ASIP for Multi-Standard Video Decoding

Jae-Jin Lee, KyungJin Byun and NakWoong Eum
Multimedia Processor Research Team
Electronics and Telecommunications Research Institute
Daejeon, Korea
{ceicarus, kjbyun, nweum}@etri.re.kr

***Abstract*—Multiple international video standards in the market have been developed successfully for many commercial products. Application-specific instruction processor is a new design methodology to develop optimized processor. This paper proposes a new application-specific instruction set processor based on 6-stage pipelined dual issue VLIW+SIMD architecture and compiler for multi-standard video decoding. The processor takes 130K in gate count at 125MHz in 130nm technology. Compared to the existing ARM processor, the proposed processor results in about 20% speed improvement as well as smaller hardware complexity.**

Keywords-multimedia processor; application-specific instruction processor; video decoding.

I. INTRODUCTION

In the implementation of embedded systems, the designers confront with decision of architectures which is combination of ASICs [1], FPGAs [2] , ASIPs (Application Specific Instruction-Set Processors) [3], DSPs [4], and GPPs (general purpose processors) [5]. These decisions are mainly based on performance, power consumption, flexibility and silicon area of systems. ASIPs are powerful solutions when the contradicting requirements such as performance and flexibility have to be jointly satisfied with a single task block. The flexibility of ASIP is caused by the necessity to support multi-standard video codecs such as AVS [6], VC-1 [7], and H.264 [8] in a single platform. On the other hand, next generation video codecs require extreme demands on throughput and processing of continuous data streams at high rates.

Video compression technologies have been dramatically evolved by many researchers and industries. Successful multiple international standards in the market have been released for last two decades. In particular, ISO/IEC WG11/MPEG and ITU-T SG16/VCEG have developed MPEG-1/2/4 and H.261/262/263 to compress raw digital videos since early 1990 [9][10][11]. Subsequently, the MPEG and VCEG jointly standardized the H.264/AVC [8] which is suitable for various network environments and gives the highest coding efficiency in 2003. In recent years, various video codecs such as VC-1 and so on have been commercialized, aside by the standard codes developed by the international standardization bodies. This leads a huge amount of multimedia contents to be compressed with the increasing number of video coding techniques and distributed over various networks and devices.

ASIP is a new design methodology to develop optimized processors for specific applications by adding specific instructions into base instructions for eliminating functional hot spot of applications [3]. In terms of video decoding [12][13][14], the ASIP has higher performance than DSP because of its optimized application specific instructions, and has better flexibility and reusability than ASIC because any applications can be implemented with software.

The remainder of this paper is organized as follows. In the next section, ASIP and compiler for multi-standard video decoding are briefly overviewed. In Section 3, we have evaluated the proposed ASIP. Finally, we summarize the paper and conclude it mentioning future works.

II. ASIP AND COMPILER

As shown in Figure 1, the proposed ASIP providing a separate data and program memory (Harvard architecture) consists of dual issue VLIW (Very Long Instruction Word) + SIMD (Single Instruction Multiple Data) core, program/data cache interface, general purpose register file consisting of 16 32-bit registers, special purpose register file for SIMD instructions and bus interface to access external memory.

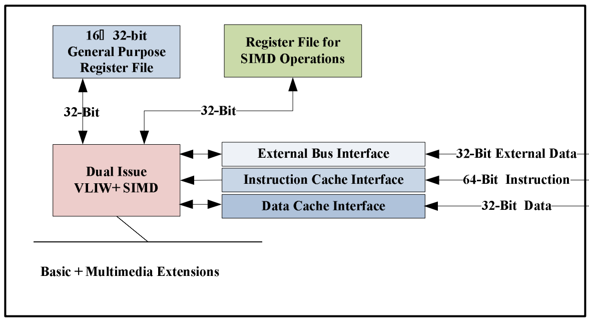

16 32-bit General Purpose Register File
Register File for SIMD Operations
32-Bit
32-Bit
Dual Issue VLIW+ SIMD
External Bus Interface
Instruction Cache Interface
Data Cache Interface
32-Bit External Data
64-Bit Instruction
32-Bit Data
Basic + Multimedia Extensions

Figure 1. Block diagram of ASIP

The behavior, the structure, and the I/O interface have been described using LISA (language for instruction set architecture) [15]. It parses the description and generates the tools and models necessary for software design and architecture implementation such as assembler, dis-assembler, linker, ISS (Instruction Set Simulator). C compiler has been generated using the C-compiler designer tool of CoWare [16]. It provides a rich set of optimization and restructuring engines that include typical high level optimizations such as copy and constant propagation, code motion, loop unrolling, loop fusion, and etc.

