Journal of VLSI Signal Processing 2007
© 2007 Springer Science + Business Media, LLC. Manufactured in The United States
DOI: 10.1007/s11265-007-0112-3

Data Reuse Exploration for Low Power Motion Estimation Architecture Design in H.264 Encoder

YU-HAN CHEN, TUNG-CHIEN CHEN, CHUAN-YUNG TSAI, SUNG-FANG TSAI AND LIANG-GEE CHEN
DSP/IC Design Lab., Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Received: 27 March 2007; Revised: 11 May 2007; Accepted: 13 June 2007

Abstract. Data access usually leads to more than 50% of the power cost in a modern signal processing system. To realize a low-power design, how to reduce the memory access power is a critical issue. Data reuse (DR) is a technique that recycles the data read from memory and can be used to reduce memory access power. In this paper, a systematic method of DR exploration for low-power architecture design is presented. For a start, the signal processing algorithms should be formulated as the nested loops structures, and data locality is explored by use of loop analysis. Then, corresponding DR techniques are applied to reduce memory access power. The proposed design methodology is applied to the motion estimation (ME) algorithms of H.264 video coding standard. After analyzing the ME algorithms, suitable parallel architectures and processing flows of the integer ME (IME) and fractional ME (FME) are proposed to achieve efficient DR. The amount of memory access is respectively reduced to 0.91 and 4.37% in the proposed IME and FME designs, and thus lots of memory access power is saved. Finally, the design methodology is also beneficial for other signal processing systems with a low-power consideration.

Keywords: parallel processing, data reuse, motion estimation, H.264

1. Introduction

Power becomes the first-class design issue nowadays [1]. One of the reasons is the emerging mobile applications. Because the capacity of battery in portable devices is limited, power should be used economically to provide longer service time. In addition, it is also important to constrain power consumption in high performance computing systems. Too much power consumption within a small silicon area leads to severe thermal problems. It increases the cost of heat disposal and reduces reliability of the system.

For multimedia applications, data access is usually a significant part of power consumption—50 to 80% in the image and video processing systems [2]. For this reason, reduction of memory access power becomes important in modern designs. In a data-driven signal processing algorithm, many data are requested from memory several times. Data read from memory for one process may also be required for other processes in the future. If the temporal data locality can be exploited, power consumption can be saved. Hence, data reuse (DR), recycling previously accessed data, is an important technique to reduce memory access power for low-power systems.

At the architecture-level, DR can be achieved with two techniques—memory hierarchy [3, 4] and parallel processing. For a memory hierarchy design, the frequently accessed data are pre-stored in the on-chip SRAMs and off-line reused for the future access. For a parallel processing design, the data read from memory are immediately shared between several processing engines (PE) and on-line reused. Therefore, memory access power of a signal processing system can be saved with customized memory hierarchy and parallel architecture. This is the reason why ASIC designs tend to be more power-efficient than processor designs. For example, an ASIC design [5] and a processor design [6] respectively consume 18 mW and 1.76 W for VGA (640×480) MPEG-4 [7] Simple Profile video encoding at 30 fps with the 0.18μm process. The ASIC design can achieve about one hundred times power efficiency.

A systematic method of DR exploration for low-power architecture design will be presented in this paper. A signal processing algorithm should be formulated as a nested loops structure. The nested loops are first decomposed, and then the data access pattern in each loop is analyzed to explore DR. Finally, memory hierarchy and parallel processing are applied to the suitable loops according to the characteristics of the algorithms, and the corresponding processing flows and memory data arrangement are also developed to realize a low-power design.

H.264 [8] is an emerging video coding standard with high coding gain and good visual quality. Multiple reference frame (MRF), variable-block size (VBS), and quarter-pel resolution ME are three useful tools to improve coding performance in H.264. However, they also greatly increase the memory access requirement, and thus corresponding DR techniques are needed for a low-power consideration. In this paper, the proposed systematic design method is applied to the integer ME (IME) and fractional ME (FME) algorithms in the H.264 encoder. With analyzing the ME algorithms, suitable hardware architectures are presented to achieve efficient DR. The amount of memory access of the IME and FME designs can be greatly reduced to 0.91 and 4.37%, respectively. Therefore, it is proven that the proposed design methodology is useful to reduce memory access power and beneficial for a low-power design.

