A High-Performance Area-Aware DSP Processor Architecture for Video Codecs

Lan-Da Van, Hsin-Fu Luo, Chien-Ming Wu, Wen-Hsiang Hu, Chun-Ming Huang, and Wei-Chang Tsai
Chip Implementation Center (CIC), National Applied Research Laboratories,
No. 1, Prosperity Rd. 1, Science-Based Industrial Park, Hsinchu, Taiwan, R.O.C.
E-mail: {ldvan, brady, wucm, wshu, wctsai, cmhuang}@cic.org.tw

ABSTRACT

In this paper, we propose a high-performance and area-aware very long instruction word (VLIW) DSP architecture using a flexible single instruction multiple data (SIMD) approach and a grouped permutation (GP) structure register file, respectively. Via the proposed data path architecture, the reduction of the execution cycles for digital filter and RGB2YUV benchmarks can be improved up to 50% compared with that of [8, 11]. For motion estimation, the number of pixels per cycle applying the proposed architecture can be four times than that of [8, 11]. For the register file, using proposed GP structure, the saving of switching network overhead could be anticipated compared with the work in [11].

1. Introduction

With continuing advances of video algorithm development in standards [1-3] as well as in very-large-scale-integrated (VLSI) technology, an increasing demand of variety of services and applications has evolved in the field of multimedia. In standard applications, DSP processor designs [4-5] are very attractive to realizing a broad range of video processing algorithms. Due to the increasing necessity for digital image transmission and storage, many image and video coding algorithms have been promoted to MPEG (Moving Picture Experts Group) [1-2], H.26x [1-3], and JPEG (Joint Photographic Experts Group) standards. Because video coding operations such as motion estimation, RGB to YUV (RGB2YUV) conversion, discrete cosine transform (DCT), digital filter, and variable length coding/decoding (VLC/VLD) are of very intensive computing loads, we are encouraged to improve the performance of DSP processors in no time. In other words, due to the complexity of these operations and the large amount of data operations in Table 1 involved with video coding systems, DSP processors will require very high processing rates for real time quality. Thus, many significant research efforts are recently to develop DSP processors for video coding [6-11]. In [7], vector DSP (VDSP) processor has been proposed to meet high processing capability; however, two large dedicated cores consisting of DCT and loop filter are needed. Thus, larger area cost is resulted. On the other hand, the performance can take advantages of the dedicated core, but the other frequent operations like RGB2YUV and motion estimation in video codecs do not work well. Pirsch's research group proposed 3-way 4 completed data-paths DSP processor in [8]. Nevertheless, since no sub-word parallelism is applied and one of functional elements including ALU, S&R, and MAC (multiply-accumulate operation) is multiplexed in a completed data path, the peak performance is limited. Recently, Pirsch's group provides a second version of programming processors with 16 data paths [9] and it is observed that area cost is a considerable issue. It is well known that TI C'64x [10] is a general-purpose DSP processor, and it is no doubt that the processor is able to achieve very high performance; however, the considerable area issue will affect costs. In [11], Lin *et al.* proposed 4-dedicated data paths (two for L/S and other two for ALU/MAC) to overcome the mentioned shortcomings. However, since just one type SIMD instruction can be provided, the performance must be limited. Thus, we are motivated to design another high-computing area-aware DSP processor for video codecs. Our approach for improving performance is to provide a flexible SIMD instruction set architecture (ISA) consisting of four 8-bit SIMD and two 16-bit SIMD capabilities to make the hardware resource utilization exhaustive. Hence, the proposed high-computing DSP processor is able to meet the high performance demands of various video coding algorithms.

The paper is organized as follows. A design of a 3.6 GOPS (giga operations per second) DSP processor is exposed in Section 2. In Section 3, benchmark studies using the proposed data path are discussed via digital filter and RGB2YUV. In section 4, comparison results are tabulated in terms of the execution cycles of digital filters, RGB2YUV, and 2-D DCT/IDCT, as well as the number pixels per cycle for motion estimation. The peak performance and register file structure type are debated. We give a brief conclusion in the last section.

