Platform independent MPEG-4 Co-processor

Bing-Fei Wu
National Chiao Tung University
bwu@cc.nctu.edu.tw

Hsin-Yuan Peng
National Chiao Tung University
sypeng.ece91g@nctu.edu.tw

Abstract

With the widespread popularity of portable devices, such as cellular phones and personal digital assistants (PDAs), in these days, people rely on the mobile technology more and more. To satisfy the low power, low cost, and high performance requirements for consumer electronics, the video encoders implemented by the VLSI architectures are much more suitable than the firmware or the software solutions. However, to integrate these ICs into a system is not easy, and these frameworks usually support dedicated frame size, frame rate and output bitrate, which will limit the utilities of the products. In order to overcome these drawbacks, a register-based platform independent MPEG-4 co-processor (RPIMC) is proposed in this paper, and it can transfer and receive the image data in all kinds of bus matrices with the suitable wrappers for being easily integrated into other platforms. RPIMC, which can be programmable to manipulate up to HD resolution and 30 frames per second, with 204K gates and 6,462 bytes of RAM is implemented, and it can adjust the data types of the input and the output streams by modifying the relative registers.

Index Terms：MPEG-4, VLSI, system.

I. Introduction

Recently, the algorithms and the architectures for processing video and audio signals are improved significantly. They are employed in various applications, such as digital TV, video conferencing and mobile multimedia systems. The markets for mobile electronics equipments, like portable PCs, cellular phones, and personal digital assistants (PDAs), are currently growing rapidly. Moreover, the higher bandwidth for wireless telecommunications now is provided for transferring moving pictures in addition to speech and data. Therefore, multimedia processing will be an essential function in such mobile-equipment applications.

The improved coding efficiency and the advanced features of MPEG-4 come with much higher computational complexity compared with previous standards. Several MPEG-4 video encoders have been reported. To satisfy rich functionalities of the future multimedia, some are implemented in firmware based on the low power DSP platform [1]. They have the highest flexibility but the cost of the hardware is too expensive. Moreover, the low power DSPs are usually operate at lower frequency, so the image quality will be degraded due to the fast algorithms of motion estimation (ME) and discrete cosine transform/ inverse discrete cosine transform (DCT/ IDCT). Therefore, the dedicated hardware methodologies are developed to achieve low power and low area cost, and it can encode the MPEG-4 video for CIF format at 15 frames per second (FPS) at 1.5V supply with 700K gates. However, lack of potential for future modification of advanced algorithms and higher design effort are the disadvantages.

Hence, in order to compromise the performance and flexibility, the hybrid software/ hardware co-design is adopted [3]-[5]. A RISC-based platform with

hardware accelerators is presented to implement MPEG-4 video encoding algorithms [3]. The optimization in both algorithm and architecture level is applied. However, the operating frequency at 40 MHz is too high for portable devices. The architectures which are developed for reducing the power consumption usually provide lower encoding complexities. Another design with the same encoding complexity as [2] based on an ARM core and AMBA is introduced, but its power consumption is not suitable for consumer electronics [4]. Except for the single purpose video encoders, an multi-functionality videophone LSI is fabricated utilizing a 0.25-um CMOS triple-well quad-metal technology [5]. Three 16-bit multimedia-extended RISC processors, dedicated MPEG-4 hardware accelerators, and a 16-Mb embedded DRAM are integrated. Although it has reasonable chip area and power consumption, it can only encode the MPEG-4 video for QCIF format at 15 FPS.

These designs can be separated into two parts. One is the architectures that only can perform video encoding [2]-[4], and the other part is the frameworks which have more functionalities like videophone [5]. Generally speaking, the single purpose encoders often provide better coding performance in image size and frame rate. Moreover, the chip area and the power consumption are also less than that of the multi-purpose one. Therefore, to integrate the MPEG-4 encoder into a system is still a critical issue. Besides, these designs are developed only for their platforms, and the encoder parameters such as image resolution, output bitrate and input frame rate are fixed. Furthermore, to integrate these frameworks into other platform is difficult because the considerations, like the timing of fetching the image data, and the output packets formats of the encoded bitstream, are usually not compatible with other platforms. When the manufacturers own their RISC and relative peripherals, such as LCD, memory card, and USB controllers, all they need is an MPEG-4 encoder which can easily integrate into their system-on-chip (SoC) design for various applications. However, the dedicated and limited functionalities of these encoders usually force the producers to establish their products by dual-chip. One is their RISC and relative peripheral controllers, and the other is an MPEG-4 encoder. The overall cost and the applications will be restricted.

In this paper, a register-based platform independent MPEG-4 co-processor (RPIMC) is proposed. The main ideas of this paper are to provide a programmable MPEG-4 encoder which can easily integrated into any platforms. To satisfy the demand for various applications, RPIMC can modify the input frame size, input frame rate, and output bitrate by adjusting the relative registers. The main controller of RPIMC will automatically calculate the internal loops of the pipelines for encoding, and will read the data in the corresponding memory with the correct image resolution based on these registers, respectively. Therefore, the manufacturers who need MPEG-4 encoders can easily integrate RPIMC into their platforms, and they can use RPIMC for various applications with proper register settings. Not only the innovative features are developed, but also the algorithms which can fit the requirements are adopted. According to the computational complexity analysis report in [6] and [7], the dominating computation-intensive tasks in MPEG-4 core profile coding are motion estimation (ME) and shape coding, which together contribute more than 90% of the overall complexity. For simple profile without shape coding tools, ME becomes the most significant one. Hence, an efficient hierarchical motion estimation algorithm (HMEA) is applied [8]. HMEA using multi-resolution frames to reduce the computational complexity and excellent estimation performance is ensured using an averaging filter to down-sample the original image. When the image resolution is larger, HMEA can reduce more sum of absolute difference (SAD) operations.

