Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 58431, 7 pages
doi:10.1155/2007/58431

Research Article

A Video Specific Instruction Set Architecture for ASIP design

Zheng Shen, Hu He, Yanjun Zhang, and Yihe Sun

Institute of Microelectronics, Tsinghua University, Beijing 100084, China

Received 9 May 2007; Revised 2 August 2007; Accepted 14 September 2007

Recommended by Sheldon Tan

This paper describes a novel video specific instruction set architecture for ASIP design. With single instruction multiple data (SIMD) instructions, two destination modes, and video specific instructions, an instruction set architecture is introduced to enhance the performance for video applications. Furthermore, we quantify the improvement on H.263 encoding. In this paper, we evaluate and compare the performance of VS-ISA, other DSPs (digital signal processors), and conventional SIMD media extensions in the context of video coding. Our evaluation results show that VS-ISA improves the processor's performance by approximately 5x on H.263 encoding, and VS-ISA outperforms other architectures by 1.6x to 8.57x in computing IDCT.

1. INTRODUCTION

For competitive markets like consumer electronics or telecommunications, application-specific integrated circuit (ASIC) often lacks flexibility and programmability. Programmable digital signal processor (DSP) is difficult to meet the cost, size, and power consumption demands. Application specific instruction processor (ASIP) can compromise advantages of custom ASIC chips and general DSP chips. In other words, ASIP chips adopt high performance and low power of ASIC chips and flexibility of DSP chips [1–5].

There has been a growing interest in the role of ASIPs in advanced multimedia system [3, 4]. Configurable ASIPs [2] combine elements form both traditional DSP and application-specific hardware accelerator into processor's architecture for media application. By exploiting data parallelism using application-specific instructions, configurable ASIPs gain significant performance improvement.

The common method to design ASIP is exploiting the design space based on a special architecture, including application-specific instruction library and function unit library developing [5]. The addition of key new instructions to the processor for target application is more efficient and easier to update.

Multimedia signal processing technology has been developed with the increasing of consumer electronics demands. Progress in multimedia technology research has led to an increasing number of standards, such as MPEG2, MEPG4, H.263, and H.264/AVC. More recent standards introduce new encoding tools to offer improved performance in terms of picture quality at certain bitrate. However, this does not mean that older standards become obsolete. The introduction of new standards to the market is a gradual process, which means that the instruction set architecture (ISA) for the ASIP design is continuously updated to address and compute requirements of new video standards.

Existing single instruction multidata (SIMD) multimedia extensions and DSPs support various instructions to execute packed operations between two registers. These operations are used for various video signal processing, such as motion estimation and compensation, discrete cosine transform (DCT), and inverse-DCT (IDCT). The key idea in multimedia extensions, such as Intel' MMX, SSE1 and SSE2, Sun's VIS, HP's MAX, Compaq's MVI, MIPS's MDMX, and Motorola's AltiVec [6], is the exploitation of subword parallelism in SIMD fashion. TMS320c64xx of Texas Instruments supports special instructions for multimedia signal processing, such as SUBABS4, AVGx. Trimedia CPU64 introduced two slot operations, collapsed load instructions with interpolation, and so forth [12–17].

This paper presents a novel ISA named VS-ISA (video specific instruction set architecture) to the THUASDSP2004 architecture, which is a video specific DSP architecture for ASIP design and scalable for ISA update [7]. We quantify the performance improvement on H.263 encoding. We also show how the new instruction can be used to optimize modules of other video standards, such as MPEG4 and H.264/AVC.