A. Pipeline and Bypass Logic

The proposed architecture is based on a pipeline with 6 stages as shown in Table I. The pipeline is fully bypassed, i.e., instructions reading from register R can directly follow the instruction writing to the same register.

TABLE I. PIPELINE OF THE PROPOSED PROCESSOR

Stages	Descriptions
PF(PreFetch)	Branch address or zero-overhead loop detection
FE(Fetch)	Fetch from the instruction memory
DC(DeCode)	Decode the instruction and read the operands
EX(EXecution)	Execute the ALU or logical operations
MEM(MEMory)	Read or write the memory
WB(WriteBack)	Write the results back into registers

The bypass logic [17][18] ensures a consistent read access between the instructions and makes sure that the latest result for a register is read by the instruction. A bypass is required when an instruction X in the execute stage (EX) is producing a result that is read by the following instruction Y. Instruction Y is at that time in the decode (DC) stage and requesting the result. A bypass allows instruction Y to access the result before it is actually written back into the main register file.

The MAC (Multiply-Accumulate) is very beneficial to speed-up many different type of applications. As shown in Figure 2, the proposed architecture has the pipelined dual cycle MAC supported by C compiler by sharing multiplier. This results in efficient micro-architecture from an area cost point of view.

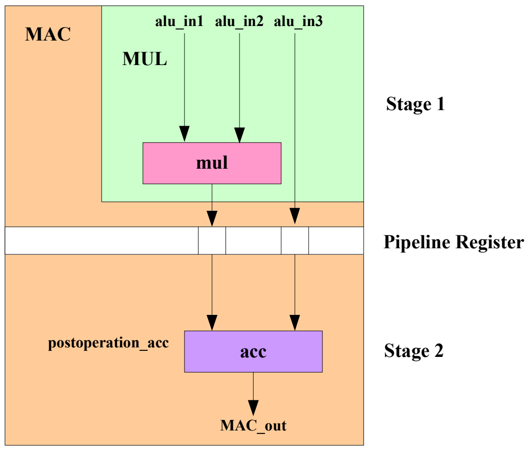


Figure 2. Dual cycle MAC

Figure 3 provides a block diagram of the bypass logic. It can be seen that the main register file is accessed in the DC stage. The operands are immediately pushed into the pipeline registers “op1”, “op2” and “op3”. If a custom instruction would need these operands already in the DC stage, the values can also be written into a signal instead of a register. Thus, those signals can be used to any combinatorial data-path in the DC stage. In EX stage, the latest operand value is written into the signal “alu_in1”, “shifter_in1” and “shifter_in2”. Once the result is computed the “writeback_dst” operation need to be activated (not shown here) and the register address “BPR” as well as the writeback value “WBV” need to be written.

B. Instruction Set

The Instruction set of the proposed processor consists of basic load/store, arithmetic, logic, branch and trap instructions, and multimedia extensions for multi-standard video decoding as shown Table II.