The main contribution of this paper is to design a high quality programmable video encoder that is suitable for every platform and every purpose with small gate counts to solve the changeless disadvantage of normal VLSI frameworks [2]-[5].

The rest of this paper is organized as follows. Section II shows the architecture of RPIMC. Section III presents the efficient motion unit (MU) framework. Section IV depicts the results of implementation and Section V draws conclusions.

II. The Architecture of RPIMC

Figure 1 shows the overall architecture of RPIMC, and it mainly contains four parts, controller, MU, texture coding engine (TCE), and bitstream generator (BG). The controller will calculate the required inter loops for different input frame resolutions, and it is also responsible for macro block (MB) level hardware scheduling, and coding mode decision. Other hardware accelerators improve the system performance by parallel processing according to the parallelism of algorithms. MU includes ME and motion compensation (MC), and can carries out ME with the search range from -16.0 to +15.5 pixel unit. Moreover, it interpolates pixels in reference frames into compensated MBs with the specific motion vector (MV). DCT, IDCT, quantization (Q), inverse Q (IQ), and AC/DC prediction on texture pixels in MBs are integrated in TCE. BG produces bitstrem headers, motion information, and texture information in the format of variable length codes. The hardware pipeline scheduling and the register bank will be described below.

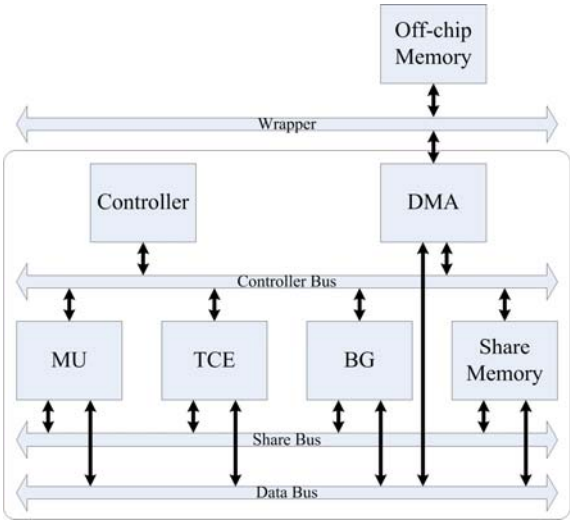

Fig. 1 The overall architecture of RPIMC

A. The hardware pipeline scheduling

After analyzing the clock cycles needed for processing one macro block, three stages pipeline scheduling, which is divided by MU, TCE and BG, is applied. As shown in Fig. 2, RPIMC will fetch the input frame MB by MB, and the pipeline processing can be separated into the intra and the inter modes since MU is not activated in the latter mode. The duration for processing one MB is called one time slot (TS), which is the period between two vertical dotted lines, and MU*i*, TCE*i*, and BG*i* denote the operation of the *i*-th MB of the image in the corresponding hardware accelerators.

In the first TS of the intra mode, only TCE is activated, and BG is in the suspend mode in order to reduce the power consumption. After TCE processing the first MB, the texture information, which are the quantization coefficients, will be input into BG, and TCE will start to manipulate the second MB in the second TS. In addition to the scheduling of the accelerators, the usage of the bus matrix is also an important issue. In order to maximize the performance, the occupation of the bus matrix of each component should be separated as possible as they can. In RPIMC, the image data is input in the format of YUV420, and one 16×16 MB denotes four 8×8 Y blocks, one 8×8 U block, and one 8×8 V block.

According to the design of TCE, these six blocks are manipulated and output in one TS with the same interval, T_{TCE}, which is shown in Fig. 2.

The first two output data of TCE will be stored into the local buffer in order to let the BG to write out the bitstream since the duration of BG occupying the bus matrix, T_{BG}, is around $2 \times T_{TCE}$. After the fourth block producing from TCE, the bus matrix will be free for a while, and the controller will start to fetch the next MB from the external memory at this moment. When all MBs of the image are computed, the operation mode will be switched from the intra mode to the inter mode.

As shown in Fig. 3, there are three pipeline stages in the inter mode. In the first TS, only MU is enabled and processes the first MB. After that, TCE will compute the motion materials while the second MB is in MU, and BG will be activated to read the texture information at the third TS. In this mode, BG will still occupy the bus matrix at the beginning of each TS, and two FIFO buffers are adopted to store the temporal reconstructed MBs from MU since the bus is busy in the first half of TS. The controller will manage the usage of the bus matrix after BG finished, and ME will have higher priority than MC. If ME wants to fetch the next MB, MC will put the reconstructed MB into FIFO. Otherwise, MC will output the data to the external memory when ME is not using the bus. In this way, the pipeline scheduling and the usage of the bus matrix will be efficient to increase the overall processing speed. TS of each MB is 1,200 cycles in average, depending on the cycle time of BG occupying the bus.

If the latency of external memory is 5 cycles at the operating frequency of 20-MHz, which is 250 ns per word, RPIMC will encode the MPEG-4 videos for CIF format at 21-MHz for real-time applications.

