A Single-Issue DSP-based Multi-standard Media Processor for Mobile Platforms

Di Wu, *Tiejun Hu and Dake Liu
Department of Electrical Engineering
Linköping University, SE-581 83, Linköping, Sweden
diwu@isy.liu.se, hutiejun@huawei.com, dake@isy.liu.se

Abstract: This paper presents the study of modifying a legacy single-issue DSP processor to provide real-time processing capacity for emerging multimedia applications. The latest video compression standards such as H.264 and SMTPE VC-1 require both high computing performance and flexibility which can not be met by legacy single-issue DSPs. Feasibility of achieving both real-time performance and flexibility for multi-standard video decoding with HW acceleration is studied and proven.

1 Introduction

Single-issue DSPs are DSPs with one-way data path which have been widely used in mobile handsets for voice coding and other applications. Because of the emerging era of mobile multimedia, more and more mobile handsets need to support performance demanding multimedia applications. Legacy single-issue DSPs aimed at voice coding lack computing power for these applications. For small DSP vendors and handset developers using legacy single DSPs, one feasible way is to attach HW accelerators for performance enhancement with acceptable cost. This solution can not only shorten the time-to-market but also lower the design cost.

H.264 (MPEG-4 part 10) [1] and VC-1(WMV-9) [4] are the latest mainstream video compression standards which provide higher compression ratio while also demanding higher computational capacity. Although an ASIC design can meet the performance requirements, it lacks the flexibility for heterogeneous video coding standards.

The remainder of the paper is organized as follows. In Sec.2, the algorithms in H.264 and VC-1 are analyzed and compared. In Sec.3, the profiling and complexity analysis of H.264 and VC-1 are presented. Sec.4 gives the details of the hardware implementation. Scheduling issues are covered in Sec.5 and Sec.6 presents the experimental results. Finally, Sec.7 concludes the paper.

2 Algorithm Features and Comparison

2.1 Entropy Decoding

Advanced entropy coding such as VLC (variable length coding) can achieve a high degree of efficiency in video coding. Although only simple VLC is used in VC-1, the use of multiple small

*Tiejun Hu is now with Huawei Technologies.

code tables can also achieve high coding efficiency. In comparison, context adaptive variable length coding has been adopted by H.264 (baseline profile) which is more complicated.

2.2 Transform

As depicted in Fig.1, in H.264 and VC-1, 4×4 block based Integer Transform (IT) which only involves shift and addition without multiplication has been used instead of the traditional Discrete Cosine Transform (DCT) used in JPEG and MPEG-4. Furthermore, a variable block size transform is introduced in H.264 and VC-1 because it has been disclosed that transform with larger block size can also achieve better performance for cases such as a periodical texture. According to this, a traditional 8×8 block can be subdivided into 8×4, 4×8 and 4×4 blocks to suit the underlying data. Details of the transform can be found in [4] and [7].

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

(a) 4x4 Inverse transform of H.264

$$\begin{bmatrix} 17 & 17 & 17 & 17 \\ 22 & 10 & -10 & -22 \\ 17 & -17 & -17 & 17 \\ 10 & -22 & 22 & -10 \end{bmatrix}$$

(b) 4x4 Inverse transform of VC-1

Figure 1: Integer Transform

H.264 interpolation
Linear:
p=(a + b + 1) >> 1
6-tap FIR:
p=a-5b+20c+20d-5e+f

VC-1 interpolation:
Linear:
p=(a + b + 1 - R) >> 1
p=(9a + 3b + c + 3d + 8 - R) >> 4
4-tap FIR:
p=(-a + 9b + 9c – d + 8 – r) >> 4
p=(-4a+53b+18c–3d + 32 - r) >> 6
(R,r are rounding control values)

Figure 2: Interpolation

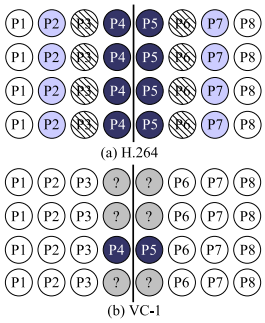


Figure 3: Loop Filter

2.3 Interpolation

An interpolation operation is used to generate intermediate sub-pixel values from full-pixel values for high accuracy motion compensation. Both in H.264 and VC-1, the maximum accuracy is 1/4 pixel. In H.264, reference blocks as small as 4×4 are supported while the smallest blocks supported in VC-1 are 8×8 because it makes a compromise between motion compensation freedom and complexity.