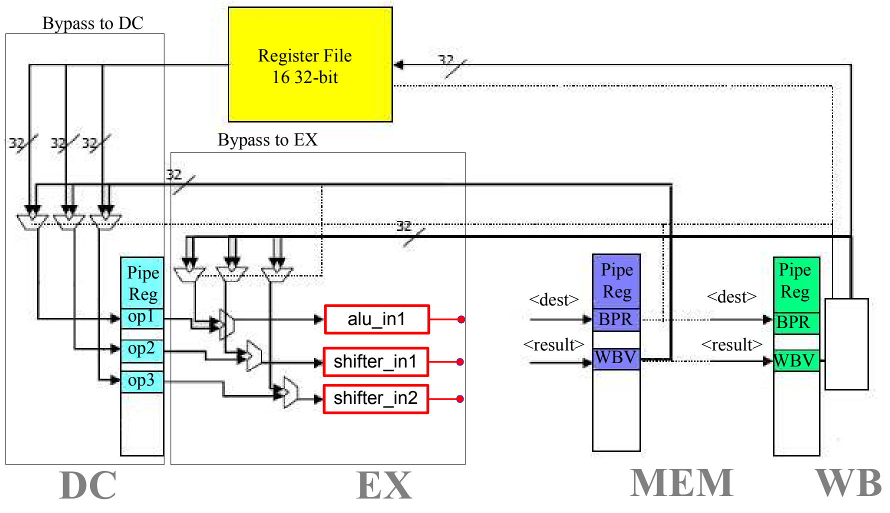


Figure 3. Block diagram of bypass logic

TABLE II. MULTIMEDIA EXTENSIONS FOR MULTI-STANDARD VIDEO DECODING

Instructions	Description
lmax	Compute maximum value of the two operands ex) int32 lmax_ckf(int32 data1, data2)
lmin	Compute minimum value of the two operands ex) int32 lmin_ckf(int32 data1, int32 data2)
smul_4	Computes the signed multiplication of 4x8-bit input operands ex) int32 smul_4_ckf(uint32 data1, uint32 data2)
umul_4	Computes the unsigned multiplication of 4x8-bit input operands ex) uint32 smul_4_ckf(uint32 data1, uint32 data2)
labs_4	Compute SIMD ABS ex) int32 labs_4_ckf(int32 data1, int32 data2)
bilf	Performs bi-linear filtering operation of 4x8-bit data1 and data2 ex) uint32 bilf_ckf(uint32 data1, uint32 data2, uint32 round_flag)
lclip	Performs clipping operation of data1 ex) uint32 lclip_ckf(int32 data1, int32 min, int32 max)
bext	Performs byte extension of data1 Extension type is determined by flag value (MSB or LSB) ex) uint32 bext_ckf(uint32 data1, uint32 flag)
add_clip4	Performs SIMD addition and clipping operations 2x16-bit data1 and data2 and 4x8-bit data3 ex) uint32 add_clip4_ckf(uint32 data1, uint32 data2, uint32 data3)
clz	Counts the leading zeros of data1 ex) int32 clz_ckf(uint32 data1)

As show in Table II, the proposed processor has special SIMD instructions for multi-standard video decoding. To extract these multimedia extensions, we have performed in depth profiling of existing multi-standard video decoding software such as MPEG-2, MPEG-4, AVS, VC-1 and H.264/AVC.

By applying the proposed multimedia extensions, we can effectively improve the performance of the video decoding algorithm. For example, the 'add_clip4' instruction is employed to add reconstructed residual values and prediction values and then clips the outcomes to 0~255 for four pixels at the same time in the 'Reconstruction' module. Since the prediction value is 8 bits/pixel and the residual value is maximum 16 bits/pixel in general video decoders, we have implemented an 'add_clip4' instruction to process four pixels at the same time, as shown in Figure 4.

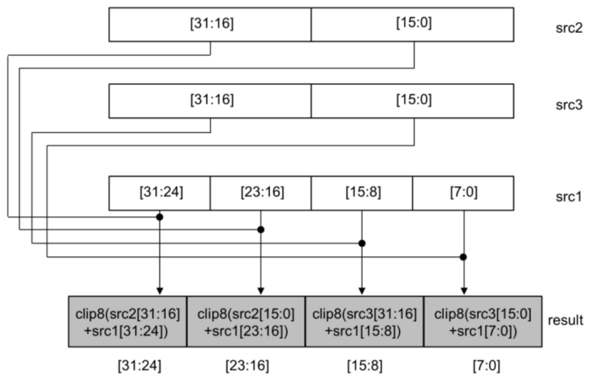

[31:16]
[15:0]
src2
[31:16]
[15:0]
src3
[31:24]
[23:16]
[15:8]
[7:0]
src1
clip8(src2[31:16] +src1[31:24])
clip8(src2[15:0] +src1[23:16])
clip8(src3[31:16] +src1[15:8])
clip8(src3[15:0] +src1[7:0])
result
[31:24]
[23:16]
[15:8]
[7:0]

Figure 4. The operation of 'add_clip4' instruction

The number of inputs for 'add_clip4' is three and each input is 32 bits. 8-bit prediction values for the successive four pixels are entered into the first 32-bit parameter (src1) and the 16-bit residual data for the successive four pixels is entered into the second and the third parameters (src2, src3). Four successive pixels with addition and clipping operation can be processed in a single cycle. 'clz' instruction is efficiently used for the optimization of context adaptive variable length decoding.