The rest of the paper is organized as follows. The concepts of DR are first described in Section 2. Then, a systematic method of DR exploration for low-power architecture design is introduced in Section 3. In Section 4, the proposed design methodology is applied to IME and FME algorithms of H.264 to realize low-power ME engines with efficient DR. Finally, we will draw a conclusion in Section 5.

2. Data Reuse

For data dominated multimedia applications, a large number of data access are required. Power consumption of data access is usually a significant part in a signal processing system. However, in a signal processing algorithm, data accessed from memory for one process may also be required for some processes in the future. If the temporal data locality is exploited, the amount of memory access can be reduced and power consumption can be saved. Data reuse (DR), reuse of previously accessed data, is an effective technique to reduce memory access power. In this paper, DR is categorized as off-line DR and on-line DR. We will introduce the two kinds of DR schemes in the following.

2.1. Off-Line Data Reuse

Memory access power is dependent on the size and the technology of the memory. A larger memory usually consumes more power than a smaller one. In addition, off-chip DRAMs usually consume more power than on-chip SRAMs. For this reason, memory hierarchy [3, 4] is common for low-power architecture design. On-chip SRAMs are added between the off-chip DRAMs and registers in the data path. The frequently accessed data are per-stored in the on-chip SRAMs and can be off-line reused for the future access. In this way, most of data access is changed from off-chip memory access to on-chip memory access, and thus memory access power will be greatly reduced. In addition, the external memory bandwidth (BW) is saved, and it is beneficial for system performance. The technique that data are pre-stored in the small memories and reused in the future is defined as off-line DR.

2.2. On-Line Data Reuse

Traditionally, parallel processing is used to reduce power with the cooperation of voltage scaling [9]. However, it can also be used to reduce memory access. Here, we assume there is a datum required for two processes. If the two processes are in sequential, two times of data access is needed.

However, if the two processes are in parallel, the datum can be only accessed once and then shared between the processing engines (PE). Hence, the amount of data access is reduced. The technique that the accessed data are immediately reused by other PEs is defined as on-line DR.

Here, bilinear interpolation is taken as an example to describe how on-line DR can be achieved with parallel processing. As shown in Fig. 1a, four reference pixels (A, B, C, and D) are required to generate an interpolated pixel (M). That is to say, four pixels of memory access are required to interpolate one pixel. If there are two interpolated pixels generated in sequential, eight pixels of memory access are required. However, if the two neighboring interpolated pixels are generated in parallel, only six pixels of memory access are sufficient as shown in Fig. 1b. It is because the middle two reference pixels can be on-line reused. Therefore, memory access is reduced by use of parallel processing.

Memory hierarchy and parallel processing both can reduce memory access power with the penalty of increase of hardware resources. As a result, leakage power [10] will be increased. In this paper, we focus on reduction of switching power and ignore the impact of leakage power. If leakage power is taken into consideration, co-optimization is required for hardware resources and DR.

3. Systematic Method of Low-Power Architecture Design

Off-line DR with memory hierarchy and on-line DR with parallel processing should both be optimized to minimize memory access power. In this section, a systematic method of DR exploration for low-power architecture design will be presented. There are two assumptions here. First, temporal data locality should exist in the algorithm. It is because DR cannot be achieved if the processes are independent to each other. Second, we assume that the algorithm can be realized as a form with nested loops. Therefore, systematic loop analysis can be applied for DR exploration. The systematic flow of low-power architecture design is shown in Fig. 2. Two steps, loop analysis and architecture mapping, are identified and will be introduced in the following.