As shown in Fig.2, bi-linear and cubic FIR interpolation operations with integer coefficients (3, 4, 9, 5, 20, 18, 53) have been employed by H.264 and VC-1. The limited number of integer coefficients makes it feasible to use shifts and adders instead of multipliers in order to generate the results for both standards.

2.4 Reducing Artifacts

Quantization in intra-coded blocks and residues in inter-coded blocks generate discontinuities along block borders. Both H.264 and VC-1 employ a loopfilter along 4×4 block borders to eliminate these discontinuities. The difference is the length of the filter (the number of pixels updated after

the filtering operation) and the filtering frequency (e.g. rows filtered along the vertical border). The loopfilter is mainly a multi-tap filter applied to pixels on the edge of 4×4 blocks in the decoded frames. In H.264, up to eight pixel values might be required in order to perform a filtering operation and up to six pixel values might be updated after this operation. In comparison, only two pixel values will be updated in VC-1.

As is shown in Fig.3 (a), according to H.264, the multi-tap filter is applied to 4 rows (columns) along 4×4 block borders. Pixels pairs including (P4,P5),(P3,P6) and (P2,P7) are updated depending on the filtering strength. As for VC-1 which is depicted in Fig.3(b), the pixel pair (P4,P5) in the third row is filtered first, and its result determines whether the other three pixel-pairs (those marked in grey) should be filtered. Thus at most two pixels in each row (column) will be updated which is less computationally intensive than that of H.264.

Other features such as the overlapped transform (VC-1) which is a simple 4×4 transform applied to 8×8 block borders in order to reduce artifacts caused by quantization errors in intra-coding, can be found in [1] and [4].

3 Profiling and Complexity Analysis

In this paper, the profiling was targeted on H.264 baseline decoding (Similar work can be found in [3]). The computational complexity (MIPS cost) of different subroutines was studied based on a software decoder developed on the ISA of a single scalar DSP. Since the MIPS cost ratio varies largely according to different video sequences and the type of frames (I, P frame), the profiling result depicted in Fig.4 is just a rough estimation based on several video sequences. However, it clearly shows that filtering operations such as interpolation and deblocking are the most computational intensive subroutines both in H.264 and VC-1 decoding. Since VLC (variable length coding) is bit level manipulation, general DSP instructions can not efficiently handle it. Thus the MIPS cost of VLC is also high. In order to achieve real-time decoding of a H.264 video sequence in QCIF (176x144) resolution at 30fps (frames per second), more than 600MHz is required. Note that the firmware can be further optimized with methods including interleaved memory storage (Sec.4) and faster VLC table searching which can improve the performance by up to 100%.

The profiling of H.264 decoding covers VC-1. Actually from the comparison in Sec. 2, it is obvious that H.264 inner-loop operations have higher complexity than those of VC-1.

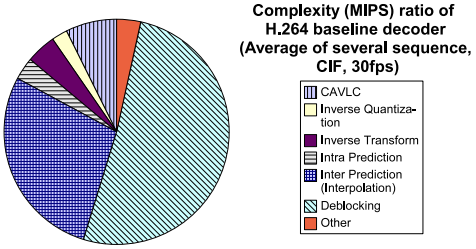


Figure 4: Complexity Analysis

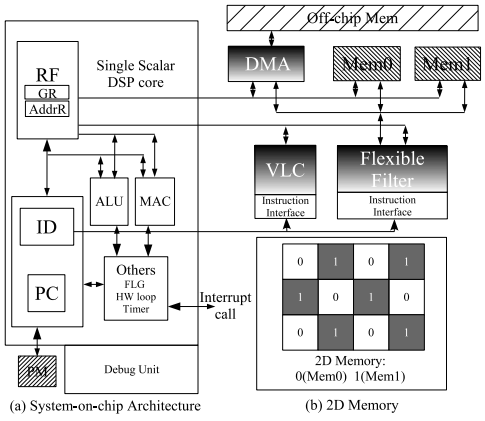


Figure 5: Processor Model

4 HW Implementation

4.1 Processor Model and Interface

As is shown in Fig.5(a), the single-issue DSP core consists of a 16×16 MAC, an ALU and a $16 \times 16b$ register file. Scalable ports are available for attaching external hardware accelerators. An accelerator is an external computational unit that can be connected to the DSP core via interfaces. All accelerators receive instructions from the DSP core through the 24bit wide instruction bus.