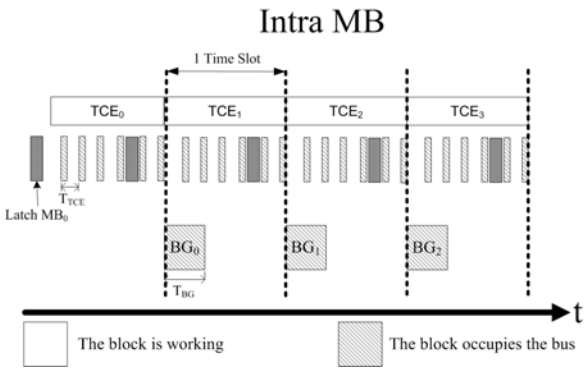

Fig. 2 The intra MB scheduling

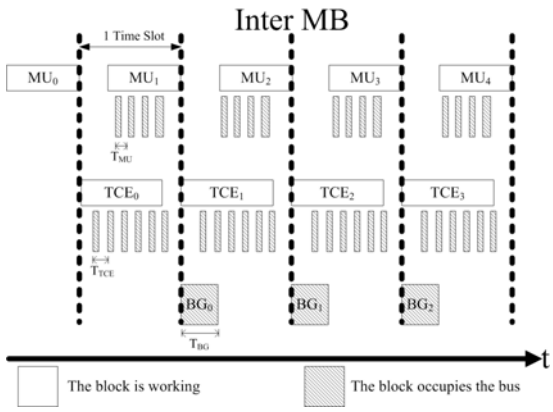

Fig. 3 The inter MB scheduling

B. The register bank

The design of the register bank makes RPIMC independent from the platforms, and it combines the control registers (CRs) and the status registers (SRs). The main feature of RPIMC is that it can program several system parameters to satisfy various applications, such as mobile video phones, digital video recorders, and high performance surveillance systems. In these utilizations, the required frame resolution, FPS, the output bitrate and the power consumption are different from each other. The manufacturers who have their own platforms can integrate RPIMC easily by setting the corresponding CRs, and RPIMC will start to encode the video with the format they want. The working flow of RPIMC is shown in Fig. 4, and CRs and SRs with their definitions are listed in Table 1.

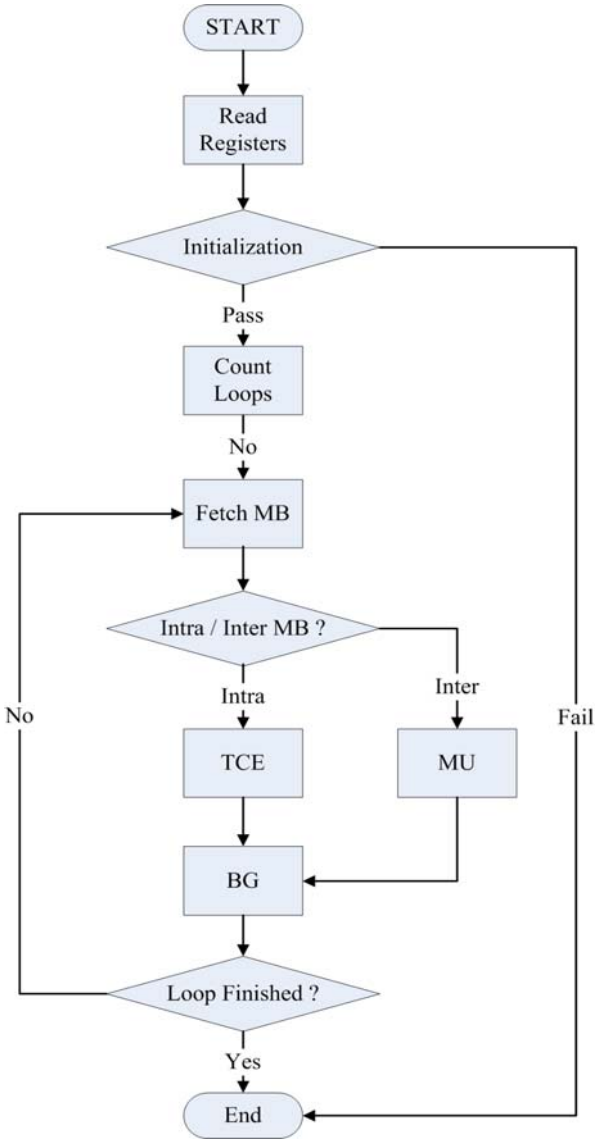

Fig. 4 The flow chart of the controller

To achieve the platform independent design, the controller has to verify the value of CRs in the initial state. At the beginning, it will fetch the values in the register bank, and check if the start memory address of the current frame, the reference frame, and the output data are overlapped by adding them with the size of each input picture which can be obtained in CRs. Since RPIMC can allow these input data be arranged in the discontinuous memory block to make the design of the platforms flexibly and efficiently, if one address of them is placed within the manipulated area of the others, RPIMC will not be ready and stays in the idle state. The decision equation is (1), where $Add_{(x)}$, img_{size}, and S represents the start place of (x), the size of the image, and the set of memory address of the current frame, the reference frame, and the output data, respectively.

$$Add_{(A)} \in \begin{Bmatrix} [Add_{(B)}, Add_{(B)} + img_{size}] \\ \cup [Add_{(C)}, Add_{(C)} + img_{size}] \end{Bmatrix},$$
$$where\ A, B, C \in S, \quad (1)$$
$$S = \begin{Bmatrix} Add_{(reference\ frame)}, \\ Add_{(current\ frame)}, \\ Add_{(output\ data)} \end{Bmatrix}, and\ A \neq B \neq C,$$

Furthermore, because the encoding cycles for one 16×16 block of RPIMC are fixed, the controller can estimate the computing ability according to the input operating frequency in CRs. If the requirement of the image resolution and the frame rate exceed the load of RPIMC, illustrated in (2), where C_{block} denotes the required cycles for encoding one block, the co-processor will not be enabled, too. On the contrary, while it has free time to wait for the image data, the controller will automatically gated the input clock and rewrite the SRs to save the power.

$$C_{block} \times \frac{img_{size}}{16 \times 16} \times fps \succ input\ \ frequency, \quad (2)$$

If the initialization procedure is passed with no errors, the controller will calculate the number of loops for encoding a frame by the desired image resolution. Then, the image data will be input one block by one block at the address assigned in the CRs, and the controller will determine whether the input block is intra or inter one. If it is an intra one, it will be sent to TCE, or else to MU. When the loops are finished, RPIMC will be back to the idle state to reduce the power consumption. In order to make the host easier to integrate with RPIMC, it will reflect the encoder conditions to the host through SRs. The mode of the encoder (intra MB or inter MB), the current working status (idle, enable, finish, or sleep), and the bitstream size of the encoded frame will be stored in SRs by the controller of RPIMC.