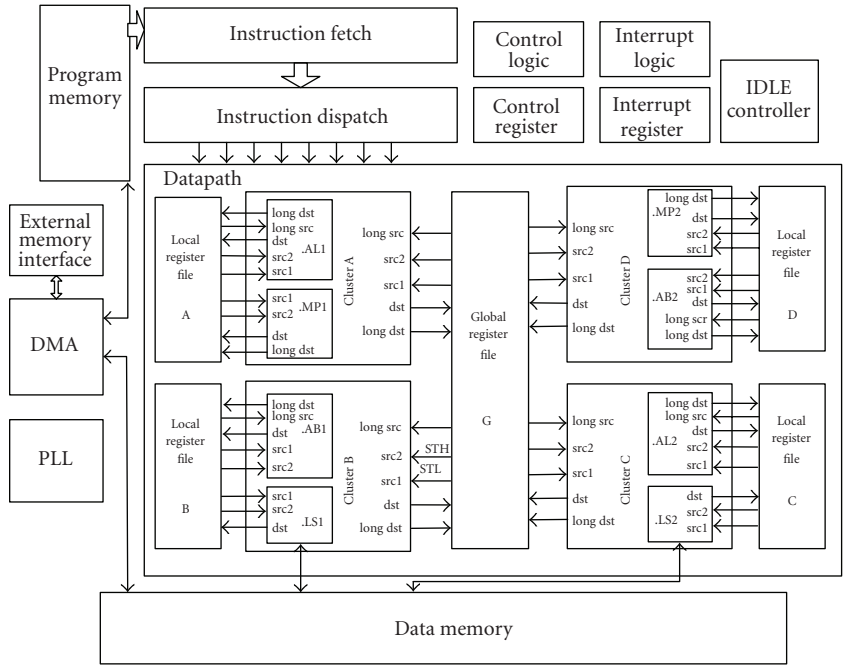


Figure 1: Whole architecture of THUASDSP2004.

The rest of this paper is arranged as follows. Section 2 introduces the enhanced THUASDSP2004 architecture, Section 3 analyzes video applications. Section 6 proposes novel instruction set architectures and Section 5 explains performance evaluations. Finally, conclusions and future work are drawn in Section 6.

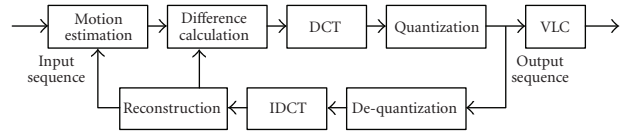


Figure 2: Block diagram of video coding.

2. THUASDSP2004 ARCHITECTURE OVERVIEW

THUASDSP2004 consists of two parts, a very long instruction word (VLIW) architecture DSP core and a hardware coprocessor part. Figure 1 shows the overview of the whole architecture. The VLIW DSP core contains one global register file and four local register files. Every local register file is corresponding to a function unit cluster. A function unit cluster can contain two function units. Every function unit in a cluster has its own access port to the corresponding local register file, and all function units in the same cluster share a set of access ports to the global register file. Every register file contains sixteen 32-bit registers [7]. The hardware coprocessor part has a variable-length coding (VLC) processor for encoding, a variable-length decoding (VLD) processor for decoding, an image coprocessor for transforming color format between YUV and RGB, and a video specific direct memory access (DMA) for video data transfer.

THUASDSP2004 ISA has basic 74 instructions and can work well at the frequency of 150 MHz. As VLIW architecture, the DSP core issues 8 instructions every cycle, and the instruction throughput is 1200 MIPS.

THUASDSP2004 has a VLIW-like architecture, which can be compared with TMS320C6000 family of Texas Instruments (TI). And the performance of THUASDSP2004 is similar as that of TMS320C62xx in video coding [7].

3. VIDEO APPLICATION CHARACTERISTICS

As seen in Figure 2, standard source video codecs such as MPEG4, H.263, and H.264/AVC consist of standard modules, such as motion estimation (ME), discrete cosine transform (DCT), quantization and dequantization, and so forth.

Within video application, the high degrees of parallelism and high memory bandwidth have been well researched [8]. The size of datapath is another factor that has yet to be determined for video processing. The integer data size for multimedia is typically believed to require only 8 or 16 bits.

We primarily focus on the data parallelism, memory bandwidth, and the small integer operands in instruction set designing. We applied certain rules while identifying new instructions.

