Chapter 1

Introduction

1.1 Background

Over the past two decades, video compression (coding) techniques have been extensively studied to reduce the large transmission bandwidth and huge storage space of uncompressed video data. As a result, a number of international video coding standards (such as MPEG and H.26X series) have been established assuring interoperability between the products of different manufactures and worldwide spreading of video coding technology. H.264/AVC (ITU-T & ISO/IEC, 2003) is the latest video coding standard, which significantly outperforms prior standards in terms of video quality and coding performance. For example, compared with MPEG-2 (ITU-T & ISO/IEC, 1994) and MPEG-4 (ISO/IEC, 1999), H.264/AVC can save 64% and 39% of bit rates in average (Joch, Kossentini, Schwarz, Wiegand, & Sullivan, 2002). Due to the higher coding performance and better subjective quality of the H.264/AVC standard, it plays pivotal roles in a wide range of video applications such as digital camcorder, multimedia phone, DVD player, digital TV broadcasting, and video conferencing.

The superior compression performance of H.264/AVC originates mainly from its inter prediction part (i.e. motion estimation (ME)) with new features including variable block-size (VBS), quarter-pixel accuracy, and multiple reference frames (MRF). However, H.264 ME requires huge computational complexity and memory bandwidth.

Therefore, optimization of the H.264 ME at algorithm level accompanied by its acceleration with efficient dedicated architectures is required for real-time applications. This thesis is concerned with the H.264 ME that consists of integer motion estimation (IME) and sub-pixel motion estimation (SME). In this thesis, we study the H.264 IME and SME algorithms and architectures, and propose efficient low cost architectures for them.

1.2 The Need for Video Compression

Digital videos are voluminous and contain huge amount of data. Therefore, video compression techniques are required to reduce transmission bandwidth as well as storage space. To explain the huge amount of data in digital video, consider a standard-definition television (SDTV) video sequence with a resolution of 720×576 and a frame rate of 30 frames per second (fps). If Red-Green-Blue (RGB) color space is used for video color representation and with one byte for each component, three bytes per pixel are required for carrying the color information. Therefore, each frame requires 720×576×3×30= 37.32 MB. As a result, in order to store one-hour video, it needs 60×60×37.32=134.35 GB storage space. For streaming of this video over a network, the required bandwidth is 37.32×8=298.56 Mbps. It is clear that for bigger resolutions such as high-definition television (HDTV), much higher storage capacity and bandwidth are required. In addition, the voluminous video data increase the complexity as well as implementation cost of the video processing systems that use uncompressed digital video. Therefore, it is essential to compress digital video even with increasingly available storage space and transmission bandwidth (Lee & Kalva, 2006).

1.3 H.264/AVC

The latest and widely accepted video coding standard, ITU-T H.264 or MPEG-4 Advanced Video Coding (AVC) that is commonly known as H.264/AVC, is the result of the collaborative work between the ITU-T Video Coding Expert Group (VCEG) and the ISO/IES Moving Picture Expert Group (MPEG) referred as the Joint Video Team (JVT). The main goals of H.264 are enhanced compression efficiency, network friendly video representation for interactive (video telephony) and non-interactive applications (broadcast, streaming, storage, and video on demand) (Ostermann et al., 2004). MPEG-2, the previous popular standard for digital TV systems, had been established 10 years before H.264/AVC and was not able to satisfy the requirement of new and emerging applications. For example, HDTV requires higher transmission bandwidth or transmission media such as Cable Modem and xDSL offers much lower data rates than broadcast channels, and thus improved coding performance was required to facilitate the use of such technologies (Wiegand, Sullivan, Bjontegaard, & Luthra, 2003). In addition, the prior video coding standards for video telecommunication such as H.261, H.263, H.263+ were not suitable for recent and future wireless or wired networks as well as their formatting and error robustness requirements.

The H.264/AVC standard consists of a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL) to achieve its goals. The H.264 VCL determines the source coding approaches and algorithms, which results in higher coding performance. The H.264 NAL formats the VCL representation of the video and provides header information to package that data for network transport (Sullivan & Wiegand, 2005). This thesis is concerned with the main part of the H.264 Video Coding Layer, which is motion estimation. However, the interested readers can find the details of the H.264 Network Layer Abstraction in (Wenger, 2003) and (Stockhammer, Hannuksela, & Wiegand, 2003) including error resilience issue.

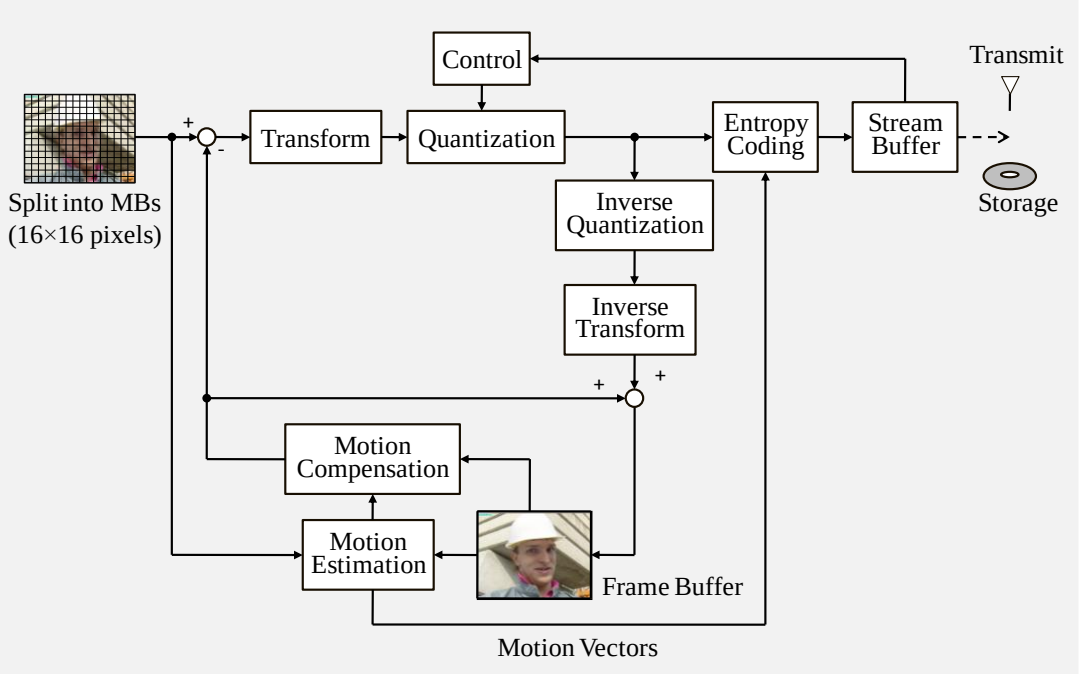


Figure 1.1 : Typical block diagram of a block-based hybrid video encoder.

As shown in Figure 1.1, the H.264/AVC video coding layer is similar to the prior video coding standards, which is a hybrid of motion compensation and transform coding techniques. However, it employs several new coding tools to improve its coding performance. Motion estimation with variable block sizes, quarter-pixel accuracy and multiple reference frames is adopted to exploit more temporal redundancy. Intra prediction with several prediction modes is applied to remove spatial redundancy by using neighboring samples of prior coded blocks. Furthermore, context-based adaptive binary arithmetic coding is used in the last stage of encoder exploiting more statistical redundancy. In-loop deblocking filter is employed to reduce blocky artifacts leading to improvement of the perceptual quality. The detailed descriptions about the new coding tools in H.264/AVC can be found in some excellent papers (Ostermann et al., 2004; Sullivan & Wiegand, 2005; Wiegand, Sullivan et al., 2003).

By using these new coding tools, H.264/AVC can outperform prior video coding standards in terms of coding performance and video quality. Table 1.1 shows the

average bit rate saving of H.264/AVC main profile (MP) compared with MPEG-2 MP @ main level (ML), H.263 high latency profile (HLP) and MPEG-4 advanced simple profile (ASP) standards using several video sequences where H.264 considerably outperforms other coders (Schafer, Wiegand, & Schwartz, 2003). That makes H.264/AVC the best standard for video compression, especially in bandwidth and storage limited applications.

Although H.264/AVC has the best performances compared with previous successful standards, its gain in coding performance comes at the price of huge computational complexity. For example, the profiling at instruction level shows that H.264/AVC is about 6 times more complex than MPEG-4 simple profile (SP) at the encoder and 2.2 times more complex at the decoder (Lian, Tseng, & Chen, 2006). The higher computational complexity of H.264/AVC increases the difficulty of its implementation, particularly for encoders that require much more computation than decoders. With the up-to-date technique, the hardwired encoder is still essential for real-time applications, especially when facing HDTV specifications. For example, the hardware encoder design of Huang et al. (2005) achieved 1200 times speed-up in comparison with Pentium-IV 3 GHz CPU (Liu et al., 2009).

Table 1.1 : Average bit rate saving of H.264 relative to prior standards (Schafer et al., 2003).

Coder	MPEG-4 ASP	H.263 HLP	MPEG-2
H.264/AVC	38.62%	48.80%	64.46%
MPEG-4 ASP	-	16.65%	42.95%
H.263 HLP	-	-	30.61%

1.4 The Importance and Challenges of Motion Estimation Design for H.264/AVC

Motion estimation is the heart of all video coding standards such as MPEG and H.26X series. Over the past two decades, hundreds of studies have been proposed to design efficient motion estimation algorithms and VLSI architectures for real-time applications. In the latest video coding standard, H.264/AVC, motion estimation is characterized with three new tools including VBS, quarter-pixel accurate motion vector (MV), and MRF. These tools significantly improve the coding performance of H.264/AVC. However, they require huge computational complexity and memory bandwidth. Therefore, there is a need for novel and efficient designs that support new features of motion estimation in H.264/AVC.

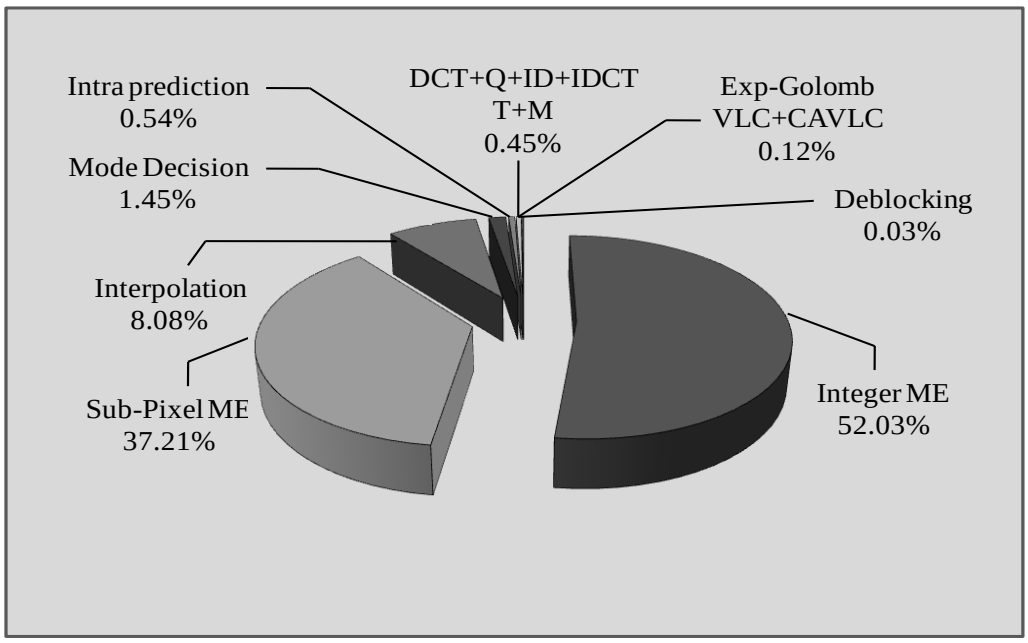
C.-Y. Chen and Fang et al. (2006) used software profiling to demonstrate the huge computational complexity and memory access of the H.264 encoder and to find its critical components. Their instruction profiling of the H.264/AVC baseline encoder shows that for SDTV (720 × 480)/HDTV720p (1280 × 720) video with four/one reference frames, 30 fps, and a search range of [-64, +63] and [-32, +31] in horizontal and vertical directions, the computational complexity is 2,470/3,600 Giga-Instructions Per Second (GIPS). As shown in Figure 1.2, the main part of the computational complexity (i.e. 97.32%) is occupied by motion estimation part. The huge computational complexity of motion estimation is much more than the capability of current general-purpose processors. Therefore, acceleration of motion estimation in H.264/AVC by dedicated hardware is required for real-time applications.

In addition, based on the profiling results, the H.264/AVC ME requires a very large memory access so that for SDTV (720 × 480)/HDTV720p (1280 × 720), it needs 3,800/5,570 Giga-Bytes per second (GBPS), respectively. Consequently, there is a need

for effective techniques to lower the memory access requirement of motion estimation for real-time applications.

In summary, the profiling results reveal that the huge computational complexity and the memory access requirement of the H.264/AVC ME are the main bottleneck of video encoding systems. Therefore, design of efficient hardware architectures for integer and sub-pixel motion estimation is of vital importance for real-time applications.

Figure 1.2 : Runtime percentage of functional blocks in H.264/AVC baseline encoder (C.-Y. Chen, Fang et al., 2006).



1.5 Research Objective

The main objective of our research is to design low cost hardware architectures for IME and SME in H.264/AVC. Due to different characteristics of IME and SME, different but appropriate design considerations and techniques should be used for each part at algorithm and architecture levels.

At algorithm level, we investigate algorithms that have good coding performance and are suitable for hardware implementations. In addition, we try to optimize their

computational complexity and memory bandwidth while maintaining their coding performances.

As for architecture level, investigation of new hardware architectures and enhancing of state-of-the-art designs are taken into consideration. These hardware architectures should have regular and simple structure, good performance in terms of speed, hardware cost, hardware utilization, memory bandwidth, and processing ability.

1.6 Research Scope and Applications

The scope of this thesis is to design the H.264/AVC IME and SME architectures for real-time encoding of medium toward large resolutions such as HDTV. Since these architectures target area-constrained applications (i.e. portable multimedia devices), their area costs, pin count, and power consumptions should be small. Regarding the IME design, since the heart of IME architectures is processing element array, our concern is to design a high performance processing element array featured with regular structure, efficient memory bandwidth, and small area cost. As for SME design, the main concern is to reduce huge computational load and memory access requirement of interpolation process, and to design an efficient SME hardware accelerator with small hardware cost and memory bandwidth.

For implementation of IME and SME architectures, there are different design alternatives that can be mainly classified into two categories including processor-based designs and application specific integrated circuit (ASIC) designs. Processor-based designs show better programmability and lower performance whereas ASIC designs provide the best performance with little flexibility (Y.-L. Lin, 2006). Between these two categories, there are other options such as digital signal processors, multimedia processors, and Field Programmable Gate Arrays (FPGAs), which provide moderate performance and flexibility. Among all of design alternatives, ASIC approach is the

most suitable choice for our research due to its best performance. In addition, most of the reported designs in literature are based on ASIC design so that we can use them as reference for evaluating the performance of our designs.

1.7 Thesis Organization

Chapter 2 describes the H.264 IME, its research methodology and common design metrics for evaluating of motion estimation designs. In addition, state-of-the-art IME architectures are reviewed and evaluated based on introduced design metrics. Chapter 3 analyzes the SME features in the H.264 standard and provides a thorough survey of SME algorithms and architectures. In Chapter 4, the proposed bit-serial architectures for the H.264 IME are presented in detail. Chapter 5 describes our SME algorithm and its hardware architecture. Finally, Chapter 6 draws a conclusion and presents future work and directions.

Chapter 2

Review of Integer Motion Estimation Designs

2.1 Introduction

Motion estimation with integer pixel accuracy, i.e. referred as motion estimation (ME) in this chapter, is used as a powerful technique in video coding systems to remove temporal redundancy between successive frames, which leads to high level of coding performance in such systems. Motion estimation is a computationally intensive process, which typically consumes more than 50% of computation of encoders. Among different motion estimation algorithms, full search (i.e., exhaustive search) block matching motion estimation algorithm is widely used due to its better coding performance and regularity compared with fast search algorithms such as four-step search (Po & Ma, 1996) and diamond search (Tham, Ranganath, Ranganath, & Kassim, 1998).

In this algorithm, each luminance frame is divided to macroblocks (MBs) of size $N \times N$ (i.e. 16×16). Then, each MB at the current frame is matched against a corresponding MB at the reference frame within a search range of *[–w, w-1]*, as depicted in Figure 2.1.

The most common matching criteria are Sum of Absolute Differences (SAD) and Sum of Square Differences (SSD), as denoted in equations (2.1) and (2.2), respectively.

$$SAD(i,j) = \sum_{m=1}^{N} \sum_{n=1}^{N} |C(m,n) - R(m+i, n+j)| \quad ; \quad -w \le i \;\&\; j < w \,. \qquad (2.1)$$

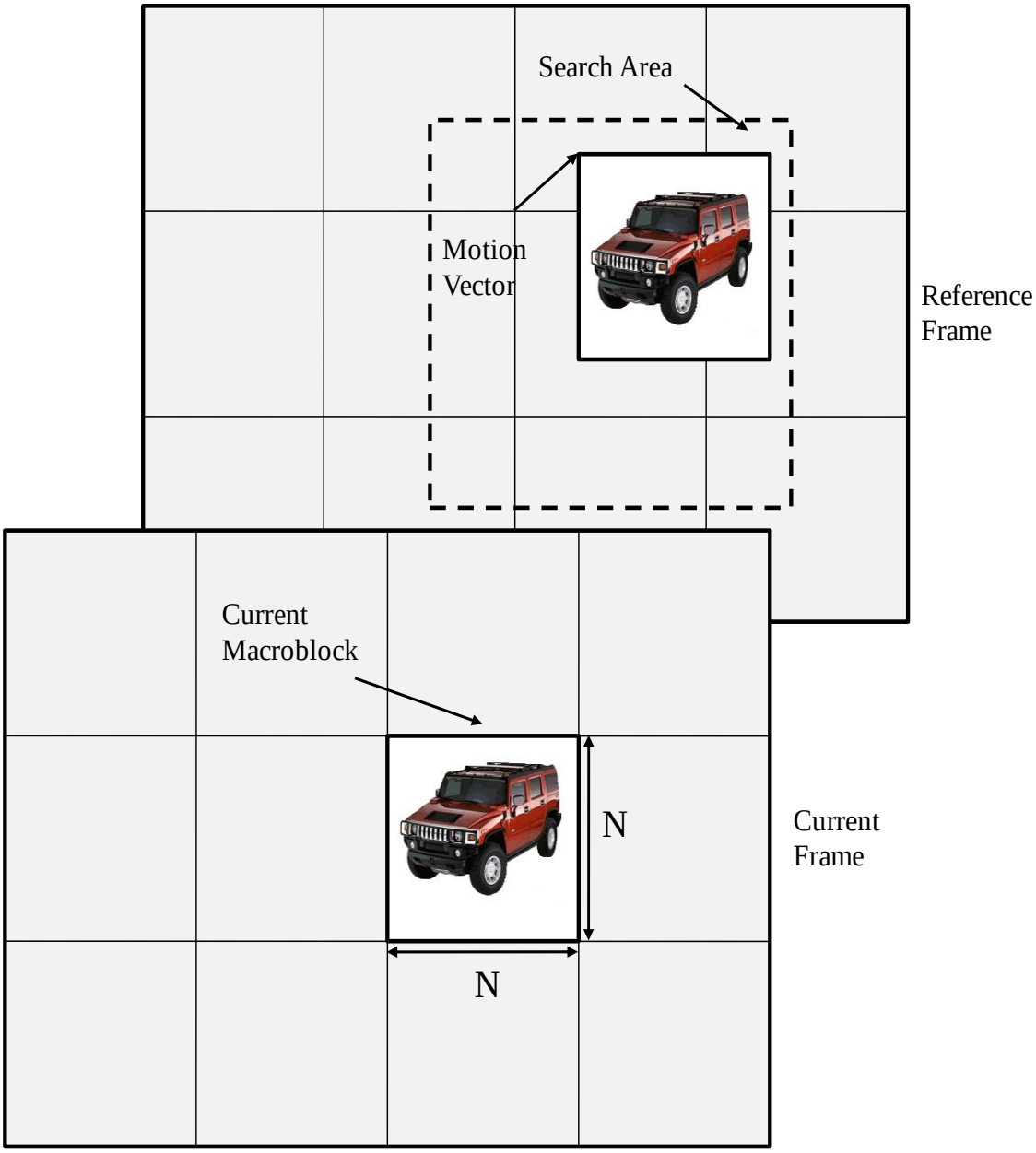


Figure 2.1 : Block matching motion estimation algorithm.

$$SSD(i,j) = \sum_{m=1}^{N}\sum_{n=1}^{N}((C(m,n) - R(m+i,n+j))^2 \quad ; \quad -w \leq i \;\&\; j < w \;. \qquad (2.2)$$

In these equations, *(i, j)* is the current MV, *C(m, n)* the current MB, and *R(m+i, n+j)* refers to the pixel values in the reference MB. After checking all search locations, the search location with the smallest SAD is selected as the MV of the current MB.

Although SSD can lead to a better prediction result, SAD is commonly used in video coding systems as matching criterion owing to its good accuracy and lower computational complexity compared to SSD (S.-H. Wang, 2007).

When there is more than one object in an MB with different moving directions, fixed block sized motion estimation decreases the coding performance. Therefore, it is more efficient to use variable block-size motion estimation to find the best match for each object. For this reason, in H.264/AVC motion estimation with variable block-size (i.e. 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 block sizes, as shown in Figure 2.2) is adapted to improve video coding performance. In addition, after conducting motion estimation for all block sizes, inter-mode decision is carried out for determining the optimal coding mode that improves the coding performance. Besides, H.264/AVC supports motion estimation with MRF, i.e. up to five prior coded frames can be used as reference in inter prediction, which can enhance the performance of H.264/AVC. Although variable block-size motion estimation and MRF contribute to the rate-distortion (RD) performance of the H.264 encoder, they increase the computation complexity, memory bandwidth, and difficulty of motion estimator implementation.

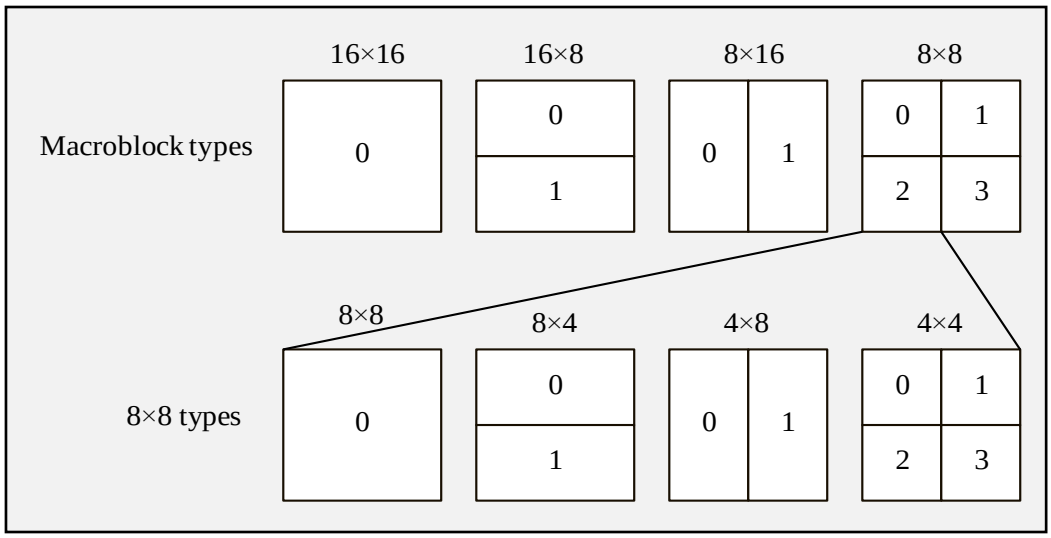


Figure 2.2 : Variable block sizes in H.264/AVC.

2.2 Research Methodology and Design Metrics Assessment

As mentioned in previous chapter design of motion estimation for H.264 consists of two parts i.e. algorithm and architecture. In this thesis, we use H.264 reference software for implementing and evaluation of our algorithms where we compare them with the full search algorithm in terms of peak signal to noise ratio (*PSNR*) which is explained in the following paragraphs. As for architectures, we use Verilog-HDL for implementation of our designs and synthesize them in Silterra 0.18 μm technology using Synopsys Design Compiler. Now we discuss the design metrics for assessment of motion estimation designs. Generally, there are a number of important design metrics for assessment of motion estimation designs including the quality of algorithm, silicon area, operating frequency, throughput, memory bandwidth, and hardware utilization (C.-Y. Chen, Chien et al., 2006; Kuhn, 1999), which are described below.

- **Algorithm quality:** This metric is used for evaluation the performance of fast search motion estimation algorithm against full search algorithm. Generally, a fast search algorithm has a lower computational cost than the full search algorithm. However, video quality degradation and irregularity are the penalties. The most common metric for evaluating the quality of a fast algorithm is peak signal to noise ratio (*PSNR*), whose formula is as follows:

$$PSNR = 10\ \log_{10}(\frac{255^2}{MSE})\ dB \quad (2.3)$$

where the Mean Square Error (*MSE*) of the current frame (I_C) and its reconstructed frame (I_R), which consist of $H \times W$ pixels, is calculated from:

$$MSE = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}(I_C(i,j) - I_R(i,j))^2\ . \quad (2.4)$$

In equation (2.4), $I_C(i, j)$ and $I_R(i, j)$ are the pixel values in current and reconstructed frames, respectively.

To evaluate the video quality of a fast algorithm, the PSNR difference between this and the full search algorithm is defined as:

$$\Delta PSNR(dB) = PSNR_{fast\ algorithm} - PSNR_{full\ search\ algorithm} \ . \quad (2.5)$$

- **Silicon area/Hardware cost:** Generally, the size of a VLSI chip is defined in terms of square millimeter/micrometer, which will be available after completing the chip design. However, the equivalent gate count of a VLSI design (in terms of 2-input NAND gate) is another widely accepted criterion for estimating the hardware cost at architecture level. Since most of the motion estimation designs in the literature use the gate counts as the measuring criterion, we utilize the gate counts for the evaluating hardware cost in this thesis.
- **Operating frequency:** The required operating frequency is dominated by the degree of parallelism in hardware. The smaller the degree of parallelism is, the higher the required frequency is (C.-Y. Chen, Chien et al., 2006). Generally, a smaller degree of parallelism results in lower hardware cost because in this case reduced number of parallel architectures are used. In addition, since dynamic power is proportional to the operating frequency, the designer can compromise between power, area cost and the required frequency by selecting an optimized degree of parallelism.
- **Throughput:** The throughput is another important metric for the performance of motion estimation designs, and it shows the processing ability of motion estimation architectures. This metric can be defined as the number of processed macroblocks per second (MB/s) by an architecture. Note that the throughput depends on other design parameters such as search range and number of reference

frames. To have a fair comparison between different motion estimation designs in this thesis, we measure the throughput under search range of [-16, 15], one reference frame, and for Common Intermediate Format (CIF) video resolution.

- **Memory bandwidth:** Memory bandwidth is defined as the number of bits, which hardware has to access from memory for each macroblock (C.-Y. Chen, Chien et al., 2006). This parameter influences the traffic of system bus and its power consumption. Memory bandwidth is influenced by data reuse technique. For providing a fair comparison, the memory bandwidth of motion estimation designs is defined for CIF video @ 30 fps, one reference frame, and a search range of [-16, 15] specifications.
- **Hardware utilization:** Besides the above-mentioned design metrics, hardware utilization is another metrics that may be used for motion estimation hardware architectures, which refers to the percentage of computing cycles/operating cycles for an MB. The operating cycles include three parts i.e. latency, computing cycles, and bubble cycles (C.-Y. Chen, Chien et al., 2006). Computation cycles are the required number of cycles for calculating one SAD. The higher the hardware utilization is, the better the hardware efficiency is.

