PLATFORM-BASED MPEG-4 VIDEO ENCODER SOC DESIGN

Yung-Chi Chang, Wei-Min Chao and Liang-Gee Chen

DSP/IC Design Lab
Department of Electrical Engineering and Graduate Institute of Electronics Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

ABSTRACT

An MPEG-4 video coding SOC design is presented in this paper. We adopt platform-based architecture with an embedded RISC core and efficient memory organization. A motion estimator supporting predictive diamond search and spiral full search is implemented for compromise between compression performance and design cost. The proposed data reuse scheme reduces required memory access bandwidth. Several key modules are integrated into an efficient platform in hardware/software co-design fashion. The cost-efficient video encoder SOC consumes 256.8mW at 40MHz and achieves real-time encoding of 30 CIF (352x288) frames per second.

1. INTRODUCTION

MPEG-4 standard is becoming the main technique of the mobile devices and streaming video applications such as smart phone and handheld PDA devices. The improved coding efficiency and advanced functionalities of MPEG-4 come with much higher computational complexity compared with previous standards. Several MPEG-4 video chips have been reported. To satisfy rich functionality of future multimedia, some are implemented in software [3] based on the low-power DSP platform. They have highest flexibility but degraded quality due to the fast algorithms of ME and DCT. Some [4] use the dedicated hardware methodology to achieve low power and low area cost. Lack of potential for future modification of advanced algorithms and higher design effort are disadvantages. Hence, some [5] [6] adopted the hybrid software/hardware co-design to compromise the performance and flexibility.

According to the computational complexity analysis reported in [1] and [2], the dominating computation-intensive tasks in MPEG-4 core profile coding are motion estimation(ME) and shape encoding, which together contribute more than 90% of the overall complexity. For simple profile without shape coding tools, ME becomes the most significant one. It belongs to highly regular low-level task, and a huge amount of data access through frame buffer is also required. So, dedicated architectures and local buffers are heavily relied for efficient implementations and data access reduction. For other coding tasks, including DCT/IDCT, Q/IQ, and MC, dedicated architectures can be adopted for these highly regular tasks. Programmable architectures are suitable for the other less-demanding but high-level task, such as system control.

In this paper, a RISC-based platform with hardware accelerators is presented to implement MPEG-4 video encoding algorithms. The optimization in both algorithm and architecture level is applied. Not only the key components but also the connection optimization and memory organization are discussed in this paper. The whole system is divided into three main subsystems. In motion subsystem, the hybrid motion estimator supporting both predictive diamond search and spiral full search with halfway termination for real-time or high compression quality applications are proposed to reduce the dominant cost in the typical coding system. In texture subsystem, the efficient interleaving schedule and substructure sharing technique among quantization and DC/AC prediction are proposed [7] to reduce the cost further. In bitstream subsystem, to handle the complex bitstream syntax and avoid inefficient bit-level storage, the hardware/software co-operations scheme is applied for the bitstream generation. By applying these optimization approaches, a low cost and high performance MPEG-4 video encoder SOC is implemented.

This paper is organized as follows. In Sec. 2, the whole system architecture will be explored. The system memory organization will be discussed in Sec. 3. In Sec. 4, we will present the algorithm, architecture, and performance of the motion estimator. The implementation result will be shown in Sec. 5. In Sec. 6, a brief conclusion will be given.

2. SYSTEM ARCHITECTURE

Fig. 1 depicts the proposed platform-based MPEG-4 video coding system. RISC takes responsibility for MB level hardware scheduling, coding mode decision, motion vector coding, and other high level procedures. Other hardware accelerators improve the system performance by parallel processing according to the parallelism of algorithms. Mo-