There are two on-chip memory banks that be accessed simultaneously by the DSP core and accelerators. Data can be transferred between the DSP core and the accelerator via general registers. Besides this, for a large amount of data such as pixel values, the on-chip memory can be used as the data buffer between the DSP core and the accelerator. The accelerators can directly access on-chip memory banks in parallel.

4.2 Design Consideration and 2D Memory

Since the pure software based solution can not provide real-time performance, hardware acceleration is required. Based on the analysis presented in Sec.3, instruction set based acceleration is proposed. First, new instructions are designed and the behavior models of the accelerators are implemented in C and embedded into the scalable cycle-accurate instruction set simulator (ISS) via the API.

For video coding which mainly consists of pixel based operations, the number of pixels that can be accessed in one cycle determines the computing parallelism that can be achieved, thus in order to exploit the parallelism exists in pixel level manipulation, a method called 2D memory has been proposed for parallel memory access which improves the performance in pixel level operations by adding a small AGU (address generator unit). Pixel data in the frame are stored in both Mem0 and Mem1 in an interleaved way, which means adjacent pixel data are stored in different memories as shown in Fig.5(b). Two pixel data can be simultaneously accessed (read or write) either vertically or horizontally. Thus the existing memory of the legacy DSP can be fully utilized.

Because the silicon cost of the DSP core and the logic part of the accelerator is much smaller than that of the memory banks, memory cost is still a major issue in our design. If the number of memory banks can be increased to four, which means four pixel data can be accessed simultaneously in one cycle, the bottleneck between the memory and the Flexible Filter will be eliminated (in case four-way parallel data path is used). However, this requires further redesign of the single-issue DSP. Thus in order to make a compromise between design cost and performance improvement, we used the legacy memory architecture without extending it.

4.3 VLC Co-processor as an Accelerator

Since the single-issue DSP is inefficient in entropy decoding which is mainly bit manipulation, instructions for bitstream parsing will improve the entropy decoding performance by 3–7 times [6]. By extending the bit manipulation instruction acceleration proposed by [6], an accelerator (VLC Co-processor) was designed to accelerate the entropy decoding. As is shown in Table 1, the instruction extension provided by VLC Co-processor covers basic and inner-loop operations in the entropy

coding, which is flexible enough for multiple standards (e.g. the instruction showzeros returns the number of leading zeros before the first '1' in the bitstream within one cycle, which is very suitable for searching binary codes of various length with large number of leading zeros).

Instructions	Functionality
bitreadreset	Initialize the VLC accelerator
showbits Rs/imm, Rt	Load bits (number stored in Rs) from bitstream to general register (Rt), without changing the position of current bit.
getbits Rs/imm, Rt	Load bits (number stored in Rs) from bitstream to general register (Rt), and change the current bit position.
flushbits Rs	Change the current bit position
showzeros Rs, imm	Count the number of leading zeros before first 1 in the bitstream, no larger than the immediate value (imm)
packbits Rs	Pack code variable length code word into 16bit registers.
Note: imm–immediate value, Rs–source register, Rt–destination register	