(a) A rule that is specific on video application characteristic

New instructions proposed are expected to get significant performance enhancement for video coding. Performance improvement should be measured at the application level, rather than the module level.

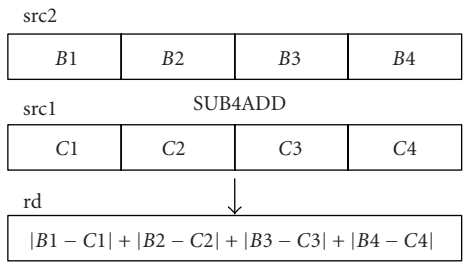


Figure 3: SUB4ADD instruction in VS-ISA.

(b) A rule that is based on the processor architecture

New instructions typically add function logic to the exiting datapath. The increase of critical path caused by addition logic should be avoided or kept to minimum.

4. VIDEO SPECIFIC INSTRUCTION SET ARCHITECTURE

This section describes VS-ISA and its enhancement to THUASDSP2004 basic ISA on video application. All modules' implementations are self-contained functions: function-call and function–return overheads are included, function inputs are read from memory, and outputs are written to memory.

As mentioned in Section 3, video application data has the character of high degree of parallelism and the integer data sizes for video typically require only 8 or 16 bits. For instances, the data sizes are 8 bits for motion estimation in almost every video codec standard, such as H.263, H.264/AVC, while the data sizes are 16 bits for integer transform. THUASDSP2004 having 32-bit datapath, within SIMD operations, packed data types store either four 8-bit values or two 16-bit values in a single 32-bit register, or four 16-bit values in a 64-bit register pair. VS-ISA proposes SIMD instructions of packed 8-bit data type for motion estimation and packed 16-bit data type for integer transform, quantization, and invquantization.

There are two destination modes in VS-ISA: single destination mode and double destination mode. When single destination mode is used, results will be written to the destination register specified by destination address. While double destination mode is used, results will be written not only to the destination register but also the associate register of the destination register. By this mean, results can be easily duplicated while it costs only one bit in instruction code to indicate the destination mode. Data duplication is very useful when the result of one operation is used by many sequent operations, especially when software pipelining is executing [9].

4.1. Proposed instructions for motion estimation

A significant part of the computation complexity of video application is found in motion estimation. Motion estimation is used to exploit the inherent temporal redundancy of a video image. In a typical motion estimation process, each frame is divided into 16 × 16 macroblocks. Most computational complexity of the algorithm is found in the "macroblock matching" module, which determines the similarity of a macroblock in the current image with a motion-displaced macroblock in the reference image. We implemented a five-step motion estimation algorithm, the search step of which is 8-, 4-, 2-, 1-, or 1/2-pixel, and the search range is 3 × 3 macroblock.

Typically, the match criterion is the sum of absolute difference (SAD) function. To complete the matching of one macroblock, SAD function would be invoked for 41 times on H.263 encoding,

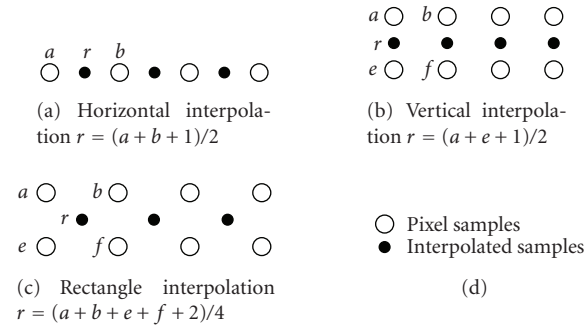
$$\mathrm{SAD}(a,b,m,n)=\sum_{j=0}^{15}\sum_{i=0}^{15}\left|Y1(a+i,b+j)-Y0(m+i,n+j)\right|, \tag{1}$$

$$A+=|B1-C1|+|B2-C2|+|B3-C3|+|B4-C4|, \tag{2}$$

where (a, b) and (m, n) are the upper left corner positions of the current macroblock and the 16 × 16 region from the searching range of previous reconstructed frame,

Table 1: SAD using VS-ISA and software pipeline.