2.3 Review of Motion Estimation Designs

Block matching motion estimation has played an important role in improving of coding performance in the block based video coding standards including MPEG and H.26X series. However, it is computationally most demanding part of video encoders and thus its acceleration with efficient algorithms and hardware architecture has been under consideration. Consequently, over the past two decades, many video coding scientists have extensively studied motion estimation to find the best efficient ways for its implementation at algorithm and architecture levels. Owing to existence of

comprehensive and thorough surveys of block matching motion estimation algorithms and/or architectures in the literature (C.-Y. Chen, Chien et al., 2006; Dufaux & Moscheni, 1995; Huang, Chen, Tsai, Shen, & Chen, 2006; Pirsch, Demassieux, & Gehrke, 1995; Pirsch & Stolberg, 1998; Tseng et al., 2005), it is not essential to provide an in depth and complete survey in this thesis. Instead, we quickly explore the design space of motion estimation and review six representative motion estimation designs, which will be used later as reference for comparison to our designs.

2.3.1 Exploration of Design Space

Design of motion estimation consists of algorithm and architecture parts. Motion estimation algorithms can be divided into two main categories, namely lossless and lossy algorithms. The full search algorithm belongs to the first category and provides the best video quality whereas fast algorithms belong to the second category having lower video quality and computational complexity compared with the full search algorithm. For the hardware implementation, the full search algorithm is much more suitable than a fast search algorithm due to its regularity, lack of data dependencies and optimality (Li & Leong, 2008). In addition, the hardware design of fast search algorithms includes the following challenges: unpredictable data flow, irregular memory access, difficult mapping to systolic arrays, low hardware utilization, and sequential procedures with data dependence that cannot be parallelized (Huang, Chen et al., 2006). Furthermore, the silicon area of architectures that are based on fast algorithms should be significantly smaller than that of full search algorithm for cost efficiency (Tseng et al., 2005). Because of the above-mentioned reasons, the full search algorithm is often adopted in motion estimation architectures.

As for hardware architectures, the main design challenges of motion estimation can be considered as meeting the huge computational complexity of motion estimation in

real-time applications with a reasonable silicon area, high hardware utilization, minimized memory bandwidth, and low I/O bit width. There are different kinds of motion estimation hardware architectures in the literature such as one-dimensional (1-D), two-dimensional (2-D) systolic/semi-systolic arrays and adder tree architecture. Generally, these architectures are composed of the same parts called processing elements (PEs) where each PE computes the absolute difference between one pixel of the current block and one pixel of the search area. *'PE array is the trend of motion estimation architectures. Tradeoffs can be made between area (number of PEs) and throughput (processing capability), SAD latency (total cycles to compute a SAD) and memory bit width/bandwidth (serial/parallel loading), PE utilization and data alignment circuits (shift registers/memory with circular addressing), and bus bandwidth. The designers have to carefully select what can be sacrificed and what must be insisted on for target applications'(Tseng et al., 2005).*

2.3.2 State-of-the-Art Motion Estimation Architectures

Now we review six representative hardware architectures from 2004 to 2009 that belong to the most frequently cited works by the H.264 motion estimation designs (C.-Y. Chen, Chien et al., 2006; J. Kim & Park, 2009; Li & Leong, 2008; Lopez, Callico, Tobajas, Lopez, & Sarmiento, 2008; Ou, Le, & Hwang, 2005; Yap & McCanny, 2004).

- **Work of Yap and MaCcany (2004):** Yap and MaCcany proposed a 1-D systolic array architecture, as shown in Figure 2.3. It consists of 16 PEs and similar to conventional 1-D architecture (K. M. Yang, Sun, & Wu, 1989). Reference pixels are broadcasted to all PEs and right reference pixel for each PE is selected by control signal. Propagating register are used for propagating current pixel and the SAD of 4×4 blocks are stored in the PEs and are then reused for producing of

bigger block sizes. Due to small hardware cost, this architecture is suitable for area-constrained applications such as portable multimedia devices.

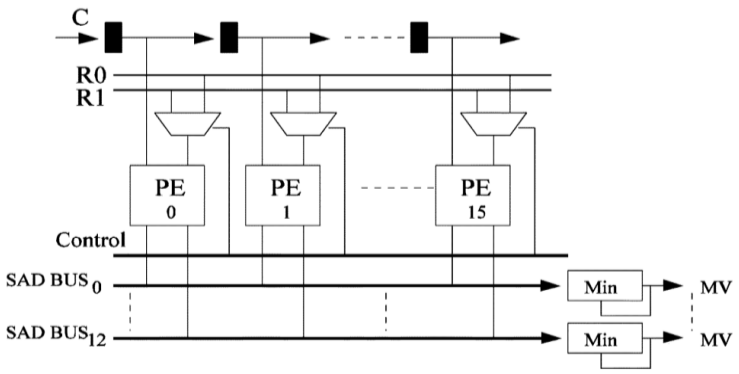


Figure 2.3 : 1-D Architecture of Yap and MaCcany.

Work of Uo et al. (2005): An efficient architecture for the H.264 motion estimation was implemented by Ou et al. This architecture consists of 16 SAD modules for 16 4×4 blocks and each module has four 1-D PE arrays. Figure 2.4 shows the structures of the SAD module and 1-D PE array where each 1-D PE array includes four PEs. As seen in this figure, each 1-D PE array includes four PEs and therefore there are 256 PEs in the architecture. The reference pixels are broadcasted into PEs and current pixels are propagating with propagation registers in a similar way to the work of Yap and MaCcany. However, because of using 256 PEs, the processing ability as well as the hardware cost of this architecture is much higher than the design of Yap and MaCcany.

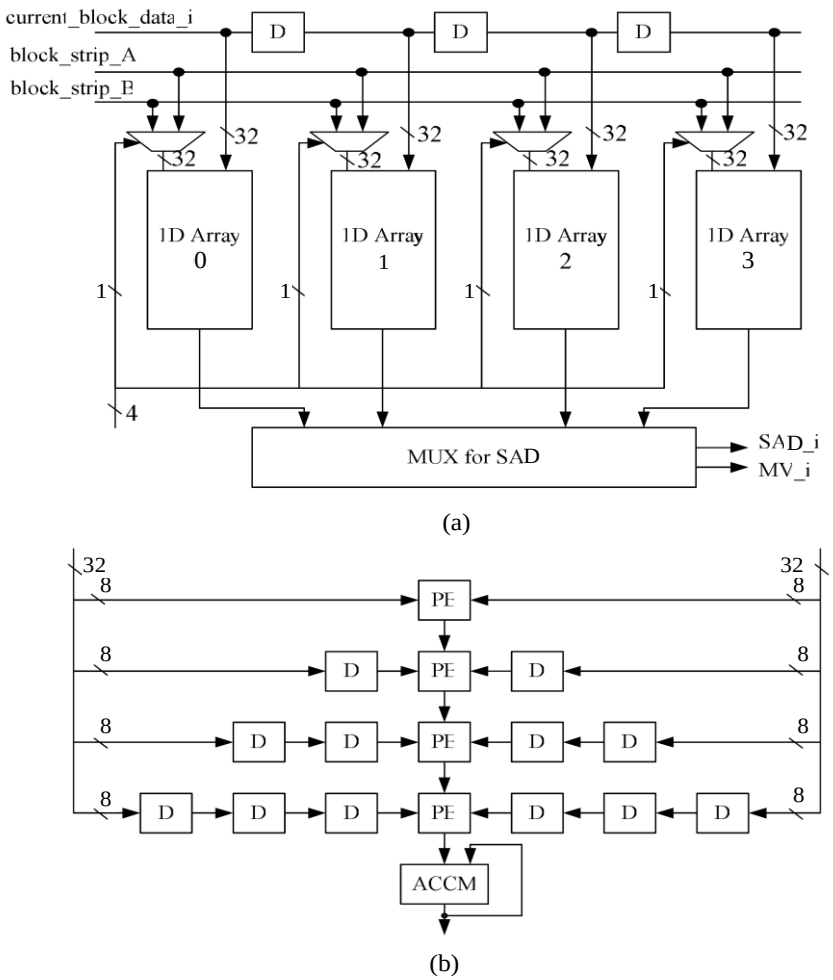


Figure 2.4 : The basic structure of the PE array in the work of Ou. et al. (a) structure of the PE array for the module i (b) structure of the 1-D array in the PE array.

- **Work of C.-Y. Chen et al. (2006):** C.-Y. Chen et al. proposed a 2-D PE array along with propagation registers and one 2-D adder tree, as shown in Figure 2.5. Propagation registers not only store the reference pixels but also enable data reusing by reconfiguration technique, which improve hardware utilization and memory bandwidth. The proposed 2-D adder tree consists of 16 2-D adder tree for generating 16 SADs of the smallest blocks, which are reused for calculating the SAD of bigger blocks.

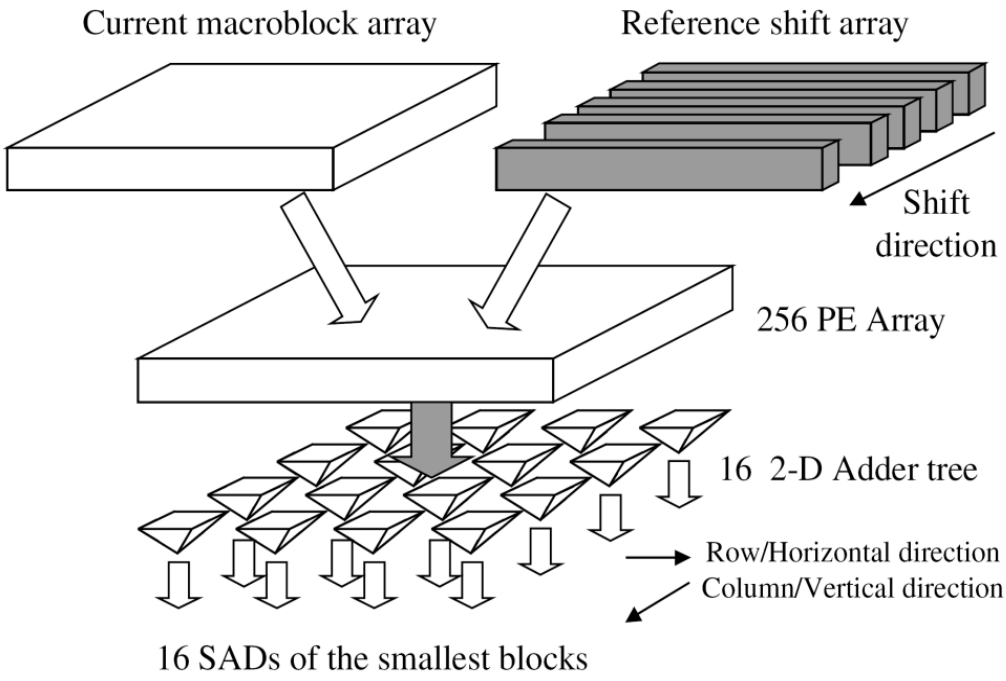


Figure 2.5 : C.-Y. Chen et al. architecture.

- **Work of Li and Leong (2008):** A low cost most significant bit (MSB) first bit-serial architecture VBS motion estimation was proposed by Li and Leong, as shown in Figure 2.6. This architecture consists of 16 4×4 SAD adder trees for computing the SADs of all 4×4 blocks and a SAD merger for producing the SAD of other bigger blocks. Due to the nature of MSB-first arithmetic, the authors employ the SAD early termination technique that enhances the average hardware performance. However, because of serial operations, the processing capability of the proposed architecture is not sufficient for medium or large resolutions.

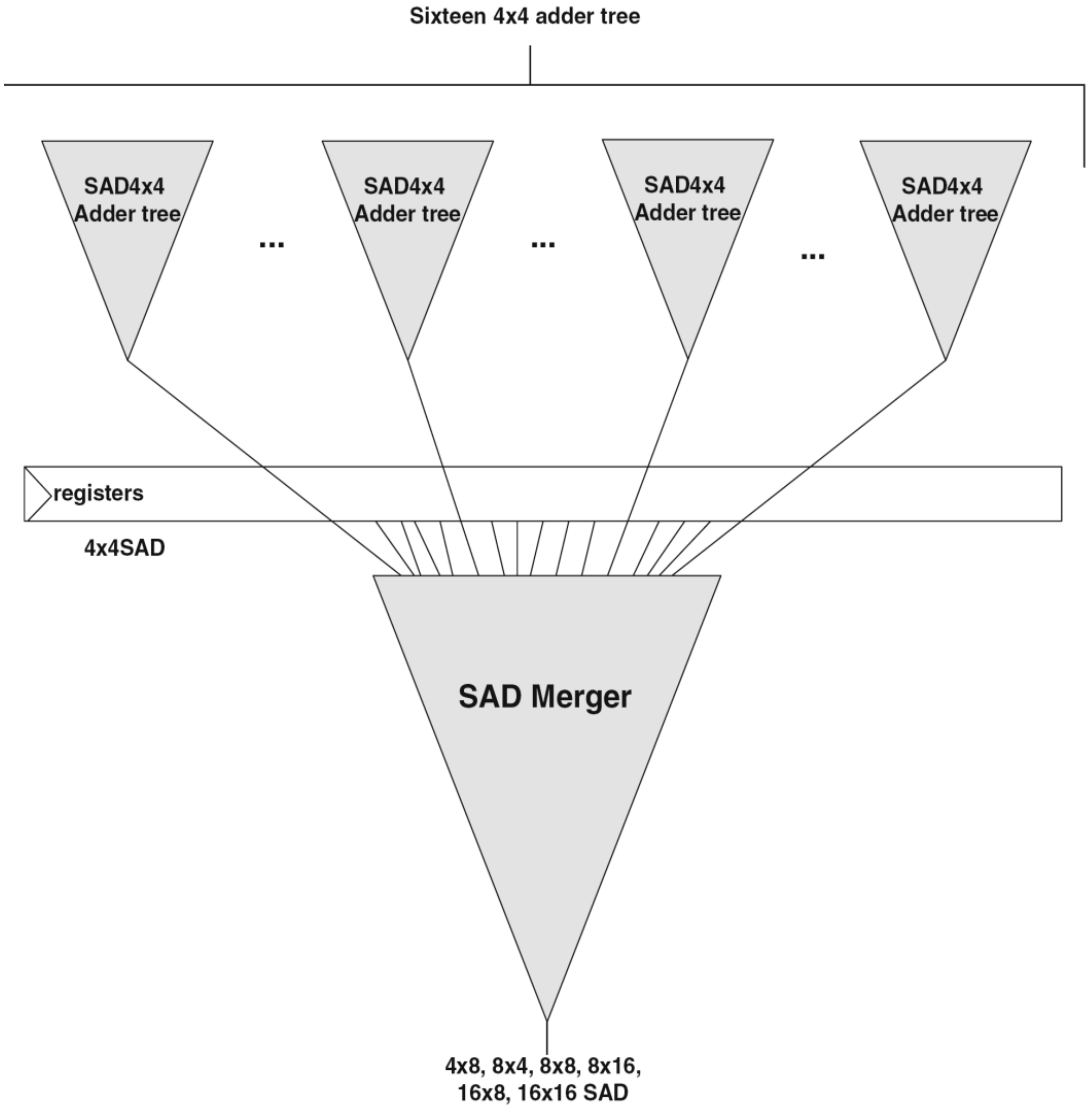


Figure 2.6 : Top level architecture of Li and Leong design.

- **Work of Lopez et al. (2008):** Lopez et al. contributed a new architectural template based on 1-D motion estimation arrays, as shown in Figure 2.7. The proposed template allows different allocation alternatives for the computational resources within motion estimation architectures. Their architecture consists of four independents groups with four PEs each that allows different design tradeoffs in terms of area cost and memory bandwidth.

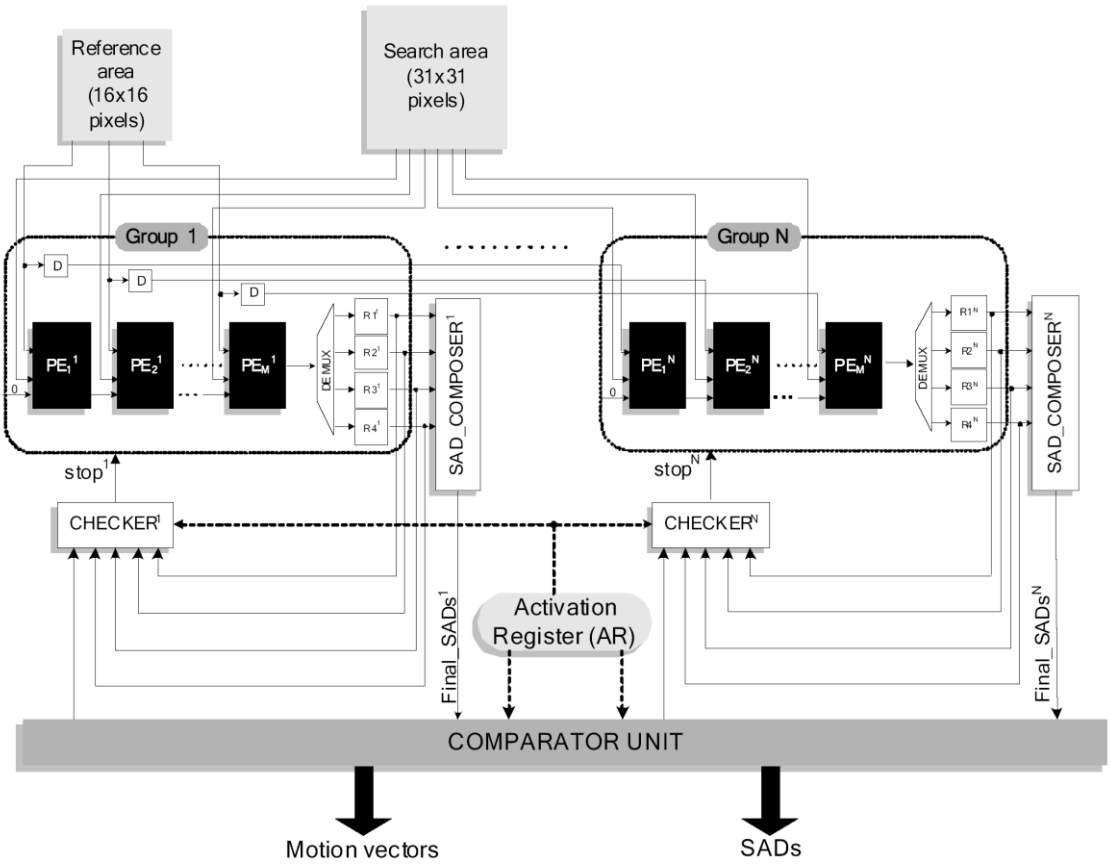


Figure 2.7 : The architectural template of Lopez et al.

- **Work of Kim and Park (2009):** A 1-D architecture was introduced by Kim and Park, as shown in Figure 2.8. This architecture has a similar structure to that of the conventional 1-D architecture with 16 PEs. The main idea in the architecture of Kim and Park is the scanning order of search candidates. While the raster scan is often adopted in motion estimation designs (Yap & McCanny, 2004), Kim and Park propose a new scan order that calculates the SAD values on "as-early-as-possible" basis, which means the new scan order makes the SAD values to be reused as early as possible. As a result, the number of required register for storing the SAD values is decreased and therefore the silicon area is saved.

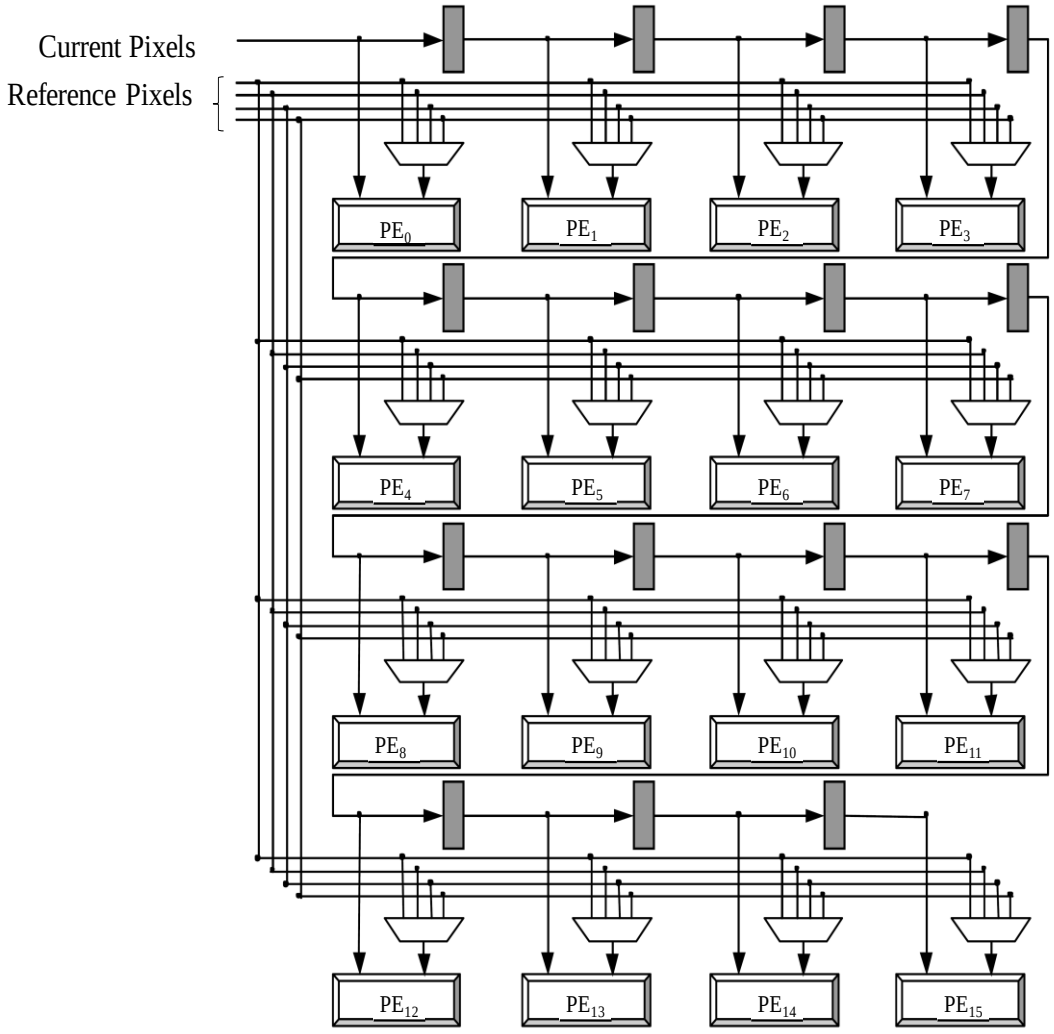


Figure 2.8 : Architecture of Kim and Park.

Table 2.1 summarizes the design metrics of the reviewed state-of-the-art motion estimation architectures for CIF video @ 30 fps, one reference frame, and a search range of [-16, 15] specifications. In Chapter 4, we will use this table as reference for evaluation and comparison of the performance of our designs relative to the surveyed architectures.

Table 2.1 : Design metrics evaluation of the reviewed motion estimation designs.

Architecture	Design Metrics	
Yap and MaCcany (2004)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	61
	Operating frequency (MHz)	294
	Throughput (MB/s)	17,944
	Memory bandwidth (Kbits/MB)	4,194
	Hardware utilization (%)	100
Ou et al. (2005)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	597
	Operating frequency (MHz)	200
	Throughput (MB/s)	195,312
	Memory bandwidth (Kbits/MB)	N/A[1]
	Hardware utilization (%)	100
C.-Y Chen et al. (2006)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	88.6
	Operating frequency (MHz)	110.8
	Throughput (MB/s)	106,250
	Memory bandwidth (Kbits/MB)	139[2]
	Hardware utilization (%)	100
Li and Leong (2008)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	55
	Operating frequency (MHz)	420
	Throughput (MB/s)	22,786
	Memory bandwidth (Kbits/MB)	4718
	Hardware utilization (%)	N/A
Lopez et al. (2008)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	33.41[3]/21.3[4]
	Operating frequency (MHz)	100
	Throughput (MB/s)	5,910
	Memory bandwidth (Kbits/MB)	541[5]/2,166[6]
	Hardware utilization (%)	100
Lim and Park (2009)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	39.2
	Operating frequency (MHz)	416
	Throughput (MB/s)	25,390
	Memory bandwidth (Kbits/MB)	4,194
	Hardware utilization (%)	100

[1] Not available
[2] Worse case
[3] "(4, 4) arch";
[4] "(16, 1) arch";
[5] "(4, 4) arch";
[6] "(16, 1) arch";

2.4 Summary

This chapter briefly explained the H.264 motion estimation as well as its block matching criteria. In addition, we introduced the design metrics for evaluation of ME designs including algorithm quality, silicon area, operating frequency, throughput, memory bandwidth, and hardware utilization. Besides, we explored the design space of motion estimation and reviewed six representative motion estimation architectures. Finally, we concluded this chapter by tabulating the performance of the surveyed motion estimation designs based on introduced design metrics, which will be used later as reference works for the evaluation of our designs.

Chapter 3

A Survey of Algorithms and Architectures for Sub-Pixel Motion Estimation in H.264/AVC

3.1 Introduction

Video coding standards make possible storage and transmission of high volume raw digital video data in limited storage and transmission bandwidth environments. The latest block based video coding standard, H.264, provides much better compression efficiency and subjective video quality compared with all the previous standards such as MPEG-2, H.263, and MPEG-4. For instance, compared with MPEG-4, H.263, and MPEG-2, H.264 can save 39%, 49%, and 64% of bit rate in average, respectively (Joch et al., 2002).

The higher compression performance and subjective quality in H.264/AVC are achieved by using new functionalities such as motion estimation with variable block-size and quarter-pixel accuracy, multiple reference frames (Wiegand, Zhang, & Girod, 1999), Lagrangian mode decision (Sullivan & Wiegand, 1998; Wiegand, Schwarz, Joch, Kossentini, & Sullivan, 2003), advanced entropy coding (Marpe, Schwarz, & Wiegand, 2003), and so on.

The heart of the H.264/AVC encoder is motion estimation that is used for removing of temporal redundancy in a sequence of images. In H.264/AVC, ME is conducted in two stages. In the first stage, integer motion estimation with different block sizes (*16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4*) is performed and up to 41 integer motion vectors (IMVs) of all blocks and sub-blocks are determined whereas up to 5

reference frames can be searched. In the second stage, SME is started which is usually conducted in two steps, as shown in Figure 3.1. In the first step, half-pixel refinement is performed by searching eight half-pixel positions around the best integer MV. Then, the quarter-pixel refinement is carried out in the same manner around the best half-pixel position. Finally, mode selection is conducted.

The H.264/AVC SME typically improves coding efficiency between 2-6 dB PSNR (Yi-Hau Chen, Chen, Chien, Huang, & Chen 2008). However, due to the use of VBS, MRF and interpolation scheme for producing sub-pixel values, two sub-pixel search steps and matching process, the computational budget and the memory access requirement of the quarter-pixel accurate ME are highly intensive. Therefore, not only efficient and fast algorithms are demanded to reduce the huge computation and memory access bandwidth of the H.264 SME but also dedicated hardware architectures are required to accelerate the SME process for real-time applications.

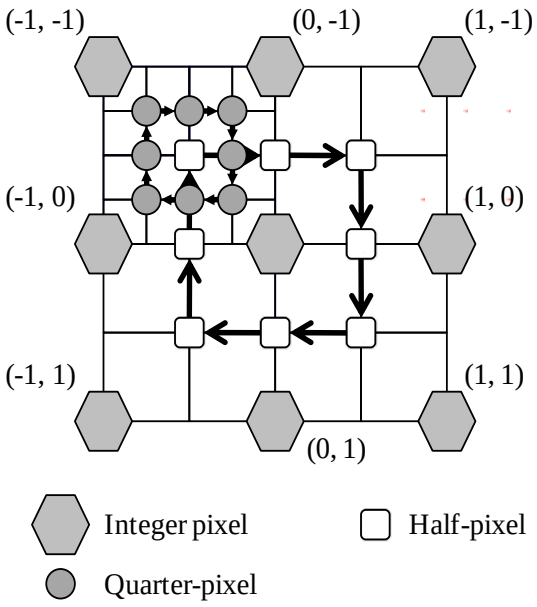


Figure 3.1 : SME refinements in H.264/AVC.