Table 1: MOPS operations needed in MPEG-2 Encoder/ Decoder [1]

MPEG CODECS [MOPS]	Encoder Complexity		Decoder Complexity	
	SIF 352x240	CCIR 601 720x486	SIF 352x240	CCIR 601 720x486
NO B-frames	738	3,020	96	395
20% B-frames	847	3,467	101	415
50% B-frames	1,011	4,138	108	446
70% B-frames	1,120	4,585	113	466

2. Design of a 3.6 GOPS VLIW Data Path

2.1. Grouped Permutation Structure Register File

The main concept we proposed is to reorder data paths as shown in Fig. 1. In this way, the more switching

network area for the register file can be saved. That is referred to as grouped permutation (GP) structure register file. The shaded region in Fig. 1 shows the GP-structure register file in our proposed 4-way VLIW DSP processor. Each sub-block has four access ports (2R/2W). Each register file composed of 16 elements (r0~r15) is partitioned into a private (r0-r7) and a shared sub-block (r8~r15). The shared sub-blocks (i.e., GP registers) are used for data exchanges among a couple of joined function units (FU's) with a 1-bit controller. Thus, the complexity of port mapping can be reduced to $O(F)$, where F denotes the number of FU's. The shared sub-blocks are all identical and each has eight 32-bit elements. The private sub-blocks (i.e., local registers) of the D_L/D_S FU's have eight 32-bit elements for general-purposed utilizations and memory addresses, while those of ALU/MAC has eight 40-bit accumulators.

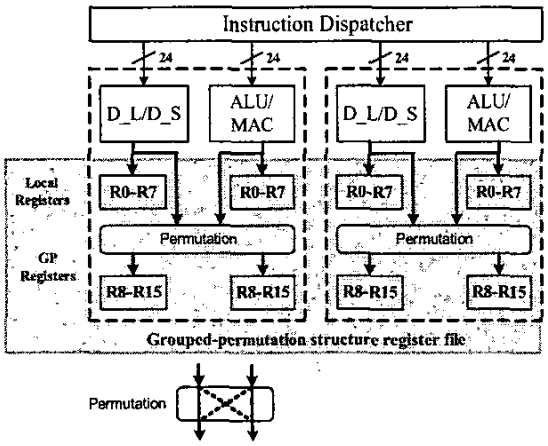

Fig. 1. Proposed 4-way VLIW data path architecture with grouped permutation register file.

2.2. SIMD Functional Unit

In video codecs, most algorithms such as digital filters, RGB2YUV, and motion estimation perform on 8-bit manipulation and each algorithm can be accomplished in one completed data-path or two dedicated data-paths. So as to fully exploit the data parallelism, we provide two types of SIMD's for the ISA to enhance the computing capability. That is, the AUL/MAC functional unit can concurrently perform two 16-bit or four 8-bit operations simultaneously. Moreover, the DSP processor supports two concurrent 16-bit and four 8-bit MAC operations with 40-bit accumulator. For conciseness, the most two frequently used instructions are illustrated as

```
MAC4 ri, rm, rn;
```

which executes $r_i.[39:20] = r_i.[39:20] + r_m.[31:24] \times r_n.[31.24]$, $r_i.[19:00] = r_i.[19:00] + r_m.[23:16] \times r_n.[23:16]$, $r_{i+1}.[39:20] = r_{i+1}.[39:20] + r_m.[15:08] \times r_n.[15:08]$, and $r_{i+1}.[19:00] = r_{i+1}.[19:00] + r_m.[07:00] \times r_n.[07:00]$ in parallel. The notation "4" deontes that this instruction has "four" 8-bit SIMD capability. Besides, the DSP also supports powerful double load and double store instructions and the former insturction is as

```
DLD rm, rn, (ri)+j;
```

where r_m and r_n are GP registers that deliver data to ALU/MAC functional units and the r_i and r_{i+1} are local address registers. Based on this mechanism, the accesses do not conflict. DLD performs two parallel memory accesses ($r_m \leftarrow Mem[r_i]$, $r_n \leftarrow Mem[r_{i+1}]$) with concurrent address updates ($r_i \leftarrow r_i+j$ and $r_{i+1} \leftarrow r_{i+1}+j$). These instructions require six concurrent register file accesses (including two reads and four writes for loads, or four reads and two writes for stores). Since these SIMD instructions exhaust all multiplications resources (i.e., our DSP totally has eight 8-bit multipliers per data path), the peak performance of 3.6 GOPS (as explained in next subsection) can be achieved.