Table 1: VLC Instruction Set

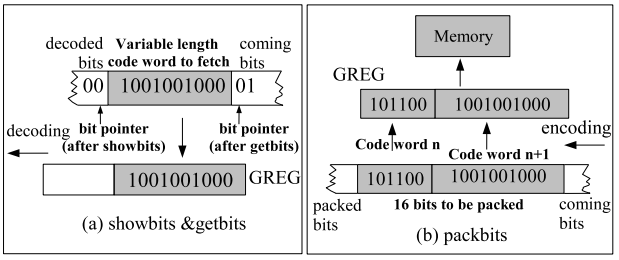


Figure 6: VLC Stream Flow

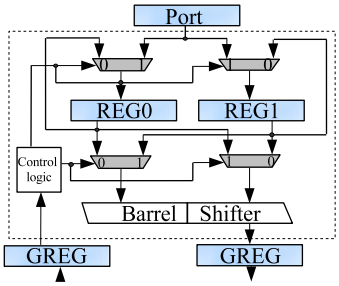


Figure 7: VLC Processor

Fig.6 gives a description of VLC stream flow. The block diagram of VLC Processor is shown in Fig.7. As the VLC sample code depicted in Fig.8 shows, the pointer of current bit in the stream is not updated until the matched codeword is found in the table, which has high efficiency for matching codeword with variable length.

```
lp_demo:
add gr0,1
;load bits (num=gr0)
;from stream to gr1
showbits gr0,gr1
;look for matched entry,
; if matches set flag=1
lkup tab(gr0),gr1
if aeq jump lp_demo
;update bit pointer to stream
flushbits gr0
```

Figure 8: VLC Sample Code

Subroutine	Filtering Features			
	H.264		VC-1	
	Num of taps	Tap Coeffs	Num of taps	Tap Coeffs
Integer Trans(4x4)	4	1,2	4	10,17,22
Integer Trans(8x8)	4	N/A	4	12,16,15,9,4,6
Interpolation	2,6	1,5,20	2,4	1,2,3,6,9,18,53
Loop Filter	4,5	1,2,3	4	2,5
Overlapped Trans	N/A		4	1,7

Table 2: FIR Filtering in Subroutines

4.4 Flexible Filter as an Accelerator

An accelerator called the Flexible Filter has been designed for multi-tap FIR filtering operations. Since according to the media processing, many subroutines comprise of FIR filtering operations, high parallelism exists and can be utilized for acceleration. The interpolation in H.264 and VC-1 consists of half-sample and quarter-sample operations, which are mainly multi-tap filtering operations (up to 6 taps). Also integer transforms such as 4×4 integer transforms and 8×8 transforms comprise of 4-tap filtering operations. Subroutines that consist of filtering operations with different number of taps are shown in table 2. Both 6-tap and 4-tap filtering operation instructions have been adopted by the instruction extension.

The data path of the Flexible Filter depicted in Fig.9 has four pipeline stages consisting of configurable filter taps and an arithmetic logic unit (ALU). During implementation of the configurable filter taps, different solutions have been compared. In the latest video coding standards, in order to reduce the computational complexity, multiplications with small constant coefficients have been widely adopted. Thus multiplications in most subroutines except the inverse quantization can be realized as a network of shifts, adders and subtractors which is called configurable tap unit (CTU) here. Instead of using four 16bit multipliers, we used four CTUs as filter taps. Compared with a 16bit multiplier, the gate count of a configurable tap unit is only half which achieves both flexibility and silicon efficiency. As depicted in Fig.9, there are six taps, the paths connected to IR0 and IR5 can be seen as taps with coefficient '1'. Paths connected to IR1–IR4 are CTUs that can be configured by instructions. Each of these four taps consists of components such as four shifts, three adders (or subtractors). Within each CTU, the input value is left shifted with different shifting depth and then added to realize the multiplication by constant coefficients.

Two working modes called Customized mode and SIMD mode are supported in the Flexible Filter. In the Customized mode, customized instructions are issued for irregular operations which are optimized for specific applications such as H.264. And the SIMD mode issues general SIMD instructions to the four-way data path. It has higher flexibility for various standards with lower performance. The instruction extension for the Flexible Filter is shown in Table 3.

4.4.1 Local Storage

Since only two data in the memory can be accessed simultaneously, the performance bottleneck exists between the Flexible Filter and the memory. For high performance, an efficient local storage is necessary to avoid the bottleneck of data exchange between the Flexible Filter and the memory. In our design, a register file with 4×9 16bit registers (LR) is adopted. Besides the LR, there are six 16bit input registers (IR). With the local register file, for subroutines such as bi-directional interpolation, intermediate values can be stored locally without data swapping between the memory and the Flexible Filter. The IRs are connected in a line, thus input data can propagate from the left to the right while a new value is loaded into the left-most IR thus to minimize data transfer between the accelerator and the memory.