C. C compiler

The compiler for the proposed architecture is developed with the C-compiler designer tool. Multimedia extensions are mapped by CKF, inline assembly and Matcher rules [19], respectively.

CKF stand for compiler known function. It is used to implement certain special instruction combination which would not usually be output by the compiler. The advantage of CKFs over inline assembly is that it gives more control to the C-compiler designer than to the user. Designers need not be aware of the assembly instructions that are required to implement the functionality.

C Compiler for ASIP provides a rich set of optimization and restructuring engines that include typical high level optimizations such as copy and constant propagation, code motion, loop unrolling, loop fusion and etc.

III. IMPLEMENTATION AND EXPERIMENTAL RESULTS

Although multimedia extensions proposed in this paper can be used to replace complicated operations efficiently, they are not enough to achieve the desirable performance by adopting only multimedia extensions. Essentially, C-level code optimization of the video codecs is required for real-time applications with minimum power consumption.

8-bit 4:2:0 format videos are widely used in the market. Most software video decoders mainly employ 8-bit memory access for the decoded picture buffers. However, most embedded processors support physical 32-bit memory access for memory load and store operations. Even though one pixel value is accessed by general software implementations, four bytes are loaded or stored into external physical memory. In video decoders, memory access is usually a bottleneck with higher resolution videos and this pixel-wire memory access could make the data access rate worse. Thus, we need to optimize the software with word aligned memory access for decoded picture buffers. For example, in the motion compensation module, a motion vector would be (5, 5) which are not multiples of four. The MPEG-2 decoder needs to load 17×17 pixels from the reference location (5, 5) because MPEG-2 employs half-pixel interpolation. However, the proposed decoder loads 20×20 pixels from the pixel (4, 4) by simultaneously considering the word-alignment memory access and the half-pixel interpolation. For H.264/AVC, as it uses the quarter-pixel interpolation with 6-tap FIR filter, two more left and upper pixels can be loaded and three more right and lower pixels are loaded. In other word, all the (21×21) pixels from (3, 3) to (23, 23) should be loaded regardless of the sub-pixel locations. As a result, (21×24) pixels from (3, 0) to (23, 23) are loaded by the proposed algorithm due to the word-alignment. The word-aligned position is calculated and the amount of alignment is defined by

$$\begin{aligned} aligned\_x &= (((mv\_x\text{-}2) >> 2) << 2) \\ aligned\_y &= mv\_y\text{-}2 \end{aligned} . \quad (1)$$

The LISA processor design platform offers the possibility to generate structured RTL model by grouping operations into functional units. Each functional unit in the LISA model represents an entity or a module in the HDL model. The generated Verilog RTL code for application specific processor is synthesized using Synopsys Design Compiler and implemented by SMIC 130nm cell library. The Figure 7 shows architecture of multi-core SoC (MOSAIC) including eight application-specific instruction processors and Table III shows features of the implemented multi-core SoC. As shown in Figure 7 the multi-core architecture consisting of four clusters including two ASIPs, DMA, TCM (Tightly Coupled Memory), inter-core buffer and communication manager. Two ASIPs are clustered for pipelined operations and each cluster is the basic unit for implementation of multi-core system. To reduce communication overhead, three types of hierarchical communication architecture such as private cache, shared cache and inter-core buffer have been proposed. PCIe interface is used for communication between host and target and video controller is for displaying decoded image.

The multi-standard video decoding algorithms are mapped into multi-core architecture by novel parallelization method called MB (macroblock) row-level parallelism [20].

Four CIF(352×288) video sequences such as ‘Foreman’, ‘Mobile’, ‘Paris’, and ‘Tempete’ are used for evaluating decoding performance of the propose processor. Table IV shows detail encoding parameters of the test sequences and Table V shows the results of speed-up achieved by adopting multimedia extensions into four video decoding algorithms in terms of the decoding cycle. Compared to other decoders, speed-up of MPEG-2 decoder is very high because a large portion of the MPEG-2 decoding algorithm has been optimized with the proposed multimedia extensions

Figure 5 shows EVM board decoding VGA video stream encoded by MPEG-4 video standard. In addition to multi-standard video decoding, we have mapped various detection algorithms such as motion, lane and face detection algorithm into multi-core SoC.