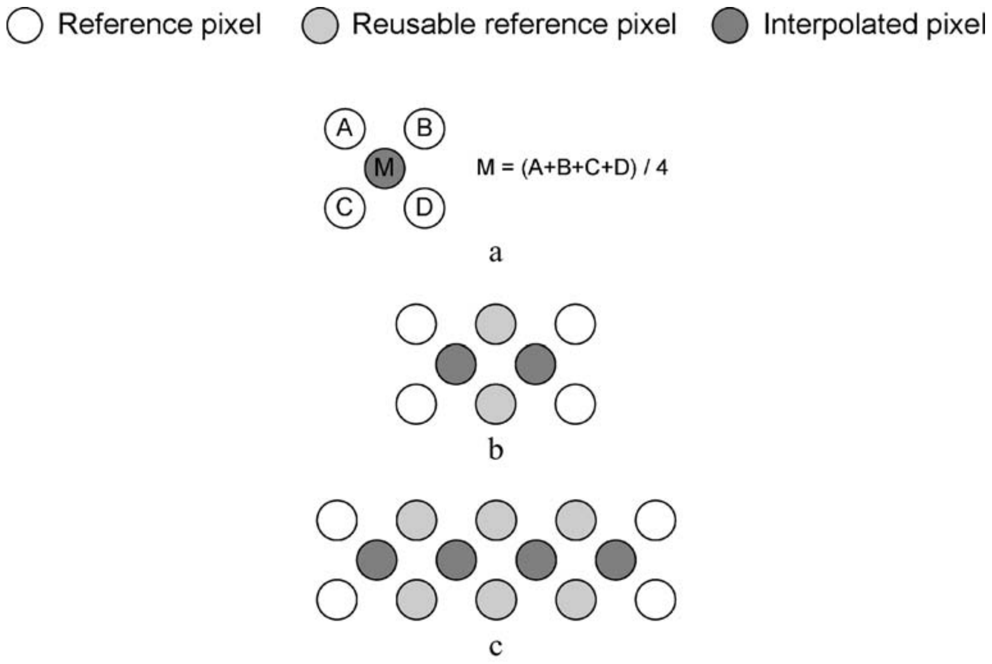


Figure 1. Data reuse from parallel processing of bilinear interpolation. **a** Illustration of bilinear interpolation; **b** two-parallel processing with two reusable reference pixels; **c** four-parallel processing with six reusable reference pixels.

3.1. Loop Analysis

For a start, we need to transform the algorithm into a structure with nested loops. Then, the nested loops are decomposed in order to apply loop analysis to each loop.

1. Data reuse exploration: Temporal data locality should be explored in order to achieve DR. At first, we have to find the data accessed pattern in each loop. Then, the overlapped accessed data in each loop can be found and are potential for DR. There is a simplified example shown in Fig. 3a. There are two loops in the algorithm. Inside the *Loop j*, four consecutive data are read. Among them, two data are overlapped in the two consecutive processes of *Loop j*. Therefore, *Loop j* data locality is found, and it is a candidates for DR. In addition, there are also lots of overlapped data in two consecutive processes of *Loop i*, and thus *Loop i* data locality is another candidate for DR.
2. Re-scheduling: In some algorithms, the original processing flow cannot achieve efficient DR. In this case, loop interchange may be an effective way to improve DR efficiency. Here comes an example. With the original schedule in Fig. 3a, two overlapped data in the consecutive processes of *Loop j* can be on-line reused if parallel processing is applied. However, if *Loop i* and *Loop j* are interchanged. Three overlapped data in the consecutive processes of *Loop i* can be reused as shown in Fig. 3b. As a result, the on-line DR efficiency in the lower loop becomes better. In addition, the memory size can also be reduced with re-scheduling when we want to achieve off-line DR. For the original schedule in Fig. 3a, eight data need to be buffered for off-line DR of all the required data in the lower loop. After rescheduling, only six pixels need to be buffered as shown in Fig. 3b.
3. Algorithm Modification: In some algorithms, there are data dependencies inside a loop. The tasks must be processed sequentially, and parallel processing is not applicable. In addition, there are data dependencies between two loops in some algorithms. The processing order is fixed and cannot be re-scheduled. In those cases, algorithm modification is required for improvement of DR efficiency, and it may lead to some penalties. In a video coding system, the penalty may be the coding performance degradation. Therefore, there will be a trade-off between coding performance and power in this case.