There are a few effective surveys on algorithms and architectures for video compression techniques that have already been published (C.-Y. Chen, Chien et al., 2006; Dufaux & Moscheni, 1995; Huang, Chen et al., 2006; Pirsch et al., 1995; Pirsch & Stolberg, 1998; Tseng et al., 2005). These surveys mainly focus on former standards such as MPEG-2 and MPEG-4 and they do not cover the new functionalities introduced later in H.264 such as SME with VBS, MRF, Lagrangian mode decision, and so forth. Motivated by the above issue, we try to provide a comprehensive survey about the most significant SME algorithms and architectures and address the SME design challenges and strategies at algorithmic and architectural levels.

The rest of this chapter is organized as follows. Section 3.2 analyzes the H.264 SME and shows the impact of its features on coding performance. Section 3.3 investigates the design space of SME algorithms reviewing recent approaches and the concepts behind them in the state-of-the-art SME algorithms. Section 3.4 explores the design challenges and choices of SME architectures and surveys the advanced SME architectures. In addition, the design metrics for evaluation of SME designs are reviewed and the evaluation of the reviewed SME architecture is provided. Finally, Section 3.5 provides a summary of this chapter.

3.2 Analysis of the H.264 SME

Design of a new and an efficient SME algorithm or hardware architecture relies on a careful analysis and an in-depth understanding of the SME features and characteristics. A thorough understanding of the SME features is essential for providing an insight into the design parameters (such as performance and subjective quality) and how to make an optimal tradeoff between them. In this section, we evaluate the performance of the H.264 SME algorithm and investigate the effect of its features on coding performance through some experiments.

Different technical features are used in the H.264 SME such as VBS, quarter-pixel resolution, and MRF, which improve the coding performance of the H.264 encoder at the price of increased computation and memory bandwidth. To evaluate the impact of different SME features on coding performance, we do several experiments. We use a set of tests consisting of several CIF and quarter CIF (QCIF) image sequences with the following conditions:

- Search range is 16.
- R-D optimization is on.
- The quantization parameter is 28.
- Entropy coding method is Context-based Adaptive Variable Length Coding (CAVLC).
- Frequency for encoded bit stream is 30.
- Number of coded pictures is 300 for all sequences and the H.264 reference software ("Joint Video Team Reference Software version 11," 2007) is used for experiments.

Tables 3.1-3.4 show gain in performance respecting to the reference conditions when extra features in the encoder are used for QCIF, CIF, 4CIF and HD videos, as described below.

(a) Integer motion estimation (IME) with 1 reference frame, SAD as distortion criteria, and only *16×16* blocks (i.e. reference conditions).

(b) Increasing motion vector accuracy to quarter-pixel.

(c) Allowing all VBS except sizes below *8×8* block.

(d) Allowing all VBS except *4×4* blocks.

(e) Using all VBS.

(f) Increasing number of reference frames to two.

(g) Using three reference frames.

(h) Using five reference frames.

(i) Changing the distortion criterion (i.e. replacing the SAD with the Sum of Absolute Transformed Differences (SATD)).

Table 3.1 : Effectiveness of the H.264 SME features on coding performances in QCIF videos.

Case	Container		Foreman		News	
	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR(dB)	ΔBR (%)	ΔPSNR(dB)
(a)	0	0	0	0	0	0
(b)	-35.20	1.654	-37.54	2.009	-20.39	1.339
(c)	-41.93	2.043	-47.51	2.788	-28.54	2.014
(d)	-42.88	2.106	-48.54	2.946	31.00	2.258
(e)	-43.20	2.113	-48.78	2.970	-31.23	2.283
(f)	-43.31	2.132	-49.92	3.188	-31.32	2.302
(g)	-45.96	2.276	-49.78	3.239	-31.62	2.324
(h)	-46.22	2.311	-50.75	3.334	-31.74	2.341
(i)	-46.76	2.355	-51.01	3.385	-32.41	2.422

Table 3.2 : Impact of the H.264 SME features on coding performances in CIF videos.

Case	Flower		Mobile		Mother-daughter	
	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR(dB)	ΔBR (%)	ΔPSNR(dB)
(a)	0	0	0	0	0	0
(b)	-47.27	2.523	-65.30	3.503	-39.69	1.958
(c)	-56.72	3.303	-69.19	3.920	-44.99	2.317
(d)	-58.20	3.479	-70.06	4.038	-45.73	2.381
(e)	-58.21	3.484	-69.99	4.043	-45.78	2.386
(f)	-59.04	3.610	-73.86	4.657	-46.42	2.449
(g)	-60.00	3.724	-74.84	4.910	-46.55	2.463
(h)	-60.87	3.845	-75.47	5.113	-46.96	2.498
(i)	-61.45	3.927	-76.39	5.229	-47.83	2.569

Table 3.3 : Effectiveness of the H.264 SME features on coding performance in 4CIF videos.

Case	City		Crew		Soccer	
	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR(dB)	ΔBR (%)	ΔPSNR(dB)
(a)	0	0	0	0	0	0
(b)	-57.74	3.703	-18.01	1.133	-36.39	1.606
(c)	-62.62	4.266	-25.96	1.770	-41.54	1.900
(d)	-63.24	4.355	-26.59	1.829	-42.31	1.949
(e)	-63.29	4.356	-26.55	1.827	-42.38	1.955
(f)	-65.37	4.832	-30.28	2.170	-42.90	1.998
(g)	-65.19	4.799	-30.89	2.235	-43.33	2.029
(h)	-65.59	4.880	-31.81	2.328	-43.83	2.061
(i)	-65.56	4.933	-32.70	2.438	-44.05	2.082

Table 3.4 : Impact of the H.264 features on coding performances in HD1080 videos.

Case	Blue_sky		Rush_hour		Station	
	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR(dB)	ΔBR (%)	ΔPSNR(dB)
(a)	0	0	0	0	0	0
(b)	-18.43	0.904	-11.72	0.340	-36.28	1.385
(c)	-23.38	1.207	-20.33	0.623	-40.71	1.636
(d)	-24.20	1.231	-20.65	0.636	-40.52	1.622
(e)	-24.31	1.239	-20.60	0.634	-40.74	1.642
(f)	-23.11	1.176	-23.53	0.743	-39.50	1.572
(g)	-23.23	1.182	-23.96	0.762	-39.29	1.565
(h)	-23.56	1.201	-25.45	0.819	-40.08	1.625
(i)	-23.84	1.223	-25.20	0.808	-37.87	1.500

In Tables 3.1-3.4, the average bit rate difference percentage (ΔBR(%)) and PSNR difference in terms of dB (ΔPSNR(dB)) between two cases are calculated (Bjntegaard, 2001), where case (a) is considered as the reference for other cases (i.e. case (b) to case

(i)). From the simulation results, we can see that when a new feature is added, better performance is achieved at the cost of extra computation, while the gain in performance is not equal for all video sequences and depends on video contents. From the simulation results, following observations can be made.

Quarter-pixel motion estimation provides considerable improvement over integer motion estimation in terms of bit rate saving and video quality.

As for VBS, the more varied the block sizes are, the better coding efficiency is. However, as seen in Tables 3.1 and 3.2, elimination of the 4×4 block will have negligible effect on coding performance whereas it leads to a lot of computation reduction. From the hardware point of view, it can result in many clock cycles reduction in the SME process. By using different VBS, a good tradeoff between the video quality and the required clock cycles can be achieved.

On the subject of MRF motion estimation, use of two reference frames usually brings a sensible coding gain compared with one reference frame. However, increasing the number of reference frames is not always advantageous. As shown in Tables 3.1-3.4 3.2, while the number of reference frames is increased to three or five, marginal coding performance improvement is achieved. However, the computational complexity and memory bandwidth are proportional to the number of searched reference frames. As a useful key point for designers, it is worth to mention that more than 80% of selected best reference frames is either one of the first two nearest reference frames (Huang, Hsieh, Chien, Ma, & Chen, 2006).

Regarding distortion cost, SATD has slightly higher video quality than SAD at the cost of increased computation. However, from the hardware point of view, SATD requires a higher hardware cost than SAD.

In summary, the SME hardware/software designers can make a good tradeoff between coding performance and computational complexity by using different SME functionalities and considering aforementioned observations.

3.3 Investigation of Algorithms

This section investigates the design space of SME algorithms and reviews the design challenges of SME algorithms and possible choices to cope with them. In addition, it surveys the state-of-the-art of SME algorithms.

To design a new and an efficient algorithm, designers should take into consideration several design issues. The new algorithm should have lower computational complexity and memory access compared with the H.264 SME. In addition, it should lead to similar coding performance compared to the H.264 SME algorithm. Moreover, it should be a hardware friendly algorithm for hardwired applications. This issue will be discussed in the next section. The designers should carefully consider all of the functions in SME algorithm and the impact of each function on the coding performance and computational complexity to identify the main bottlenecks and possible solutions. Although the proposed analysis in the previous section covers some of aforementioned issues, the details of SME bottlenecks and solutions are still not given, which will be discussed in the following paragraphs.

The VBS feature is one of the main contributors to the computation and memory bandwidth. Because in the H.264 SME each block has its own motion vector, and therefore it should process each block separately with multiple reference frames, which increases the computational complexity, memory bandwidth, and processing time. A simple but effective way for lowering computation, memory access and processing time is to reduce the number of block sizes before or during SME stage. For example, removing all block modes of size below 8×8 leads to 42.9% reduction in computation

complexity at price of small quality drop (Song et al., 2007). However, this technique is not suitable for low or medium frame size and results in more video quality degradation. In other studies (Abdelazim, Yang, & Grecos, 2009; Su et al., 2006), different mode filtering methods are introduced that select only some of the best modes for SME refinement. As a result, a lot of computation, memory access, and processing time are saved at the price of small video quality drop.

Interpolation is a computationally intensive process in SME, which requires a lot of memory access as well as computation to produce half- and quarter-pixels. Matching error surface function can provide a good way for reducing computation and memory bandwidth of the interpolation process. *'It is generally believed that the fast ME algorithm works best if the error surface inside the search window is unimodal. As shown in Figure 3.2, the error surface of the integer pixel ME is not unimodal due to the large search window and complexity of video content. Therefore, the IME search would easily be trapped into a local minimum. On the other hand, since the sub-pixels are generated from the interpolation of integer pixels, the correlation inside a sub-pixel search window is much higher than that of the integer pixel search window. Thus, the unimodal error surface will be valid in most cases of the sub-pixel. As a result, the matching error decreases monotonically as the search point moves closer to the global minimum'* (Y.-J. Wang, Cheng, & Chang, 2007).

Under unimodality assumption of error surface at sub-pixel resolution, the matching costs at sub-pixel precision can be calculated. Accordingly, the interpolation process is avoided and thus the computation and memory requirement are considerably saved. Several models for error surface function have been developed with different levels of accuracies and complexities. For instance, Suh and Jeong (2004) proposed five mathematical models of error surface with distinct performances. Among their models, the first and the second models are based on an inverse matrix solution that have better

coding performances, and are more suitable for hardware implementations. However, the above-motioned work only supports half-pixel accuracy. In a later study (Hill, Chiew, Bull, & Canagarajah, 2006), an algorithm has been proposed to support the quarter-pixel precision. It shows that the unimodal assumption does not necessarily work for all sub-pixels resolution. That is why the resulting performance of quarter-pixel interpolation free algorithm is less than the full search SME. To compensate the quality drop, the authors have introduced the "complete-system model," which provides near full search SME performance at the price of an increased complexity due to use of interpolation. However, this model is not a hardware friendly algorithm.

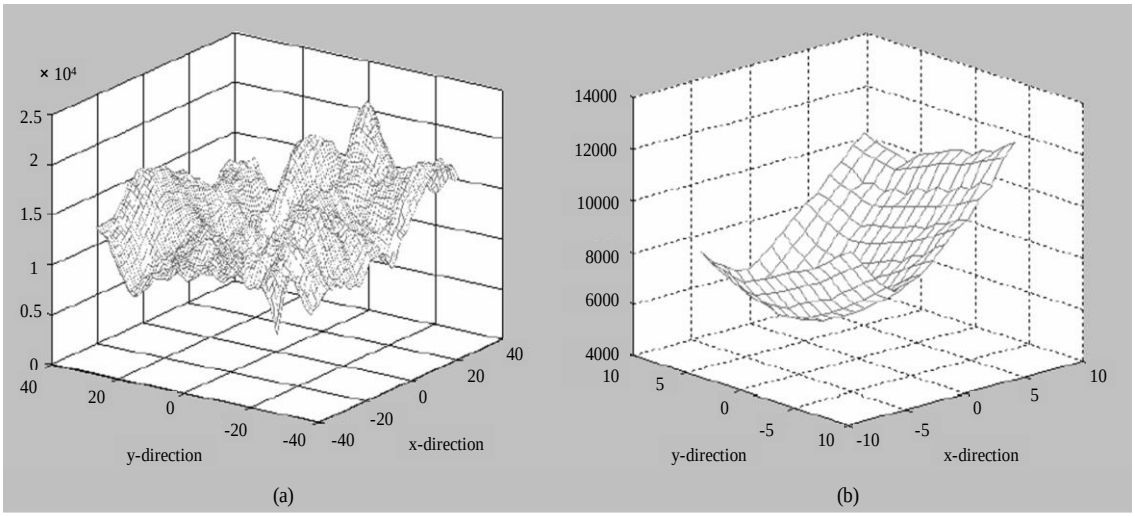


Figure 3.2 : Error surface of (a) integer motion estimation (search range: 32) (b) sub-pixel motion estimation with 1/8 pixel accuracy (Y.-J. Wang et al., 2007).

Another way for reducing the computation and memory requirement of the interpolation process is to use an interpolation filter with reduced complexity. H.264 uses a 6-tap finite impulse response (FIR) filter with (1/32, −5/32, 20/32, 20/32, −5/32 and 1/32) coefficients and a 2-tap bilinear filter with (1/2, 1/2) coefficients for producing half-pixel and quarter-pixel positions, respectively. On the other hand, the computation cost and memory requirement of a filter are proportional to its tap length.

Accordingly, the half-pixel interpolation is the bottleneck of the interpolation process in the H.264 SME. To address this problem, R. Wang, Huang, Li, and Shen (2004) proposed a 4-tap filter with (-1/8, 5/8, 5/8 and -1/8) coefficients which decreased the computation and the memory bandwidth compared with the standard 6-tap filter.

In addition to the tap length of an interpolation filter, the suitability of its coefficients for hardware/software implementation is another important issue. As seen in the above paragraph, some of the coefficients in the H.264 6-tap FIR filter and the proposed filter of R. Wang et al., (2004) are not a power of two. As a result, they need multiplication, addition, subtraction, and shift operations. Among these operations, multiplication is the most expensive one and demands more overhead for implementation. To cater to this issue, Tsai, Chen, Chen, and Chen (2005) proposed a multiplication free filter with 6-tap length. In a later study (Hyun & Sunwoo, 2009), both the tap-length and the multiplication issues were considered by introducing a multiplication free 4-tap filter that led to computation and memory bandwidth reduction.

Due to the existence of two-iterative search for the half-pixel and quarter-pixel refinements in the H.264 SME that are usually carried out in a sequential manner, the processing time is increased. From the hardware point of view, this problem poses a restriction on required clock cycles for high-resolution video such as HDTV and higher resolutions. To bring a solution for this problem, the single-iteration algorithm for HDTV has been proposed by T.-Y. Kuo, Lin, and Chang (2007). Compared with conventional two-iteration scheme, the cycle count in the single-iteration method is halved. Therefore, the long computation latency is reduced and HD size requirement can be achieved. In addition, the number of the search points is reduced to 6 points that leads to computation and memory bandwidth reduction. In this algorithm, the initial search center is determined by the motion information of neighboring blocks, which enhances the coding performance. In recent studies (Yi-Hau Chen, Tzu-Der Chuang et

al., 2008; Y.-K. Lin et al., 2008), the single-iteration technique is widely adapted as one of the effective methods to reduce the long processing time in high resolution specification as well as the computation and memory bandwidth.

Bit-width reduction is another choice for the complexity and memory bandwidth saving. The main idea is to represent each pixel with a lower number of bits (usually one or two bits) using bit transformation. As a result, SME can use bit transformed data and therefore the computation and memory access are reduced. Akbulut, Urhan, and Erturk (2006) proposed the first bit transformed SME. After the interpolation of reference frame with 8 bits/pixel bit-depth, they used 1-bit transform to represent each pixel with 1 bit. Then, the SME refinement was carried out as usual. However, they used 8-bit data at the interpolation process and their method added an extra complexity due to transforming 8-bit data to 1-bit samples using multi-band pass filter. Motivated from the above drawback, Celebi, Akbulut, Urhan, Hamzaoglu, and Erturk (2008) introduced all-binary SME algorithm. In order to reduce the computation of interpolation, 8-bit pixel values were transformed before the interpolation. Furthermore, by replacing the multi-band pass filter with multiplication free one-bit transform (Erturk, 2007), further computation was saved. However, for motion compensation it should be noted that residue data should be transformed back to pixel domain requiring additional computation.

3.4 Exploration of SME Architectures

In this section, the design challenges and strategies of the H.264 SME are discussed. Besides, representative SME hardware architectures are surveyed. In addition, reviewed SME architectures are evaluated based on designs metrics.

3.4.1 Hardware Architecture Design: Challenges and Strategies

Due to the huge computation and memory access requirement of the H.264 SME, acceleration of the SME process by dedicated hardware architectures is of vital importance for real-time application. Nevertheless, design of such architectures is a challenging job. The reason is that besides the extra ordinary huge computation and memory bandwidth, the SME algorithm in the H.264 reference software has been developed without architectural considerations. In the H.264 reference software, the SME is based on a sequential flow with a great deal of data dependency. From the hardware point of view, the sequential operations (processing) restrict the parallel processing, which in turn reduce the processing power. Besides, the data dependency increases the memory requirement at the price of an added storage space, area cost, and power consumption as well. Therefore, the data flow of the full search SME algorithm should be carefully reordered and optimized to permit concurrent operations with a reduced data dependency, which consecutively enables the parallel processing under the restrictions of sequential SME procedure. An example for analysis of the H.264 SME flow is the work of Yi-Hau Chen, Chen, Chien, Huang, and Chen (2008) where the authors simplify its complex encoding procedure into several encoding loops. In addition, they propose two decomposing techniques to parallelize the SME algorithm while maintaining high hardware utilization and achieving data reuse.

The main building blocks of the H.264 SME consists of three parts i.e., interpolation unit, processing unit that is responsible for search and cost calculation, and mode decision. Among them, the interpolation unit is computationally intensive while processing and mode decision units demand less computation. Therefore, design of hardware architecture is performed by a particular emphasis on interpolation unit as the most computational intensive unit. However, design of efficient hardware architectures

for less computational demanding units is still important as they can affect the overall performance of the design.

The main design challenges for the interpolation unit are hardware utilization and huge memory bandwidth that arise from the VBS, and the 6-tap FIR filter that is used for generation of half-pixels. Due to the seven different block sizes, the interpolation unit should be carefully designed to achieve maximum hardware utilization. The common solution for this issue is the use of 4×4 interpolation unit. As all bigger blocks can be decomposed into 4×4 blocks, the complete hardware utilization can be achieved. However, since each 4×4 block needs a 9×9 interpolation window, the 4×4 based interpolation increases the memory bandwidth. On a contrary way, the use of a 16×16 interpolation unit leads to the lowest memory bandwidth, which is due to the covering of all block sizes, at the price of low hardware utilization. Examples that are based on 4×4 and 16×16 interpolations are the designs of T.-C. Chen, Huang, and Chen (2004) and C. Yang, Goto, and Ikenaga (2006), respectively. As for the FIR filter, the previous reported papers in the literature (Park, Muhammad, & Roy, 2003; Samueli, 1988) are not suitable for the H.264 interpolation unit because of their too low input bandwidth (Lei, Wen, Zeng, & Ji, 2004). In another work (Song, Liu, Goto, & Ikenaga, 2005), an architecture for the H.264 interpolation unit is presented that has a 6-tap FIR filter with a pipelined architecture, which increases its processing ability. In a recent paper (Lu, McCanny, & Sezer, 2009), by exploiting the mixed use of parallel and serial-input FIR filters, a high throughput rate and efficient silicon utilization are achieved so that the proposed design can support SDTV and HDTV applications.

On the subject of less computationally demanding units, i.e. absolute operation, SATD and inter mode decision, the discussion on this subject is not provided here. However, the interested readers are referred to the following studies for more information. Vanne, Aho, Hamalainen, and Kuusilinna (2006) and T.-C. Wang, Huang,

Fang, and Chen (2003) proposed efficient architectures for absolute difference operation and SATD function, respectively. As for the inter mode decision, it is difficult to find an architecture design with details in literature. However, the interested readers are referred to the work of H.-C. Kuo (2006) for a detailed example.

In addition to the hardware architectures for the SME building blocks, efficient memory design is another important issue. The reason is that the SME uses the data-intensive interpolation method to generate sub-pixels within up to five reference frames that calls for a great deal of data access through the frame memory. This data access is a slow process and in addition, it leads to a lot of power consumption and decreases the overall performance of the design. The common solution for this problem is data reusing technique (Soudris et al., 2000; Wuytack, Catthoor, Nachtergaele, & De Man, 1996; Wuytack, Diguet, Catthoor, & De Man, 1998). In this technique, the temporal data locality is exploited and the previous accessed data is reused in future, which result in data access reduction. Accordingly, the power consumption is decreased. The data reuse method can be performed offline or online. In the offline scenario, frequently accessed data is stored in on-chip memory and will be recycled offline in future access. In this way, the data access from the frame memory can be decreased. In the online data reuse scheme, the current accessed data is shared between different places that work in parallel and therefore data access reduction is achieved. Y.-H. Chen, Chen, Tsai, Tsai, and Chen, (2008) introduced an effective data reuse methodology, which formulated the SME algorithm as the nested loops structures and explored data locality by the use of loop analysis. Based on their technique, the amount of memory access is significantly reduced that leads to a lot of saving in memory access power. Haihua, Zhiqi, and Guanghua (2007) have proposed a hardware implementation for interpolation unit that uses single-step interpolation method along with data reusing scheme, which can save the memory bandwidth, and processing cycles.

3.4.2 Hardware Architectures of the H.264 SME

This sub-section surveys the state-of-the-art SME architectures and classifies them into full and fast SME architectures. A full SME architecture is based on the full search SME algorithm whereas a fast SME architecture is a hardware design of a fast SME algorithm. Three full SME architectures (T.-C. Chen et al., 2004; Wu, Kao, & Lin, 2010; C. Yang et al., 2006) and four fast search SME architectures (T.-Y. Kuo et al., 2007; Song et al., 2007; Tsung, Chen, Ding, Tsai et al., 2009; Y.-J. Wang et al., 2007) are selected as the representative designs. The details as well as the comparisons of which are provided in the following paragraphs.

T.-C. Chen et al. (2004) proposed the first H.264 SME architecture. To bring a solution for the complex sequential procedure flow of the H.264 SME, they presented the 4×4 block decomposition and vertical integration techniques to enable the mapping of the SME flow in hardware with features of regular schedule, full utilization, parallel processing, and reusability. As shown in Figure 3.3, the main important parts of the proposed architecture are interpolation engine, nine 4×4 processing units, and Lagrangian mode decision engine. The interpolation engine produces half-pixels and quarter-pixels by using 1-D FIR and bilinear filters, respectively. The 4×4 processing units generate residues and SATD values. Since there are nine positions in each sub-pixel refinement, nine 4×4 block processing units are used. The "Lagrangian mode-decision" is responsible for finding the best mode among 41 modes in each macroblock. The area cost of the proposed design is 79 K gates and it can support SDTV format with 30 fps, one reference frame and all MB modes at the clock frequency of 100 MHz. The maximum performance of this design is 52 K MB/s.

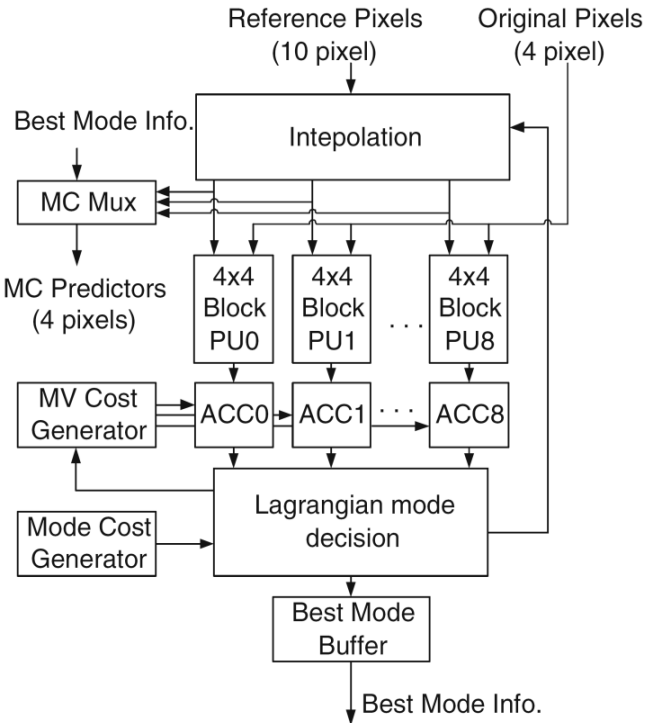


Figure 3.3 : Block diagram of Y.-C. Chen's design.

Based on the work of Y.-C. Chen et al. (2004), Yang et al. (2006) introduced a high performance architecture of the H.264 SME for HDTV application with two new techniques. Firstly, while Y.-C. Chen et al. (2004) used 4-pixel interpolation unit, they proposed a 16-pixel interpolation to increase processing capability. However, four times increase of the data width of the interpolation and the search engine only brought 52.5% of clock cycle saving. Secondly, they replaced 1-D filters in design of Y.-C. Chen el al. with diagonal filter and fully pipelined 1-D filters in the interpolation unit. As a result, the delay path was decreased and higher clock frequency was achieved without a considerable increase of area cost. The clock frequency of the design of Yang et al. is 285 MHz that supports HD1080p format in 30 fps for one reference frame. The area cost and the maximum throughput of the proposed design are 189 K gates and 250K MB/s.

Wu et al. (2010) presented a three-engine parallel SME architecture to support high-resolution application. To cope with the high bandwidth input data problem, which led

to high bandwidth memory and a high hardware cost, they used a reference pixel scheduling method and an efficient memory organization to ensure that each engine can fetch the right reference pixel at the right time with minimum hardware cost. In addition, they dispatched 41 sub-blocks of each MB to the three engines in a load-balancing and interlaced ways to increase the throughput. Furthermore, they introduced a resource-sharing scheme for SATD generators to reduce the number of the required SATD generators from three units to two units. Their proposed SME architecture takes 311.7 K gates and can support HD1080 30fps at clock frequency of 154MHz.

Based on the work of Y.-C. Chen et al. (2004), Y.-J. Wang et al. (2007) contributed a fast SME architecture. Their architecture was a hardware realization of a fast algorithm with reduced number of search points. As a result, the number of 4×4 processing units decreased to five instead of nine in design of Y.-C. Chen et al. Compared with the work of Y.-C. Chen et al., it saves 40% and 14% of area cost and searching time at the price of less than 0.2 dB PSNR quality drop. However, due to the use of the 4-pixel interpolation units and 1-D FIR filters, it has a limited power processing and can only support SDTV format.

Song et al. (2007) proposed a fractional motion estimation engine for HD1080p resolution. To achieve real-time encoding, they presented three techniques. Firstly, they removed all modes below 8×8 block and supported one reference frame. Therefore, 88.6% of the computations were saved with 0.1 dB loss. Secondly, they used two reusing methods, namely lossless inside-mode and cross-mode (Shao, Liu, Goto, & Ikenaga, 2007), which saved about 65% pixel generation and SATD calculation. Thirdly, they removed the pipeline bubbles between the half-pixel and the quarter-pixel stages by optimization the SME scheduling. The area cost of the proposed engine is 203.2 K gates and it can encode real-time HDTV1080p at 30 fps under the clock frequency of 200 MHz.