Table 1 The register banks

Types	Name	Description
CR	W	The width of the input image
	H	The height of the input image
	I_Size	The size of the input image
	I_FPS	The input frame rate
	Bitrate	The output bit rate
	Clock	The input operating frequency
	MEM1	The start address of MEM_1 for current frame
	MEM2	The start address of MEM_2 for reconstructed frame
	O_FPS	The output frame rate
	Out	The start address of output bitstream
SR	Status	The current state of RPIMC (Idle, Enable, Sleep or Finish)
	O_Size	The size of the output bitstream
	Mode	The current operating mode of RPIMC (Intra / Inter)

C. Memory organization

RPIMC requires the off-chip memory (OFFM) and several on-chip memory (ONM) blocks to complete the whole MPEG-4 video encoding procedure. OFFM contains source frames and reconstructed frames, and ONM is used as local buffers to reduce the bus bandwidth. In order to increase the speed of TCE for fitting the pipeline scheduling, the transformed coefficients for AC/DC prediction and the transpose memory for DCT/IDCT are integrated into ONM. Besides, for the data fetching performance and the information reuse efficiency, ONM allocates the space for storing the current MB and the search area for MU, and it also includes the input and output buffer for both TE and BG, respectively.

The input video source and the reference frames are stored in OFFM, and the direct memory access (DMA) plays an important role to control the memory interface to read data from or write them out to OFFM in a specified sequence after being initialized by the controller. In the OFFM design, two main parts, MEM1 and MEM2 are used to store two frames and they act as the ping-pong buffer to increase the encoding speed. The operations for MEM1 and MEM2 can be separated into two modes. First, in the intra frame mode, the input images are always stored in MEM1, and the reconstructed frame produced from TCE will be saved in MEM2. Second, in the inter frame mode, since the current frame will be the reference frame in the next encoding loop, these two parts will switching their status mutually until the next intra frame mode, and the scheme is illustrated in Fig. 5.

Frame Number and Frame Type	Mem_1 Status	Mem_2 Status
Intra Frame	Current Frame	Reference Frame
1st Inter Frame	Current Frame	Reference Frame
2nd Inter Frame	Reference Frame	Current Frame
3rd Inter Frame	Current Frame	Reference Frame
⋮	⋮	⋮
Intra Frame	Current Frame	Reference Frame
1st Inter Frame	Current Frame	Reference Frame
2nd Inter Frame	Reference Frame	Current Frame
3rd Inter Frame	Current Frame	Reference Frame
⋮	⋮	⋮

Fig. 5 The state of the external memory

III. The Efficient HEMA

HMEA uses multi-resolution frames to reduce the computational complexity, and excellent estimation performance is ensured using an averaging filter to down-sample the original image. At the smallest resolution, the least two motion vector candidates are selected using a full-search block matching algorithm (FSBMA). At the middle level, these two candidate motion vectors are employed as the center points for small range local searches. Then, at the original resolution, the final motion vector is obtained by performing a local search around the single candidate from the middle level. HMEA exhibits regular data flow and is suitable for hardware implementation. An efficient VLSI architecture that includes an averaging filter to down-sample the image and two 2-dimensional semi-systolic processing element arrays to determine the sum of absolute difference (SAD) in pipeline is also presented. Simulation results indicate that HMEA is more area-efficient and faster than many full-search and multi-resolution architectures while maintaining high video quality. HMEA can be divided into two parts. One is the averaging filter for down-sampling, and the other is the MV search procedure. The complete algorithm is described as below.

A. Hierarchical Frame Structure

The HMEA comprises three resolution levels, from zero to two. Level 0 is the top level, and the level 2 is the lowest. Numerous ways are available to down-sample an image. Based on the test results and the easy VLSI implementation, the averaging filter is selected. Moreover, the downsampling elements can be re-used for interpolating the half accuracy pixels during the half-pel search. For the k-th input frame, $I_k^{(2)}(\cdot)$, the upper level images are computed by executing the following down-sampling:

$$I_k^{(l-1)}(i,j)=\frac{1}{4}\cdot\sum_{m=2i}^{2i+1}\sum_{n=2j}^{2j+1}I_k^{(l)}(m,n),\ \textit{for } l=1,2\ , \quad (3)$$

where $I_k^{(l-1)}(i,j)$ represents the value at position (i,j) of the *k*-th frame at level $l-1$.

The test results presented in Table 2 indicates that the estimation performance of adopting averaging filter is significantly exceeds that of the method that considers only the left-top pixel and the two-dimensional discrete wavelet transform (2D-DWT), and can be used to design an efficient down-sampling hardware architecture. Antonini 9/7 DWT requires higher computational power, but it provides poor quality in the downsampling stage of HMEA. Moreover, if the scaling factor of Haar DWT is replaced by 1/2, the results are exactly the same as the averaging filter, and can get rid of the dynamic range problem. The reason of the averaging filter outperforms the Haar DWT is that $1/\sqrt{2}$ is chosen as its scaling factor, and this will cause the inaccuracy of the values of downsampled pixels. Considering both the coding performance and the hardware design, the averaging filter is chosen to down-sample the image in HMEA. Therefore, the averaging filter is chosen to down-sample the image.

The number of pixels at the next lower level is reduced to one quarter the number at the upper level. Figure 6 shows the hierarchical frame structure. The MB size changes from 16×16, through 8×8, to 4×4 at levels 2, 1 and 0, respectively.