3.2. Architecture Mapping

After loop analysis, we should first decide what kinds of DR techniques should be applied to the loops with temporal data locality. There are some circumstances that parallel processing cannot be applied. The memory addresses in some algorithms are dependent on a variable which will be changed in some loops. In that case, the amount of overlapped data will be variable. The total numbers of required data accessed in parallel are not fixed, and it leads to a problem of parallel memory access. Therefore, parallel processing is not suitable for those loops with variable addresses. On the other hand, it is more

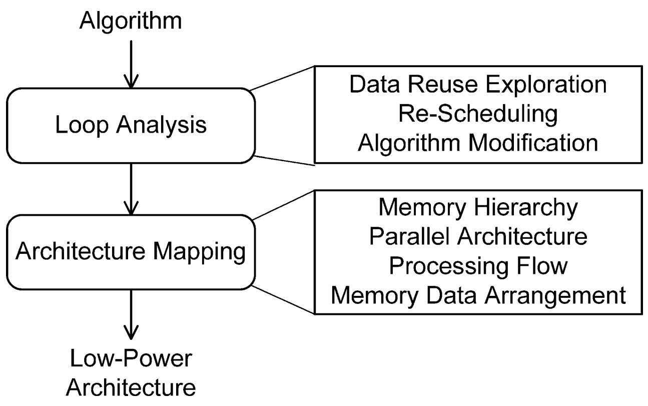


Figure 2. Systematic flow of low-power architecture design.

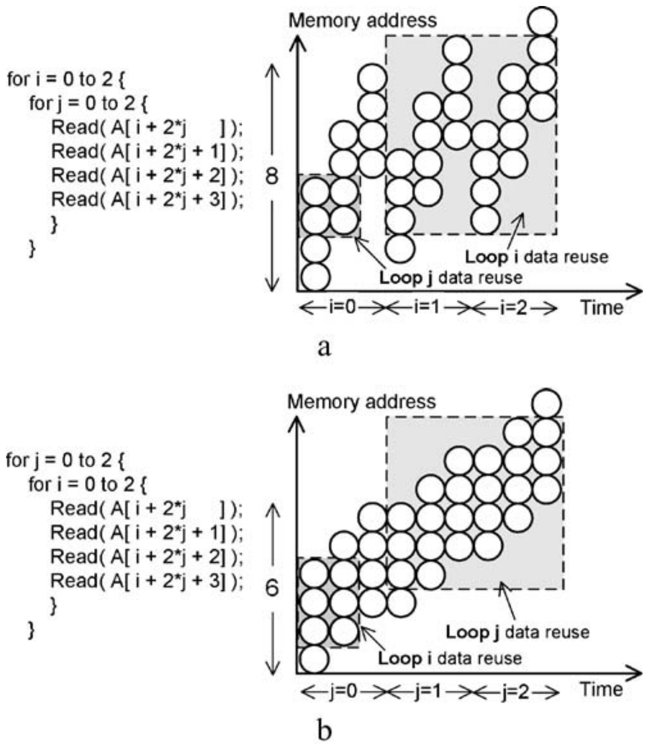


Figure 3. Data reuse exploration with loop analysis. **a** Original nested loops; **b** *Loop i* and *Loop j* are interchanged.

suitable to apply parallel processing in the lower loops. It is because fewer data should be accessed in parallel, and fewer PEs are required. As shown in Fig. 3a, six pixels are required to be accessed in parallel for two parallel processes in *Loop j* , but nine pixels are required for two parallel processes in *Loop i* . In conclusion, the overlapped data in the lower loops are more suitable to be on-line reused with parallel architecture design. On the contrary, the overlapped data in the higher loops are more suitable to be off-line reused by means of memory hierarchy design.