The proposed processor results in about 20% speed-up in terms of processing cycles, compared to conventional ARM1020E processor [25]. Figure 6 shows the number of cycles required for ARM1020E and the proposed ASIP to process inverse transform and quantization of H.264 CIF test sequences.

TABLE III. SPEED-UP BY MULTIMEDIA EXTENSIONS

Sequences	**MPEG-2**	**MPEG-4**	**AVS**	**H.264**
Foreman	2.30	1.82	2.07	1.01
Mobile	2.07	1.58	1.68	1.01
Paris	2.29	1.76	1.84	1.01
Tempete	2.15	1.64	1.73	1.01
Average	**2.20**	**1.7**	**1.83**	**10.1**

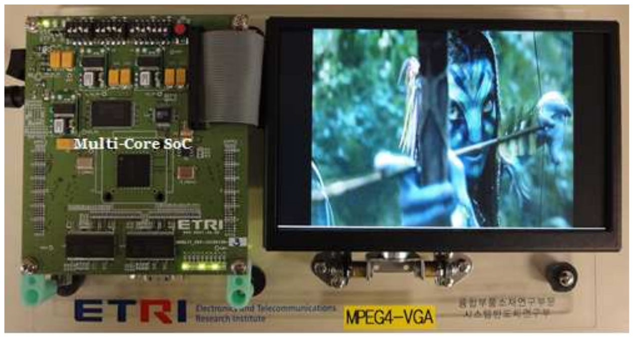


Figure 5. EVM board of multi-core SoC (MOSAIC)