T.-Y. Kuo et al. (2007) presented an SME architecture for HDTV and high profile. They proposed two techniques for solving the associated problems with HDTV video size and high profile. Firstly, the proposed architecture was based on the single-iteration algorithm. Consequently, its processing time was halved compared with conventional two-step full search algorithms. In addition, the number of its search candidates was reduced to six points that resulted in reduction of processing units and hardware cost. Secondly, since the high profile of H.264 uses the costly 8×8 SATD, they replaced it with 4×4 SATD leading to silicon area reduction. The proposed architecture supports HD720p 30 fps with a clock frequency of 70 MHz, 62.2 K gates, and 0.043 dB PSNR video quality degradation.

In a later study (Tsung, Chen, Ding, Tsai et al., 2009), an SME architecture for quad full high definition (QFHD) was introduced that was based on the single-iteration algorithm. To cater to the huge computation and memory bandwidth requirement of the QFHD specification, several optimization techniques were used. Firstly, the 6-tap interpolation filter was replaced by a bilinear filter and thus memory bandwidth and area cost were saved. Secondly, based on the distributive law in SATD, the residues in the quarter-pixel SME were estimated while the half-pixel residues were known. Therefore, the calculations of sub-pixels were decreased. Thirdly, a cache-based memory (Tsung, Chen, Ding, Chien, & Chen, 2009) was used that resulted in bandwidth reduction. The clock frequency and throughput of the proposed design are 280 MHz and 1659 K MB/s, which are sufficient for the 4096×2160 QFHD real-time processing.

3.4.3 Design Metrics and Evaluation of the Reviewed Architectures

The illustrated design metrics for evaluation of the H.264 IME designs in Chapter 2 can be used for evaluation of the H.264 SME designs as well. These design metrics are algorithm quality, silicon area, operating frequency, throughput, memory bandwidth, and hardware utilization. However, although the memory bandwidth and hardware utilization are two important evaluation metrics, none of the reviewed SME architectures provides any data regarding these two parameters except the work of Y.-C Chen et al. (2004), which only gives the details of hardware utilization.

Therefore, since these two metrics are not available for the surveyed architectures in the literature, we only use four design metrics for the evaluation of the SME designs including algorithm quality, silicon area, operating frequency, and throughput. Please note that for a fair evaluation of the throughput in SME architectures, we should consider the number of modes that is supported by each architecture. The reason is that the number of block modes influences the throughput of SME architectures, which is due to sequential processing of block modes in SME architectures.

Table 3.3 provides an evaluation of the aforementioned SME architectures based on the design metrics. From this table, three trends can be summarized as follows. First, under similar throughput condition, a full SME architecture has a higher area cost than a fast SME architecture with the benefit of the highest video quality. Second, fast SME architecture designs are generally preferred due to stringent real-time constrains. Finally, the single-iteration based fast SME architectures (T.-Y. Kuo et al., 2007; Tsung, Chen, Ding, Tsai et al., 2009) are suitable candidates for using in high-resolution application with a negligible quality drop and reasonable area cost.

Table 3.5 : Design metrics evaluation of the reviewed H.264 SME architectures designs.

Architecture	Design Metrics	
Y.-C. Chen et al. (2004)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	79.3
	Operating frequency (MHz)	100
	Throughput (MB/s)	49 K[7]
C. Yang et al. (2006)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	189
	Operating frequency (MHz)	200
	Throughput (MB/s)	250 K[8]
Wu et al. (2010)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	311.7
	Operating frequency (MHz)	154
	Throughput (MB/s)	250 k[9]
Y.-J Wang et al. (2007)	Quality loss (dB PSNR)	-0.1 to -0.2
	Silicon area (K gates)	48
	Operating frequency (MHz)	100
	Throughput (MB/s)	50 K[10]
Song et al. (2007)	Quality loss (dB PSNR)	-0.1
	Silicon area (K gates)	203.2
	Operating frequency (MHz)	200
	Throughput (MB/s)	250 K[11]
T.-Y. Kuo et al. (2007)	Quality loss (dB PSNR)	-0.043
	Silicon area (K gates)	62.2
	Operating frequency (MHz)	70
	Throughput (MB/s)	71.3 K[12]
Tsung, Chen, Ding, Tsai et al et al. (2009)	Quality loss (dB PSNR)	-0.02
	Silicon area (K gates)	448
	Operating frequency (MHz)	280
	Throughput (MB/s)	830 K[13]

[7] All block modes and for SDTV resolution.

[8] All block modes and for HD1089 resolution.

[9] All block modes and for HD1080 specification.

[10] All block modes and for SDTV resolution.

[11] 8×8 and above modes and for HD1080 resolution.

[12] All block modes and for HD720 resolution.

[13] Number of supported modes is not available, QFHD resolution, and 24 fps.

3.5 Summary

This chapter reviewed recent state-of-the-art H.264 SME algorithms and architectures. First, the H.264 SME was analyzed and the impact of its functionalities on coding performance was investigated. Then, design space of SME algorithms was explored representing design problems, approaches, and recent advanced algorithms. Besides, design challenges and strategies of SME hardware architectures were discussed and promising architectures were surveyed. Finally, design metrics as well as evaluation of the surveyed SME architectures were given.

Chapter 4

Analysis and Design of the Proposed Low-Cost Bit-Serial Architecture for Integer Motion Estimation in H.264/AVC

4.1 Introduction

H.264/AVC is the latest video coding standard that outperforms all of previous standards in terms of coding efficiency. Compared with previous video standards, H.264/AVC can provide up to 50% coding gains on different bit rates and video resolutions (Wiegand, Sullivan et al., 2003). To achieve a higher coding performance and better subjective visual quality, H.264 uses many new features such as variable block-size motion estimation (VBSME), multiple reference frames, in-the-loop deblocking filtering, weighted prediction, and so on.

H.264/AVC uses variable block-size motion estimation, which consists of seven block sizes (i.e. 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 block sizes). Compared with traditional fixed block-size motion estimation, variable block-size motion estimation results in a more accurate prediction that enhances the coding performance. Although the coding performance of variable block-size motion estimation is higher than traditional fixed block-size motion estimation, it requires more computation and its hardware implementation is more difficult than a fixed block-size hardwired motion estimator. The instruction profile of an H.264/AVC encoder shows that the computational load of a CIF video with 30 fps is 315 GIPS where integer variable

block-size motion estimation is computationally most demanding part of it, which amounts to 74.29% (234 GIPS) computation of the encoder (Song, Liu, Ikenaga, & Goto, 2006). Therefore, acceleration of integer motion estimation by hardware architectures is required for real-time application.

To meet the real-time constraints of the H.264/AVC IME, several architectures have been proposed, most of which use the full search algorithm due to its regularity, simple control unit, and coding performance. Generally, IME architectures are based on 1-D (J. Kim & Park, 2009; Lopez et al., 2008; Yap & McCanny, 2004), 2-D (Ou et al., 2005) systolic or semi-systolic and adder tree (C.-Y. Chen, Chien et al., 2006) architectures. Most of these architectures use bit-parallel data path to increase their processing ability at the price of an extra silicon area and I/O bit width. Conventionally, 1-D architectures are used for area-constrained application such as mobile phones because of their small hardware cost and power consumption. However, 1-D designs have a lower processing ability than 2-D architectures, and thus may not be suitable for medium and large resolutions. Therefore, for current and future area-constrained applications such as portable multimedia devices with greater resolution toward high definition, new low cost IME architectures with higher processing capability are required.

Bit-serial architectures can be a solution for the above-mentioned problem. In bit-serial approach, each bit of a word with n-bit length is processed at a time. As a result, all of the bits are passed through the same logic leading to a significant reduction in area and pin count. In addition, due to the small path delay between registers in bit-serial architectures, they can operate at a higher clock frequency, and thus achieve a higher throughput.

Li and Long (2008) have proposed an MSB bit-serial architecture for the H.264/AVC IME with small area cost. This architecture benefits from an early termination scheme for calculating SAD to enhance its performance. However, the performance of the

proposed architecture is not sufficient for medium or high resolutions. In addition, compared with the least significant bit (LSB) arithmetic, MSB scheme has longer latency and needs extra hardware to convert its data into redundant representation.

In this chapter, we introduce two 2-D systolic array LSB bit-serial IME architectures, which are suitable for area restrictive portable systems. Both designs are based on FS algorithm and can support VBSME for video with different resolutions. The first design uses a 2-D systolic array along with 1-D data broadcasting and partial result propagating techniques. The second design has a 2-D bit-serial adder tree together with a reconfigurable reference buffer, which can take advantage of the data reusing between successive search candidates. As a result, it is an appropriate choice for higher resolutions when using hardware parallelism. To decrease the computational load as well as memory bandwidth and to increase the overall performance of the proposed designs, several optimization techniques have been applied. By using pixel truncation scheme and proposing the word length reduction technique, not only the required cycles for computing the SAD of one search position is reduced from 16 cycles to 5 cycles (i.e. 67.75% reduction) but also the silicon areas, power consumption, memory bandwidth and latencies of the proposed design are decreased. In addition, to reduce the area cost, computational complexity, memory bandwidth, and power consumption further, three techniques have been used as follows: mode filtering, 1/2-subsampling and a power reduction method for the reconfigurable reference buffer.

The rest of this chapter is organized as follows. In Section 4.2, advantages and challenges of bit-serial architectures are explored. Section 4.3 describes the proposed bit-serial architectures in details. In Section 4.4, the proposed optimization techniques are discussed showing the performance improvement caused by them. Section 4.5 gives the experimental results and provides a comparison with the previous architectures. Finally, Section 4.6 gives a summary of this chapter.

4.2 Bit-Serial Approach: Advantages and Challenges

As described in introduction, bit-serial architectures result in lower area cost and pin counts compared with bit-parallel architectures, which in turn lead to smaller package size and thus lower production cost. In addition, bit widths of signals in bit-serial architectures are one bit and often to one destination, which simplify the routing and reduce propagation delay and interconnection density. However, due to the spreading of operations and data over the time, design of bit-serial architectures are often more complicated than their equivalent bit-parallel architectures, especially in complicated designs. Besides, in the bit-serial scheme, a higher clock frequency can be achieved, which increases the processing ability. Furthermore, small interconnection density and area cost of bit-serial architectures can be advantageous for VLSI designs with deep submicron technologies.

In deep sub micron regime and beyond, as the scale of process technologies steadily shrinks, the interconnect (i.e. the interconnection parameter) will be determining cost, delay, power, and reliability of the future LSI designs (Sakurai, 2000). Therefore, a lower interconnection density could improve the overall performance of the designs.

From the power consumption viewpoint, the bit-serial operation has its own advantages and drawbacks. The total power consumption of a CMOS circuit can be modeled as:

$$P_{total} = C.V_{DD}^2.F + I_{leakage}.V_{DD} \qquad (4.1)$$

where the first term ($C.V_{DD}^2.F$) denotes the dynamic power and the second ($I_{leakage}V_{DD}$) represents the leakage power. Since bit-serial architectures are usually working at a higher clock frequency (i.e. F), their dynamic power is higher. On the other hand, the total capacitance load (i.e. C) of a bit-serial architecture, which is

approximately proportional to the area cost, is less than a bit-parallel design, which decreases the dynamic power. However, note that a bit-serial architecture generally consumes higher dynamic power than its bit-parallel counterpart, as will be explained later in the last paragraph of this section.

Regarding the leakage power, which is proportional to the number of transistors (i.e. area cost), a smaller circuit results in lower leakage power. Therefore, the bit-serial scheme demonstrates lower leakage power than its bit-parallel matching part. Lower leakage power may be beneficial to portable multimedia devices and electronic consumer systems that are often in standby mode, where only the leakage power is consumed. Considering the size of process technology, as it steadily shrinks, the leakage current is becoming higher and will be a major contributor to total power consumption. Therefore, for recent and future VSLI designs with deep submicron technology size, the use of bit-serial architecture, which has small silicon area, can provide lower leakage power compared with bit-parallel architecture. Interested readers are referred to some excellent papers (Elgharbawy & Bayoumi, 2005; N. S. Kim et al., 2003; Sakurai, 2000) for thorough and in depth information on leakage power.

One of the major problems associated with the bit-serial approach is its long processing time. For example, a bit-serial full adder (FA) requires n clock cycles to add two n-bit inputs, whereas the counterpart bit-parallel architecture with n-bit data path takes one clock cycle. The number of the required clock cycles in bit-serial arithmetic can be even bigger than the bit length of the input operands. Actually, the bit length of the output result determines the required clock cycles in bit-serial structure. For instance, consider the 8-bit input pixel data in the IME process. Since there are 256 pixels in an MB, the maximum final SAD value may have up to 16 bits. Therefore, the required cycles for calculating a SAD will increase to 16 cycles instead of 8 cycles.

From the hardware point of view, this problem can increase the hardware cost and power consumption too.

Mostly, there are many serial shift registers inside bit-serial architectures working at a high clock frequency. At each clock cycle, all of the data are shifting inside the shift register consuming a significant part of total dynamic power (Tan, Eriksson, & Wanhammar, 1994). This problem is another drawback of bit-serial architectures especially for using them in low power applications. In the next sections, we describe our two bit-serial architectures providing solutions for the associated problems with the bit-serial approach.

4.3 The Proposed Bit-Serial Architectures

4.3.1 Hardware Architecture of the First Design

Figure 4.1 shows the 2-D structure of our first design, which consists of serial 4×4 processing units (PUs), 1-D serial adder trees, and serial propagation registers. Our architectures is featured with reusing of partial SAD results and broadcasting of reference pixel data techniques in a similar way employed by partial propagate SAD IME designs (Huang, Wang, Hsieh, & Chen, 2003; Liu, Huang, Song, Goto, & Ikenaga, 2007). The proposed design utilizes 16 4×4 PUs for calculating the SADs of 4×4 blocks that these partial SADs are reused to generate the SAD of bigger blocks (i.e. 4×8, 8×4, 8×8, 8×16, 16×8, and 16×16). Each 4×4 PU has four rows of serial processing elements (PEs), four 1-D serial adder trees and serial full adders (FAs), and propagation serial registers, as shown in Figure 4.2. Serial PEs are responsible for calculating the absolute difference between current MB pixels and search area pixels. 1-D serial adder trees add the absolute differences of rows where the sum result of each row is propagating down and adds with sum result of next row using a serial FA.

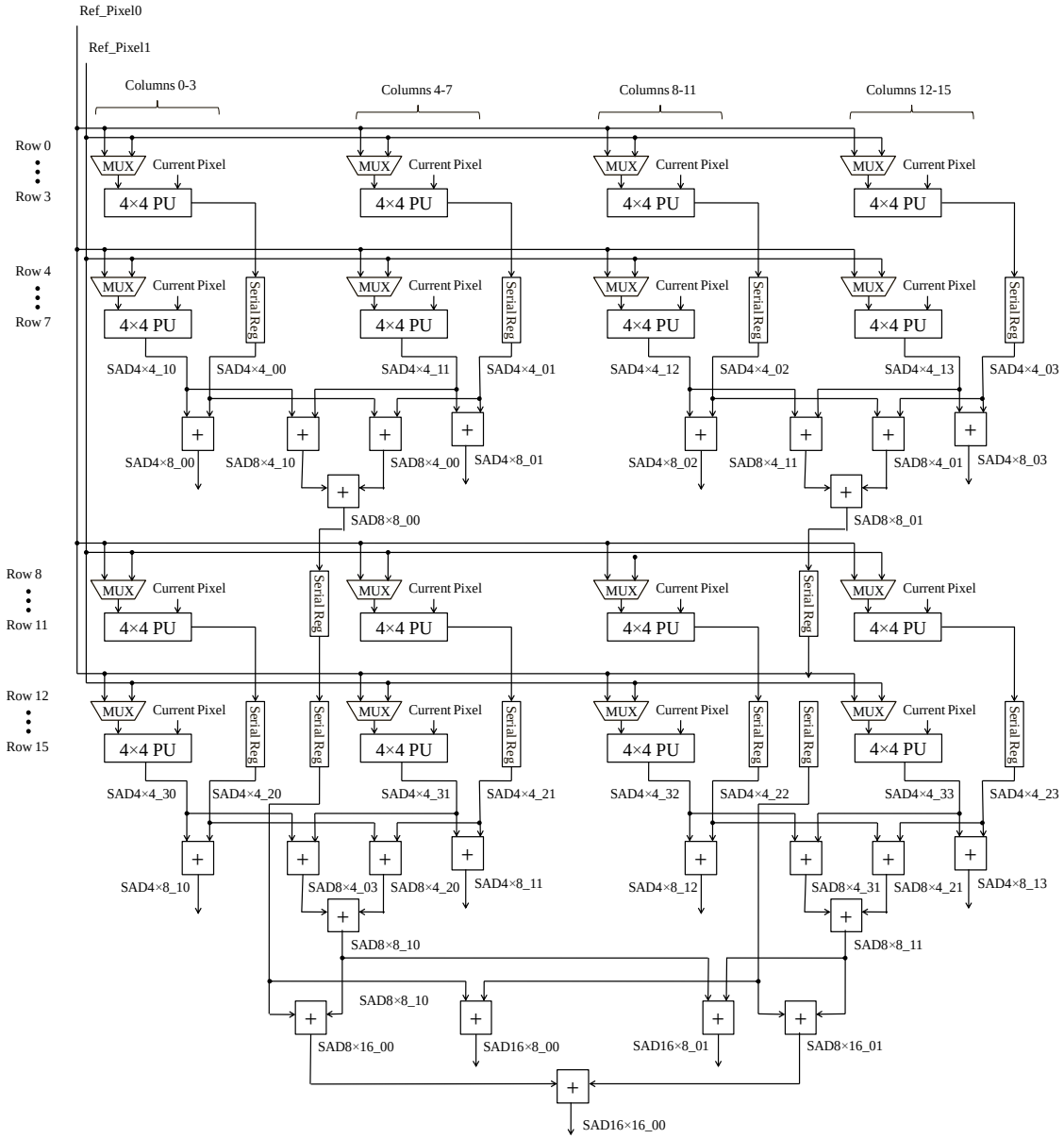


Figure 4.1 : Our proposed 2-D bit-serial architecture.

There are 256 serial PEs in our architecture, which have great effect on its performance. Therefore, design of an efficient serial PE architecture is important. To address this issue, we present a new serial PE architecture, which is based on an efficient absolute difference algorithm (Vanne et al., 2006). In this algorithm, the absolute difference operation is expressed as:

$$|R - C| = \begin{cases} 1 + R + C', & R > C \\ (R + C')', & R \le C \end{cases} \qquad (4.2)$$

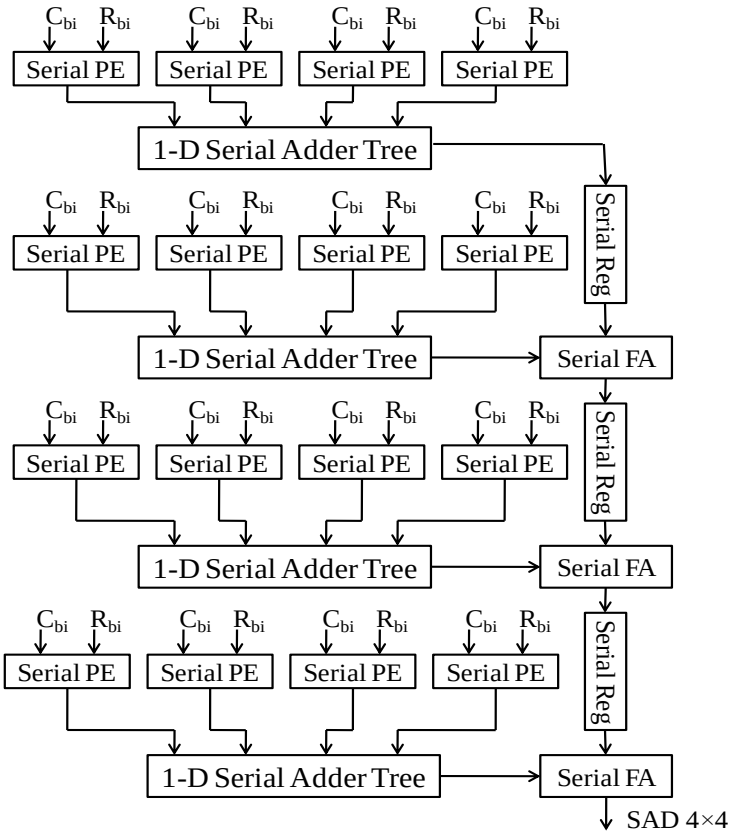


Figure 4.2 : Hardware architecture of the 4×4 PU.

where *R* indicates the reference pixel and *C* represents the current pixel.

The proposed bit-serial architecture of this algorithm is shown in Figure 4.3. At each clock cycle, one bit of the current and reference pixels are inputted starting from LSB. As a result, their difference (i.e. *R*-*C*) is generated in n clock cycles where n is the bit length of *R* and *C*. At the first cycle of a new operation, an initial signal (*ini1*) is used to set the carry-in (C_i) to zero leading to a correct result. In the proposed architecture, the n^{th} carry-out (C_o) bit is latched in a D flip flop (DFF) and its inverse is used to XOR with the *R*-*C* bits that are stored in an n-bit serial DFF. In addition, the nth C_o has to sum with the XORed bits to produce the correct absolute difference using serial FA.

The sum operation can be eliminated to save the hardware cost of 256 serial FAs of PEs, which can lead to maximum error of 256 in the worst case (Liu et al., 2007). To solve this problem, we use serial half adder (HA) to save the hardware cost and

guarantee an error free calculation, as shown in the proposed serial PE architecture (see Figure 4.3).

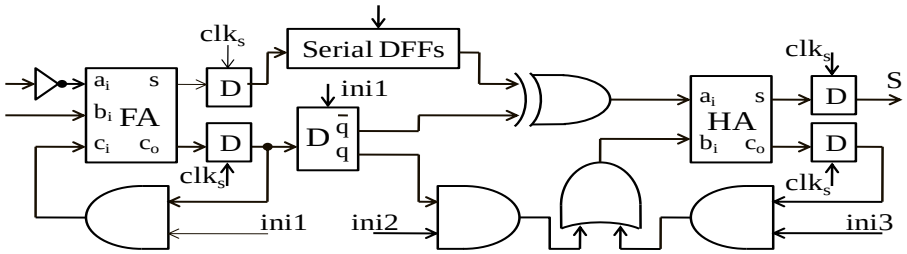


Figure 4.3 : Structure of the proposed serial PE.

4.3.2 Data flow of our first design.

The data flow of the first architecture for a 48×48 search area is given in Table 4.1, where the $C_{bn}(i, j)$ points to the pixels of the current MB, $R_{bn}(i, j)$ indicates the reference pixels in the search area, *bn* refers to the n^{th} bit of each pixel, *i* signifies row's index and *j* specifies column's index. Here, the clk_p and clk_s represent parallel and serial clocks so that clk_s is equal to $n \times clk_p$ and *n* is the bit length of each pixel. Note that *n* should be equal to the bit length of the largest possible SAD of a 16×16 block to lead to a correct result, which for our case *n* is equal to 16. The clk_s is the main clock of the design and clk_p is used to make easier the description of the data flow.

During a clk_p cycle, 16 reference pixels of the left side of zeroth row are loaded so that their zeroth bits (16 b_0s) are inputted in the zeroth clk_s cycle and their 15^{th} bits are proceeded at the 15^{th} clk_s cycle. Note that we use the sum ($\sum$) to show that there are 16 bits, which are from 16 different columns. In a similar way, during the following cycles of clk_p the corresponding rows of reference pixels are loaded so that at the 15^{th} cycle, the left side reference pixels of 15^{th} row are inputted. When ignoring the pipeline latency, the SAD of the (0, 0) to (0, 31) search candidates are sequentially produced at 16^{th} to 48^{th} clk_p cycles.

The scan order is column by column so that after loading the last row pixels in each column, the first row data of the next right side column should be loaded at the

following cycle. This type of data flow reduces the hardware utilization because when starting a new column, 15 cycles are needed for reading the search candidate block pixels. To solve this problem, it is required to lead the reference pixels of next column too, when loading the reference pixels of 33th row of each column (as shown in Table 4.1). In Figure 4.1, two sets of reference pixels are read and multiplexers select the right row of reference data and therefore 100% hardware utilization is approached. Due to the broadcasting of reference pixel data in vertical direction, all of PEs of a column use the same data. Therefore, the vertical data reuse is achieved leading to memory bandwidth reduction.

Table 4.1 : The data flow of our first design.

Clk_p	**Clk_s**	**PEs of 0th Row**	**PEs of 1st Row**	⋯	**PEs of 14th Row**	**PEs of 15th Row**
0	0	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(0,j)\rvert$	-	⋯	-	-
	1	$\sum_{j=0}^{15}\lvert C_{b1}(0,j)-R_{b1}(0,j)\rvert$				
	⋮	⋮				
	n-1	$\sum_{j=0}^{15}\lvert C_{bn-1}(0,j)-R_{b0}(0,j)\rvert$				
1	n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(1,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(0,j)\rvert$	⋯	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	14n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(14,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(1,j)-R_{b0}(13,j)\rvert$	⋯	$\sum_{j=0}^{15}\lvert C_{b0}(14,j)-R_{b0}(0,j)\rvert$	-
15	15n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(15,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(1,j)-R_{b0}(14,j)\rvert$	⋯	$\sum_{j=0}^{15}\lvert C_{b0}(14,j)-R_{b0}(1,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(15,j)-R_{b0}(0,j)\rvert$
16	16n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(16,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(1,j)-R_{b0}(15,j)\rvert$	⋯	$\sum_{j=0}^{15}\lvert C_{b0}(14,j)-R_{b0}(2,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(15,j)-R_{b0}(1,j)\rvert$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
32	32n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(32,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(1,j)-R_{b0}(31,j)\rvert$	⋯	$\sum_{j=0}^{15}\lvert C_{b0}(14,j)-R_{b0}(18,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(15,j)-R_{b0}(17,j)\rvert$
33	33n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(0,j+1)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(1,j)-R_{b0}(32,j)\rvert$	⋯	$\sum_{j=0}^{15}\lvert C_{b0}(14,j)-R_{b0}(19,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(15,j)-R_{b0}(18,j)\rvert$
34	34n	$\sum_{j=0}^{15}\lvert C_{b0}(0,j)-R_{b0}(1,j+1)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(1,j)-R_{b0}(0,j+1)\rvert$	⋯	$\sum_{j=0}^{15}\lvert C_{b0}(14,j)-R_{b0}(20,j)\rvert$	$\sum_{j=0}^{15}\lvert C_{b0}(15,j)-R_{b0}(19,j)\rvert$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.3.3 The Second Design: Bit-Serial Adder Tree Architecture

Figure 4.4 shows the proposed bit-serial architecture, which is based on Adder Tree architecture (C.-Y. Chen, Chien et al., 2006). This architecture has a 2-D structure that is composed of a reconfigurable reference buffer, an PE array, 16 2-D bit-serial 4×4 adder trees, and one 2-D VBS bit-serial adder tree that their details are explained in the following paragraphs.