(a) Function units operations in SAD module				
LS1	LDDWNA(.LS1) *B_address, B1_reg:B2_reg			
LS2	LDDWNA(LS2) *C_address, C1_reg:C2_reg			
AL1	SUB4ADD(.AL1) sad1_reg, B1_reg, C1_reg,			
	//SUB4ADD(.AL1) rd, src1, src2			
AL2	SUB4ADD(.AL2) sad2_reg, B2_reg, C2_reg,			
AB1	ADD(.AB1) SAD1, SAD1, sad1_reg			
	// SAD = SAD + sad_reg			
AB2	ADD(.AB2) SAD2, SAD2, sad2_reg			
(b) SAD assemble code with software pipeline				
Cycle				
1	LS1, LS2			
2		LS1, LS2		
3			LS1, LS2	
4	AL1, AL2			—
5	AB1, AB2	AL1, AL2		
6		AB1, AB2	AL1, AL2	
7			AB1, AB2	—



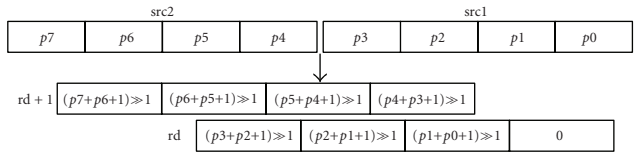
(a) Horizontal interpolation $r = (a + b + 1)/2$

(b) Vertical interpolation $r = (a + e + 1)/2$

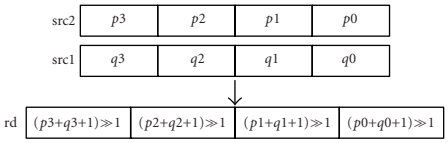
(c) Rectangle interpolation $r = (a + b + e + f + 2)/4$

(d)

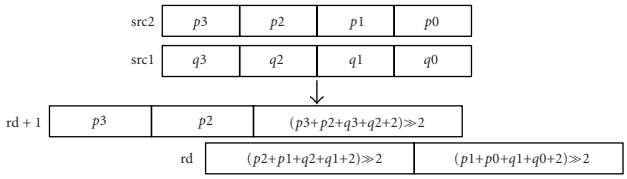
Figure 4: Interpolation modes.



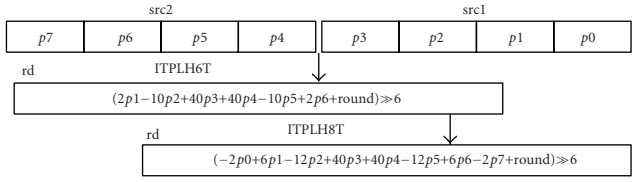
(a) ITPL8PH(.AL1) rd + 1 : rd, src2, src1, .AL1, .AL2



(b) ITPL8PV(.AL1) rd, src2, src1, .AL1, .AL2

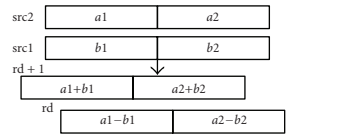


(c) ITPL8PR(.AL1) rd + 1 : rd, src2, src1, .AL1, .AL2

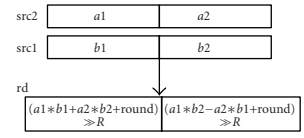


(d) ITPLH6T(.AL1) rd, src2, src1, .AL1, .AL2. ITPLH8T(.AL1) rd, src2, src1, .AL1, .AL2

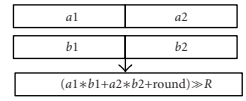
Figure 5: ITPL instruction in VS-ISA.