2.3. Unified Strategy for Peak Performance Evaluation

Many research works have claimed that high peak performance (PP) values can be obtained; however, the method for calculating GOPS is not unified. In case we directly extract the values of GOPS from these articles for comparison, it is not fair to judge which one possesses higher computing power. So, we are motivated to provide a unified strategy to evaluate the PP values for different DSP processors. In view of our architecture, the MAC4 and DLD instructions consume most processing resources in D_L/D_S and ALU/MAC data paths, respectively. For MAC4 instruction, the number of simultaneous operations as depicted in Fig. 2(a) can be distributed as: 2 operations for read, 4 operations for multiplications (MUL), 4 operations for accumulation (ACC), finally, 1 operation for write back. Thus, total number of operations will be 11. In similar behavior, DLD instruction consumes 7 operations. Assume the proposed 4-way DSP processor operates at 100 MHz, the PP can be calculated as

$$PP = (7 \times 2 + 11 \times 2) \times 100 = 3.6\,\text{GOPS}. \quad (1)$$

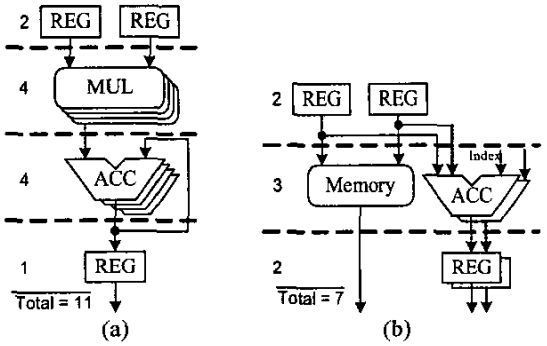

Fig. 2. Unified strategy for peak performance evaluation (a) MAC instruction, and (b) double load instruction.

3. Benchmark Studies

In this section, we further explain the programming models of our proposed data path through two benchmarks: digital filter and RGB2YUV.

Benchmark 1: 1024-Sample 64-Tap FIR Filter for 8-Bit Data Manipulation

```
0;   MOV r0,COEF;          MOV r0,0;          0;   MOV r0,COEF;          MOV r0,0;
0;   MOV r1, X;            NOP;               0;   ADD r1,X,1;          NOP;
0;   MOV r2, Y;            NOP;               0;   ADD r2,Y,2;          NOP;
     RPT 512,9;
0;   DLD r8,r9,[r0]+4;     MOV r1,0;          0;   DLD r8,r9,[r0]+4;    MOV r1,0;
     RPT 7,2
1;   DLD r8,r9,[r0]+4;     MAC4 r0,r8,r9;     1;   DLD r8,r9,[r0]+4;    MAC4 r0,r8,r9;
0;   DLD r8,r9,[r0]+4;     MAC4 r0,r8,r9;     0;   DLD r8,r9,[r0]+4;    MAC4 r0,r8,r9;
1;   DLD r8,r9,[r0]+4;     MAC4 r0,r8,r9;     1;   DLD r8,r9,[r0]+4;    MAC4 r0,r8,r9;
0;   MOV r0,COEF;          MAC4 r0,r8,r9      0;   MOV r0,COEF;         MAC4 r0,r8,r9;
0;   SUB r1,r1,62;         ADD2 r8,r0,r1;     0;   SUB r1,r1,62;        ADD2 r8,r0,r1;
0;   NOP;                  ADDs2 r8,r8;       0;   NOP;                 ADDs2 r8,r8;
1;   STh [r2]+4,r8;        MOV r0,0;          1;   STh [r2]+4,r8;       MOV r0,r0;
//=============================================================================================
//   ADD2 r8,r0,r1;        r8.Hi = r0.Hi + r1.Hi
//                         r8.Lo = r0.Lo + r1.Lo
//   ADDs2 r8,r9;          r8.Hi = 0
//                         r8.Lo = r9.Hi + r9.Lo
//   STh  [r2]+4,r8;       [r2]=r8.[15:0], then r2+4
//=============================================================================================
```