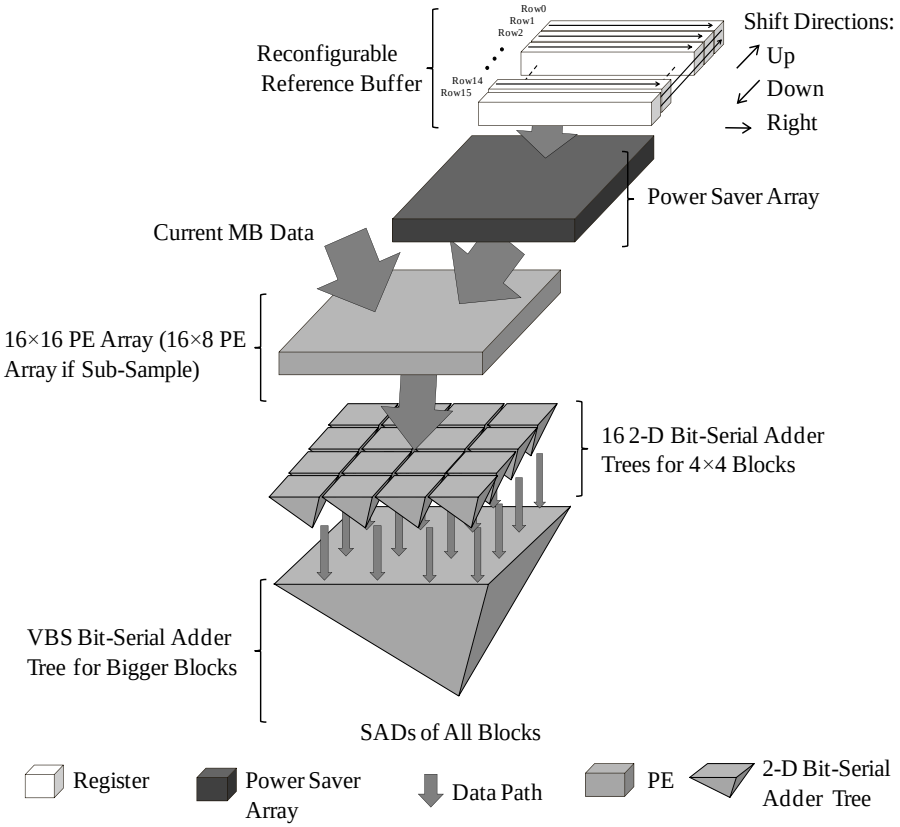


Figure 4.4 : Hardware architecture of the proposed Bit-Serial Adder Tree.

The reconfigurable reference buffer stores reference pixels enabling data reuse feature that will be discussed in subsection 4.3.4. At each cycle, 256 bits of current and reference pixels are sent to the PE array. Concurrently, a row of reference pixels are loaded into the reference buffer updating reference pixel data. The PE array consists of 256 serial PEs for calculating absolute difference of current and reference pixels whose structures are similar to the PE structures of the first design. Sixteen 2-D bit-serial 4×4

adder trees are used for computing the SADs of 4×4 blocks at the same time, as the hardware architecture of a 2-D bit-serial 4×4 adder tree is given in Figure 4.4. The SADs of 4×4 blocks are reused in the 2-D VBS bit-serial adder tree to generate the SADs of bigger blocks in parallel.

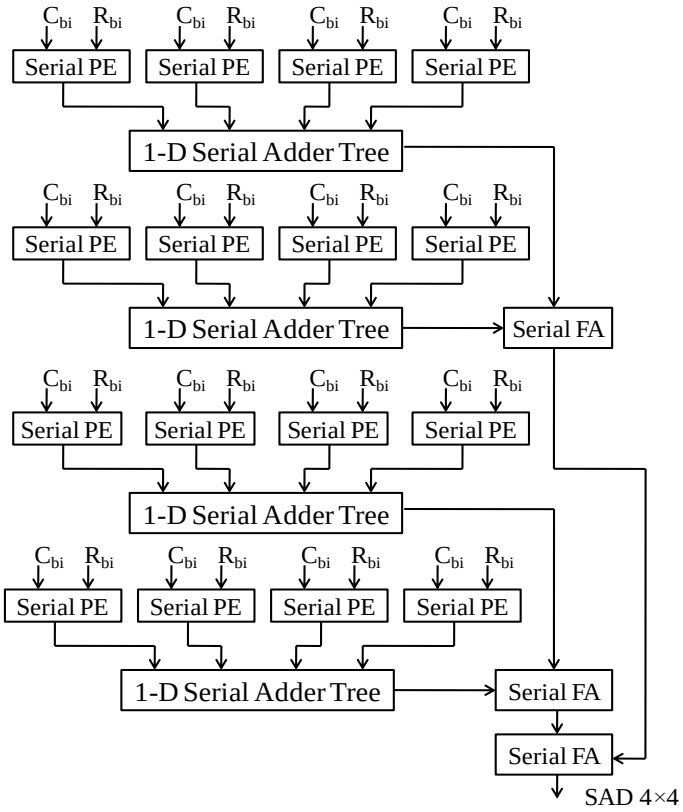


Figure 4.5 : Architecture of the proposed 2-D bit-serial 4×4 adder tree.

4.3.4 Data Flow of the Proposed Reconfigurable Reference Buffer

The data flow of our reconfigurable reference buffer is similar to SAD Tree architecture (C.-Y. Chen, Chien et al., 2006) except for one feature. The original reconfigurable reference buffer has a bit-parallel structure, and therefore we cannot directly use it in our bit-serial architecture. To integrate the original reconfigurable

reference buffer into the proposed Bit-Serial SAD Tree, its architecture has to be modified so that it can support serial operation.

The original reconfigurable reference buffer supports downward, leftward, and upward shift configurations and uses snake scan order that result in a high level of hardware utilization and data reuse. We add the bitwise shift right configuration to the original reconfigurable reference buffer to use it in our Bit-Serial Adder Tree.

Table 4.2 shows the data flow schedule of the proposed reconfigurable reference buffer for a 48×48 search area where the clk_p, clk_s, and $R_{bn}(i, j)$ were defined for Table 4.1. At each clk_p cycle and during 0^{th} to 31^{th} cycles, one row of 16 reference pixels is loaded to the proposed reconfigurable reference buffer where the shift configuration is downward. When inputting the last 16 rows of an even column (i.e. 32^{th} to 47^{th} rows of reference pixels), an additional reference pixel is loaded which is used for data reusing later. When finishing an even column, the shift configuration is switched to leftward for one cycle. In the following cycles, the scan order continues from the last row data of the next right side column (i.e. an odd column) and the shift configuration is changed to upward where the loaded rows have 16 reference pixels. Please note that as the reference pixels have already been stored in the reconfigurable reference buffer, the SAD calculation can continue lacking bubble cycles. In addition, when loading the last 16 rows of an odd column (i.e. 15^{th} to 0^{th} rows of reference pixels) an extra pixel is loaded. The same procedure will continue column by column with snake scan order, which leads high level of hardware utilization and makes possible the data reuse between two consecutive search candidates.

Table 4.2 : The data flow of our second design.

Clkp	Clks	Row 0	Row 1	⋯	Row 14	Row 15
0	0	$\sum_{j=0}^{15} R_{b0}(0,j)$	-	⋯	-	-
1	n	$\sum_{j=0}^{15} R_{b0}(1,j)$	$\sum_{j=0}^{15} \lvert R_{b0}(0,j) \rvert$	⋯	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	14n	$\sum_{j=0}^{15} R_{b0}(14,j)$	$\sum_{j=0}^{15} R_{b0}(13,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(0,j)$	-
15	15n	$\sum_{j=0}^{15} R_{b0}(15,j)$	$\sum_{j=0}^{15} R_{b0}(14,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(1,j)$	$\sum_{j=0}^{15} R_{b0}(0,j)$
16	16n	$\sum_{j=0}^{15} R_{b0}(16,j)$	$\sum_{j=0}^{15} R_{b0}(15,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(2,j)$	$\sum_{j=0}^{15} R_{b0}(1,j)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
31	31n	$\sum_{j=0}^{15} R_{b0}(31,j)$	$\sum_{j=0}^{15} R_{b0}(30,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(17,j)$	$\sum_{j=0}^{15} R_{b0}(16,j)$
32	32n	$\sum_{j=0}^{16} R_{b0}(32,j)$	$\sum_{j=0}^{15} R_{b0}(31,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(18,j)$	$\sum_{j=0}^{15} R_{b0}(17,j)$
33	33n	$\sum_{j=0}^{16} R_{b0}(33,j)$	$\sum_{j=0}^{16} R_{b0}(32,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(19,j)$	$\sum_{j=0}^{15} R_{b0}(18,j)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
47	47n	$\sum_{j=0}^{16} R_{b0}(47,j)$	$\sum_{j=0}^{16} R_{b0}(46,j)$	⋯	$\sum_{j=0}^{16} R_{b0}(33,j)$	$\sum_{j=0}^{16} R_{b0}(32,j)$
48	48n	$\sum_{j=0}^{16} R_{b0}(47,j)$	$\sum_{j=0}^{15} R_{b0}(0,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(0,j)$	$\sum_{j=0}^{15} R_{b0}(0,j)$
48	48n	$\sum_{j=0}^{16} R_{b0}(47,j)$	$\sum_{j=0}^{15} R_{b0}(0,j)$	⋯	$\sum_{j=0}^{15} R_{b0}(0,j)$	$\sum_{j=0}^{15} R_{b0}(0,j)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.4 Proposed Techniques for Performance Improvement

Our bit-serial architectures suffer from two major problems including dependency of the processing time to the bit length of the final SAD result and dynamic power consumption in the shift registers. Due to dependency of the bit length of the input data and the required cycles for computing the SAD of one search candidate to the bit length of the largest possible SAD (i.e. 16 bits for a 16×16 block), the performances of the proposed bit-serial architectures in terms of power consumption, processing ability, throughput, and latency are decreased. In addition, each PE has an n-bit shift register (i.e. n is the bit length of largest possible SAD) and the proposed reconfigurable reference buffer consists of N×N n-bit shift registers, which consume a significant amount of dynamic power. We have employed five optimization techniques to cope with these problems and improve the performances of the proposed designs further.

The first technique is pixel truncation (Bahari, Arslan, & Erdogan, 2009; He & Liou, 1997). We use current and reference pixels with only 5-bit precision. Consequently, the largest possible SAD of a 16×16 is reduced from 2^{16} to 2^{13}. As a result, the required cycles for calculating the SAD of one search point as well as the PEs' serial registers are decreased from 16 to 13, which lead to reduction of silicon area, memory bandwidth, and dynamic power as well as throughput improvement.

The second technique is word length reduction by parallelism (Zhou & Kornerup, 1995). In this technique, two FAs are used to obtain a higher computing rate in computing the SADs in a serial manner. As the final SAD of a 16×16 block can have up to 16 bits, by using two FAs, one for the low order 8 bits and another for the high order 8 bits, the processing speed is doubled at the price of an added hardware cost. Taking into account the 5-bit truncated input pixels that may lead to a 13 bits SAD, we can use three FAs to triple the throughput, where the first, second and third FAs are responsible for the low order, middle order and high order 5 bits, respectively. However, replacing

of all FAs with a set of three FAs, will approximately increase the hardware cost of FAs by three. To increase the processing ability with an acceptable hardware cost, we propose several circuit optimization techniques as follows.

Since the input pixels are truncated in our designs, we first configure their proposed PEs from 13 bits to 5 bits operation because the output result for absolute difference operation may have 5 bits of maximum bit length. This leads to a significant hardware cost reduction of the PEs and their propagation registers in the proposed architectures (see Figure 4.3). Then, for adding two absolute difference results in the first stages of 1-D serial adder trees in Figure 4.2 and Figure 4.5, we propose an efficient serial adder that can add two 5-bit inputs while the processing time is still 5 cycles instead of 6 cycles, as shown in Figure 4.6(a). When the sum result is less than 2^6, the output result is presented with 5 bits using sl output that is the low order sum result. When the sum result is bigger than 2^6-1, the sum result needs 6 bits, where the sixth bit is the carry-out signal in the 5th clk. By using an AND gate, the higher order sum result (i.e. sh) is calculated so that its bit length is only one. Due to the use of 256 PEs in each design, 128 FAs are used in the first stages 1-D serial adder trees. Accordingly in the new designs, we only use 128 extra AND gates whereas the processing time for one searching candidate is reduced from 13 to 5 cycles. In Figure 4.6(a), please note that the ini signal is set to zero at the beginning of a new add operation for one cycle to obtain a correct result.

For the second stages of the 1-D serial adder trees and other serial FAs in Figure 4.2 and Figure 4.5, we use two FAs, one for the low order 5 bits and another for the high order 5 bits, as shown in Figure 4. 6(b). Note that in this figure the carry-in (ci) of the high order 5 bits should be the 5th carry-out (co) of the low order part. By using two 5-bit FAs, the maximum value of 2^{11}-1 can be calculated. Therefore, the SAD of 4×4 and 4×8 and 8×4 block sizes can be computed with the proposed circuit in Figure 4. 6(b). As

for a 4×4 block seven sets of presented circuit in Figure 4.6(b) are used; for all block sizes smaller than 8×8 blocks, 16×7 +8 sets of this circuit are required. By using three FAs in a similar way as shown in Figure 4.6(b), we can calculated the SAD values of the bigger blocks (i.e. 8×8, 8×16, 16×8 and 16×16 blocks). Therefore, we only need nine sets of three FAs whereas the calculating of a search candidate is reduced to 5 cycles. In summary, with the proposed word length reduction scheme the required cycles for processing of one search candidate and the bit length of the PEs' shift registers are reduced from 16 bits to 5 bits, and thus hardware cost and dynamic power are significantly decreased.

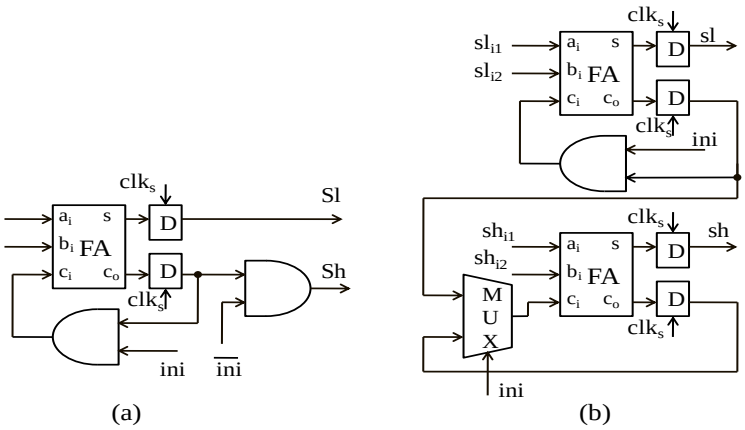


Figure 4.6 : Hardware architecture of (a) the proposed serial adder for adding 5-bit operands and (b) the proposed two parts serial adder.

The third technique is 1/2-subsampling at the MB level. Since in the H.264 standard the video frames are processed MB by MB, with 1/2-subsampling at the MB level, we can roughly reduce the hardware cost of the PE array, computational load and memory bandwidth by a factor of two. Because with 1/2-subsampling, the number of the required reference pixels, current pixels, and PEs is reduced two times.

The forth method is mode reduction in which by removing the 4×4 modes, 16 modes of 41 modes in the H.264 IME are reduced. Consequently, their corresponding circuits for obtaining minimum SADs and related MVs are saved.

The last method is a power reduction technique proposed in response to the dynamic power consumption of the bitwise shift right configuration in the proposed reconfigurable reference buffer. To facilitate the description of the proposed power saving method, consider an n-bit serial shift right register, as shown in Figure 4.7(a). This shift register supports downward, leftward, and upward shift configurations, where the reference data are in parallel from. When configuring in the bitwise shift right case, at each clk_s cycle, n bits are shifted consuming a lot of power whereas the corresponding PE circuit only uses one bit.

To avoid the serial shift operation, we employ a multiplexer. In this way at each cycle, the appropriate bit is selected and sent to PE array circuit, as shown in Figure 4.7(b). As a result, the power consumption is considerably decreased at the price of an extra hardware cost of multiplexer. Taking into account the bit length of reference pixel after applying the proposed optimization technique (i.e. 5 bits), we need to use an 8 to 1 multiplexer, where three of its inputs are not used. To mitigate the hardware cost of this multiplexer, we utilize a 4 to 1 multiplexer connected to a 2 to 1 multiplexer to provide a 5:1 multiplexer, as shown in Figure 4.7(c).

In summary, to cope with the long processing time and high dynamic power consumption problems of the proposed bit-serial IME designs, we apply several optimization techniques. By the word length reduction and bit-truncation techniques, we reduce the required processing time of a search candidate from 16 to 5 cycles, where the other design parameters such as power consumption, silicon area, and throughput are improved too. In addition, mode filtering and 1/2-subsampling methods are used to reduce the hardware costs further. Besides, the dynamic power of the proposed bit-serial reconfigurable reference buffer is decreased by using a power saving method at price of an extra area cost.

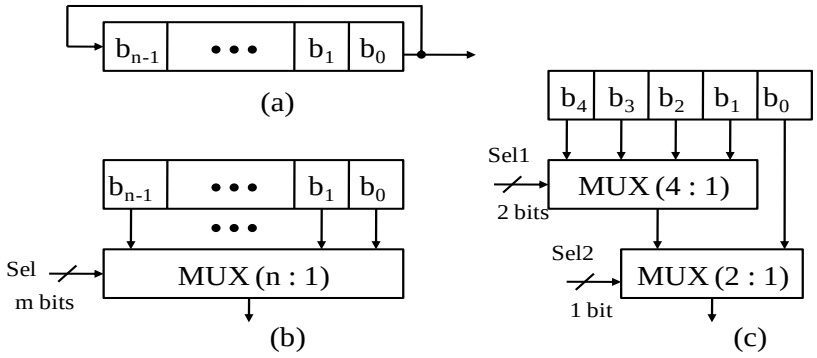


Figure 4.7 : Hardware architecture of (a) shift register, (b) the proposed power reduction circuit, and (c) the proposed improved power reduction circuit.

4.5 Experimental Results and Analysis

4.5.1 Evaluation of Rate-Distortion Performance

In order to evaluate the impact of the proposed optimizations techniques on R-D performance, we incorporate them into the reference software (JM version 11) using several test sequences. Tables 4.3-4.6 give average bit rate and PSNR difference for case (a) to case (e) where the test conditions are similar to that of Chapter 3 for Akiyo (QCIF), Container (QCIF), Mobile (CIF), and News (CIF) test sequences.

(a) FS IME with all block sizes and one reference frame.

(b) 4×4 mode filtered variable block-size FS IME with one reference frame.

(c) 3 bits truncated FS IME with all block sizes and one reference frame.

(d) 1/2-subsampled FS IME with all block sizes and one reference frame.

(e) Optimized FS IME (4×4 mode filtered + 3 bits truncated + 1/2-subsampled) with one reference frame.

In Tables 4.3-4.6, each case is described by four points with PSNR and bit rate values where the average bit rate difference percentage (ΔBR(%)) and the PSNR difference in terms of dB (ΔPSNR(dB)) of each case are calculated with respect to the reference case (i.e. case (a)). In addition to the average bit rate difference percentage and the PSNR difference, Figure 4.8 and Figure 4.9 show the R-D plots for Carphone

(QCIF), Mother-Daughter (CIF), City(4CIF), and Blue-Sky(HD1080) test sequences where the performance of the FS IME algorithm and its optimized version that are similar to case (a) and case (e), correspondingly. As seen in Table 4.3-Table 4.6 as well as Figures 4.8-4.11, the proposed optimization techniques lead to very competitive results compared with the FS IME algorithm. The proposed techniques are simple but effective and reduce the computational complexity and the memory bandwidth of the FS algorithm.

Table 4.3 : The impact of the proposed optimization methods on coding performance for Akiyo (QCIF) sequence.

Criterion \ Case	(a)	(b)	(c)	(d)	(e)
Bit rate1 (kbps)	10.2	10.1	10.8	10.1	10.1
PSNR1 (dB)	30.02	30.07	30.06	29.99	30.01
Bit rate2 (kbps)	20.0	20.0	20.1	20.1	20.4
PSNR2 (dB)	33.45	33.61	33.52	33.51	33.58
Bit rate3 (kbps)	30.1	30.1	30.1	30.4	30.8
PSNR3 (dB)	35.47	35.50	35.44	35.47	35.47
Bit rate4 (kbps)	60.0	60.1	60.0	60.0	60.3
PSNR4 (dB)	38.20	38.23	38.20	38.23	38.20
ΔBR (%)	0.00	-2.03	-0.56	-0.71	-1.19
ΔPSNR (dB)	0.00	0.08	0.02	0.03	0.04

The Parameters for used sequences are 30 fps, ±16-pel search range, high complexity mode decision and 300 encoded frames.

Table 4.4 : The impact of the proposed optimization methods on R-D performance for Container (QCIF) sequence.

Criterion \ Case	(a)	(b)	(c)	(d)	(e)
Bit rate1 (kbps)	15.0	15.5	15.0	15.0	15.0
PSNR1 (dB)	29.69	29.73	29.79	29.76	29.67
Bit rate2 (kbps)	30.0	30.1	30.0	30.1	30.1
PSNR2 (dB)	32.18	32.17	32.17	32.15	32.11
Bit rate3 (kbps)	60.0	60.7	60.1	60.1	60.0
PSNR3 (dB)	34.40	34.38	34.39	34.34	34.40
Bit rate4 (kbps)	90.1	90.0	90.0	90.1	90.1
PSNR4 (dB)	36.34	36.34	36.34	36.41	36.34
ΔBR (%)	0.00	0.21	0.01	0.68	1.58
ΔPSNR (dB)	0.00	-0.002	0.004	-0.016	-0.054

Table 4.5 : The impact of the proposed optimization methods on coding performance for Mobile (CIF) sequence.

Criterion \ Case	(a)	(b)	(c)	(d)	(e)
Bit rate1 (kbps)	1599.7	1599.7	1599.7	1599.5	1599.6
PSNR1 (dB)	28.05	28.04	28.03	27.98	27.99
Bit rate2 (kbps)	2499.3	2499.5	2499.5	2499.4	2499.8
PSNR2 (dB)	30.55	30.53	30.54	30.51	30.49
Bit rate3 (kbps)	3499.4	3499.4	3499.1	3499.3	3499.4
PSNR3 (dB)	32.97	32.95	32.96	32.94	32.92
Bit rate4 (kbps)	4998.8	4998.2	4998.8	4998.4	4998.1
PSNR4 (dB)	35.92	35.90	35.91	35.89	35.85
ΔBR (%)	0.00	0.26	0.18	0.59	0.85
ΔPSNR (dB)	0.00	-0.019	-0.012	-0.041	-0.060

Table 4.6 : The impact of the proposed optimization methods on coding performance for News (CIF) sequence.

Criterion \ Case	(a)	(b)	(c)	(d)	(e)
Bit rate1 (kbps)	40.1	40.1	40.1	40.1	40.1
PSNR1 (dB)	28.59	28.53	28.57	28.59	28.56
Bit rate2 (kbps)	80.2	80.2	80.2	80.2	80.2
PSNR2 (dB)	31.45	31.48	31.38	31.45	31.42
Bit rate3 (kbps)	150.5	150.5	150.5	150.5	150.6
PSNR3 (dB)	33.99	33.98	33.96	33.99	33.91
Bit rate4 (kbps)	250.8	250.7	250.7	250.8	250.7
PSNR4 (dB)	36.26	36.21	36.24	36.24	36.15
ΔBR (%)	0.00	0.03	1.10	0.05	1.31
ΔPSNR (dB)	0.00	-0.003	-0.046	-0.002	-0.054

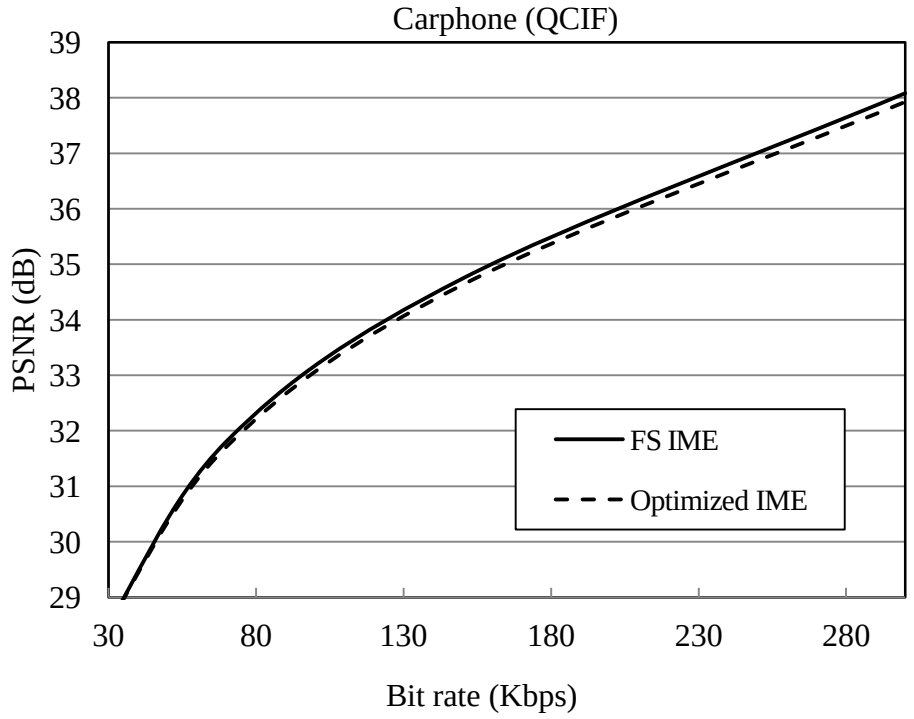


Figure 4.8 : R-D plot of Carphone test sequence for FS IME and its optimized version.

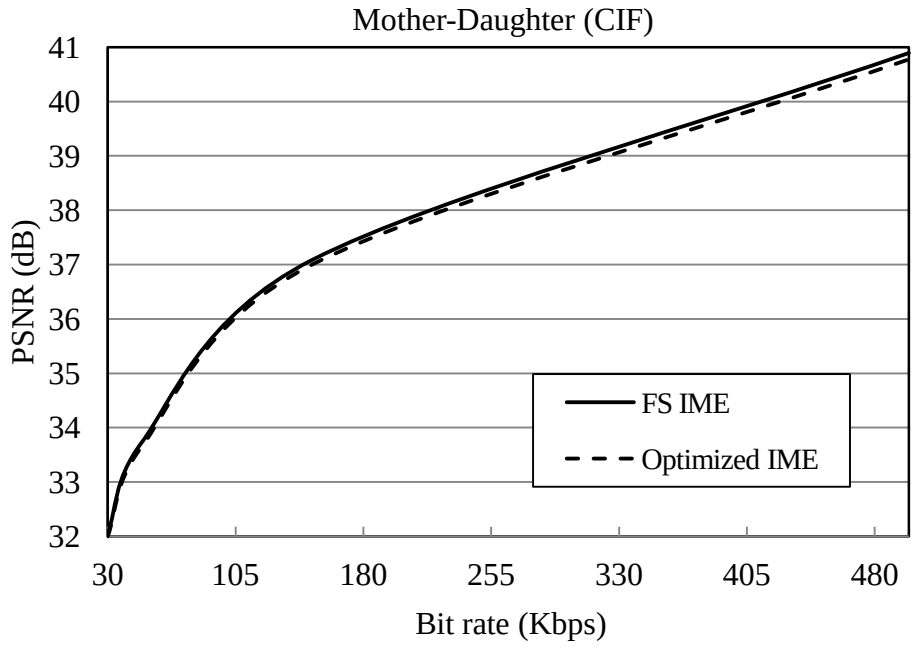


Figure 4.9 : R-D plot of Mother-Daughter test sequence for FS IME and its optimized version.