4.4.2 Conditional Execution

Subroutines such as deblocking involve a large number of branches in inner-loop operations. General branch execution such as JUMP is not efficient enough to achieve high performance. In order to achieve data level parallelism and reduce branch penalty, conditional execution has been applied to

the Flexible Filter. Based on the execution flag, the data path decides whether it should write the result back to the register file (to update the destination registers with the new result) or just skip it. Although with conditional execution, the cycle cost of certain subroutine is fix which is the worst case, it is still more efficient compared with the traditional branch execution because both extra cycles for JUMP are saved and the data parallelism is exploited along the 4-way data path.

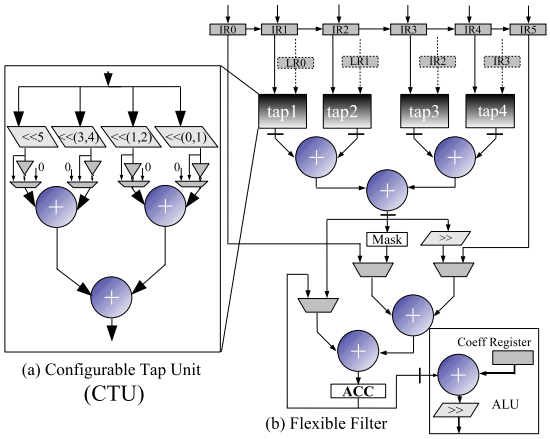


Figure 9: Flexible Filter and CTU

Instructions	Functionality
TAP6	6-tap filtering operation
TAP4	4-tap filtering operation
PADD	4 way addition per cycle
PSUB	4-way subtraction per cycle
CMPXX	CE (conditional execution)
FPOP	CE stack pop
FPUSH	CE flag stack push
FCLR	CE flag stack clear

Table 3: Instructions of the Flexible Filter

4.4.3 Algorithm Mapping

A piece of assembly code (depicted in Fig.10) for H.264 half-pel interpolation is shown in order to explain how the instruction TAP6 (mode0) works. As it can be seen from the data flow depicted in Fig.11, six full-pel values (a0...a5) are loaded to the IRs in the first three cycles (two values in each cycle from the 2D memory) and then issued along the data path of the Flexible Filter, receiving 6-tap filtering, scaling and rounding operations in three pipeline stages. The generated half-pel value (b0) is stored in the LRs as the output. The taps (CTU) have been configured to generate multiplication coefficients (multipliers such as -5 and 20) for filtering operation. For example, as it has been depicted in Fig.11, when counting from the left, the first and last taps have coefficient '1'. The second tap has the coefficient '5' which is generated by configuring the shift s2 to left shift 2 bits and s3 to be bypassed ($5 \times a = a << 2 + a$) while the left-half part of the tap (including s0 and s1) is disabled.

The ALU is used to perform scaling and rounding operations. Since the data in the IRs can propagate to the right, in the coming three cycles, three new pixel values (a6...a8) can be shifted into the IRs to generate three new half-pel values (b1...b3). The advantage of data propagation in the IRs is that five out of six old full-pel values can be reused in the calculation of next half-pel value without being reloaded. Thus only one new full-pel value needs to be loaded every cycle. For example, in order to generate b1, only a6 needs to be loaded, while a5...a1 just need to be shifted to the neighboring IR on the right.

```
;Mutiple modes are defined for tap6 instruction.
;For exampe, mode(0) is one directionaly half-pel
;interpolation in H.264.
;Firmware code of interpolation
...
;load coordinates (x,y) of up-left corner
ldpos gr4,gr5
;load 6 pixel values in a row to IR
;first operand (0:horizontally; 1:vertically)
ld2ir 0,4
ld2ir 0,2
ld2ir 0,0
;lr[0][0] =((a-5b+20c+20d-5e+f)+16)>>5
;ir[i]=ir[i-1], ir[0]=new pixel value
;instruction of 3 pipeline stages
tap6 lr[0][0], mode(0)
tap6 lr[0][1], mode(0)
tap6 lr[0][2], mode(0)
tap6 lr[0][3], mode(0).
```