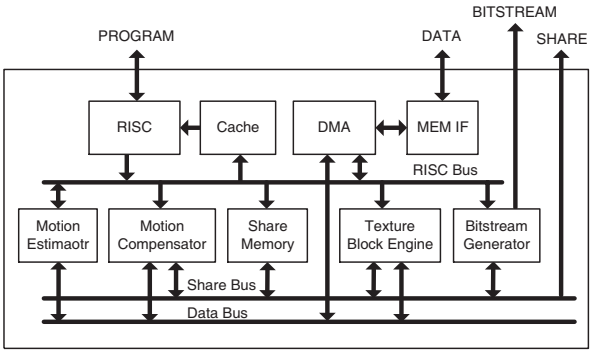

Fig. 1. System Architecture

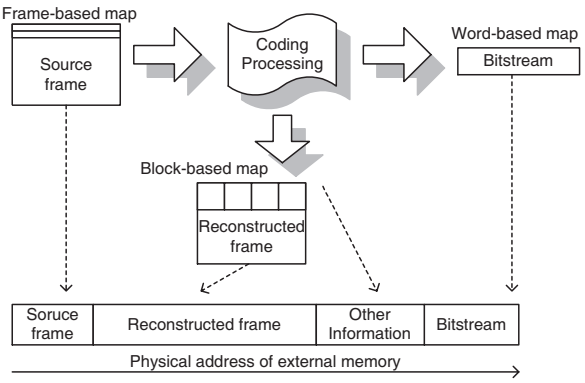

Fig. 2. Heterogeneous memory organizations

tion estimator (ME) carries out motion estimation with the search range -16.0 to +15.5 pixel unit. Motion compensator (MC) interpolates pixels in reference frames into compensated blocks by specified motion vectors. Texture block engine (TBE) carries out discrete cosine transform (DCT), inverse cosine transform (IDCT), quantization (Q), inverse quantization (IQ), and AC/DC prediction on texture pixels in block unit. Bitstream generator (BTS) produces headers, motion information, and texture information in the format of variable length codes. In addition, share memory builds the direct channels from MC to TBE and BE to BTS to decrease the traffic of the data bus. DMA involved in dedicated commands efficiently generates the proper addresses issued by RISC. Four global bus channels are used in this system. First, RISC bus broadcasts controlling information to each hardware modules. After applying operations issued by RISC, hardware modules respond processed side information for MB coding mode decision at RISC. At the same time source, reference, and reconstructed frames required by hardware modules are passed through DMA and then provided by DATA bus. Hardware modules efficiently access the data automatically according to pre-determined scheduling. These parts are integrated into a single chip with the firmware stored outside for programmability through PROGRAM bus after taped out. SHARE bus can transfer DCT coefficients, quantized coefficients, or other immediate information in the testing mode. The developing time and effort can be reduced through this information.

The RISC core contains four stages pipeline with separated program and data memory. Its instruction set is 21 bits. The special 2-operand MAX and MIN instruction is included for the median operations for MV predictor decision. Besides, a hardwired datapath for multiplication and division is also provided. We also propose an immediate store instruction (SWI) to send a specified data to memory. Compared to traditional approach, which requires one instruction to move data into a register and then the other one to store it from the register to the memory, it results in significant reduction in the code size. To achieve cycle-accurately controlling, an inner-timer and polling technique are introduced. A special instruction, WAIT, is used to support this functionality. While the RISC encounters the WAIT instructions, it waits until the next trigger events.

3. MEMORY ORGANIZATION

We have off-chip memory and several on-chip memory blocks. Off-chip memory contains source frames, reconstructed frames, and AC/DC information. On-chip memory is used as local buffers to reduce the bus bandwidth. Due to the penalty of irregular accessing to and from off-chip RAM, we access off-chip RAM more successively by using random access on-chip RAM. For MPEG-4 video coding, block-based memory organization is efficient to burst reading a block of data for video processing. However, the common video input/output devices usually adopt the raster scan direction. It makes addressing more regular if frame data is arranged in frame-based scheme. Therefore, we use heterogeneous memory organization for off-chip RAM as shown in Fig. 2. The source frames are stored in the frame-based way, while the reconstructed frames are store in the block-based way for processing in the future. The bitstream data and AC/DC information is arranged as traditional 1-D addressing. After this arrangement, the data access to/from off-chip will more consecutive.

Fig. 3 shows four types off-chip memory access. Each store unit (word) consists of four pixels. In case (a) the search window (SW) data of 48x48 pixels for ME are loaded from the previous reconstructed frame. In case (b) the 9x9 reference blocks are required for half-pixel MC. Reading in vertical direction can reduce the frequency of crossing neighboring blocks. In case (c) data are read out from source frames in frame-based organization. In case (d) the 8x8 re-

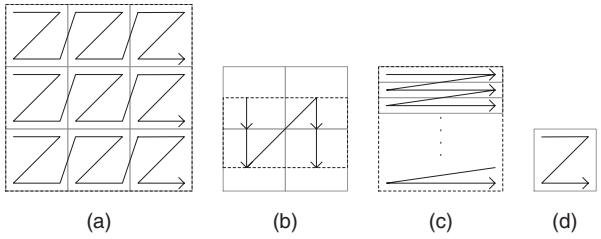

Fig. 3. Memory Access Scheme (a) SW for ME (b) Block for MC (c) Block to TBE (d) Block from TBE

constructed block is burst written to the block-based organization region of external memory without any segmentation.