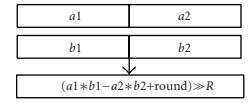
(a) BTF2(.AL1) rd + 1 : rd, src2, src1, .AL1, .AL2, .AB1, .AB2



(b) RTT2SRMx(.MP1) rd, src2, src1, .MP1, .MP2. mode0: $R = 9$, mode1: $R = 13$, mode3: $R = 16$



(c) DPD2SRMx (.MP1) rd, src2, src1, .MP1, .MP2



(d) DPDN2SRMx (.MP1) rd, src2, src1, .MP1, .MP2

Figure 6: Instructions for integer transform.

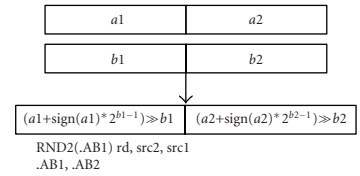


Figure 7: Instructions for quantization and dequantization.

Table 2: Architecture overhead of VS-ISA.

Critical path overhead		Hardware cost (area) overhead	
AL	1.2%	AL	4.3%
AB	3.2%	AB	5.6%
MP	5.1%	MP	7.3%
LS	1.1%	LS	6.8%

respectively, Y1 and Y0 are the pixel luminance values from current frame and reconstructed frame. Equation (1) is composed of sixty four (2). With THUASDSP2004 basic ISA, (2) needs eight 8-bit loads, four subtractions, four absolutions, four additions, totally 20 operations. As Figure 3 shows, with SUB4ADD and LDDWNA (load nonaligned doubleword) instructions in VS-ISA, (2) just needs an LDDWNA, a SUB4ADD, and an addition, totally three operations. As seen in Table 1, using software pipeline, one loop can implement a doubleword operation (64-bit, eight 8-bit pixels), so the execution cycles of SAD is reduced from 263 cycles to 37 cycles, and the performance gets an enhancement of 7.1x.

Whereas the H.263 and MPEG2 standard allows for fractional motion vectors at half pixel granularity, the MPEG4 and H.264/AVC standards support quarter pixel granularity. The calculation of reference data at fractional positions is more involved. As the reference data position is not word align, so the processor with LDDWNA (load nonaligned doubleword) can load 64 bits (eight 8-bit pixels) every cycle.

Interpolation is another key module in motion estimation, which is used in subpixel prediction. For H.263, as seen in Figure 4, interpolation can be classified to 3 modes of operations: horizontal interpolation, vertical interpolation, and rectangle interpolation. With ITPL8PH, ITPL8PV, ITPL8PR, shown in Figure 5, the performance gets an enhancement of 3.4x.

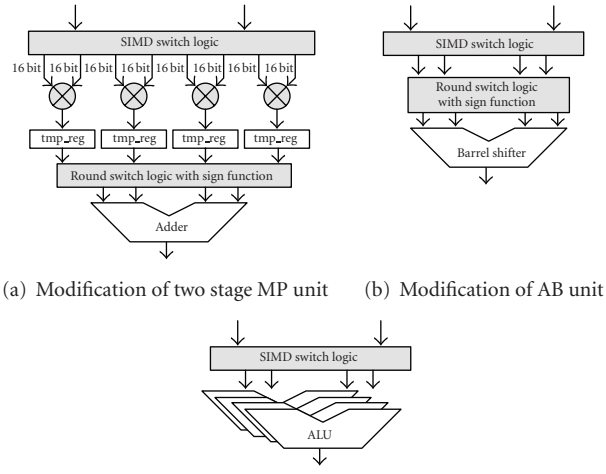
More complex interpolation is efficiently implemented with ITPLH6T, ITPLH8T instructions. Given 8 horizontal neighboring image pixels, $p0, p1, \ldots, p7$, with pi located at horizontal position I, as seen in formula (3), H.264/AVC standard calculates r at horizontal position 3(1/2) using a 6-tap filter; while in formula (4), MPEG4 standard calculates r at horizontal position 3(1/2) using an 8-tap filter,

$$\begin{aligned} r &= 2p1 - 10p2 + 40p3 + 40p4 - 10p5 + 2p6 + 32 \\ r &= \mathrm{Max}(\mathrm{Min}(p \gg 6, 0), 255), \end{aligned} \tag{3}$$

$$\begin{aligned} r = {} & -2p0 + 6p1 - 12p2 + 40p3 + 40p4 \\ & - 12p5 + 6p6 - 2p7 + \mathrm{rounding} \\ r = {} & \mathrm{Max}(\mathrm{Min}(p \gg 6, 0), 255). \end{aligned} \tag{4}$$