Figure 10: Sample Code

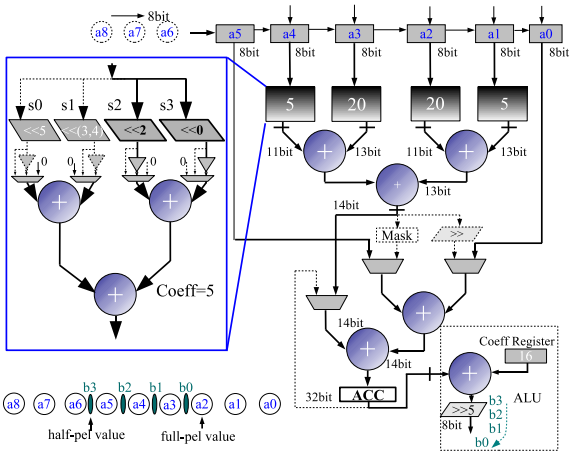
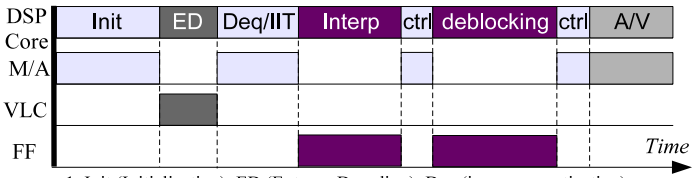


Figure 11: Data Flow of TAP6 (mode 0)

5 Scheduling

In our solution, the DSP core works as the task manager. It issues extended instructions to the accelerator in the same way as it issues core instructions. Thus the data dependency problem is avoided. Since tasks are executed sequentially, the implementation is much easier compared with multi-core solutions. Scheduling of the accelerated solution is depicted in Fig.12.



1. Init (Initialization); ED (Entropy Decoding); Deq (inverse quantization); IIT (inverse integer transform); Interp (Interpolation); A/V (Audio/Voice)
2. M/A (MAC/ALU of the DSP core); VLC (VLC Co-processor); FF(Flexible Filter)

Figure 12: Scheduling of the DSP Core and Accelerators

6 Experimental Results

Based on the cycle-accurate simulator, the performance improvement achieved by the hardware acceleration was evaluated. Since the complexity of H.264 baseline profile (not to mention main profile) is higher than VC-1 (1.5 times or so), the benchmarking was only performed based on H.264 decoder. However, all intensive inner-loop operations in VC-1 decoding have been covered by the instruction acceleration (actually they are similar while simpler compared with those of H.264) which means VC-1 real-time decoding with even higher resolution can be supported as well.

Five video sequences with different scenarios have been coded by H.264 baseline profile encoder in CIF (352x288) resolution at the frame rate of 30fps. They are used as test sequences for the benchmarking. As shown in Fig.13, sequences with difference scenarios exert different computational requirements on the decoder. For example, the sequence Akiyo has the lowest computational complexity because its scenario is a news reporter sitting in front of the camera without dramatic movement which means the number of residues coded in entropy coding is small and in motion compensation, a lot of blocks can be simply copied without interpolation. Thus it has the lowest MIPS costs both in VLC decoding and vector processing among five test sequences. While the sequence Coastguard has the highest computational complexity because its scenario is a boat sailing along the river with relatively high motion. For all these test sequences, real-time H.264 decoding performance of CIF resolution (352×288, 30fps) together with audio decoding (roughly 30MIPS) can be achieved at 150 MHz working clock frequency with hardware acceleration proposed by us. Even compared with the optimized software implementation mentioned in Sec.3, the performance with hardware acceleration is more than eight times higher.