The input video source, reconstructed frames, and transformed coefficients for AC/DC prediction are stored in the external memory. Direct Memory Access (DMA) plays a role to control memory interface (MIF) to read data from or write data to the external memory in a specified sequence after being initialized by RISC. For this kind of data-intensive applications, DMA always have a heavy load to handle the traffic through the data bus. Therefore, three special functions are involved in DMA to reduce addressing overhead and to provide pixel data more efficiently. It not only can improve the data access but also decrease the complexity of address generation in other hardware modules. First, the addressing generation combines the conversion process of 2-D to 1-D address. Second, the advanced prediction mode allows motion vectors to point out of the VOP and the data is padded from the boundary pixels in this situation. DMA handles this problem of boundary data for ME and MC units that can focus on the current processing MB. Third, special addressing for half-pixel precision compensation is supported. Due to the half-pixel precision for motion compensation, the compensated block is read out in 9 by 9 pixels and may occupy the four blocks in the block-based memory organization. This kind of fixed addressing is designed in the control unit of the DMA to improve the performance.

4. MOTION ESTIMATOR DESIGN

4.1. Algorithm

To meet the requirement of various applications under the acceptable cost, we adopt two kinds of algorithms for the motion estimation of 16x16 block size at integer-pixel precision. One is the spiral full search with halfway termination (called fast full search, FFS) which can achieve the same compression efficiency as the full search algorithm. The other is the diamond search starting from the predictor derived from neighboring MBs (called predictive diamond search, PDS) and it meets the real-time specification under the visual quality degradation. Afterwards, the hierarchy scheme is applied for the motion estimation for four 8x8 pixels blocks in a MB around +2 to -2 positions of the previous best motion vector. The half-pixel refinement is also applied for all found integer-pixel motion vectors. The whole stages of motion estimation is described as follows. The predictor is determined from neighboring MBs. The PDS mode or FFS mode is employed to find the integer pixel motion vectors. The half-pixel refinement is applied around the motion vector found in the phase 2. For four 8x8 pixel blocks in a MB, the spiral search around -2 to +2 is applied to obtain four optimal motion vectors. Four times of half-pixel refinement is applied around the motion vectors found in the previous phases.

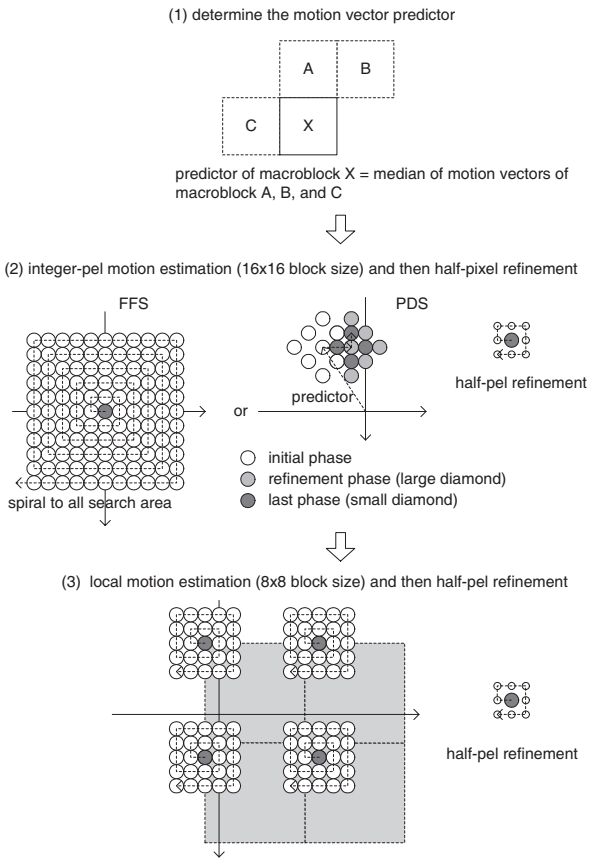

Fig. 4. Algorithms of motion estimation

Fig. 4 depicts the whole stages of motion estimation and describes as follows. The predictor is determined from neighboring MBs. The PDS mode or FFS mode is employed to find the integer pixel motion vectors. The half-pixel refinement is applied around the motion vector found in the phase 2. For four 8x8 pixel blocks in a MB, the spiral search around -2 to +2 is applied to obtain four optimal motion vectors. Four times of half-pixel refinement is applied