4.2. Proposed instructions for integer transform

The 8×8 2D-DCT and 2D-IDCT are row-column separated into 8-point 1D transforms with 16-bit width data. We use the Chen algorithm [9] for DCT and Loeffler algorithm [10] for IDCT. Both algorithms make frequent use of butterfly



(a) Modification of two stage MP unit (b) Modification of AB unit

(c) Modification of AL unit

Figure 8: Function unit modification.

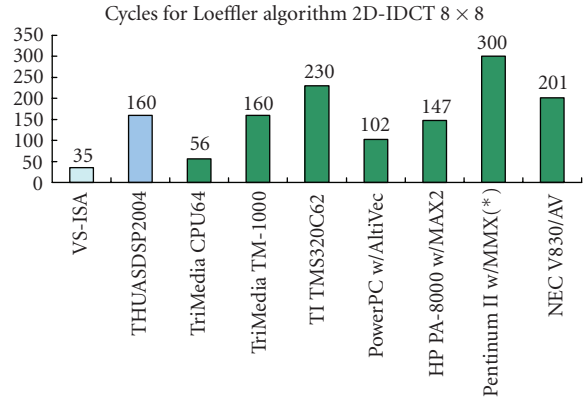


Figure 9: Cycles for 2D-IDCT.

and rotate operators, defined by formula (5). Either operator has two inputs and produces two outputs,

$$\begin{aligned} \text{butterfly}(a,b): r1 &= a+b; \\ r2 &= a-b; \\ \text{rotate}(a,b): r1 &= a^* \cos(\theta) - b^* \sin\theta; \\ r2 &= a^* \sin(\theta) + b^* \cos(\theta). \end{aligned} \tag{5}$$

VS-ISA includes specific instructions for butterfly and rotate operators. As seen in Figure 6, BTF2 (two 16-bit subwords butterfly) could implement two 16-bit conjugated operators in one clock cycle, and two 16-bit rotate operators using RTT2SRMx (two 16-bit subwords rotate with signed round in mode x) could be calculated in parallel. Signed rounding has 3 clip modes; each mode has a given r. VS-ISA also uses dot product instructions to process rotate operator. Two-dot production instructions, as shown in Figures 6(c) and 6(d), could implement rotate operator in two-clock cycles.

The DCT code is compiled and scheduled manually into 35 VLIW instructions, including function call/return overhead, loading the 8×8 pixels, performing the 2-dimensional DCT, and storing the 8×8 coefficients back in memory.

4.3. Proposed instructions for quantization and dequantization

Quantization module produces a block of quantized frequency coefficients, based on a matrix of quantized value, as (6). While dequantization produces a block of dequantized frequency coefficients, based on inverse quantized value, as (7). Both algorithms make frequent use of round operator, as seen in (8). With THUASDSP2004 basic ISA, round operation needs two instructions to implement sign function, taking up a condition register, NEG instruction to negative coefficient, ADD, and ASHR to add round and shift right, totally 5 operations. As seen in Figure 7, with RND2 instruction in VS-ISA, two 16-bit subwords round operation just needs 1 operation,

$$\begin{aligned} \text{level} = \text{sign}(\text{COF}) \times [(|\text{COF}| \times \text{reciprocals} + \text{round}) \\ \gg Q\_pt]; \end{aligned} \tag{6}$$

$$2QP = \text{reciprocals} \gg Q\_pt;$$

$$\text{COF}' = \text{sign}(\text{level}) \times QP \times (2 \times |\text{level}| + 1) \tag{7}$$

$$\begin{aligned} a &= (b + \text{sign}(b) \times 2^{R-1}) \gg R; \\ &\text{if } (R = 0), a = b + \text{sign}(b). \end{aligned} \tag{8}$$