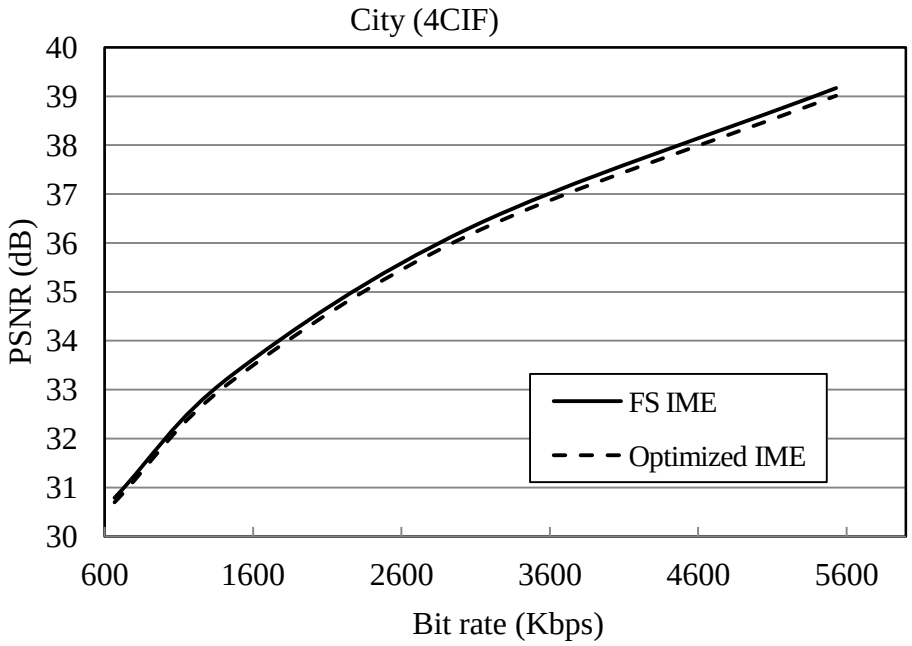


Figure 4.10 : R-D plot of City test sequence for FS IME and its optimized version.

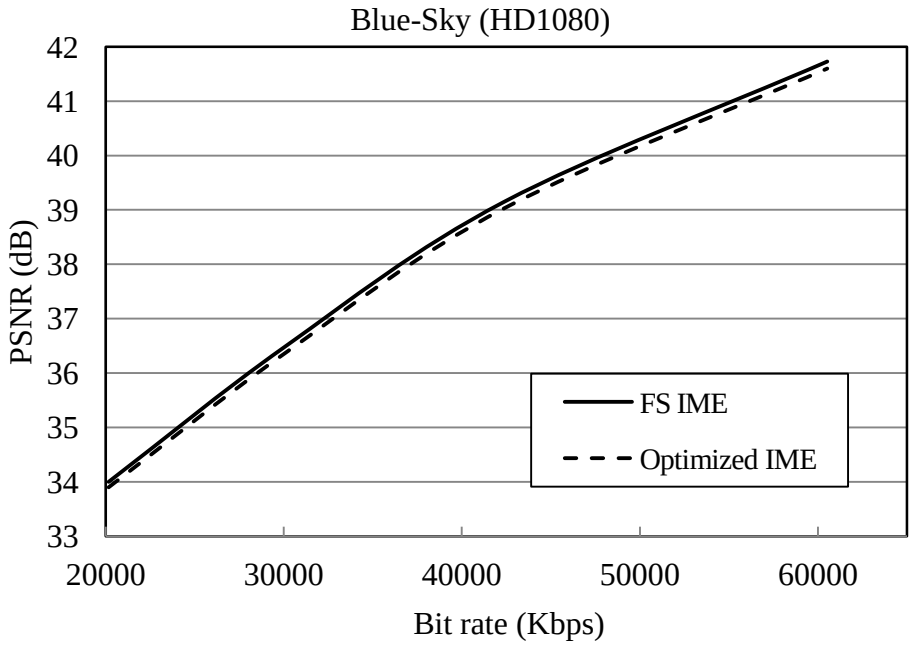


Figure 4.11 : R-D plot of Blue-Sky test sequence for FS IME and its optimzed version.

4.5.2 Implementation Results and Reusability Discussion

The proposed bit-serial IME architectures for H.264 have been implemented in Verilog-HDL and synthesized in Silterra 0.18 μm technology using Synopsys Design Compiler. Table 4.7 lists the implementation results of the proposed architectures. Due to bit-serial structure and applying of several optimization techniques, the hardware cost of the first and the second architectures are 29.28 K gates and 31.5 K gates, respectively making them good candidates for using in area-constrained designs. Taking into consideration a search range of [-16, +15] in the horizontal and vertical directions and the required cycles for processing one search candidate (i.e. 5 cycles), the proposed architectures can process an MB at 5120 clock cycles. As a result, CIF resolution with 30 fps can be processed at 60.82 MHz where the first and the second designs consume 19.94 mW and 11.77 mW under slow operating conditions (i.e. 1.62 V and 125°C). With the clock frequencies of 440 MHz and 427 MHz, the maximum throughputs of the first and the second proposed designs are 85.93 K MB/s and 83.39 K MB/s, respectively. Therefore, higher resolutions can be met with higher clock frequencies. For instance, real-time VBS FS IME of 720×480 video @ 30 fps with a [-16, 15] search range and one reference frame is satisfied at 207 MHz. Note that due to having of reconfigurable reference, the memory bandwidth of the second design is lower than the first design.

Table 4.7 : Implementation results of the proposed designs.

	Our first design	Our second design
Technology	Silterra 0.18um	Silterra 0.18um
Voltage	1.62 V	1.62 V
Gate counts	29.28 K gates	31.5 K gates
Required frequency	60.82 MHz	60.82 MHz
Max Throughput	85.93 K MB/s	83.39 K MB/s
Memory Bandwidth	163 Kbits/MB	87 Kbits/MB
Power consumption	19.94 mW	11.77 mW

In addition, since the first design has many propagation registers and the proposed reconfigurable reference buffer of the Bit-Serial Adder Tree benefits from power saving technique, the second design consumes a lower power consumption than the first design.

The proposed designs can be reused for higher resolutions such as HD format by adopting the parallelism technique whereas the processing ability as well as the memory bandwidth of them can be improved further. For example, consider HD720 resolution with one reference frame, [-32, +31] search range, and 30 fps. The required frequency for satisfying the real-time IME for the above specifications with one set of our bit-serial architecture is 2,211 MHz. By using five sets of the first or the second designs, which work in parallel, these specifications can be met under 368.5 MHz. When using the first architecture, we need to five sets working in parallel. As a result, the processing ability increase 500% at price of the hardware cost of five sets of the first design. Please note that the reference pixels in each row can be reused by these parallel architectures, and therefore the memory bandwidth is decreased as well. If we use the second design for the above specification, we need one reconfigurable reference buffer with only seven extra reference pixels in each row and five sets of PE array. Consequently, the processing ability increases by a factor of five and 80% of the memory bandwidth is roughly saved. When using the parallelism, the second design shows a better performance compared with the first design in terms of hardware cost. Because in parallelism, the second design only requires one reference reconfigurable buffer with extended size and the required number of its PE array is defined by the degree of the parallelism (i.e. N). Whereas for the first design, we need to use N sets of the first architecture that increases its hardware cost by a degree of N.

4.5.3 Design Metrics Evaluation and Comparison

Table 4.8 lists the design metrics for the proposed bit-serial architectures and the state-of-the-art architectures that were surveyed in Chapter 2 including the designs of Yap and MaCcany (2004), Ou et al. (2005), Y.-C. Chen et al. (2006), Li and Leong (2008), Lopez et al. (2008), and Lim & Park (2009). Since the required operating frequency, maximum throughput, and memory bandwidth are affected by the search range, frame size, the number of reference frames, and frame rate, and in order to provide a reasonable comparison, the given design metrics in Table 4.8 are for the same specifications i.e. a [-16, 15] search range, 352×288 frame size, one reference frame, and 30 fps.

Regarding the algorithm quality metric, because of using the FS IME algorithm, all of the architectures in Table 4.8 achieve the highest coding performance. Note that owing to the use of optimization methods, our bit-serial architectures have a very negligible quality degradation that is only about 0.05 dB PSNR in average. However, when considering the advantages of the optimization methods that were explained in Section 4.4, this quality loss can be ignored.

As seen in Table 4.8, due to the optimized bit-serial architectures, the area costs of our designs are lower than the previous architectures except for "(16, 1) arch" design of Lopez et al. However, "(16, 1) arch" requires much higher memory bandwidth and has much lower throughput than our designs. In addition, compared with prior low cost designs (i.e. 1-D architectures (J. Kim & Park, 2009; Lopez et al., 2008; Yap & McCanny, 2004) and MSB bit-serial architecture (Li & Leong, 2008), our bit-serial architectures need the lowest number of cycles for processing one search candidate (i.e. 5 cycles), and thus they outperform other designs in terms of the required frequency and the maximum throughput. When our design is compared with 2-D designs (i.e. the designs of Ou et al. and Y.-C. Chen et al.), our architectures need a higher clock

frequency for real-time encoding of the above-mentioned specifications due to serial operation. However, the hardware costs of our designs are much lower than 2-D architectures. The reason is that these 2-D architectures consist of 256 bit-parallel PEs and therefore need to lower clock frequencies to meet the real-time constrains of Table 4.8. In addition, these 2-D architectures have higher throughputs relative to our designs at the price of higher silicon areas. Please note that by using hardware parallelism, as described in subsection 4.5.2, we can achieve higher throughput rate with reasonable hardware cost.

Another important design parameter is the memory bandwidth that is defined as the required number of bits that an MB has to read from memory. In fact, the memory bandwidth is affected by the level of data reusing so that a higher level of data reusing leads to a lower memory bandwidth. Among all of architectures in Table 4.8, the proposed Bit-Serial Adder Tree design has the lowest memory bandwidth due to pixel truncation and data reusing in the vertical and horizontal directions enabled by the reconfiguration reference buffer. In addition, the first design outperforms the 1-D and MSB bit-serial architectures as it uses the pixel truncation and broadcasting of reference data whereas the 1-D designs only use broadcasting technique and the MSB bit-serial architecture does not benefit from the reconfigurable reference buffer or data broadcasting technique.

Regarding the hardware utilization, when ignoring the start up cycles, all of the architectures in Table 4.8 are approaching 100% hardware utilization except the design of Li and Leong (2008) that includes 761 cycles on-line delay for calculating the SAD of each MB that takes 18432 cycles.

Table 4.8 : Design metrics evaluation and comparison.

Architecture	Design Metrics	
Yap and MaCcany (2004): 1-D architecture with 16 PEs.	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	61
	Operating frequency (MHz)	194.6
	Maximum Throughput (MB/s)	17,944
	Memory bandwidth (Kbits/MB)	4,194
	Hardware utilization (%)	100
Ou et al. (2005): 2-D architecture with 256 PEs	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	597
	Operating frequency (MHz)	12.16
	Maximum Throughput (MB/s)	195,312
	Memory bandwidth (Kbits/MB)	N/A
	Hardware utilization (%)	100
Y.-C. Chen et al. (2006): 2-D parallel adder tree architecture with 256 PEs.	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	88.6
	Operating frequency (MHz)	12.16
	Maximum Throughput (MB/s)	106,250
	Memory bandwidth (Kbits/MB)	139
	Hardware utilization (%)	100
Li and Leong (2008): MSB bit-serial design with 256 PEs.	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	55
	Operating frequency (MHz)	274
	Maximum Throughput (MB/s)	22,786
	Memory bandwidth (Kbits/MB)	4718
	Hardware utilization (%)	95.87
Lopez et al. (2008): 1-D architectural template	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	33.41/21.3
	Operating frequency (MHz)	100
	Maximum Throughput (MB/s)	5,910
	Memory bandwidth (Kbits/MB)	541/2,166
	Hardware utilization (%)	100
Lim and Park (2009): 1-D architecture with 16 PEs.	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	39.2
	Operating frequency (MHz)	195
	Maximum Throughput (MB/s)	25,390
	Memory bandwidth (Kbits/MB)	4,194
	Hardware utilization (%)	100
Our first design: 2-D bit-serial architecture with 128 PEs.	Quality loss (dB PSNR)	-0.05
	Silicon area (K gates)	29.28
	Operating frequency (MHz)	60.8
	Maximum Throughput (MB/s)	85930
	Memory bandwidth (Kbits/MB)	163
	Hardware utilization (%)	100
Our second design: Bit-serial adder tree architecture with 128 PEs.	Quality loss (dB PSNR)	-0.05
	Silicon area (K gates)	31.5
	Operating frequency (MHz)	60.8
	Maximum Throughput (MB/s)	83390
	Memory bandwidth (Kbits/MB)	87
	Hardware utilization (%)	100

4.6 Summary

In this section, we have presented two low cost bit-serial H.264 IME architectures and addressed the advantages and challenges of the bit-serial structure. Our architectures benefit from SAD and data reusing techniques reducing their memory bandwidth. The first design has a 2-D structure featured with broadcasting of reference pixel data and propagating of partial sum and SAD results. The second design uses 2-D bit-serial adder tree connected to a reconfigurable reference buffer making it suitable for hardware parallelism. With the aim of solving the design challenges of bit-serial architectures and in order to improve the performance of the proposed designs, we have used several optimization methods including pixel truncation scheme and word length reduction technique, mode filtering, 1/2-subsampling and a power reduction method for the reconfigurable reference buffer. These optimization methods significantly improve the performance of our designs in terms of area cost, memory bandwidth, throughput, and power consumption. The experimental results show that the proposed designs can process real-time VBS IME of CIF resolution with [-16, 15] search range, one reference frame and 30 fps at 60.8 MHz and with less than 32 K gates. Finally, the performance evolution and comparison with the previous representative designs have been provided showing the advantages and weaknesses of our designs relative to them.

Chapter 5

Algorithm Analysis and Bit-Serial Architecture Design for Sub-Pixel Motion Estimation in H.264

5.1 Introduction

H.264/AVC employs SME with quarter-pixel accuracy that can improve R-D efficiency by 4 dB PSNR (T.-C. Chen et al., 2004). However, as described in Chapter 3, the computational load and the memory bandwidth of the quarter-pixel accurate ME are highly extensive due to the use of VBS, MRF, interpolation scheme for producing sub-pixel values, two sub-pixel search steps, and matching process. In addition, as seen in the H.264/AVC reference software, MB processing in the SME algorithm is usually sequential with a lot of data dependency, which is not suitable for hardware implementation. From the hardware point of view, the above problem restricts the degree of parallelism at the architecture level, which in turn decreases the processing capability (i.e. throughput) of the design. Consequently, the hardware architecture may not be able to meet the required throughput of real-time applications, especially for HDTV resolution. Therefore, design of fast algorithms and efficient hardware architectures for the H.264/AVC SME are required for real-time applications. Besides, all of the SME designs in the literature (i.e. the reviewed designs in Chapter 3) use parallel data path to increase their processing capability at the price of higher area costs and pin counts. Consequently, they may not be suitable for area-constrained applications.

In this chapter motivated by the above-mentioned issues, we propose a fast SME algorithm and its low cost architecture requiring low computational complexity and memory access requirement. The proposed algorithm is based on parabolic interpolation free algorithms (Suh & Jeong, 2004). We review parabolic interpolation free algorithms and extend their accuracy from half-pixel to quarter-pixel. In addition, we analyze the computational complexity and the memory bandwidth requirement of the H.264 SME in the reference software and the proposed algorithm. According to our analysis, the proposed algorithm significantly reduces the computational budget and the memory requirement in comparison with the interpolate and search method in the reference software with an acceptable video quality. Besides, to lower the computational complexity and the memory bandwidth further, we propose a fast version of the proposed algorithm along with SAD truncation and mode filtering techniques. For the hardware architecture design, we choose bit-serial structure for implementing our algorithm to benefit from its advantages. In addition, we use reusability, source sharing, and power saving techniques in our architecture that lead to area saving and power consumption reduction. Our architecture can support real-time HD1080 format with 20.3 K gates at the operation frequency of 88.3 MHz.

The rest of this chapter is organized as follows. Section 5.2 reviews the parabolic based SME algorithms and presents the details of the proposed algorithm. In Section 5.3, the computational budget and the memory access requirement of the proposed algorithm and the standard interpolate and search method are analyzed. Section 5.4 presents the proposed optimization techniques that reduce the computational complexity of the proposed algorithm. In Section 5.5, the proposed bit-serial architecture for quarter-pixel accurate ME is described and our power saving technique is presented. In Section 5.6, the experimental results and comparison between our design and the

previous architectures are provided. Finally, Section 5.7 gives a summary of this chapter.

5.2 The Proposed Low Complexity Algorithm

5.2.1 Review of Parabolic Interpolation Free Algorithms

Suh and Jeong (2004) introduced five mathematical models (i.e. Model 1, Model 2, Model 3, modified Model 2, and modified Model 3) for calculating of the sub-pixel motion vectors, which the complexity as well as the performance of each model depended on its order. To compute the coefficients of models, the SAD values of the neighboring positions around the best integer motion vector were used. After calculating the coefficients, the sub-pixel SAD values around the best integer motion vector were obtained. Then, the lowest sub-pixel SAD value determined the best sub-pixel MV. Due to avoiding of sub-pixel interpolation in the proposed models, their computational load and memory bandwidth were significantly reduced. Among the proposed models, Model 1 achieves the best coding performance due to considering all eight neighboring positions around the best integer motion vector at the price of more complexity. From the hardware point of view, Model 1 and Model 2 are more suitable than the other models. Please note that all of these five models only support half-pixel accuracy and do not involve the new coding tools of the H.264/AVC standard such as quarter-pixel accuracy motion vector, VBS, and Lagrangian mode decision. Therefore, for supporting the H.264/AVC SME with quarter-pixel accuracy, a new parabolic interpolation free algorithm is required. In the next sub-section, we explain our SME algorithm with quarter-pixel accuracy, which is based on Model 1 due to its good coding performance and suitability for hardware implementation.

5.2.2 The proposed Nine-Five Model Algorithm

Our algorithm is based on Model 1, which we extend its accuracy form half-pixel to quarter-pixel featured VBS and Lagrangian mode decision coding tools. Due to the use of nine and five search points in half and quarter-pixel refinements, respectively, we name it nine-five model (NFM) algorithm. In Figure 5.1, suppose that after coarse IME, the SAD values of the best integer MV (IMV)candidate and its eight neighboring integer-pixels are known (i.e. (0,0) and ((-1,-1), (0,-1), (1,-1), (-1,0), (1,0), (-1,1), (0,1), (1,1)), respectively).

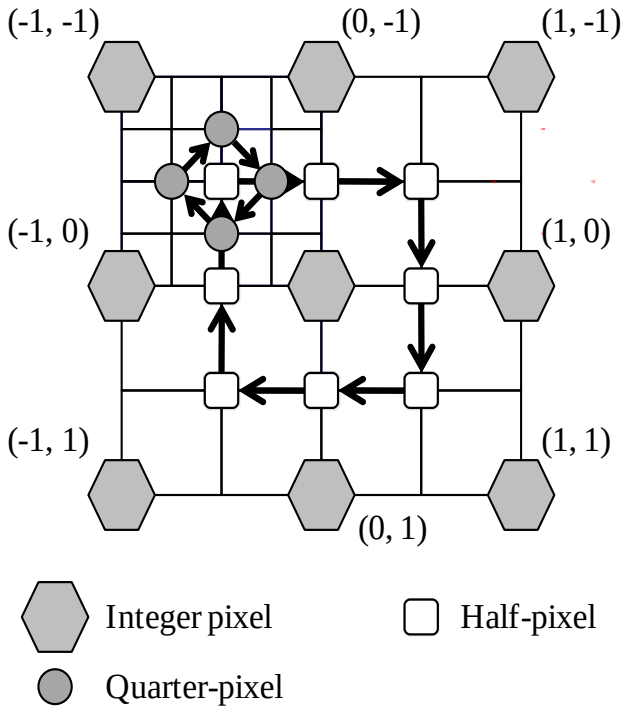


Figure 5.1 : The search pattern of the proposed algorithm for sub-pixel motion estimation.

We define the following equation as a parabolic surface function of SAD values.

$$SAD(x,y) = A_1x^2y^2 + A_2x^2y + A_3xy^2 + A_4xy + A_5x^2 + A_6x + A_7y^2 + A_8y + A_9 =$$
$$ISAD(x,y) = HSAD(x,y) = QSAD(x,y). \quad (5.1)$$

In this equation *ISAD*, *HSAD*, and *QSAD* represent integer, half-pixel, and quarter-pixel SAD amounts, respectively. By using the nine integer SAD values, all nine coefficients of equation (5.1) (i.e. A_1-A_9) can be calculated from equation (5.2) and equation (5.3).

$$\begin{bmatrix} ISAD(-1,-1) \\ ISAD(\ 0,-1) \\ ISAD(\ 1,-1) \\ ISAD(-1,\ 0) \\ ISAD(\ 0,\ 0) \\ ISAD(\ 1,\ 0) \\ ISAD(-1,\ 1) \\ ISAD(\ 0,\ 1) \\ ISAD(\ 1,\ 1) \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \\ A_9 \end{bmatrix} . \quad (5.2)$$

By calculating the inverse matrix of equation (5.2), A_1-A_9 coefficients are obtained from equation (5.3).

$$\begin{bmatrix} 1/4 & -1/2 & 1/4 & -1/2 & 1 & -1/2 & 1/4 & -1/2 & 1/4 \\ -1/4 & 1/2 & -1/4 & 0 & 0 & 0 & 1/4 & -1/2 & 1/4 \\ -1/4 & 0 & 1/4 & 1/2 & 0 & -1/2 & -1/4 & 0 & 1/4 \\ 1/4 & 0 & -1/4 & 0 & 0 & 0 & -1/4 & 0 & 1/4 \\ 0 & 0 & 0 & 1/2 & -1 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & -1 & 0 & 0 & 1/2 & 0 \\ 0 & -1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} ISAD(-1,-1) \\ ISAD(\ 0,-1) \\ ISAD(\ 1,-1) \\ ISAD(-1,\ 0) \\ ISAD(\ 0,\ 0) \\ ISAD(\ 1,\ 0) \\ ISAD(-1,\ 1) \\ ISAD(\ 0,\ 1) \\ ISAD(\ 1,\ 1) \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \\ A_9 \end{bmatrix} . \quad (5.3)$$

Now by substituting A_1-A_9 and the values of -0.5, 0, and 0.5 for *x*, *y* in equation (5.1), the motion compensation (MC) prediction errors at the neighboring half-pixel positions (i.e. $HSAD_1$-$HSAD_9$) can be calculated from equation (5.4).

$$\begin{bmatrix} HSAD(-0.5,-0.5) \\ HSAD(\ 0,-0.5) \\ HSAD(\ 0.5,-0.5) \\ HSAD(-0.5,\ 0) \\ HSAD(\ 0,\ 0) \\ HSAD(\ 0.5,\ 0) \\ HSAD(-0.5,\ 0.5) \\ HSAD(\ 0,\ 0.5) \\ HSAD(\ 0.5,\ 0.5) \end{bmatrix} = \begin{bmatrix} 1/16 & -1/8 & 1/4 & -1/8 & 1/4 & -1/2 & 1/4 & -1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & -1/2 & 1 \\ 1/16 & -1/8 & 1/4 & 1/8 & -1/4 & 1/2 & 1/4 & -1/2 & 1 \\ 0 & 0 & 1/4 & 0 & 0 & -1/2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1/4 & 0 & 0 & 1/2 & 0 & 0 & 1 \\ 1/16 & 1/8 & 1/4 & -1/8 & -1/4 & -1/2 & 1/4 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/2 & 1 \\ 1/16 & 1/8 & 1/4 & 1/8 & 1/4 & 1/2 & 1/4 & 1/2 & 1 \end{bmatrix} \times \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \\ A_9 \end{bmatrix}. \qquad (5.4)$$

Then, the total coding costs of all half-pixel candidates are calculated using the estimated SAD and the coding cost of the corresponding MV:

$$J=HSAD + \lambda \times R\ . \qquad (5.5)$$

where λ is the Lagrangian multiplier, which is a function of quantization parameter (QP) and R is the total number of bits required for coding the MV information. The minimum J determines the optimal half-pixel accurate MV. After finding the optimal half-pixel accurate MV, we use a cross pattern around that to find the quarter-pixel accurate ME similarly (see Figure 5.1). By substituting the values of -3/4, -1/2, 0, 1/2, and 3/4 for x, y in equation (5.1), the SAD's values at quarter-pixel positions (*QSADs(x, y)*) can be calculated. Finally, by using the equation (5.5) (where *HSAD* is replaced by *QSAD*), the optimal quarter-pixel accurate MV is obtained.

5.3 Computational Complexity and Memory Access Analysis

This section presents a direct approach for calculating the computational complexity (*CC*) and the memory access (*MA*) requirement of the SME algorithm in the reference software and the proposed algorithm. All of the calculations are based on macroblock (MB) as the MB processing is adopted in the H.264/AVC video coding systems. Based on our analysis, we can approximately estimate the required computation and the memory access of a certain specification (*CS*) in terms of its MB rate. The results of our analysis reveal that the computational budget and the memory access requirement of the proposed algorithm is much lower than the interpolate and search method in the reference software.

5.3.1 Analysis of Computational Budget

Here, the computational complexity is calculated in terms of the number of the required operations for one n×m block (i.e. *4×4 or 4×8 or 8×8 or 8×16 or 16×8 or 16×16* block) in the quarter-pixel accurate ME process. The computational complexity of the interpolate and search method in the reference software for one n×m block consists of the following terms: interpolation, search, and matching process for the half-pixel (*HP*) and the quarter-pixel (*QP*) refinements. The whole computational complexity can be formulated as follows:

$$CC_{total}(n\times m) = CC_{HP}(n\times m) + CC_{QP}(n\times m). \quad (5.6)$$

The total numbers of half and quarter-pixels in the interpolation process for one n×m block can be calculated from equations (5.7) and (5.8), respectively.

$$H_{n\times m}=3n\times m+2(n+m)+1. \quad (5.7)$$

$$Q_{n\times m}=12n\times m+10(m+n)+10. \quad (5.8)$$

Note that in these equations, $H_{n\times m}$ and $Q_{n\times m}$ represent the number of half-pixels and quarter-pixels within a search range of *[-0.5, 0.5]* and *[-0.75. 0.75]* for an *n×m* block, correspondingly.

By considering the 6-tap filter, which is used in the half-pixel refinement, producing one half-pixel requires six add/subtract operations and one shift operation. For the quarter-pixel refinement with 2-tap linear filter, generating one quarter-pixel needs two add operations and one shift operation. The search range in the quarter-pixel ME is limited to *(-1, +1)* and *17* search points in total are checked. Therefore, one *n×m* block requires n×m×17 subtractions, *n×m×17* absolute operations to calculate the absolute differences, *n×m×17-17* additions to sum up the differences, and at least two comparisons for finding the best position with minimum distortion. Therefore, the computation complexity of one *n×m* block can be approximated, as given in equation (5.9).

$$CC_{(n\times m)} \approx 7\,H_{n\times m}+3\,Q_{n\times m} + (n\times m\times 51-15)_{search\ and\ match} \approx 108\ n\times m + 37\ (m+n) + 22. \quad (5.9)$$

As for the NFM algorithm, its computational complexity for one *n×m* block is only *84* add, *16* shift and one comparison operations in the half-pixel stage, and, *168* add, *24* shift, and one comparison operations in the quarter-pixel stage. The computational complexity of different block sizes and for an MB with 41 modes in the interpolate and search method and the NFM algorithm are given in Table 5.1.

Taking into account 41 blocks in each MB, the computational complexity of an MB can be expressed by equation (5.10).

$$CC_{MB} \approx \sum_{mode(n\times m)=1}^{41} CC_{n\times m}\,. \quad (5.10)$$

Table 5.1 : The computational complexity of the interpolate and search method and the NFM algorithm for different block sizes.

Block size	Number of operations (1)	Number of operations (2)
16×16	28854	294
8×16	14734	294
8×16	14734	294
8×8	7526	294
8×4	3922	294
4×8	3922	294
4×4	2046	294
MB (41 modes)	213382	12054

(1) Interpolate and search method; (2) NFM algorithm

Now, by using equation (5.10), the computational complexity of a certain specification (*CS*) can be formulated, as shown in equation (5.11).

$$CC_{CS} \approx CC_{MB} \times ((W \times H)/256) \times (\#\ reference\ frame) \times (\#\ frame\ per\ second)). \quad (5.11)$$

where the W and H stand for width and height of the given frame size and (W×H)/256 term determines the number of MBs in each frame. Table 5.2 lists the computation complexity of the NFM algorithm and the search and interpolate SME algorithm in the H.264 reference software for some specifications. From the results of Table 5.2, we can see that when all VBS are considered, the NFM algorithm can save 94.35% of the computational complexity in comparison to the interpolate and search method for the same specification. When a lower number of VBS modes are used, which are usually higher block sizes, the NFM algorithm can even save more computations. As an example consider the last row of Table 5.2 where only 8×8 and above modes are employed. In this case, the NFM algorithm saves 97.76% of the computational complexity relative to interpolate and search scheme. The reason is that the NFM algorithm requires a certain number of operations (i.e. 294 operations) for each block size whereas the interpolate and search method requires more computations for the higher block sizes, as shown in Table 5.1.