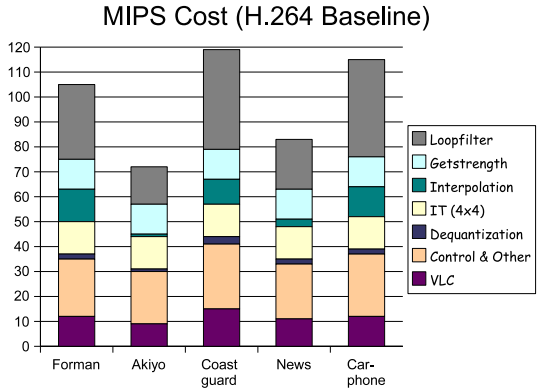


Figure 13: Total MIPS Cost of H.264 Baseline Decoding

Applications	Standards
Voice	AMR, G.723, G.729
Audio	MPEG Layer-3 (MP3), MPEG AAC (Advanced Audio Coding), OGG
Image	JPEG, JPEG2000
Video	Motion JPEG, MPEG4, H.264, VC-1

Table 4: Application Coverage

7 Conclusion

Because of the parallel data path (though without multipliers), the Flexible Filter very well exploits the parallelism in filtering based operations. Its estimated gate count is no more than 30K gates. When synthesized with UMC $0.18\mu m$ process the silicon cost of the accelerators is less than $0.5mm^2$. It provides enough flexibility for the latest standards such as H.264 and VC-1 with small silicon area (compared to 4-way SIMD with 16×16 multipliers which has been elaborated in Sec 4.4). At the same time, since general multiplication is avoided, lower power consumption is achieved which is more important for mobile applications with limited battery capacity.

In this paper, a programmable solution has been proposed for media processing. The programmable single-issue DSP processor works as the task manager and performs less intensive tasks. The low-cost and configurable accelerators perform computational intensive tasks with high parallelism. Real-time decoding performance can be achieved at resolution which is high enough for all mobile terminals. The memory of the legacy DSP is fully utilized with parallelism. It has been proven that the solution of a single-issue DSP processor plus low-cost accelerators in media processing is

not only practical but also advantageous in time-to-market and silicon cost. HW reusability has also been explored and it has been proven that configurable accelerators can accommodate different computing tasks with similar computing and data access patterns.

As it is depicted in Table 4, since the single-issue DSP was originally designed for voice coding, most of the voice coding standards are already supported. With the VLC accelerator, it well supports the current audio coding standards such as MP3 and OGG. Besides the latest video coding standards such as H.264 and VC-1 which have the highest computational complexity, other legacy image and video coding standards such as MPEG4 and JPEG are supported as well. To be noticed is that though the JPEG2000 coding is supported, it is not in real-time, because it is generally used for compressing still images captured by the camera which can take several seconds. Thus with the low-cost hardware acceleration, the legacy single-issue DSP platform covers most of the latest multimeida applications for mobile terminals.

8 Future Work

Since the Flexible Filter also supports 4-way SAD (sum of absolute difference) operations, a video (e.g. H.264) encoder can be implemented together with the decoder which enables video conference at lower resolution (e.g. QCIF). This part will be included in our future work.

References

[1] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, *JVT-G050,Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC)*, March 2003.

[2] Ostermann, J.; Bormans, J.; List, P.; Marpe, D.; Narroschke, M.; Pereira, F.; Stockhammer, T.; Wedi, T., *Video coding with H.264/AVC: tools, performance, and complexity*, Circuits and Systems Magazine, IEEE, Volume: 4, Issue: 1, First Quarter 2004

[3] Horowitz, M.; Joch, A.; Kossentini, F.; Hallapuro, A., *H.264/AVC baseline profile decoder complexity analysis*, Circuits and Systems for Video Technology, IEEE Transactions on, Volume: 13, Issue: 7, July 2003

[4] Sridhar Srinivasan, Pohsiang (John) Hsu, TomHolcom b, Kunal Mukerjee, *Windows Media Video 9: Overview and Applications*, Signal Processing: Image Communication, ELSEVIER, 2004

[5] Karsten Sühring. *H.264/AVC Software Coordination* (http://iphome.hhi.de/suehring/tml/)

[6] Berekovic, M.; Stolberg, H., Kulaczewski, M., Pirsch, P.; Möller, H; Runge, H.; Kneip, J.; Stabernack, B., *Instruction Set Extension for MPEG-4 Video, Journal of VLSI Signal Processing, Volume 23,issu 1, 27-49(1999)*

[7] Malvar, H.S.; Hallapuro, A.; Karczewicz, M.; Kerofsky, L., *Low-complexity transform and quantization in H.264/AVC*, Circuits and Systems for Video Technology, IEEE Transactions on, Volume: 13, Issue: 7, July 2003