Secondly, we need to decide how many degrees of parallelism are required. Bilinear interpolation in Fig. 1 is taken as an example. Here, we define a factor AAP_{BI} , which stands for average accessed pixels for one bilinear-interpolated pixel. AAP_{BI} can be calculated by

$$AAP_{BI} = \frac{Total\ numbers\ of\ accessed\ pixels}{Total\ numbers\ of\ interpolated\ pixels} \quad (1)$$

AAP_{BI} is 4 for sequentially interpolated pixels. If two pixels are interpolated in parallel as shown in Fig. 1b, AAP_{BI} is decreased to 3 (6/2). Furthermore, AAP_{BI} will be reduced to 2.5 (10/4) if four horizontally adjacent pixels are generated in parallel as shown in Fig. 1c. Thus it can be seen that more degrees of parallelism will result in more memory access reduction, but it also leads to more hardware resources. Finally, the degrees of parallelism will be dependent on both the area and power constrains.

Memory access requirement is different between distinct parallel architectures. As shown in Fig. 1, there are 2×2, 2×3, and 2×5 reference pixels needed to be accessed in parallel for 1-PE, 2-PE, and 4-PE architectures, respectively. More degrees of parallelism usually means more memory bitwidth requirement. Therefore, memory data should be arranged according to the parallel architecture to achieve the memory access requirement.

4. Design Examples

In this section, we will first introduce the motion estimation (ME) algorithm of H.264. Then, the systematic method of DR exploration is applied to the IME and FME algorithms in H.264. Finally, the corresponding hardware architectures are presented.

4.1. Motion Estimation Algorithm of H.264

ME is a powerful coding tool used to remove temporal redundancy in a video sequence. For each macroblock (MB) in the current frame, a best matching block insides the search window (SW) of the reference frame will be found as depicted in Fig. 4. Then, the motion vector (MV) and the residues, the differences between the current and the best matching MBs, are coded. In H.264, multiple reference frame (MRF), variable block-size (VBS), and quarter-pel resolution ME algorithms are adopted to further improve coding performance. In the following, we will introduce these three coding tools.

1. Multiple reference frame: In the previous video coding standard, only one reference frame is supported in the process of ME. However, MRF-ME in H.264 provides at most five reference frames, and it is useful to improve coding performance for uncovered backgrounds, repetitive motions, and highly textured areas [11]. Although MRF is good for coding performance, the amount of memory access is increased with the number of reference frames. Without effective DR techniques, memory access power will be greatly increased.
2. Variable block-size: In the conventional ME algorithm, only one block size, 16×16, is supported. In H.264, there are seven kinds of block-sizes, comprising 4×4, 4×8, 8×4, 8×8, 8×16, 16×8, and 16×16. VBS-ME can obtain good compression efficiency in a region with complex motion [12]. In the reference software [13], ME of seven block-sizes are performed sequentially. For IME, all the sum of absolute difference (SAD) costs of the 4×4 block-size are pre-calculated and saved in a buffer. Then, the results can be off-line reused for all other larger block-sizes. However, the buffer of the 4×4 SAD costs is too large to be implemented in a dedicated hardware. For a 32×32 search range, the buffer size is *192* k-bit ($32 \times 32 \times 16 \times 12$). Therefore, on-line SAD calculation for VBS IME should be adopted. In this manner, the SAD costs cannot be reused, and the amount of memory access becomes seven times as compared to the conventional single-block-sized algorithm. Memory access power is greatly increased and becomes a serious problem.
3. Quarter-pel resolution: Half and quarter-pel refinement is supported in the FME algorithm of H.264 and performed sequentially as illustrated in Fig. 5a. Sum of absolute transformed difference (SATD) is used as the matching cost of FME. Block matching of eight half-pel and one integer-pel candidates are conducted in the first pass of refinement. Then, the matching costs of eight quarter-pel candidates around the best candidate in the first pass are computed for quarter-pel refinement. Finally, the best matching block with quarter-pel resolution is found. A six-tap filter is applied to interpolate the pixels at the half-pel position as shown in Fig. 5b, and the quarter-pel data are generated with bilinear interpolation. (I, I), (I, H), (H, I), and (H, H) respectively stand for the search candidates in the horizontal-integer and