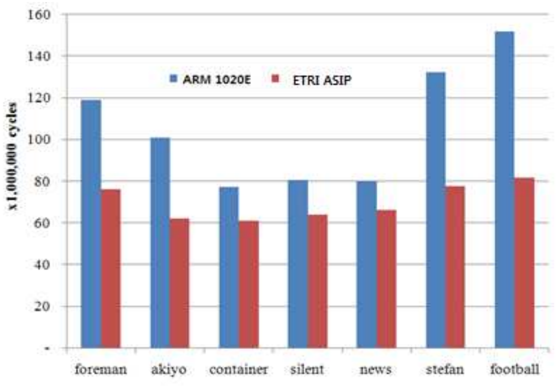


Figure 6. ARM 1020E vs. ETRI ASIP

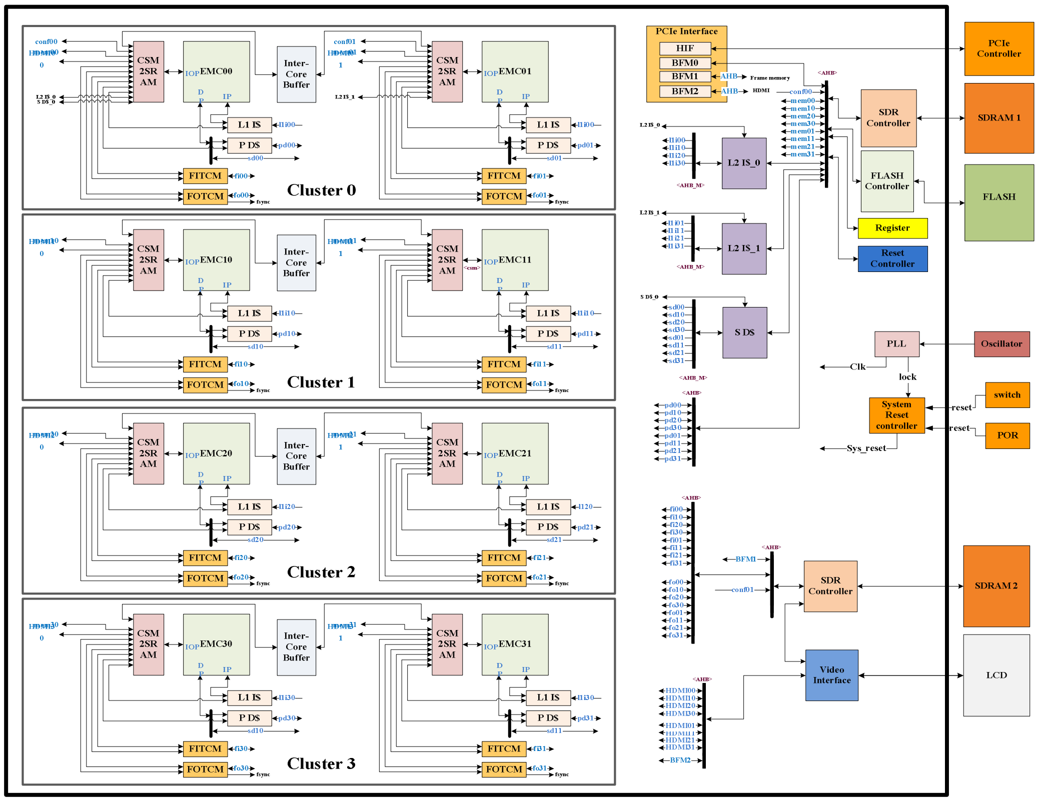


Figure 7. Architecture of multi-core SoC

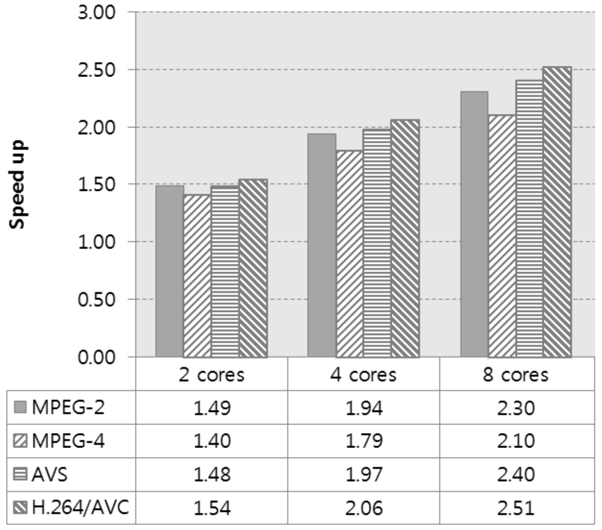
TABLE IV. CHARACTERS OF MULTI-CORE SOC

Features	Features
Desig Rule	SMIC 130nm , 1P8M CMOS, Core 1.2V/Pad 3.3 V
Frequency	Core/SDRAM : 125MHz, PCIe : 62.5 MHz, Video Interface : 27 MHz
Internal PLL	8 ~ 175MHz/Programmable
Gate Count	1.3 M
Internal SRAM	789.7 KB
Power Consumption	225mA, 1.2V@125MHz
ChipSize/Package	8.12x8.12 mm2/308 FBGA

TABLE V. ENCODING CONDITIONS

Features	MPEG-2	MPEG-4	AVS	H.264/AVC
Profile@Level	Main @High	Advanced simple@L5	Jizhun@6.0	High@4.2
Coding Structure	IPPP	IPPP	IPPP	IPPP
Number of Frames	30	30	30	30
Encoder	TM 5 [21]	XviD [22]	RM5.2j [23]	JM 16.2 [24]
QP	rate control	rate control	rate control	27(I), 28(P)
Bitrate (foreman)	48 KB/s	44 KB/s	56 KB/s	48 KB/s
Bitrate (mobile)	238 KB/s	224 KB/s	252 KB/s	240 KB/s
Bitrate (paris)	80 KB/s	80 KB/s	88 KB/s	84 KB/s
Bitrate (tempete)	173 KB/s	172 KB/s	172 KB/s	176 KB/s

Figure 8 shows the decoding time speed-up according to the number of processing cores.



	2 cores	4 cores	8 cores
■ MPEG-2	1.49	1.94	2.30
▨ MPEG-4	1.40	1.79	2.10
▤ AVS	1.48	1.97	2.40
▧ H.264/AVC	1.54	2.06	2.51

Figure 8. Speed-up according to the number of processing cores

In the case of MPEG-2 decoder, speed-up is 1.49x on 2 cores, 1.94x on 4 cores and 2.30x on 8 cores. In the case of MPEG-4, a speed-up is 1.40x on 2 cores, 1.79x on 4 cores and 2.10x on 8 cores. The AVS and H.264/AVC decoders also yielded a similar speed-up in accordance with the number of cores. Even with a multi-core implementation for video decoders, it is not easy to achieve more than 3x speed-up due to the sequential entropy decoding part and MB-to-MB dependency.