Fig. 3. Benchmark of 1024-sample 64-tap FIR for 8-bit data manipulation.

```
0;   MOV r0,COEF;          NOP;               0;   MOV r0,COEF;         NOP;
0;   MOV r2,BLK;           NOP;               0;   ADD r2,BLK,3;        NOP;
0;   MOV r3,NR;            NOP;               0;   MOV r3,NR;           NOP;
0;   MOV r4,OUT;           NOP;               0;   ADD r4,OUT,3;        NOP;
0;   LD r8,[r0]+4;         NOP;               0;   LD r8,[r0]+4;        NOP;
1;   LD r8,[r0]+4;         MOV4 r0,r8;        1;   LD r8,[r0]+4;        MOV4 r0,r8;
0;   LD r8,[r0];           MOV4 r1,r8;        0;   LD r8,[r0];          MOV4 r1,r8;
1;   NOP;                  MOV4 r2,r8;        1;   NOP;                 MOV4 r2,r8;
     RPT 8,8;
     RPT 4,6;
0;   LD  r9,[r2]+6;        NOP;               0;   LD  r9,[r2]+6;       NOP;
1;   NOP;                  SRL r3,r9,8;       1;   NOP;                 SRL r3,r9,8;
0;   NOP;                  MADDs4 r10,r3,r0;  1;   NOP;                 MADDs4 r10,r3,r0;
1;   STb [r4]+1,r10;       MADDs4 r10,r3,r1;  0;   STb [r4]+1,r10;      MADDs4 r10,r3,r1;
0;   STb [r4]+1,r10;       MADDs4 r10,r3,r2;  0;   STb [r4]+1,r10;      MADDs4 r10,r3,r2;
1;   STb [r4]+4,r10;       NOP;               1;   STb [r4]+4,r10;      NOP;
0;   ADD r2,r2,r3;         NOP;               0;   ADD r2,r2,r3;        NOP;
//=============================================================================================
//   BLK : Address of 8x8 Block
//   NR  : Next row of Block
//   COEF = {{0,0.299,0.587,0.114},{0,-0.1687,-0.3313,0.5},{0,0.5,-0.4187,-0.0813}}
//   MADD4s r10,r7,r9;     r10[31:16] = 16'b0;
//                         r10[15:00] = r7[31:24] x r9[31:24] + r7[23:16] x r9[23:16] +
//                                      r7[15:08] x r9[15:08] + r7[07:00] x r9[07:00];
//   SRL  r3,r9,8;         r3 = r9 >> 8
//   STb  [r3]+1,r10;      [r3]=r10.[07:00], then r3+1
//=============================================================================================
```

Fig. 4. Benchmark of RGB2YUV with N pixels.

For video computation [3], many huge filter-like operations are used to increase the image quality; thus, an efficient treatment is required. Based on the proposed data path architecture, the execution cycles of filter for 8-bit data manipulation compared to that of [9, 11] can be diminished. Now, consider the input sequence $x(n)$, the filter output $y(n)$ can be represented as

$$y(n) = \sum_{i=0}^{L-1} h(i)x(n-i), \tag{2}$$

where n and L denote time index and the number of taps of the filter, respectively. Using our developed ISA, Eq. (2) can be implemented in Fig. 3. Each output sample $y(n)$ can be accomplished in one grouped data path (i.e., one is D_L/D_S and the other is ALU/MAC) depicted in dash-line box of Fig. 1. Since the architecture features four 8-bit SIMD and two grouped data paths, the execution cycles of N samples can be reduced to $NL/8$.

Benchmark 2: RGB2YUV with N Pixels

The human eye is more sensitive to luminance than to chrominance. That is why we convert RGB color space to YUV space. Moreover, in most video codecs, the RGB2YUV conversion is one of huge consuming operations. The conversion equation can be written as

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{3}$$

In similar behavior, the programming model can be implemented in Fig. 4. Since the syntax notations are appended to the bottom row, herein, we bypass the descriptions.

4. Comparison Results for DSP Architectures

In this section, we give concise comparison results as listed in Table 2 in terms of the execution cycles of digital filters, RGB2YUV, and 2-D DCT/IDCT, as well as the number pixels per cycle for motion estimation. The PP and register file structure type are also addressed in Table 2. For more clarity, we normalized the performance as shown in Fig. 5 to reveal that 50% refinement can be achieved via the proposed data path architecture for 8-bit FIR and RGB2YUV applications. Four times performance

improvement can be obtained for motion estimation benchmarks. Hence, the real time video processing can be approached. For the register file, the ring structure of [11] consists of F permutation units, each permutation unit including F inputs for distributing data into register. The proposed GP structure is made up of $F/2$ 2-input permutation units. In summary, the design in [11] has the interconnection complexity $O(F^2)$, whereas our design yields a much lower interconnection complexity of $O(F)$. This can further reduce the interconnection overhead for hardware requirements between FU and register file, especially for the large value of F and data width.