In block matching algorithm, SAD is an important procedure, and its value at level l can be defined as

$$SAD_{MB}^{(l)}(p,q)=\sum_{i=0}^{[16/2^{(2-l)}]-1}\sum_{j=0}^{[16/2^{(2-l)}]-1}\left|\begin{matrix}I_k^{(l)}(i,j)\\-I_{k-1}^{(l)}(i+p,j+q)\end{matrix}\right|, \quad (4)$$

where *l* is the level number and *l*=0, 1, 2.

Table 2 The comparison of the video quality between various downsampling methods for left-top, Haar's DWT, Atonini's 9/7 DWT, and the averaging filter in dB.

Video Sequence	Left-top	Antonini 9/7 DWT	Haar DWT	Averaging filter
	PSNR	PSNR	PSNR	PSNR
News	32.79	33.84	35.42	35.45
Flower garden	21.43	24.23	26.55	26.62
Foreman	30.53	28.74	33.14	33.18
Table tennis	31.02	32.17	33.05	33.07
Stefan	23.81	22.35	25.67	25.82
Mobile	23.07	21.42	24.49	24.53

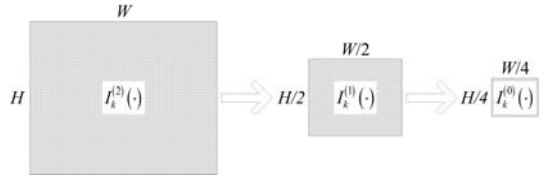

Fig. 6 The hierarchical frame structure

In the above equation, the computational complexity of the matching process can be enormously reduced. At level 1, the computational complexity is only one quarter that on level 2, and that at level 0 is one quarter that at level 1.

B. *Framework of HMEA*

The overall searching process can be separated into three levels. As presented in Fig. 6, when level 2 receives an input image $I_k^{(2)}(\cdot)$, the image will be down-sampled to $I_k^{(1)}(\cdot)$ and $I_k^{(0)}(\cdot)$, where the resolutions of $I_k^{(1)}(\cdot)$ and $I_k^{(0)}(\cdot)$ are one quarter and one sixteenth of that of $I_k^{(2)}(\cdot)$, respectively. Let the entire search range at level 2, or $\Omega^{(2)}$, be $[-w, w-1]$. After the original image $I_k^{(2)}(\cdot)$ has been down-sampled, the search procedure, illustrated in Fig. 7, begins. Let $Cur^{(l)}$, $Pre^{(l)}$, $SA_k^{(l)}$ and $MV_n^{(l)}$ denote the current MB, the previous frame search area, the k-th search area and the n-th MV candidate at level l, respectively. The MV searching process is completed when MV_{MB}, defined in (5), has been determined.

$$MV_{MB} = 4 \times MV_i^{(0)} + 2 \times MV_0^{(1)} + MV_0^{(2)}, \quad (5)$$

C. *Half-pel Search*

After MV_{MB} is manipulated, the half-pel search is started. Therefore, the neighboring half accuracy pixels of the MV_{MB} have to be calculated, and a total of 833 pixels and 8 SADs are necessary. The complexity of the half-pel search, C_{half} is defined as (6), and it is combined with the pixels and the SAD operations.

$$C_{half} = (833 \times M + 8 \times N) \times \frac{W \times H}{16^2} \times R_f, \quad (6)$$

where M, N and R_f are the number of operations required to compute a half accuracy pixel, the SAD operation, and the frame rate, respectively. Fortunately, the downsampling stage of HMEA has already calculated 144 pixels for half-pel search so the complexity of half-pel search for HMEA, C_{half_HMEA}, can be reduced as (7).

$$C_{half\_HMEA} = (689 \times M + 8 \times N) \times \frac{W \times H}{16^2} \times R_f, \quad (7)$$