4.4. Hardware modification for proposed instructions

As seen in Figure 8, for implementing SIMD instructions, SIMD switch logic is added to all function units. Within SIMD operations, packed data types store either four 8-bit values or two 16-bit values in a single 32-bit register, or four 16-bit values in a 64-bit register pair. For packed two 16-bit data, the switch logics dispatch the single 32-bit register value as two independent 16-bit values to function units. And for packed 8-bit data, the switch logics dispatch four independent 8-bit values to function units.

Function units (FUs) need some extra logic for the proposed instructions. The most significant modification is in MP unit. As seen in Figure 8(a), for supporting 32×32 multiplication as well as parallel 16×16 multiplication, there are four 16×16 multipliers with round logic, to implement video specific instructions, such as BTF2, RTT2SRMx. The MP unit is the critical path of the whole processor. To support RND instructions, a sign function unit is added to AB unit, as seen in Figure 8(b). And to support interpolate operations, the SIMD combine logic of AL unit needs combining the eight 8-bit addition results to a 64-bit value, and storing to a 64-bit register pair.

Table 2 shows the hardware overhead for supporting proposed SIMD and video specific instructions. Comparing with the basic architecture, the critical path of modified function units increase from 1.1% to 5.1%, and the modified function unit area increases from 4.3% to 7.3%.

Table 3: Performance on H.263 encoding with normalized data.

Sequence	Bitrate	H.263 encoding (fps), working on 150 MHz			
		TMS320-C62xx	TMS320-C64xx	THUASDSP2004	VS-ISA
Foreman QCIF	32 k	124	253	96	451

Table 4: Performance with THUASDSP2004 ISA and VS-ISA.

H.263 encoding module/function	THUASDSP2004 (cycles)	VS-ISA (cycles)	Enhancement
DCT_8 × 8	160	35	4.57x
IDCT_8 × 8	168	38	4.42x
Quant_8 × 8	138	22	6.27x
Dequant_8 × 8	170	18	9.44x
SAD_8 × 8	263	37	7.12x
Interpolate_MB	913	268	3.41x
Motion estimation_MB	11604	2385	4.87x
Difference calculation_MB	962	286	3.36x
Reconstruction_MB	1230	375	3.28x
I-frame	3992	692	5.77x
P-frame	17976	3822	4.70x

5. PERFORMANCE EVALUATION

We use assembly code written manually to evaluate the performance of VS-ISA. The processor operates at 150 MHz, synthesized with SMIC 0.18 library under worst case condition, with 128-Kbytes onchip memory.

We encoded the "Foreman" sequence at QCIF resolution at 451 frames per second, as shown in Table 3, with a target bitrate of 32 kbps on H.263 encoding. Frames are represented in a 4 : 2 : 0 format, resulting in six blocks per macroblock, while THUASDSP2004 encodes the video sequence at 96 frames per second and TI's TMS320C64xx encodes at 253 frames per second.

Table 4 gives a performance comparison between VS-ISA and THUASDSP2004. As seen, VS-ISA enhances the processor performance by 5.77x in I-frame and 4.70x in P-frame.

As an example of the power of the combined approach of VLIW, subword parallelism, and an extensive set of operations, the inverse-discrete cosine transform (IDCT) [11] computation is an interesting benchmark from the targeted application domain. Figure 9 shows the comparison of the number of clock cycles to compute the IDCT of an 8 × 8 block between VS-ISA and other different architectures. As seen, VS-ISA outperforms both DSPs and SIMD multimedia extensions architectures by 1.6x to 8.57x in computing the IDCT [12–17].