IV. CONCLUSION AND FUTURE WORK

This paper proposes a new application specific processor based on 6-stage pipelined dual issue VLIW+SIMD architecture and compiler for multi-standard video decoding. The proposed processor whose approximate gate count is about 130K runs at 125MHz in SMIC 130nm technology. The proposed processor results in about 20% speed-up in terms of processing cycles, compared to conventional ARM1020E processor without quality degeneration for the decoding of the H.264 CIF test sequences. For Full HD multi-standard video decoding, multi-core platform consisting of 64 ASIPs is under development.

V. ACKNOWLEDGEMENT

This material is supported by Ministry of Knowledge and Economy (MKE) and Korea Evaluation Institute of Industrial Technology (KEIT), Republic of Korea under Contract No. 10035152, Energy Scalable Vector Processor - Basic Technology.

REFERENCES

[1] Keith Barr, "ASIC Design in the Silicon Sandbox: A Complete Guide to Building Mixed-Signal Integrated Circuits," McGraw Hill, Dec. 2006.

[2] Arifur Rahman and Jason H. Anderson, "FPGA Based Design and Applications (Integrated Circuits and Systems)," Springer, Nov. 2012.

[3] A. Hoffmann, T. Kogel, A. Nohl, G. Braun, O. Schliebusch, O. Wahlen, A. Wieferink, and H. Meyr, "A Novel Methodology for the Design of Application-Specific Instruction-Set Processors (ASIPs) Using a Machine Description Language," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 20, pp. 1338-1354, Nov. 2001.

[4] Richard G. Lyons, "Understanding Digital Signal Processing (3rd Edition)," Prentice Hall, Nov. 2010.

[5] John L. Hennessy and David A. Patterson "Computer Architecture, Fifth Edition: A Quantitative Approach," Morgan Kaufmann, Sep. 2011.

[6] AVS-Group, "Information Technology - Advanced Coding of Audio and Video - Part 2: Video," advanced Audio and Video Standard (AVS1-P2), 2005.

[7] "VC-1 Compressed Video Bitstream Format and Decoding Process (SMPTE 421M-2006)", SMPTE Standard, 2006.

[8] ISO/IEC 14496-10 International Standard (ITU-T Rec. H.264)

[9] ISO/IEC 11172: "Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s," Geneva, 1993.

[10] ISO/IEC 13818-2: "Generic coding of moving pictures and associated audio information-Part 2: Video," 1994, also ITU-T Recommendation H.262.

[11] ISO/IEC 14496-2: "Information technology-coding of audiovisual objects-part 2: visual," Geneva, 2000.

[12] F. Pescador, C. Sanz, M.J. Garrido, E. Juarez and D. Sampler, "A DSP Based H.264 Decoder for a Multi-Format IP Set-Top Box," IEEE Trans. Consumer Electronics, vol. 54, pp. 145-153, Feb. 2008.

[13] Y. Chen, E. Li, X. Zhou and S. Ge, "Implementation of H.264 encoder and decoder on personal computers." Journal of Visual Communication and Image Representation, vol. 17, pp. 509-532, April 2006.

[14] Y.-L. Lee and T.Q. Nguyen, "Analysis and Efficient Architecture Design for VC-1 Overlap Smoothing and In-Loop Deblocking Filter", IEEE Trans. Circuits Syst. Video Technol. vol.18 , pp. 1786-1796, Dec. 2008.

[15] S. Pees, A. Hoffmann, V. Zivojnovic and H. Meyr, "LISA-Machine description language for cycle-accurate models of programmable DSP architectures," Design Automation Conf., pp. 933–938, June 1999.

[16] C-compiler Design Guide, CoWare, 2011

[17] Processor Designer Training Manual, CoWare, 2011.

[18] LISA Language Reference Manual, CoWare, 2011.

[19] Compiler Designer Reference Mannual, CoWare, 2011.

[20] J.Y. Lee, J.J. Lee and S.M. Park, "Multi-core platform for an efficient H.264 and VC-1 video decoding based on macroblock row-level parallelism," IET Circuits, Devices & Systems, vol. 4, pp. 147-158, Mar. 2010.

[21] http://www.mpeg.org/MPEG/video/mssg-free-mpeg-software.html

[22] MPEG-4 XviD, http://www.xvid.org/Xvid-Codec.2.0.html

[23] AVS RM5.2j, http://www.avs.org.cn/fruits/en/softList.asp

[24] Joint Video Team (JVT) reference software JM16.2, http://iphome.hhi.de/suehring/tml/

[25] ARM, http://www.arm.com