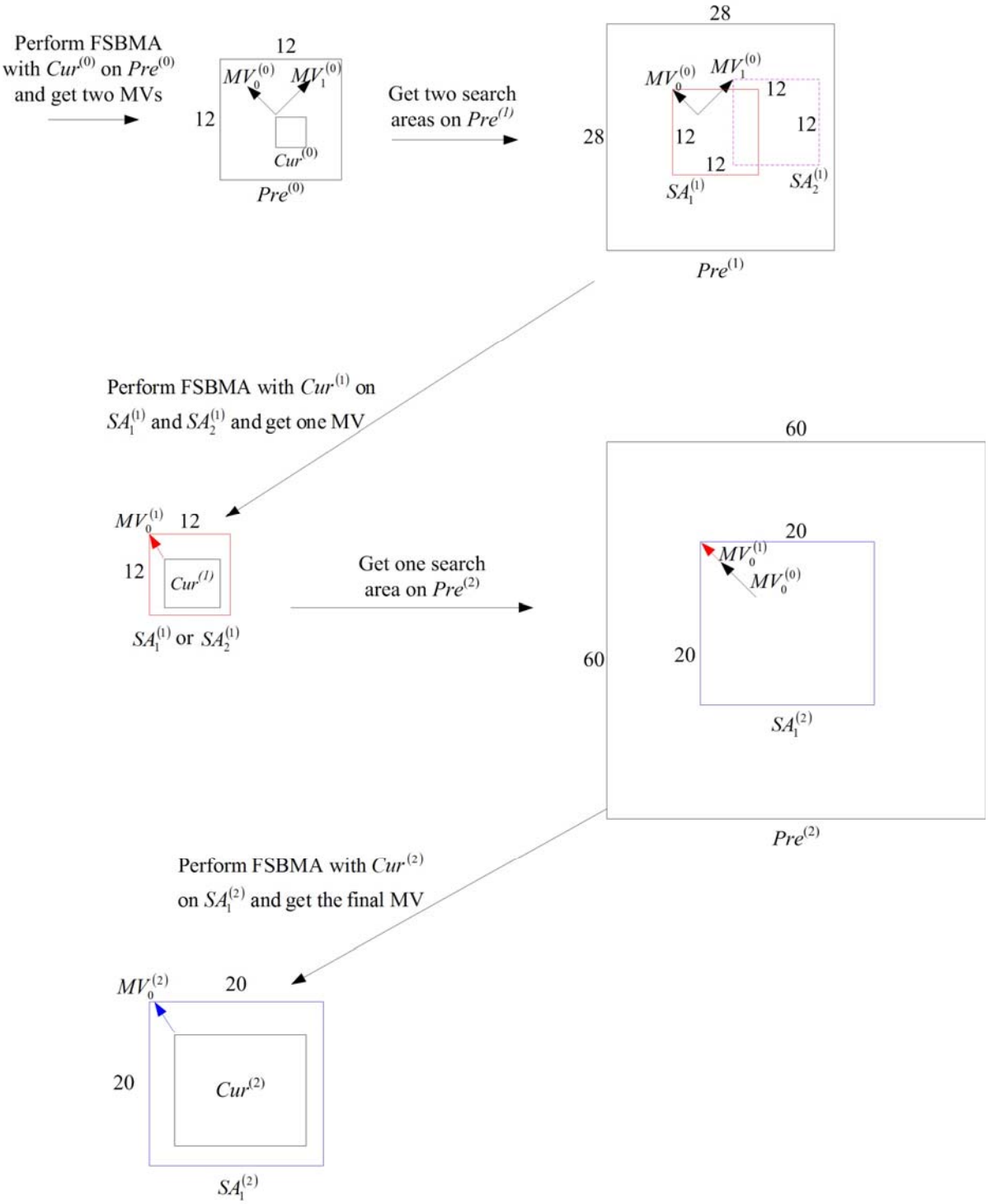

Fig. 7 The hierarchical search procedure

D. Complexity Analysis

The overall search procedure includes the downsamping stage, $C_{downsample}$, the integer-pel search, and the half-pel search, and $C_{downsample}$ is defined as (8). The half accuracy pixels only need one addition to manipulate where the pre-processed pixels for HMEA requires three of them, and the shift operation can be reduced by reading the higher bits of the pixel. Therefore, the cycles for downsampling a pixel are three times to them for the half accuracy pixels.

During the overall search procedure, the search complexity is described as (9):

$$C_{downsample} = \left[\left(\frac{W \times H}{2^2} + \frac{W \times H}{4^2} \right) \times 3 \times M \right], \quad (8)$$

$$C_{HMEA} = C_{downsample} + C^{(0)} + C^{(1)} + C^{(2)} + C_{half\_HMEA}$$
$$= \left\{ \left(\frac{w}{2} + 1 \right)^2 \times \left(\frac{1}{2^2} \right)^2 + 2 \times 5^2 \left(\frac{1}{2^1} \right)^2 + 5^2 \times \left(\frac{1}{2^0} \right)^2 \right\}, \quad (9)$$
$$\times N \times 16^2 \times \frac{W \times H}{16^2} \times R_f + C_{downsample} + C_{half\_HMEA}$$

where $C^{(l)}$ represents the search complexity in level l. In the case of FSBMA, computational complexity is given by (10).

$$C_{FSBMA} = (2w+1)^2 \times N \times 16^2 \times \frac{W \times H}{16^2} \times R_f + C_{half}, \quad (10)$$

The SAD operation for a pixel, which is described in (4), needs 256 additions, and 256 subtractions, and the manipulation for a half accuracy pixel only requires one additions. Therefore, the relationship between M, and N can be illustrated as (11).

$$M = \frac{N}{256 + 256}, \quad (11)$$

From the equations (9) to (13), they demonstrate that the computational complexity of HMEA will be only 3.9% and 1.3% of that of it of FSBMA for *w* of 16 and 32, respectively.

E. The comparison between the ME architectures

The MPEG test video sequences: "News," "Foreman," "Flower garden," "Table tennis," "Stefan," and "Mobile" are used to evaluate the performance of HMEA.

All the sequences consist of 300 frames; the frame rate is 30 FPS, and the image size is CIF. The search range is defined as $[-w, w-1]$, where *w*=16. The PSNR is used for the measurement of performance, and the PSNR is defined as (12).

$$PSNR = 10\log_{10} \frac{255^2}{\frac{1}{W \cdot H} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left[I_k(i,j) - \hat{I}_k(i,j) \right]^2}, \quad (12)$$

where $\hat{I}_k(\cdot)$ is the *k*-th motion compensated image, respectively.

The performance of HMEA is compared to that of two well-known algorithms: FSBMA and n-step search (nSS) [9], and two MMEA algorithms, MRMC-m [10] and MRMCS [11]. nSS is a general version of the 3SS to cover the increased search ranges ($n = 3$, 4, 5 for $w = 8$, 16, 32, respectively). MRMC-m is a MMEA based on multiple candidates, and it has m-candidates at each resolution. MRMCS uses three MV candidates at level 1, and two of the MV candidates that are obtained on the basis of minimum matching error at level 0, and the other one is based on the spatial MV correlation. MRMC-m and MRMCS are both using left-top method for downsampling the images, and they also keep multiple winners at the top level.