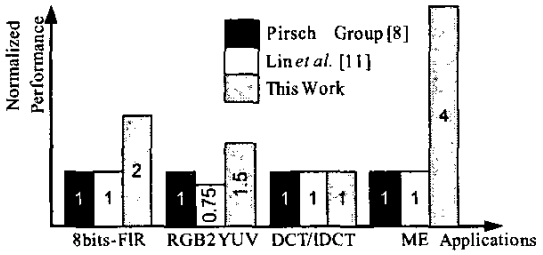

Fig. 5. Normalized performance for various applications.

5. Conclusion

We have proposed a high-performance area-aware DSP architecture based on the flexible SIMD's and GP register file. Via the proposed data path architecture, the reduction of the execution cycles for digital filter and RGB2YUV benchmarks can be improved up to 50% compared with that of [8, 11]. For motion estimation, the number of pixels per cycle applying the proposed architecture can be four times than that of [8, 11]. As discussed above, the interconnection complexity is significantly reduced based on our proposed GP register file structure. The silicon proof of the DSP processor architecture is our future work.

References

[1] B. Furht, Editor-in-Chief, *Handbook of Multimedia Computing*, CRC Press, Boca Raton, Florida, 1999, chap 21.

[2] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, Kluwer Academic Publishers, 1997.

[3] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, pp. 704-716, July 2003.

[4] Y. H. Hu, *Programmable Digital Signal Processors: Architectures, Programming, and Applications*, Marcel Dekker Inc., 2002.

[5] P. Pirsch and H. J. Stolberg, "VLSI implementations of images and video multimedia processing systems," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, pp. 878-891, Nov. 1998.

[6] R. K. Kolagotla, *et al.*, "High performance dual-MAC DSP architecture," *IEEE Signal Processing Magazine*, pp. 42-53, July 2002.

[7] K. Aono, M. Toyokura, T. Araki, A. Ohtani, H. Kodama, K. Okamoto, "A video digital signal processor with a vector-pipeline architecture," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1886-1893, Dec. 1992.

[8] W. Hinrichs J. P. Wittenburg, H. Lieske, H. Kloos, M. Ohmacht, and P. Pirsch, "A 1.3-GOPS Parallel DSP for High-Performance Image-Processing Applications," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 946-952, July. 2000.

[9] H. Kloos, *et al.*, "HiPAR-DSP 16, a scalable highly parallel DSP core for system on a chip video-and image processing applications," *IEEE IGARSS*, pp. 3112-3115, 2002.

[10] *TMS320C64x DSP Library Programmer's Reference*, Texas Instruments Inc., Apr. 2002.

[11] T. J. Lin, C. C. Chang, C. C. Lee, and C. W. Jen, "An efficient VLIW DSP architecture for baseband processing," *IEEE Int. Conf. Computer Design*, pp. 307-312, Oct. 2003.

Table 2: Performance Comparison Results among Various DSP Processors

Performance	Pirsch's Group	TI C'64x	Lin *et al.*	This Work
N-Sample L-Tap FIR for 8-Bit Data Manipulation (Execution Cycles)	$NL/4$	$NL/8$	$NL/4$	$NL/8$
N-Sample L-Tap FIR for 16-Bit Data Manipulation (Execution Cycles)	$NL/4$	$NL/4$	$NL/4$	$NL/4$
RGB2YUV with N Pixels (Execution Cycles)	$9N/4$	$3N/2$	$3N$	$3N/2$
2-D DCT/IDCT for NxN pixels	$N^3/2$	$N^3/2$	$N^3/2$	$N^3/2$
Motion Estimation (Pixel per Cycle)	2#	2	2	8
Peak Performance @ 100 MHz	2.0 GOPS 1*	>5.0 GOPS	2.8 GOPS 2*	3.6 GOPS
# of Function Units	4 (3-Way)	8 (8-Ways)	4 (4-Ways)	4 (4-Ways)
RF Type	N.A.	Centralized	Ring	Grouped Permutation

1*: Original performance is 1.3 GOPS @ 66 MHz [8].

#: Assume that there does not exist SAD-like instruction.

2*: Original performance is 3.0 GOPS @ 100 MHz [11].