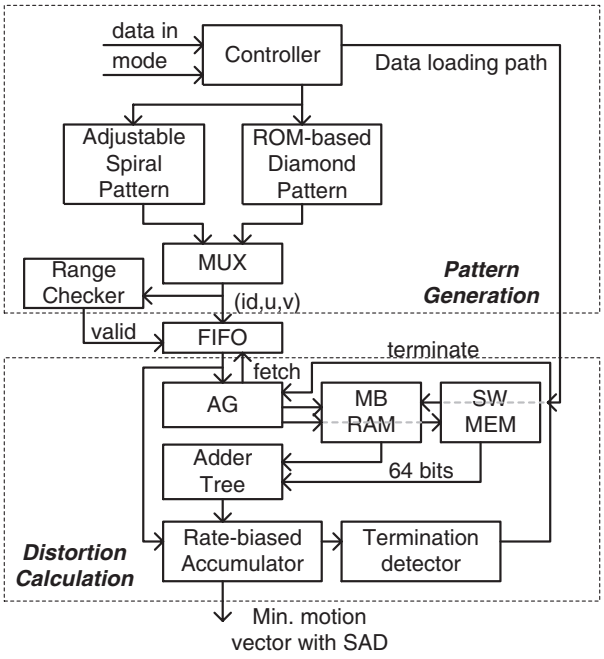

Fig. 5. Architecture of motion estimator

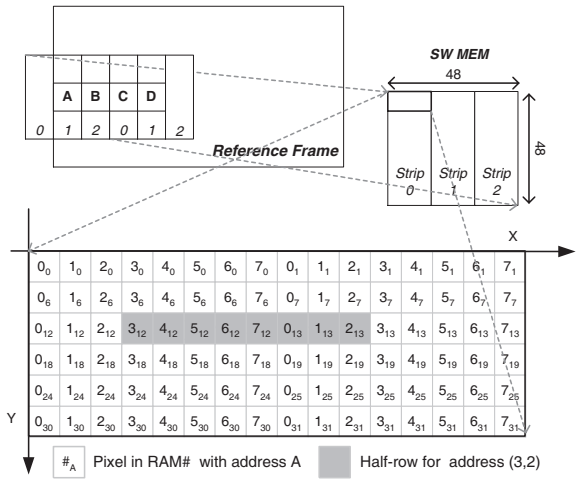

Fig. 6. Memory organization of motion estimator

around the motion vectors found in the previous phases.

4.2. Architecture

Fig. 5 depicts the hardware architecture of the motion estimator supporting PDS and FFS. This architecture mainly includes three processing stages and two buffers to store current MB and the search window. Before performing motion estimation, the video coding system transfers data from external memory into these buffers to eliminate the bus bandwidth for calculating of sum of absolute difference in the following. Meanwhile, the adder tree accumulates the sum of the pixels in the current MB to save it into a register for the mode decision in the future. To speed up the data loading and reduce the bus traffic, the search window buffer can be loaded using column-by-column data-reuse scheme. After motion estimation starts, the pattern generation (PG) stage generates the valid candidate positions. Then these positions are passed through the FIFO stage and fetched by the distortion calculation (DC) stage. The DC stage is responsible for calculating SAD of candidate positions and finds the minimum one. The accumulation comparison elimination (ACE) unit performs the PDE algorithm to reduce the computational complexity.

4.3. Data Reuse Scheme

The eight-way interleaved memory organization is used to dynamically fetch eight pixels in one cycle without reading collision in the same memory bank. Fig. 6 depicts this organization with the search range -16 to +15 pixels. Before performing motion estimation for each MB labeled as A, B, or C, it needs to load colocated 48 by 48 pixel size of search windows in the reference frame to search window memory (SWMEM) which is divided into three strips and each one contains exclusive sixteen pixels. Under the leftmost MB of the frame, all data located in these three strips are required to be loaded. However, the left MBs in the same row can reuse two-thrid of search window of the immediately previous MB.

