply	146	283	464
8x8 DCT	125	126	238

Table 1: Machine cycles

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 5 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 on chip program memory

Process	MxP	TMS320c 64	TMS320c 55
8x8 Multiply	84	416	215
8x8 DCT	88	976	480

Table 2: Code size in bytes

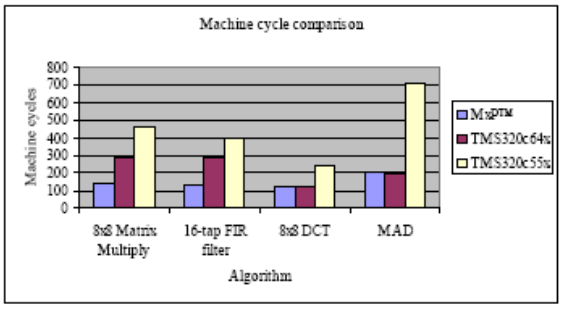

Figure 4 Machine cycle comparison

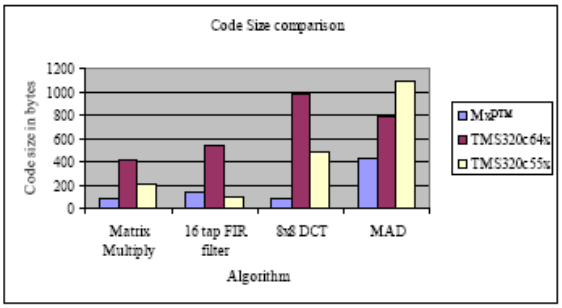

Figure 5: Code size comparison

CONCLUSION

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.

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 Signal Processing and Coding, Wiley, ISBN:0-471-79147-4, March 2007.

[6] 2.ITU-T Draft Recommendation G.729, "Coding of speech at 8Kbps using the Conjugate Structure Algebraic Code Excited Linear – Prediction (CS-ACELP)".

[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

[12] MxP Architecture, Rev 0.1.0, Doc. No. GS\ Programmer's Reference, January 2004. 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