Tables 3 and 4 present the results. Table 3 describes the complexity of these four algorithms in the various search area, and Table 4 shows the performance in terms of PSNR. According to these tables, HMEA provides a prospective PSNR performance that is close to that of FSBMA, and a greater search range corresponds to a lower complexity. Although the averaging filter has higher computational complexity than the left-top method which MRMCS and MRMC-m adopted, the number of the MV candidates in level 1 of HMEA is less than the other two MMEAs. Therefore, the overall complexity of HMEA is smaller than MRMCS and MRMC-m. In Table 3, nSS exhibits the lowest computational complexity with consistency that is proper for hardware implementation. However, it can be observed that nSS provides the lower PSNR especially for the sequences that have fast motion. Besides, although MRMC-m also needs a consistent computational complexity, it contributes the worse PSNR than MRMCS and HMEA for similar

Table 3 The comparison of the complexity, including half-pel search, between FSBMA, *n*SS, MRMC-4, MRMCS and HMEA in different search ranges.

Search Range	FSBMA	*n*SS	MRMC-4	MRMCS	HMEA
8	100	10.05	18.00	17.66	13.53
16	100	3.22	4.96	4.78	3.91
32	100	1.02	1.52	1.52	1.32

Table 4 The PSNR comparisons of various fast-search algorithms in dB.

Video Sequence	HMEA	FSBMA	4SS	MRMC-4	MRMCS
News	35.45	35.85	34.83	35.01	35.15
Flowergarden	26.62	27.22	26.39	26.57	26.71
Foreman	33.18	33.70	32.15	32.97	33.14
Table tennis	33.07	34.08	32.16	32.45	33.05
Stefan	25.82	26.43	25.13	25.41	25.98
Mobile	24.53	25.18	23.96	24.11	24.65

computational complexity. Meanwhile, the PSNR of HMEA is slightly less than MRMCS in the video sequences that contain high motions since MRMCS applies an MV candidate based on spatial correlation in an MV field. However, MRMCS needs many more cycles to manipulate the MV candidate. Based on the computational complexity resulted determined by the tests, HMEA is the most suitable algorithm for VLSI implementation.

V. Implementation Results

The hardware architecture of RPIMC, as described in Section II, with the efficient HMEA [8], illustrated in Section III, is successfully implemented. The VLSI circuits were described in VHDL and synthesized by SYNOPSYS Design Analyzer using UMC 0.18um CMOS standard cell library. The chip implementation results and the performance comparison will be depicted in this Section.

A. The chip implementation results

The total gate counts of RPIMC are 204K gates, and it contains 6,462 bytes on-chip memory. Since the major features of RPIMC are that it is platform-independent and is programmable to encode videos with different resolutions and frame rate, RPIMC usually faces the situation of waiting the platforms. Therefore, RPIMC introduce two kinds of power saving techniques, sleep mode and clock gating. Sleep mode offers the greatest power savings to the user, and during this mode, RPIMC watches for a wake-up event which is asserted by the external pin. On the other hand, the clock gating can reduce a large percentage of the power since the logic activity in RPIMC is very high (~90%). Each clock of the blocks in RPIMC can be gated by the controller to manage the power down mode according to the operations. The chip layout and the specification are shown in Fig. 11, and Table 5. When RPIMC is operating at 21 MHz to encode the images in CIF at 30 FPS, it consumes 262 mW. RPIMC has been confirmed that the maximum operating frequency of 140 MHz in a typical condition by the Shmoo Utility, and it means that RPIMC can perform the video compression

up to the complexity of 1280x720 (HD) at 20 FPS which is enough for the usage of any consumer electronics and even surveillance systems.

B. The performance comparisons

The encoding performance of RPIMC is compared with the XviD MPEG-4 software encoder using various test sequences provided by MPEG, and the result is shown in Table 6. It can be observed that the quality of RPIMC is a little less than XviD because the data precision of the hardware design is not as good as the software model, especially in the TCE block. However, the quality decrease of RPIMC is not easy to be distinguished by the human eyes.

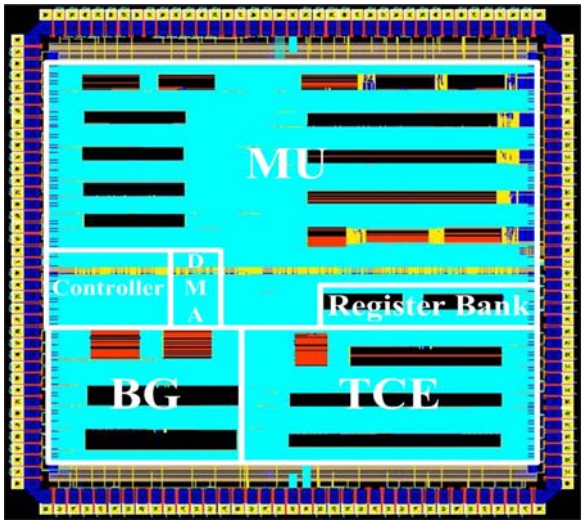

Fig. 11 The chip layout

Table 7 demonstrates the comparison between some MPEG-4 video encoders proposed before. In [2], it is full dedicated hardware video codec design, and it uses MVFAST for ME with search range equals to -16~+15.5 with extremely low operating frequency and power dissipation. However, the full dedicated design lacks of flexibility for future integration. The platform-based designs, including [12] and [3]-[5], are in hardware/software co-design fashion with performance and flexibility. In [5], it adopts 3SS for ME with the search range of -32~+31.5, and several LSI, such as the logic for H.223, the speech DSP, and an 16-Mb embedded DRAM, are also integrated. Although it contains many functions, its die size is too large and it can only encode the images of QCIF at 15 FPS. As for [4] choose a coarse ME with search range -8~+7.5, and it contains an ARM embedded microprocessor and it consumes 500 mW to compute the images of CIF at 15 FPS. In the above designs, the encoding complexities are too low, and their gate counts are too high for consumer electronics. In [3], the cost-efficient video encoder SoC consumes 256.8 mW at 40 MHz and achieves real-time encoding of CIF at 30 FPS. All of these designs before only works for specific resolution of the input images, and are not easy to integrate to other platforms. RPIMC with the programmable register bank and the efficient HMEA [8] which can manipulate various image sizes, up to HD, and it is designed to be a platform-independent co-processor. In the viewpoint of video encoder part, RPIMC has the richest functionalities, the highest encoding complexity and the lowest cost.