Table 5.2 : The computational complexity of the interpolate and search method and the NFM algorithm for some specifications.

Specifications	Features	$CC_{Interpolate\ \&\ search}$	CC_{NFM}
QCIF(15 fps)	VBS(all), MRF(5)	$1{,}584.3\times10^6$	89.5×10^6
CIF(30 fps)	VBS(all), MRF(5)	$12{,}674.8\times10^6$	716×10^6
SD480p(30 fps)	VBS(all), MRF(5)	$43{,}209.8\times10^6$	$2{,}440.9\times10^6$
HD1080p(30 fps)	VBS(all), MRF(5)	$259{,}259.1\times10^6$	$14{,}646\times10^6$
HD1080p(30 fps)	VBS(8×8-16×16), MRF(2)	$57{,}296.4\times10^6$	$1{,}286\times10^6$

The low computational complexity characteristic of the NFM algorithm provides an attractive opportunity for the design of low cost and low power VLSI hardware architectures for the H.264/AVC SME targeting resource-constrained applications.

5.3.2 Analysis of Memory Access

In addition to the computational complexity, the memory access requirement is another important issue especially for designs with limited resources. That is because the memory access requirement affects the traffic load of reading/writing data from/into system memory, I/O, and interconnection bandwidth, and thus the power consumption and hardware utilization of the system. We define the memory access requirement in terms of the number of bits that one $n\times m$ block has to access memory and calculate it for the interpolate and search SME algorithm in the reference software and the proposed NFM algorithm. Not to mention that the memory access requirement can be reduced by careful data reusing and data scheduling schemes.

In general, the memory access of one $n\times m$ block in the interpolation method can be divided into two main parts: producing the sub-pixels of reference frame(s) and the search and match process in the half and quarter-pixel refinements. For producing the half-pixels of one $n\times m$ block, each half-pixel requires six read operations and one write operation to save the produced half-pixel in the memory. While for producing one quarter-pixel, it decreases to two read and one write operations. Besides, in total, there

are 17 search points in the half and quarter-pixel refinements that each one at least requires $2\times n\times m$ read and 17 write operations for saving the search point results. Therefore, the memory access requirement of one n×m block in the interpolate and search method can be calculated by equation (5.12), where each pixel is represented by 8 bits.

$$MA_{(n\times m)} = MA_{HP} + MA_{QP} = [(8\times 6\times n\times m + 8\times 9\times n\times m)_{rd} + (8\times n\times m + 8\times 9))_{wr}]_{HP} + \\ + [(8\times 2\times n\times m + 8\times 8\times n\times m)_{rd} + (8\times n\times m + 8\times 8)_{wr}]_{QP}. \quad (5.12)$$

Table 5.3 : The memory access of the interpolate and search method and the NFM algorithm for different block sizes.

Block size	MA interpolate and	MA NFM
16×16	55432	144
16×8	27784	144
8×16	27784	144
8×8	13960	144
8×4	7048	144
4×8	7048	144
4×4	3592	144
MB (41modes)	392648	5904

As for the NFM algorithm, the memory requirement for one n×m block is reduced to nine read operations that are needed for reading nine integer SAD values with 16-bit width. The memory access requirement of different block sizes and for an MB with 41 modes in the interpolate and search method and the NFM algorithm are given in Table 5.3. According to our analysis, the memory access requirement of the NFM algorithm is only 144 bit per block, which is independent of the block size. Consequently, it can save 98.5% of the memory access requirement compared to the interpolate and search method for each MB with all its 41 modes.

Similar to the computational complexity, we can calculate the memory access requirement of a certain specification, as follows:

$$MA_{MB} \approx \sum_{mode(n\times m)=1}^{41} MA_{n\times m}. \quad (5.13)$$

$$MA_{CS} \approx MA_{MB} \times ((W\times H)/256) \times (\#\ reference\ frame) \times (\#\ frame\ per\ second). \quad (5.14)$$

In Table 5.4 the memory access requirement of the interpolate and search algorithm as well as the NFM algorithm for some specifications are given. As seen in this table, the NFM algorithm can save 98.5% of the memory access requirement relative to the interpolate and search method when all 41 modes of MB is enabled. The reason is that the NFM algorithm avoids the interpolation process that significantly reduces its memory access requirement. From the hardware point of view, a higher memory access leads to a higher I/O data bus traffic and interconnection requirements, which have a significant effect on the performance of the hardware architecture. Besides, reduction of memory access is very beneficial to power constrained applications because in the image and video processing systems the data access requirement takes 50%-80% of power consumption (Y.-H. Chen et al., 2008).

Table 5.4 : The memory access of the interpolate and search method and the NFM algorithm for some specifications.

Specifications	Features	$MA_{interpolate\ \&\ search}$	MA_{NFM}
QCIF(15 fps)	VBS(all), MRF(5)	$2{,}915.4\times10^6$	43.8×10^6
CIF(30 fps)	VBS(all), MRF(5)	$23{,}323\times10^6$	350.7×10^6
SD480p(30 fps)	VBS(all), MRF(5)	$79{,}511.2\times10^6$	$1{,}195.6\times10^6$
HD1080p(30 fps)	VBS(all), MRF(5)	$477{,}067.3\times10^6$	$7{,}173.4\times10^6$
HD1080p(30 fps)	VBS(8×8-16×16), MRF(2)	$108{,}090.2\times10^6$	$1{,}574.6\times10^6$

5.4 Optimization Techniques for Computational Complexity Reduction

5.4.1 Fast NFM Algorithm

To reduce the computational complexity of the NFM algorithm, we propose fast NFM (FNFM) algorithm, which is based on exploiting the associativity and the distributivity characteristics of the equations in the NFM algorithm. By a careful rearrangement of the NFM's equations, finding and reusing the common terms, which participate in calculation of equations, more computation can be saved. In the FNFM algorithm, during the half-pixel refinement when the coefficients (i.e. A_1 - A_9) are produced, they are concurrently reused in the quarter-pixel stage to produce the SAD values of the quarter-pixels. Further, in the quarter-pixel stage like the half-pixel stage, the equations are categorized and all common terms are extracted. Then, these common terms are reused anywhere which is needed. For example in quarter-pixel refinement and from equation (5.1), QSAD (-1/4,-1/2) is as follow:

$$QSAD(-1/4,-1/2) = A_1/64 - A_2/32 - A_3/16 + A_4/8 + A_5/16 - A_6/4 + A_7/4 - A_8/2 + A_9 \quad (5.15)$$

After extracting the common terms in (5.15), the new representation is as follows:

$$QSAD(-1/4,-1/2) = (A_1/64 - A_2/32) - (A_3/16 - A_4/8 + A_6/4) + A_5/16 - HSAD(-1/2, 0). \quad (5.16)$$

Where not only $(A_7/4 - A_8/2 + A_9)$ term is replaced by $HSAD(-1/2, 0)$, which is calculated in half-pixel refinement, but also it and $(A_1/64 - A_2/32)$, $(A_3/16 - A_4/8 + A_6/4)$, terms can be reused in calculation of $QSAD(1/4, -1/2)$:

$$QSAD(1/4,-1/2) = (A_1/64 - A_2/32) + (A_3/16 - A_4/8 + A_6/4) + A_5/16 - HSAD(-1/2, 0). \quad (5.17)$$

Table 5.5 shows the computational complexity of the NFM and the FNFM algorithms. In average, the FNFM algorithm save 46.28% or 96.92% of the computational complexity for each MB with 41 modes in comparison with the NFM algorithm or the interpolate and search method. However, both algorithms have similar coding efficiency. Note that lower computational load leads to lower hardware cost and power consumption.

Table 5.5 : Complexity comparison of the NFM[1] and the FNFM[2] algorithms.

Block size	Number of operation[1]	Number of operation[2]
16×16	294	160
16×8	294	160
8×16	294	160
8×8	294	160
8×4	294	160
4×8	294	160
4×4	294	160
MB (41 modes)	12054	6560

5.4.2 SAD Pixel Truncation and Mode Filtering Techniques

The pixel truncation is well known in literature as a powerful technique for complexity reduction of block matching ME algorithm with acceptable video quality degradation. All previous works have been focused on the effect of pixel truncation for fix and variable block size IME (Bahari et al., 2009; He & Liou, 1997), but none for SME. In video coding systems, each pixel is generally represented by 8 bits. It has shown that up to 4 bits can be truncated with a little video quality degradation. Because of using integer SAD values in the FNFM algorithm, the pixel truncation cannot be used here. Therefore, we have proposed the SAD truncation technique to reduce the complexity of the FNFM algorithm. The maximum bit length of integer SAD values is 16 bits, which happens for the SAD calculation of a 16×16 MB. According to our evaluation, up to 4 bits of SAD values can be truncated with very little video quality degradation. However, from the hardware point of view, the SAD truncation technique can lead to area saving and power consumption reduction at hardware implementation.

The reason is that the hardware cost of an operation is proportional to its operands bit length. Therefore, reducing the bit length can lead to lower area and power consumption.

Our last optimization technique is 4×4 mode filtering that can save 39% of the computational complexity, memory access, and the processing time of each MB in the FNFM algorithm. The reason is that there are 16 4×4 modes in each MB (i.e. 39% of the modes in each MB). Besides, the computational complexity and memory access requirement of the FNFM algorithm does not depend on the size of blocks and all block modes are processed in a sequential manner. From a hardware point of view, 4×4 mode filtering increases the processing ability and lowers the power consumption.

5.5 Bit-Serial Hardware Architecture Design for SME in H.264/AVC

Generally after selection and evaluation of an algorithm, it is decomposed into its components to provide insight into how to design the best possible hardware architecture for it. In our particular case, the FNFM algorithm is decomposed into addition, subtraction, and shift components. These components can be efficiently implemented in hardware, either using bit-parallel or bit-serial architectures. Whereas bit-parallel architectures process n bits (where n is the world length of each data) at every clock cycle, bit-serial architectures process one bit at each clock cycle. As a result, bit-parallel architectures are suitable for applications that require high processing capacity whereas bit-serial architectures are good candidates for application with low to moderate throughput. Since the FNFM algorithm has very low computational complexity and memory access requirement, it can be embedded in bit-serial architecture with small hardware cost while it demonstrates a high processing ability.

Due to the above-mentioned advantages and other advantages that were described in Section 4.2, we choose bit-serial architecture for implementing the FNFM algorithm.

5.5.1 The Proposed Bit-Serial Architecture

The block diagram of the proposed bit-serial architecture for the FNFM algorithm is shown in Figure 5.2. It consists of four main modules: half-pixel ME, quarter-pixel ME, control, and MV predictor modules that are described in the following sub-sections.

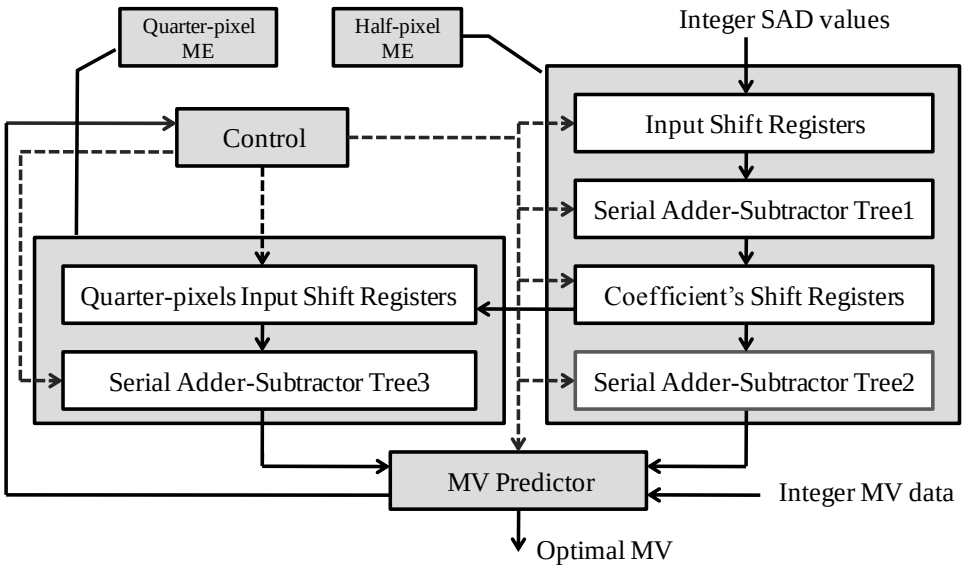


Figure 5.2 : The bock diagram of the proposed bit-serial design for the FNFM algorithm.

5.5.2 Half-Pixel ME Module

This module is responsible for calculating SAD values for half-pixel accurate MVs. We have proposed bit-serial pipeline architecture for this module, which is an efficient and compact implementation of the FNFM's equations. The half-pixel ME module consists of two parts. In the first part A_1 - A_9 coefficients are produced that are used in second part for calculating $HSAD_1$ - $HSAD_9$. The schematic diagrams for the hardware architecture of these two parts are shown in Figure 5.3 and Figure 5.4, respectively.

They only use four basic components: shift registers, bit-serial adders, bit-serial subtractors, and delay elements.

Shift register are responsible for division operations to produce the fractional terms that work serially. To avoid overflow and for having the correct calculations, the bit length of input data (i.e. integer SAD values) are extended by three bits. The next two components are bit-serial adder and bit-serial subtractor, which are responsible for addition and subtraction operations.

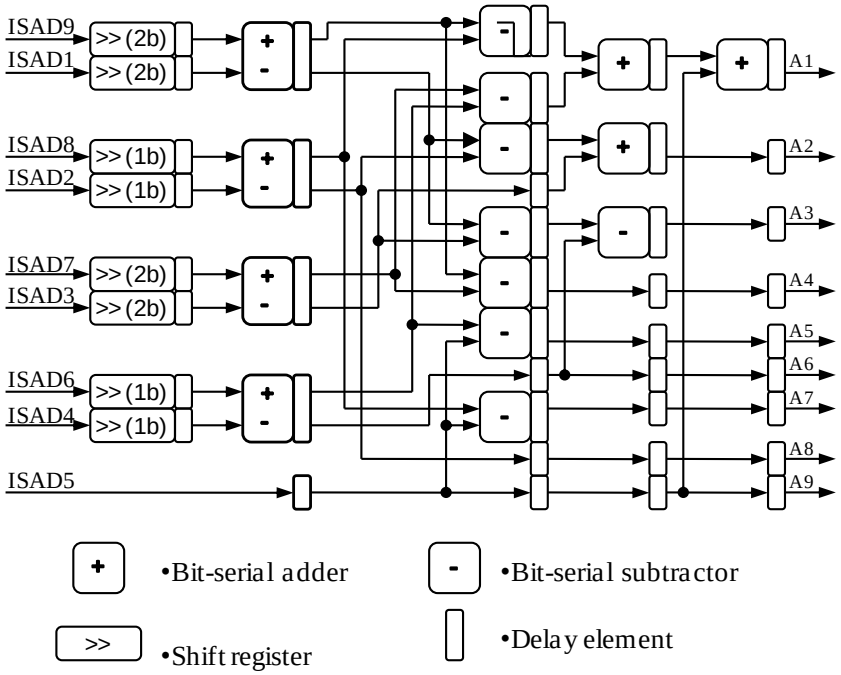


Figure 5.3 : The schematic diagram of the half-pixel module: part one.

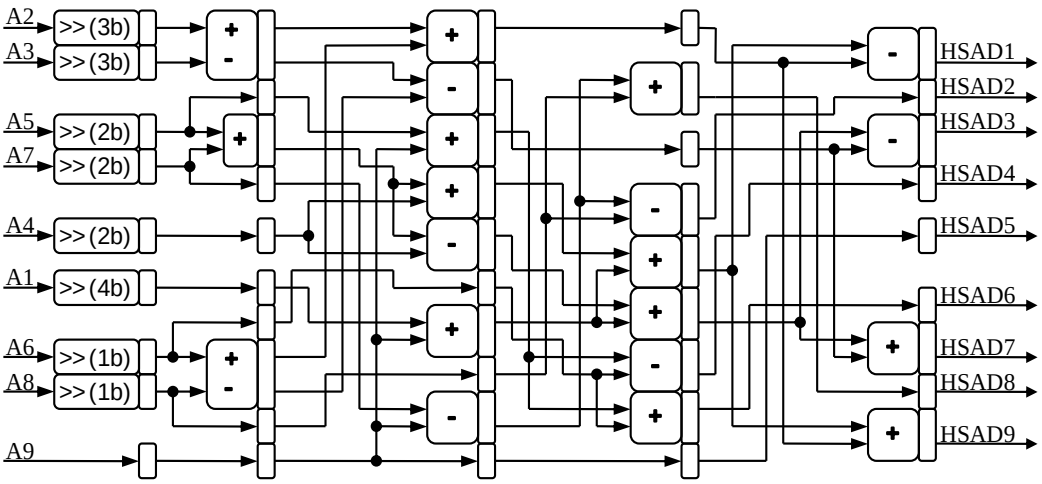


Figure 5.4 : The Schematic diagram of the half-pixel module: part two.

The schematic diagrams of the bit-serial adder and the bit-serial subtractor are shown in Figure 5.5 and Figure 5.6, respectively. In each clock cycle one bit of A and the B inputs are fed into the bit-serial adder, starting from the least significant bits (LSBs). The produced carry-out (C_{out}) is kept in a flip flop and is used as carry-in (C_{in}) to add with the next coming bits. At the beginning of two new words addition, the C_{in} should be set to zero because there is no carry before theirs LSBs to add. In terms of time, the addition of two n-bit words takes n clock cycles. Due to the use of the two's complement representation in our architecture, the subtraction of *A-B* is replaced by $A+\overline{B}+1$. The bit-serial subtractor architecture is shown in Figure 5.6 where B input is inverted and C_{in} is set to one at the beginning of two new words subtraction.

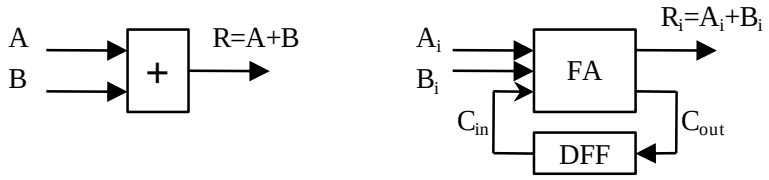


Figure 5.5 : The schematic diagram of the bit-serial adder architecture.

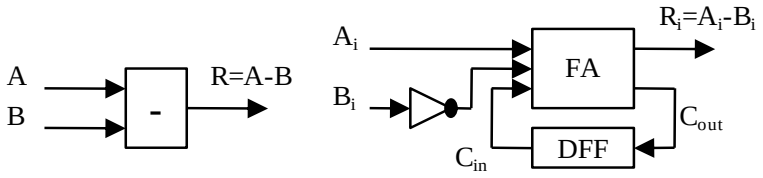


Figure 5.6 : The schematic diagram of the bit-serial subtractor architecture.

The last components which are used in our architecture are D flip flops. Adequate numbers of D flip flops are used in each part as delay elements to avoid data hazard and prevent any interference between the successive stages.

5.5.3 Quarter-Pixel ME Module

The proposed hardware architecture of the quarter-pixel ME module is shown in Figure 5.7, which its structure is similar to the half-pixel accurate ME module. This module is responsible for calculating the quarter-pixel accurate SAD values. These values are used in MV predictor module to find the best quarter-pixel accurate MV. To save the area further and increase the speed of the design, the calculated coefficients in the half-pixel accurate ME module (i.e., $A_1 \sim A_9$) are reused in this module. In addition, similar to the half-pixel accurate ME module, the quarter-pixel accurate ME module is benefitted from the use of the simplified equations that reuse the common terms.

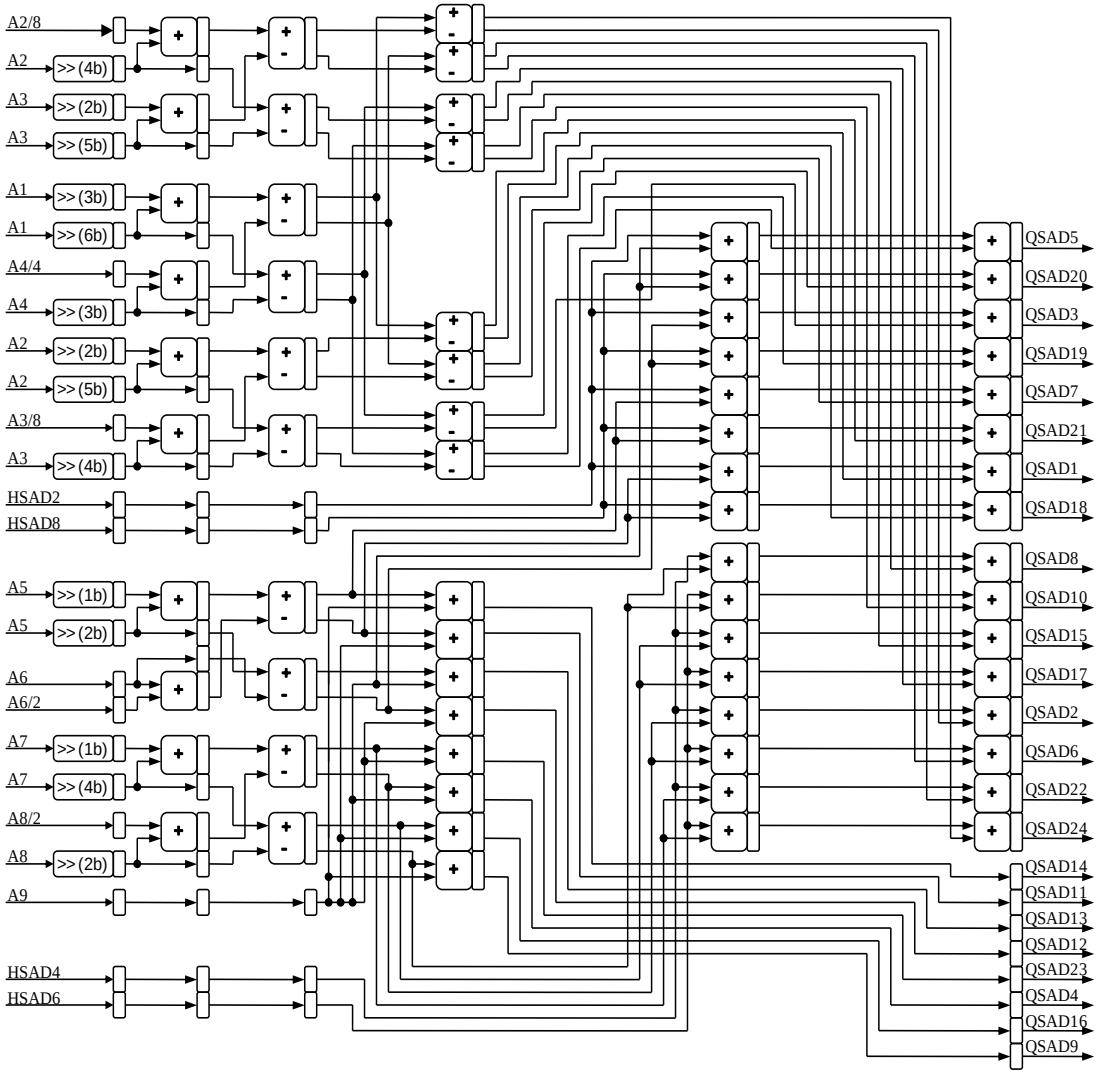


Figure 5.7 : The schematic diagram of the quarter-pixel module.

5.5.4 Control and MV Predictor Modules

The last two modules of our bit-serial architecture are the control and the MV predictor modules. The control module is responsible for coordinating the tasks of the other modules, synchronization between them, and management of the data flow within the system. In general, Finite State Machine (FSM) is used for implementing the control module, which uses counters for transition between its states. These counters are usually increased or decreased by one at a time and their outputs are used for controlling the other modules through the system. This approach has two disadvantages: first, the transition number between the state bits is so large that leads to more dynamic power consumption. Second, due to the distribution of control signals throughout the system with different propagation delays, it is possible that the received codes go through unwanted state. To deal with these disadvantages, in our design we use Gray counter in which only one bit is changed at a time during counting. As a result, only one transition is done between states, which not only reduces the power consumption but also decreases the probability of the wrong states.

The MV predictor module is responsible for finding the optimal MVs with half-pixel and quarter-pixel accuracies, which is based on equation 5.5. The bit-serial hardware architecture design for this module is a tough problem due to its complex structure and consequently it has lower throughput and performance in comparison to the previous discussed modules. Therefore, the MV predictor module requires a dedicated architecture to avoid performance degradation of the whole design. To cope with this problem, we have designed bit-parallel pipeline architecture for the MV predictor module at the price of more area cost and lower operating frequency.

5.5.5 The Proposed Power Reduction Technique

Two kinds of shift registers are used in our architecture including shift right arithmetic registers (SRARs) and shift right register (SRRs). Nine shift registers as the inputs of Figure 5.3, nine shift registers as the inputs of Figure 5.4, and 25 shift registers as the inputs of Figure 5.7 are SRARs. Nine shift resisters as the outputs of Figure 5.4 and 24 shift registers as the outputs of Figure 5.7 are SRRs. At each clock cycle, our architecture only needs one bit from every shift register, but 15 bits are shifted inside each shift registers that increase the power consumption.

To save the power consumption, we propose a new power reduction technique for SRRs, which is named as Demultiplexers-Registers Combination (DRC). The proposed power reduction technique saves a significant part of power consumption in SRRs. In Figure 5.8, the DRC technique is illustrated by an example for nine shift resisters as the outputs of Figure 5.4. In the left side of Figure 5.8, m n-bit SRRs are working in parallel that for the above-mentioned example m and n are nine and 15, respectively. In each SRR, at the first clock cycle one bit is shifted to the right and this will be repeated until n^{th} clock cycle. At the next clock cycle, all bits of each SRR are placed at their outputs, which are used in the MV predictor module and this procedure is periodically repeated. Therefore, at each clock cycle 9×15 bits are shifted that lead to a lot of power consumption. To reduce the power consumption, we propose the DRC technique with the same functionality of SRRs, which its architecture is shown in the right side of Figure 5.8. In this architecture R_1-R_n are m-bit registers with enable signals, and for the above-mentioned example, m and n are 9 and 15. At the first clock cycle, 9 bits are placed into the one register starting from R_1. Note that the inputs of R_1-R_n are connected together and the right register at the proper clock cycle is chosen by enable signal. In this method, the job of demultiplexer is done by the enable signals that leads to area saving and power reduction.

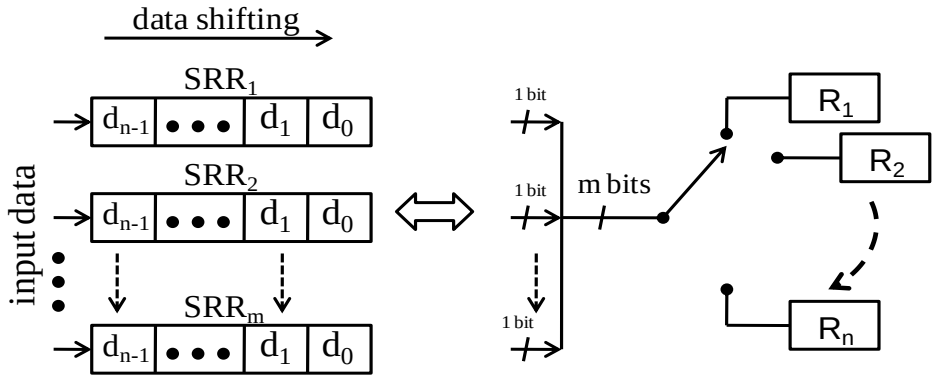


Figure 5.8 : Illustration of DRC technique.