With rotation and modulation operations for addressing, this column-by-column data reuse scheme is applied to this motion estimation architecture. The bus traffic for loading search window is then reduced from 26.10 to 9.49 Mbytes per second for CIF format with the search range of -16 to +15 pixel. In each strip, eight horizontal neighboring pixels (a half row) are stored into eight separated memory with the linear addressing. While reading a half-row of pixels randomly, two consecutive addresses are calculated first from the two-dimension coordinates. Then the proper circular rotative operations are applied to the data read out from the memory banks.

4.4. Performance

The PDS mode can satisfy the real-time specification while the FFS mode can achieve the same compression quality as MPEG-4 software verified model (VM) [8]. To explore the degradation in the PDS mode, four sequences with different features are used as test patterns. The average difference between PDS and VM in PSNR is only 0.136 dB and the maximum PSNR drop through the testing sequences is only 0.618 dB. Even in the frames whose the difference in PSNR

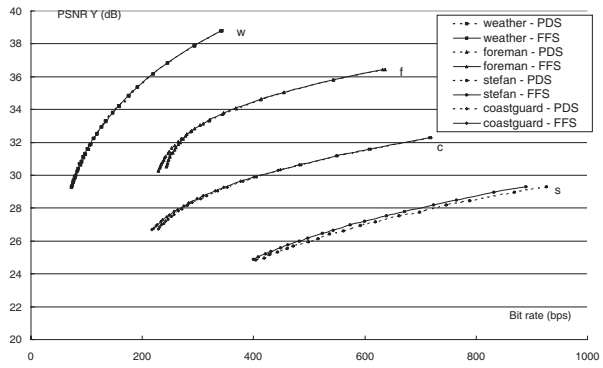

Fig. 7. RD curves with PDS and FFS modes

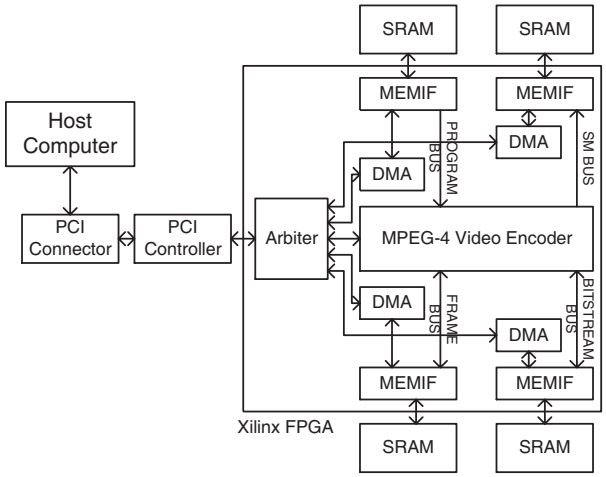

Fig. 8. Configurable platform

are maximum, it is still indistinguishable between these two in subject view. While encoding in the FFS mode, the PSNR and bit-rate of the reconstructed frames are almost the same as that encoded by VM. The average PSNR are even better than 0.00625 dB. The general R-D curves for testing sequence are simulated and shown in Fig. 7.

5. IMPLEMENTATION

A configurable platform shown in Fig. 8 is used to verify the functionality of our architecture design. This prototyping board is connected through the PCI interface to the host computer. Four separated memory with DMA modules are used to handle PROGRAM, DATA, SHARE, and BITSTREAM bus from our design. An arbiter is responsible for the memory access through PCI and memory. The MPEG-4 video encoder design is synthesized and placed on the FPGA chip. The RISC program is compiled to machine codes by the host computer and then sent to the program memory. Raw image data is transferred from the host computer to the frame memory on the prototyping board. Video encoding is processed concurrently. Afterwards, bitstream data are stored in the bitstream memory and then read from the host computer. Besides, the share memory can record the immediate information for debugging in the testing mode.

Fig. 9 shows a micrograph of the encoder and Table 1 depicts its characteristics. It contains 828K transistors and is fabricated on a 5.02 x 5.13 mm^2 with 0.35 μ m and single-poly quadruple-metal CMOS process. The chip is tested and works successfully. The supply voltage is 3.3V and consumes 256.8mW at 40MHz working frequency. Table 2 shows the number of transistors, the area, and the size ratio to the chip of each unit.