Figure 9 has been included for comparison of instruction sets, but is not suited for performance comparisons. The VS-ISA is still in development, whereas the other processors are already in production. Moreover, clock frequency differences are significant, and the hardware cost is not under consideration.

6. CONCLUSION AND FUTURE WORK

This paper proposes an efficient instruction set named VS-ISA and its hardware architecture to implement the video application. A processor with VS-ISA based on THUASDSP2004 6-issue VLIW architecture is implemented to quantify the performance. The power of VLIW parallelism in combination with SIMD subword parallelism and the video specific instruction set have been demonstrated with an extremely compact IDCT standard module and the whole H.263 baseline encoding, and a satisfied result achieved.

In the future, we will optimize the datapath with full customized designing, rather than being synthesized with standard cell library, to improve the performance of the processor. The work will be done to quantify the performance improvement on H.264/AVC and MPEG4 standard.

ACKNOWLEDGMENTS

This paper is supported by the National Natural Science Foundation of China (NSFC) under Grant no. 60236020. The project funded by Tsinghua Basic Research Foundation.

REFERENCES

[1] I. Kuroda and T. Nishitani, "Multimedia processors," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1203–1221, 1998.

[2] R. E. Gonzalez, "Xtensa: A Configurable and Extensible Processor," *IEEE Micro*, vol. 20, no. 2, pp. 60–70, 2000.

[3] M. Gries and K. Keutzer, *Building ASIPs: The MESCAL Methodology*, Springer, New York, NY, USA, 2005.

[4] P. Ienne and R. Leupers, "Customizable Embedded Processor: Design Technologies and Applications," in *Elsevier Morgan Kaufmann*, San Francisco, Calif, USA, 2006.

[5] K. Kucukcakar, "ASIP design methodology for embedded systems," in *Proceedings of the 7th International Workshop on Hardware/Software Codesign (CODES '99)*, pp. 17–21, May 1999.

[6] R. B. Lee, "Multimedia extensions for general-purpose processors," in *Proceedings of IEEE Workshop on Signal Processing Systems: Design and Implementation (SIPS '97)*, pp. 9–23, November 1997.

[7] Y. Zhang, H. He, Z. Zhou, X. Yang, and Y. Sun, "A scalable DSP System for ASIP design," in *Proceedings of the 8th International Conference on Solid-State and Integrated Circuit Technology (ICSICT '06)*, San Francisco, Calif, USA, October 2006.
[8] H. Liao and A. Wolfe, "Available parallelism in video applications," in *Proceedings of the 13th Annual International Symposium on Microarchitecture*, pp. 321–329, December 1997.
[9] Y. Zhang, H. He, and Y. Sun, "A New Register File Access Architecture for Software Pipelining in VLIW processors," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, January 2005.
[10] W. Chen, C. Harrison, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004–1011, 1977.
[11] C. Loeffer, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '89)*, vol. 2, pp. 988–991, May 1989.
[12] SPRA018, Texas Instruments Inc.
[13] N. Seshan and W. Sites, "High velociTI processing," *IEEE Signal Processing Magazine*, vol. 15, no. 2, pp. 86–101, 1998.
[14] J. T. J. van Eijndhoven, F. W. Sijstermans, K. A. Vissers, et al., "TriMedia CPU64 architecture," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '99)*, pp. 586–592, Austin, Tex, USA, October 1999.
[15] Intel MMX technology application notes, "Using MMX Instructions in a Fast iDCT Algorithm for MPEG Decoding," http://www.intel.com/drg/mmx/appnotes/ap528.htm.
[16] PowerPC: Altivec Performance Table, http://developer.apple.com/hardware/altivec/performance-Table.html, 1997.
[17] R. Lee, "Effectiveness of the MAX-2 multimedia extensions for PA-RISC 2.0 processors," in *Hot Chips IX Symposium*, pp. 135–148, Palo Alto, Calif, USA, August 1997.