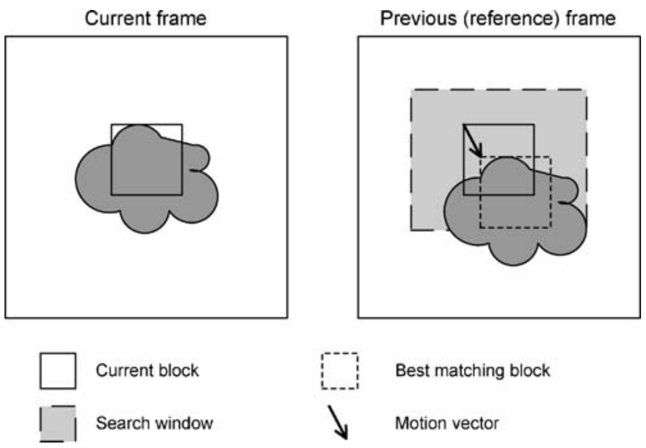


Figure 4. Illustration of ME algorithm. A best matching block will be found inside a search window of the reference frame.

vertical-integer, horizontal-integer and vertical-half, horizontal-half and vertical-integer, and horizontal-half and vertical-half positions. In H.264, six integer pixels are required to generate one (I, H) or one (H, I) interpolated pixel, and 6×6 integer reference pixels are needed for an (H, H) one. As a result, the amount of memory access is greatly increased. In the reference software [13], all the half-pel and quarter-pel interpolated pixels in a frame are pre-computed, saved in a buffer, and off-line reused. But for dedicated hardware realization, the buffer cost (16× frame size) is too high to be implemented, and on-line fractional-pel interpolation is inevitable. Without efficient DR, lots of memory access will be required.

4.2. *Integer Motion Estimation*

1. Loop analysis: The nested loops of the IME algorithm in the reference software [13] is shown in Fig. 6. AAP_{IME}, average accessed pixels for one IME search candidate, is defined to indicate DR efficiency of memory access for IME and can be calculated by

$$AAP_{IME} = \frac{Total\ numbers\ of\ accessed\ pixels}{Total\ numbers\ of\ search\ candidates} \qquad (2)$$

Loop 1 : the MB index in a current frame. For each MB, block matching is applied inside a SW. However, the SWs of the neighboring current MBs are overlapped, and thus the overlapped reference data can be reused. Because the amount of data in the SW is large, memory hierarchy design is adopted. SW SRAMs are commonly used in most of the ME designs for off-line DR. Level-C DR is a technique that can achieve horizontally inter-MB DR, and Level-D DR can achieve both horizontally and vertically inter-MB DR. They are both frequently used to reduce external memory BW. Please reference [14] for the details.

Loop 2 : the block index for VBS-ME. In H.264, there are *41* blocks with seven kinds of block-sizes in a MB. As mentioned in Sec. 4.1.2, a sequential flow is adopted for VBS IME in the reference software [13], and it cannot achieve efficient DR for hardware implementation. To solve this problem, a modified parallel VBS IME algorithm has been adopted in many designs [15–18]. Parallel VBS IME algorithm computes all matching costs of different block-sizes in parallel. For a search candidate, the costs of 4×4 is