VI. Conclusions

This paper has addressed the implementation of RPIMC, which can encode various format of the input image by setting the corresponding register bank, and its system-level design. With the programmable controller and the efficient HMEA, RPIMC can compress the real-time video at low operating frequency, 21 MHz, for the real-time application for CIF images to reduce the power consumption, compared with other MPEG-4 chips. The proposed architecture is designed to be easily integrated into other platforms by modifying the wrapper to achieve the platform-independent purpose to wider its applications.

Acknowledgment

This work was supported by National Science Council under Grand no. NSC

Table 5 The chip specification of RPIMC

Supported image format	YUV 4:2:0
Supported VOP type	I frame & P frame
Supported image size	Programmable, Up to HD (1280 x 720)
Encoding frame rate	Programmable, Up to 30 frames/sec
Maximum operation frequency	140 MHz
Voltage	1.8 V
Power consumption	262 mW @ 21 MHz
Gate count	204K gates
On-chip memory	6462 bytes
ME Algorithm	HMEA, 4MV mode, Search range -16.0 to +15.5

Table 6 The performance comparison between the software model and RPIMC in dB

Video sequences	XviD official version software encoder	RPIMC
Akiyo	42.38	41.45
Foreman	30.81	29.92
Table tennis	31.51	29.66
Mobile	21.31	20.93
Flower garden	22.25	21.88

Table 7 The performance comparison between other MPEG-4 chips and RPIMC

Designer	[2]	[3]	[4]	[5]	[12]	RPIMC
Encoding Complexity	CIF, 15FPS	CIF, 30 FPS	CIF, 15FPS	QCIF, 15 FPS	CIF, 30FPS	Programmable Up to HD, 20FPS
Operating Frequency (MHz)	13.5	40	27	60	36	Maximum 140
Power (mW)	29	256.8	500	240	N/A	262 @ 21 MHz
Gate Counts (K)	700	278	1,700	6,800 (DRAM)	170	204
Process (μm)	0.18	0.35	0.35	0.25	0.18	0.18

95-2752-E-009-012-PAE, and Aiming for the Top University Plan of the National Chiao Tung University and Ministry of Education, Taiwan, under Grant 95W803E.

References

[1] A. Hatabu, T. Miyazaki, and I. Kuroda, “QVGA/CIF resolution MPEG-4 video codec based on a low-power and

general-purpose DSP," in *IEEE Workshop on Signal Processing Systems (SIPS)*, 2002, pp. 15-20.

[2] H. Nakayama, T. Yoshitake, H. Komazaki, Y. Watanabe, H. Araki, K. Morioka, J. Li, L. Prilin, S. Lee, H. Kubosawa, and Y. Otobe, "An MPEG-4 Video LSI with an Error-Resilient Codec Core Based on a Fast Motion Estimation Algorithm," in *IEEE internal Solid-State Circuits Conference (ISSCC)*, 2002, Vol. 1, pp. 368-370.

[3] Ying-Chi Chang, Wei-Min Chao and Liang-Gee Chen, "Platform-based MPEG-4 Video Encoder SoC Design," in *IEEE Workshop on Signal Processing Systems (SIPS)*, 2004, pp. 251-256.

[4] J. H. Park, I. K. Kim, S. M. Kim, S. M. Park, B. T. Koo, K. S. Shin, K. B. Seo, and J. J. Cha, "MPEG-4 Video Codec on an ARM Core and AMBA," in *Workshop and Exhibition on MPEG-4*, 2001, pp. 95-98.

[5] M. Takahashi, T. Nishikawa, M. Hamada, T. Takayanagi, H. Arakida, N. Machida, H. Yamamoto, T. Fujiyoshi, Y. Ohashi, O. Yamagishi, T. Samata, A. Asano, T. Terazawa, K. Ohmori, Y. Watanabe, H. Nakamura, S. Minami, T. Kuroda, and T. Furuyama, "A 60-MHz 240-mW MPEG-4 Videophone LSI with 16-Mb Embedded DRAM," *IEEE Journal of Solid-State Circuit*, Vol. 35, No. 11, pp. 1713-1721, Nov. 2000.

[6] P. M. Kuhn and W. Stechele, "Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation," in *International Conference on Visual Communications and Image Processing*, 1998.

[7] H. C. Chang, L. G. Chen, M. Y. Hsu, and Y. C. Chang, "Performance analysis and architecture evaluation of MPEG-4 video codec system," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2000, Vol. 2, pp. 449-452.

[8] Bing-Fei Wu, Hsin-Yuan Peng, and Tung-Lung Yu, "Efficient Hierarchical Motion Estimation Algorithm And Its VLSI Architecture," accepted by *IEEE Trans. On VLSI Syst.*, Sep. 16, 2007.

[9] J. Lu and M. L. Liou, "A simple efficient search algorithm for block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*,vol. 7, Apr. 1997, 429-433.

[10] X. Song, T. Chiang, X. Lee, and Y.-Q. Zhang, "New fast binary pyramid motion estimation for MPEG2 and HDTV encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1015-1028, Oct. 2000.

[11] Jae Hun Lee, Kyoung Won Lim, Byung Cheol Song, and Jong Becom Ra, "A Fast Multi-resolution Block Matching Algorithm and its LSI Architecture for Low Bit-Rate Video Coding," *IEEE Trans. On Circuits and Systems for Video Technology, Vol. 11, No. 12*, Dec. 2001.

[12] Amphion CS6701 is available on http://www.amphion.com/