Table 3 gives a comparison of some MPEG-4 video codec proposed before. In [4], it is a full dedicated hardware video codec design. It uses MVFAST for ME with search range -16$\sim$+15.5. In [5], it is a platform-based video/speech codec design. It uses 3-step hierarchical search for ME with search range -32$\sim$+31.5. In [6], it is a platform-based video codec design with ARM/AMBA. It uses a coarse ME with search range -8$\sim$+7.5. All chip designs adopts fast algorithms for motion estimation. In the viewpoint of video encoder parts, our work has highest encoding complexity and the lowest cost meanwhile.

Table 1. Characteristics of the encoder chip

Technology	TSMC 0.35 μm 1P4M CMOS
Die Size	5.02 x 5.13 mm^2
Transistor count	828,692 trans.
On-chip memory	39,080 bits
Off-chip memory	2,027,527 bits
Clock frequency	40 MHz
Voltage	3.3V
Power consumption	256.8mW
Package	208 CQFP
ME algorithm	PDS/FFS, 4MV mode
	Search range -16.0 to +15.5
Encoding complexity	352 x 288 at 30 fps

6. CONCLUSION

An efficient platform architecture design with hardware accelerators for MPEG-4 Simple Profile@Level 3 video encoder SOC is proposed in this paper. With the proposed hybrid motion estimation and efficient memory organization, the system are implemented with 0.35 μm CMOS technology. It works at 40MHz and consumes 256.8mW with 5.03x5.13 mm^2 die size to meet the real-time encoding spec-

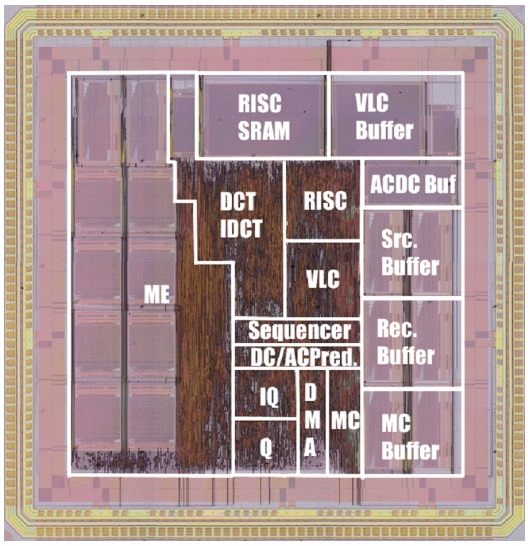

Fig. 9. Micrograph of this encoder

Table 2. Cost distribution

	Trans. (k)	Area (mm^2)	Size ratio (%)
ME	288	5.8	22.6
MC	53	0.3	1.2
DCT/IDCT in TBE	126	1.6	6.2
Q/IQ in TBE	64	0.7	2.9
ACDCP in TBE	22	0.8	3.0
RISC	112	1.8	7.0
DMA	19	0.3	1.2
VLC	95	0.7	2.7
Share MEM	68	2.8	10.9
Others (PAD etc.)	49	10.9	42.3
Total	829	25.8	100.0

ification. The proposed design achieves high performance with low design cost, which proves that a cost-effective MPEG-4 coding system implementation is realized.

7. REFERENCES

[1] 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.

[2] 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.

Table 3. Architectures Comparison

Designer	[4]	[5]	[6]	Proposed
Encoding Complexity	CIF, 15fps	QCIF, 15fps	CIF, 15fps	CIF, 30fps
Frequency (MHz)	13.5	60	27	40
Power (mW)	29	240	500	256.8
Transistor (K)	3,150	20,500 (DRAM)	1,700	829
Process (μm)	0.18	0.25	0.35	0.35
Chip area (mm^2)	28.048	117.506	110.25	25.801

[3] 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.

[4] H. Nakayama, T. Yoshitake, H. Komazaki, Y. Watanabe, H. Araki, K. Morioka, J. Li, L. Peilin, 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 International Solid-State Circuits Conference (ISSCC)*, 2002, vol. 1, pp. 368–474.

[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] 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.

[7] C. W. Hsu, W. M. Chao, Y. C. Chang, and L. G. Chen, "Cost-Effective Scheduling Of Texture Coding For MPEG-4 Video," *IEEE International Conference on Multimedia and Expo(ICME'02)*, Aug 2002.

[8] T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 19–31, Feb 1997.