5.6 Experimental Results and Comparison

5.6.1 Simulation Results

To evaluate the performance of the proposed algorithm and optimization methods, we embed them into the H.264 reference software and assess them with several image sequences videos at the following test conditions:

- The search range is *[-16, 16]*.
- The number of reference frames is 1.
- Rate-distortion optimization is on.
- Frequency for encoded bit stream is 30.
- Number of coded pictures is 300 for all sequences.

Table 5.6 shows the simulation results of the FNFM algorithm compared with the integer motion estimation of H.264 and Model 1 algorithm (Suh & Jeong, 2004) with half-pixel accuracy where our algorithm significantly outperforms them. For instance, compared to the full search IME and Model 1 algorithms, the proposed algorithm can typically lead between (0.9-4.2)/(0.3-1.6) dB PSNR improvement or (19%-50%)/(9%-26%) in bit rate reduction, respectively.

Table 5.6 : Comparison between the proposed SME algorithms and the full search IME.

Sequence	FNFM vs. IME		FNFM vs. Model 1	
	Δbit rate	ΔPSNR	Δbit rate	ΔPSNR
Akiyo	-31.20%	1.92 dB	-14.18%	0.79 dB
Carphone	-24.04%	1.29 dB	-9.16%	0.46 dB
Foreman	-42.67%	2.02 dB	-15.07%	0.60 dB
Mobile	-50.74%	4.19 dB	-25.58%	1.57 dB
News	-24.53%	1.47 dB	-11.52%	0.63 dB
Salesman	-19.24%	1.04 dB	-8.90%	0.47 dB
Stefan	-34.32%	2.69 dB	-13.42%	0.90 dB
Tennis	-21.78%	0.93 dB	-9.00%	0.36 dB

In addition to the original half-pixel interpolation free algorithm and the full search integer motion estimation algorithm, we compare our algorithm to the quarter-pixel accurate interpolate and search algorithm of the H.264 reference software for 4CIF and HD1080 test sequences, as shown in Figures 5.9 and 5.10. Besides, since our algorithm has a low complexity feature, we compare it to the all-binary algorithm (Celebi et al., 2008) as another low complexity algorithm. Figures 5.11 and 5.12 show the comparisons of rate-distortion curves between the FNFM, optimized FNFM, all-binary, and search and interpolate SME algorithms for Tennis (CIF) and Foreman (Source Input Format (SIF)) test sequences where all of these algorithms have quarter-pixel accuracy. In this figures, the rate-distortion curves of the all-binary algorithm are borrowed from the work of Celebi et al. (2008).

As seen in Figures 5.9 and 5.10, the FNFM algorithm has a small quality drop relative to the H.264 interpolate and search method. In addition, it significantly outperforms the all-binary SME algorithm in terms of video quality. According to our simulation results, the average PSNR loss of our algorithm compared to the interpolate and search method is less than 0.38 dB. However, as shown in Section 5.3, our algorithm reduces more than 96% of the computation budget and the memory access compared with the interpolate and search method. Compared to the FNFM algorithm, its

optimized version leads to a negligible quality drop (i.e. less than 0.06 dB PSNR). In addition, from the hardware point of view, the proposed optimization techniques lead to area cost reduction and power consumption saving at the hardware side.

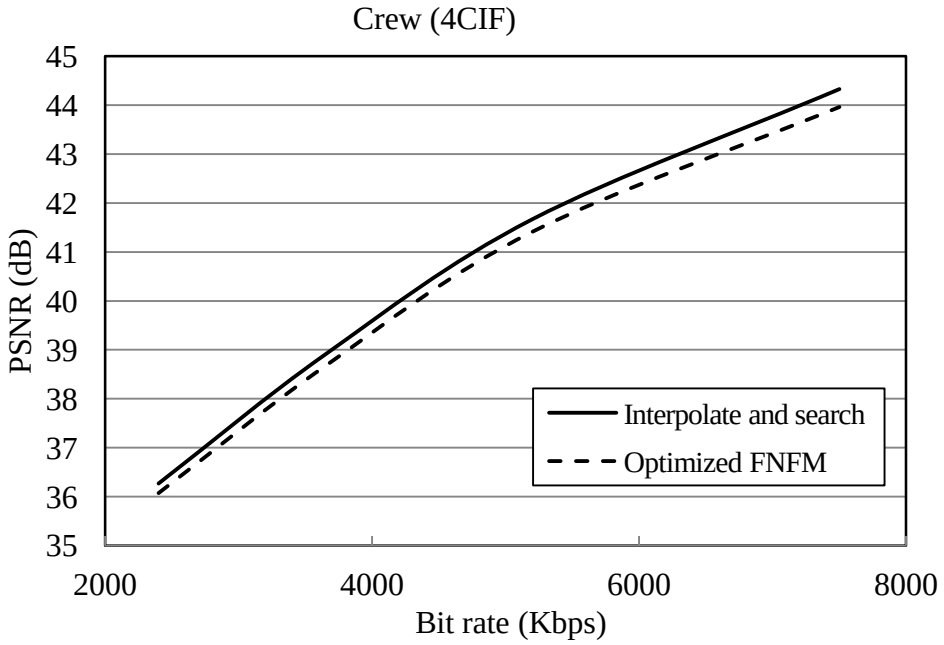


Figure 5.9 : R-D plot of Crew test sequence for interpolate and search SME and optimized FNFM algorithms.

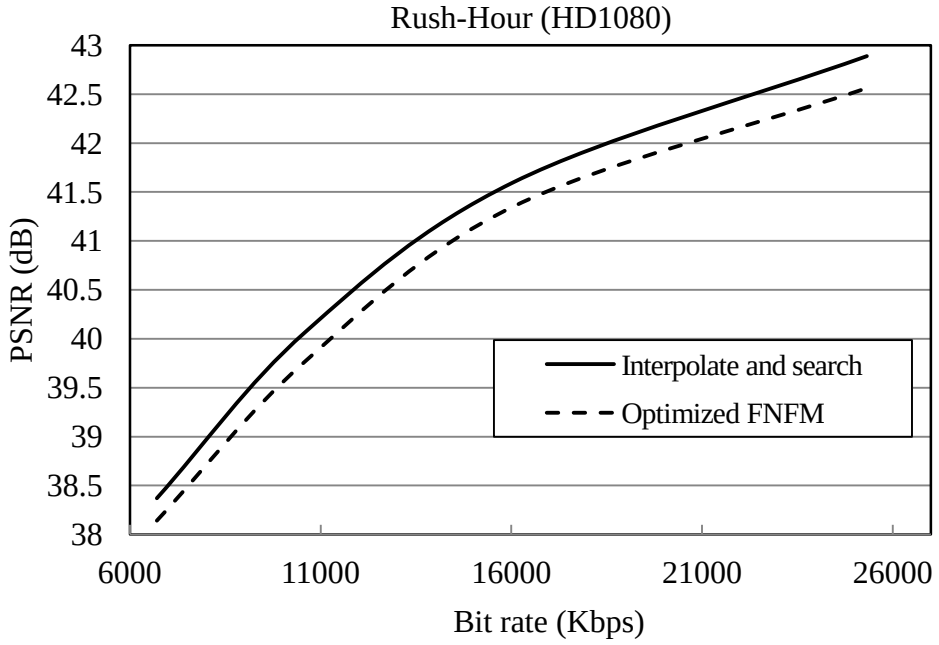


Figure 5.10 : R-D plot of Rush-Hour test sequence for interpolate and search SME and optimized FNFM algorithms.

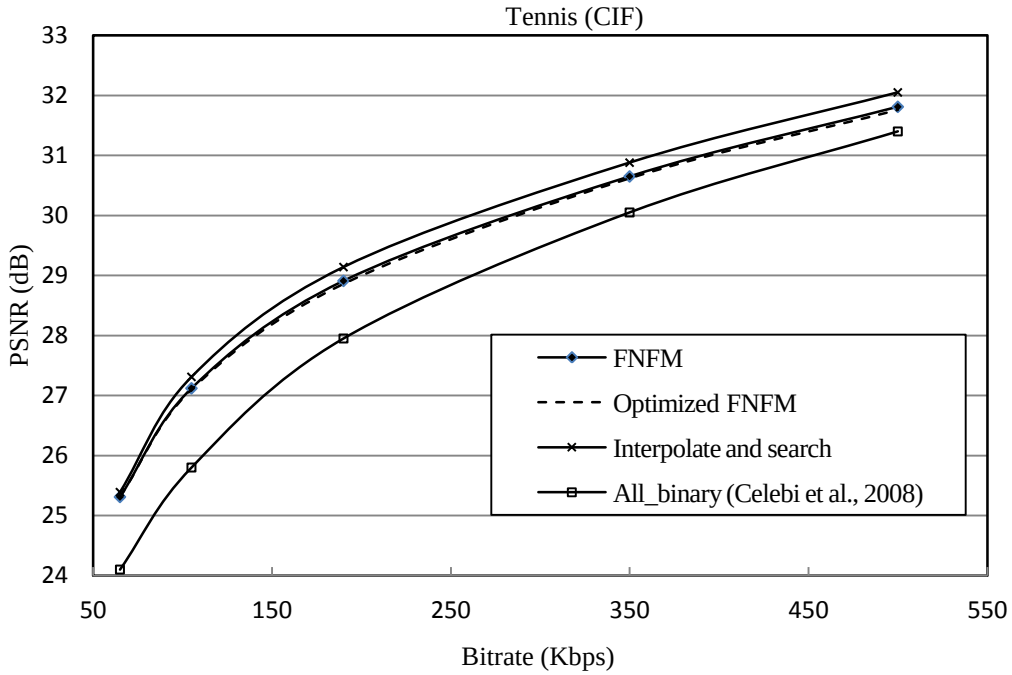


Figure 5.11 : Comparison of RD curves between FNFM, optimized FNFM, interpolated and search, and all-binary algorithms for Tennis test sequence.

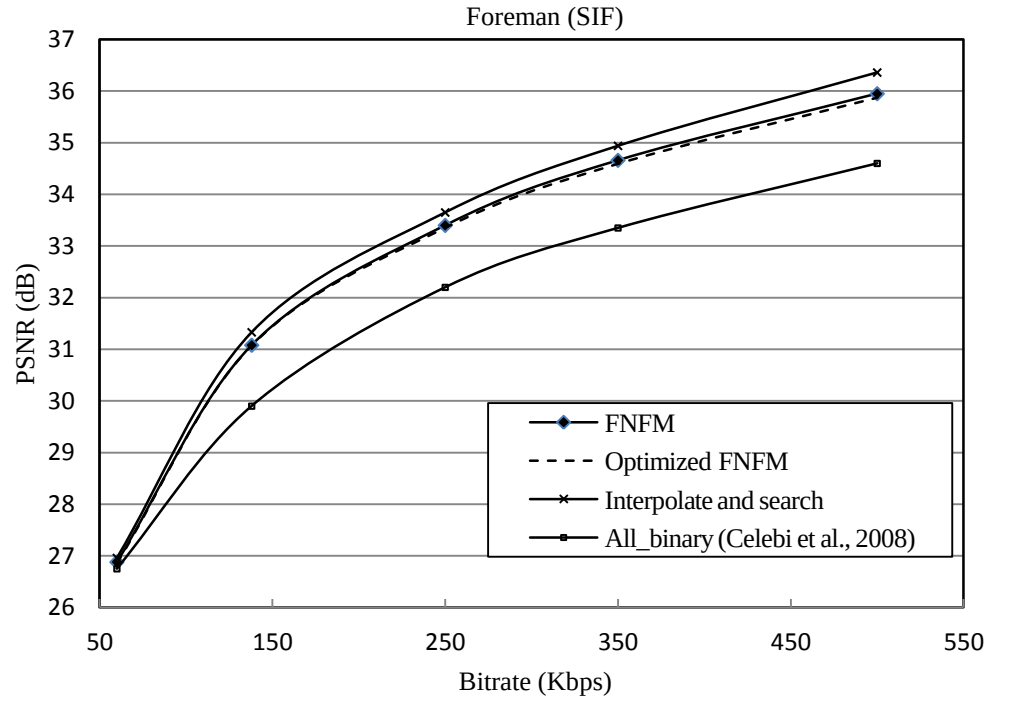


Figure 5.12 : Comparison of RD curves between FNFM, optimized FNFM, interpolated and search, and all-binary algorithms for Foreman test sequence.

5.6.2 Implementation Results and Comparison

The proposed SME architecture with the quarter-pixel accuracy for H.264/AVC is implemented in Verilog HDL and synthesized in Silterra 0.18 μm technology using Synopsys Design Compiler. The proposed architecture costs at 20.3 K gates and approaches 100% hardware utilization, which is suitable for area-constrained applications. Due to the interpolation free feature of the proposed algorithm, the memory bandwidth of the proposed design is very small (i.e. 2.7 Kbits). The reason is that for calculating the SME motion vector of each mode, the proposed algorithm needs 9 12-bit SAD values. Therefore, the memory bandwidth of each MB with 25 modes required $9 \times 12 \times 25 = 2.7$ Kbits. The proposed design can process one block with any given size in 15 clock cycles. As a result, the proposed architecture is able to process one MB with 41 modes or 25 modes (i.e. when 4×4 mode filtering is applied) in 615 or 375 clock cycles, respectively. With a maximum clock frequency of 255 MHz, our design can process up to 414 K MB/s or 680 K MB/s with 41 or 25 modes in each MB, respectively. Consequently, our design is able to support SME of HD1080 format with one reference frame, 30 fps and with 41 or 25 modes in each MB at the clock frequency of 149.44 MHz or 88.3 MHz.

In addition to HD1080 format, the proposed architecture can be reused for other formats with different specifications. Table 5.7 shows some examples and their specifications. Not to mention that by using the hardware parallelism, the proposed architecture can support higher resolutions. For example, as shown in case (g) of Table 5.7, two sets of the proposed design can support quarter-pixel accurate ME of quad full HD (QFHD) resolution (i.e. 4096×2160) with all blocks modes except 4×4 modes, one reference frames and 30 fps under the clock frequency of 199.4 MHz.

Comparison between the proposed architecture and the previous SME hardware architectures are shown in Table 5.8. This comparison is based on the introduced design

metrics of Chapter 3 including algorithm quality, silicon area, operating frequency, and throughput. Regarding the algorithm quality, among all of the SME architectures, the proposed design has the highest quality lost that is about 0.37 dB on average. However, this amount of quality loss can be acceptable especially when considering the advantages of the proposed algorithm. For instance, compared with other designs the proposed design has the lowest computational complexity and memory access requirement due to its interpolation free characteristic.

On the subject of silicon area, the proposed design significantly outperforms the other architectures. To be more specific, we provide the percentage of the improvement for our architecture relative to the previous designs in terms of area reduction. This improvement is obtained from the difference of the areas of the two designs divided by the area of the used architecture. According to our calculation, the proposed architecture provides between 57.71%-95.47% improvement in terms of area reduction relative to the prior designs. The reason is that our architecture is based on the interpolation free algorithm with very small computational complexity and uses bit-serial structure for hardware implementation. Please note that the lower hardware cost is beneficial to power consumption and can lead to dynamic power and static power reduction, as described in Section 4.2.

Table 5.7 : Reusing of the proposed SME architecture in other specifications.

Case	Specifications[(1)]	Throughput K MB/s	Frequency MHz
(a)	QCIF(VBS(all) and MRF(5))	14.85 K MB/s	9.13
(b)	CIF(VBS(all) and MRF(5))	59.4 K MB/s	36.53
(c)	SD576(VBS(all) and MRF(5))	243 K MB/s	149.44
(d)	HD720(VBS(all) and MRF(1))	108 K MB/s	66.4
(e)	HD720(VBS(all except 4×4) and MRF(5))	540 K MB/s	202.50
(f)	HD1080(VBS(all except 4×4) and MRF(2))	486 K MB/s	182.25
(g)	QFHD(VBS(all except 4×4) and MRF(1))	1036 K MB/s	194.4[(2)]

[(1)] With 30 fps. [(2)] Using two SME architectures.

The next design metric is operating frequency that is affected by the frame size and the number of the supported block modes in each design. When a higher frame size and more block modes are used, the processing time is increased. As a result, for meeting the real-time constraint, the clock frequency should be increased too. Please note that a lower operating frequency usually means a lower dynamic power consumption. Therefore, lowering the working frequency can be advantageous for low power applications. Among all of the designs in Table 5.8, only the work of T.-Y. Kuo et al. (2007) has a lower operating frequency than the proposed design. The reason is that their design is able to support HD720 format but not higher resolutions. In a similar scenario, our design can support HD720p format at the operating frequency of 66.4 MHz (i.e. case (d) in Table 5.7).

The last design metric is throughput that depends on the number of processed modes in each MB as well as the operating frequency. As seen in Table 5.8, only the work of Tsung et al. (2009) achieves a higher throughput than ours. However, this advantage comes at the price of the highest area cost. For example, the design of Tsung et al. (2009) occupies 22.06 × higher area cost relative to our design. In addition, this design avoids half-pixel interpolation to reduce the processing time. As a result, it needs another unit for handling the luminance motion compensation. Besides, by using hardware parallelism, our design can achieve a higher throughput than the work of Tsung et al. (2009) with much lower area cost, as shown in case (g) of Table 5.7.

Compared to the previous SME designs, the proposed design demonstrates a good performance in terms of design metrics, computational complexity and memory access requirement, however; it has a disadvantage. Since our algorithm avoids interpolation, our approach needs an interpolation unit for motion compensation. Fortunately, this is not a serious problem. Analysis of SMV by Y.-J Wang et al. (2007) reveals that more than 90% of the SMV is at the search center in all kinds of video content. This means

that in most cases after the SME the SMV and the IMV are the same. Therefore, only a small portion of the sub-pixels is required to be interpolated after the SME. As a result, the required interpolation unit of our method could be small. In other words, it should roughly save about 90% of area and power consumption compared to the interpolation units of the previous designs.

Table 5.8 : Design metrics evaluation and comparison of SME architectures.

Architecture	**Design Metrics**	
Y.-C. Chen et al. (2004)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	79.3
	Operating frequency (MHz)	100
	Throughput (MB/s)	49 K
C. Yang et al. (2006)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	189
	Operating frequency (MHz)	200
	Throughput (MB/s)	250 K
Wu et al. (2010)	Quality loss (dB PSNR)	-0.00
	Silicon area (K gates)	311.7
	Operating frequency (MHz)	154
	Throughput (MB/s)	250 K
Y.-J Wang et al. (2007)	Quality loss (dB PSNR)	-0.1 to -0.2
	Silicon area (K gates)	48
	Operating frequency (MHz)	100
	Throughput (MB/s)	50 K
Song et al. (2007)	Quality loss (dB PSNR)	-0.1
	Silicon area (K gates)	203.2
	Operating frequency (MHz)	200
	Throughput (MB/s)	250 K
T.-Y. Kuo et al. (2007)	Quality loss (dB PSNR)	-0.043
	Silicon area (K gates)	62.2
	Operating frequency (MHz)	70
	Throughput (MB/s)	71.3 K
Tsung, Chen, Ding, Tsai et al et al. (2009)	Quality loss (dB PSNR)	-0.02
	Silicon area (K gates)	448
	Operating frequency (MHz)	280
	Throughput (MB/s)	830 K
Our design	Quality loss (dB PSNR)	-0.37
	Silicon area (K gates)	20.3
	Operating frequency (MHz)	88.3
	Throughput (MB/s)	680 K

5.7 Summary

The sub-pixel motion estimation, together with the interpolation of reference frames, is a computationally extensive part of the H.264 encoder that increases the memory access requirement 16-times for each reference frame. Due to the huge computational complexity and the memory access requirement of the H.264 SME, its hardware architecture design is a challenging job especially for area-constrained applications with medium and large resolutions. To solve the above difficulties, we proposed an algorithm and its hardware architectures along with several optimization techniques. In algorithm level, we presented the NFM for SME with quarter-pixel accuracy, which reduced the computational budget by 94.35% and the memory access requirement by 98.5% in comparison to the standard interpolate and search method. In addition, a fast version of the proposed algorithm (i.e. the FNFM algorithm) was contributed that reduced the computational budget 46.28% further while maintaining the video quality. In the architecture level, we introduced a novel bit-serial architecture for our algorithm. Due to advantages of the bit-serial architecture, it has a low gate counts, high speed operation frequency, low density interconnection, and a reduced number of I/O pins. In addition, several optimization techniques including the SAD truncation, mode filtering, source sharing exploiting, and power saving techniques were applied to the proposed architecture that improved the performance of the proposed architecture. Our design can save between 57.71%-95.47% of the area cost when compared to the previous designs and can provide higher throughput rate at a lower frequency for the same specification. Implementation results show that our design can support real-time SME of HD1080 format with 20.3 K gates at the operation frequency of 88.3 MHz.

Chapter 6

Conclusion, Future Work and Directions

6.1 Conclusion

This thesis has presented efficient IME and SME designs for H.264/AVC, which are tailored for area-constrained applications. All of the proposed designs are based on bit-serial scheme due to its advantages such as small hardware cost, low pin count and so on. To cope with the huge computational complexity and the memory access requirement of IME and SME, different but appropriate design optimization techniques are used at the algorithm and the architecture levels.

Regarding the H.264 IME, two low cost bit-serial architectures have been proposed based on the full search algorithm owing to its regularity and coding performance. Our architectures use SAD and data reusing techniques to reduce their memory bandwidth in different ways. The first design has a 2-D structure featured with the broadcasting of the reference pixel data and the propagating of the partial sum and SAD results. Due to the broadcasting of the reference pixels, it requires a small memory bit width and therefore is suitable for I/O constrained applications. The second design uses a 2-D bit-serial adder tree architecture connected to a bit-serial reconfigurable reference buffer making it suitable for hardware parallelism. The proposed reference buffer enables reference pixel data reusing in vertical and horizontal directions leading to a signification reduction of memory bandwidth. In addition, we have proposed a power saving technique to reduce the power consumption of the proposed bit-serial reconfigurable

reference buffer. To improve the overall performance of both designs, we have presented several optimization techniques. By using the pixel truncation method and presenting the world length reduction technique, 68.75% of the power consumption and the required time for processing of each search point are saved, where the latency, silicon area, and memory bandwidth are significantly decreased as well. Besides, we have employed 1/2-subsampling technique that approximately saves 50% of the hardware cost of PE arrays and memory bandwidth. In addition, 4×4 mode reduction technique is used to reduce the hardware cost further. The proposed designs can support full search VBSME of 720×480 video with 30 frames per second, two reference frames, and [-16, 15] search range at a clock frequency of 414 MHz with less than 32 K gates.

As for the H.264 SME, first we have provided a through and in-depth survey on SME algorithms and architectures due to the lack of surveys for them in the literature. In addition, to solve the problems of the huge computational complexity and the memory access requirement of the H.264 SME, we have introduced a low complexity algorithm with quarter-pixel accuracy. The proposed algorithm uses a parabolic model to estimate SAD values for predicting optimal sub-pixel motion vector. According to our analysis, the proposed algorithm can save 94.35% of the computational complexity and 98.5% of the memory access requirement relative to the interpolate and search method in the reference software of H.264/AVC. In addition, the fast version of the proposed algorithm (i.e. the FNFM algorithm) has been presented that reduces the computational budget 46.28% further while maintaining the video quality. For the hardware architecture, the first bit-serial architecture of SME has been efficiently implemented, which greatly benefits from its attractive bit-serial architecture, pipelining, source sharing, and power saving techniques. Furthermore, due to the use of SAD truncation and mode filtering techniques, not only the latency and the gate count of the proposed architecture are reduced but also its throughput is increased. Compared

with the prior designs, the proposed architecture can save between 57.71%-95.47% of the area cost and can achieve higher throughput rate at lower frequency for the same specification. The implementation results show that our design can support real-time SME of HD1080 format with 20.3 K gates at the operation frequency of 88.3 MHz.

6.2 Future Work and Directions

In future, we will try to design a complete H.264/AVC encoder where we will integrate our IME and SME designs into it. For this aim, we need to design other building blocks of the H.264 encoder such as transform, inverse transform, quantization, inverse quantization, motion compensation, and entropy coding units.

Although our architectures achieve very satisfactory results compared with the prior designs, they may not meet the required performance of the new multimedia applications and emerging video coding standards, which will unsurprisingly pose new design challenges. As a result, there may be some areas in the proposed designs that may need further research to improve their performances for real-time applications. In the following paragraphs, motion estimation (i.e. IME and SME) design challenges in some of the emerging multimedia applications and the next advanced video coding standard, H.265, are reviewed.

Digital video systems will continue to evolve toward higher resolutions to provide better realistic images on displays. Ultra-high definition (UD) TV (Nakasu et al., 2007; Sugawara et al., 2003) is an emerging visual media, which undoubtedly requires a much higher computation and memory bandwidth than current applications. When video resolution increases, the number of MBs increases and the search range (i.e. is proportional to the frame size) should be increased too. As a result, the memory bandwidth and the computational complexity of IME and SME are increased which

make their design more difficult for real-time scenarios, especially for resource-constrained applications.

Another interesting emerging application is 3-D TV that provides a 3-D impression of 3-D natural scenes by using multiview video sequences. Such sequences are produced by capturing a 3-D scene using multiple cameras simultaneously. Motion estimation for multiview sequences suffers from two major problems. Firstly, it demands a lot of computation and memory bandwidth that are directly proportional to the number of capturing cameras. Secondly, due to the redundancy among closer cameras, compressing multiview sequences independently with H.264 is not efficient (Bilen, Aksay, & Akar, 2003). As a result, there may need new IME and SME algorithms to consider the redundancy problem that causes an added complexity in the algorithm flow.

Although motion estimation in H.264 significantly improves the video quality, demands for higher coding performance are relentless. Consequently, development of new IME and SME algorithms will ceaselessly continue. Furthermore, IME and SME with enriched and improved features in future video coding standards will keep arriving, which will unsurprisingly increase the computation and the memory bandwidth. For instance, in H.265, the number of block modes as well as their sizes are increased, which result in more computation and memory bandwidth. In addition, the H.265 SME will use the adaptive interpolation filter instead of the H.264 interpolation filter with fixed weights to improve motion compensation prediction. The reason is that the time variation of image signal is not taken into account in the H.264 interpolation filters. Therefore, the prediction efficiency of motion compensation is limited. In the adaptive interpolation, based on the characteristics and statistics of each image, a set of coefficients for the interpolation filter is calculated which improves the coding

performance at the price of an added computation and irregularity (Vatis & Ostermann, 2009; Wittmann & Wedi, 2008).

Derived from the above discussion, motion estimation in the emerging and future multimedia applications will be characterized by massive computation, memory bandwidth, which will bring new challenges for designing of IME and SME hardware architectures. On the other hand, the continuous advances of microelectronic technology will provide more processing capability, which in turn will facilitate the hardware implementation of IME and SME architectures. However, conventional architectural approaches may not be able to fulfill the hard real time constrains of the emerging and future applications. As a result, besides of the conventional and current advanced algorithmic and architectural techniques, the video coding engineers may need to develop new and innovative methods for low cost, low power and efficient IME and SME hardware realizations. Furthermore, IME and SME hardware designers should consider some new challenges that will be brought by advanced VLSI technologies such as high static (leakage) power consumption and interconnection problems explained in Chapter 4. In the current and future VLSI designs, as process technology is steadily scaling down to deep submicron area; that is 90 nm and beyond, the static power that is proportional to the hardware cost, will grow much faster than dynamic power and it will be the main contributor to total power consumption. In addition, in deep submicron area, the interconnection parameter will be a very important design parameter and it will determine cost, delay, power, reliability and turn-around time of the future LSI's rather than MOSFET's (Sakurai, 2000). On the other hand, as the future multimedia applications are data intensive with a high bandwidth memory, parallelism at hardware level will be a must for meeting the real-time constrains and a lot storage space will be required for data saving. Consequently, the hardware cost and the number of wide length interconnections will be increased, which will lead to added static power and

interconnection density. Therefore, the use of static power reduction techniques together with reducing of area cost and memory bandwidth, which result in (static) power and interconnection reduction, are vitally important for the future advanced IME and SME architectures, especially for portable multimedia devices with limited resources.