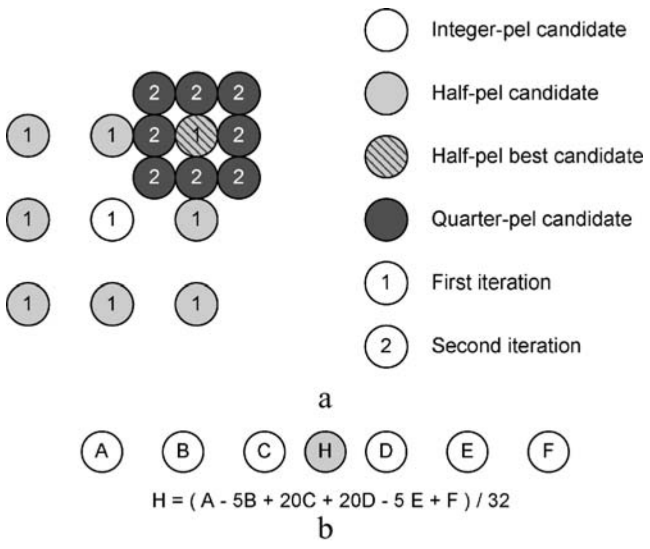


Figure 5. Illustration of quarter-pel resolution ME. **a** Flow of fractional motion estimation with two-pass iterations; **b** six-tap interpolation filter for the half pixels.

```
Loop 0 (current frame index)
  Loop 1 (MB index in a current frame)
    Loop 2 (41 block index for VBS ME)
      Loop 3 (reference frame index)
        Loop 4 (candidate index in the search range)
          Loop 5 (pixel index in a candidate)
          {
            SAD operations;
          }
```

Figure 6. Nested loops of integer motion estimation.

computed first, and all other cost of larger block-sizes are on-line calculated by summing up the corresponding 4×4 costs. As a result, computation and memory access is saved. AAP_{IME} is reduced from 1792 (256×7) to 256, and 85.71% memory access is reduced. The technique that the matching costs of the smaller block-sizes are reused by larger block-sizes is called inter-VBS DR of IME.

Loop 3 : the reference frame index. In H.264, MRF-ME is supported. For a current MB, ME is applied to several SWs of the reference frames. With the conventional multiple reference frames single current MB schedule, the required on-chip SW memory size and the amount of off-chip memory access is proportional to the number of reference frames. In order to reduce memory power, DR is required for the MRF-ME algorithm. Inter-frame DR can be achieved by use of frame-level rescheduling–interchange of *Loop 0* and *Loop 3*. The new schedule is called single reference frame multiple current MBs. Multiple current MBs of different current frames can reuse the data inside the SW of a reference frame. In this way, only one SW buffer is required, and the external memory BW is greatly reduced. Please reference [19] for the details.

Loop 4 : the candidate index in the search range. For a search candidate, 16×16 reference pixels are needed to compute the matching cost. However, a large portion of reference pixels are overlapped for the neighboring candidates. Here comes the examples. For two vertically adjacent candidates as shown in Fig. 7(a), 15×16 reference pixels are overlapped. If those overlapped data can be reused, a large amount of memory access can be saved. In this case, AAP_{IME} is reduced from 256 to 136 $(17 \times 16 \div 2)$. DR can also be achieved for the horizontally adjacent candidates in the same way as shown in Fig. 7b. The technique that the overlapped reference pixels for the neighboring search candidates are reused is called inter-candidate DR of IME.

2. Architecture mapping: Parallel 2-D adder tree architecture [18] is adopted as the basic architecture in our design and shown in Fig. 8. The current MB is stored in "16× 16 current pel buffer." The reference data in a SW are stored in the SW SRAMs for inter-MB DR. Reference pixels are read row-by-row from SW SRAMs and input to "16× 16 ref-pel systolic array." The original reference pixels in "16× 16 Ref-Pel Systolic Array" are shifted forward while a new row of data are input. To compute the SAD cost of the first candidate, 16 rows of reference pixels are input in 16 cycles. Then, only a new row of reference pixels are needed to compute a new search candidate below the first one, and 15×16 reference pixels are reused. In this way, inter-candidate DR can be achieved between the vertically adjacent candidates. Residues are generated in "256 processing unit array," and summed up by "2-D SAD tree." Two hundred fifty-six degrees of parallelism is provided here to generate 16×16 absolute difference values simultaneously for SAD summation. Sixteen SAD costs of 4×4 blocks are on-line computed by "16 2-D adder tree for 4×4-blocks." Then, all the SAD costs of larger block-sizes are calculated by "one VBS tree for larger blocks" with the on-line generated 4×4 costs. Therefore, inter-VBS DR can be achieved in "2-D adder tree."

For the full search algorithm, after the latency of 15 cycles, this architecture can process one candidate per cycle until the end of a column. However, if we want to compute the SAD costs of the search candidates in the next column, another 15-cycle latency is required to fill the reference data in "16×16 ref-pel systolic array." The scan order is shown in Fig. 9a. For a SW with a 32×32 search range, 47×16 reference pixels

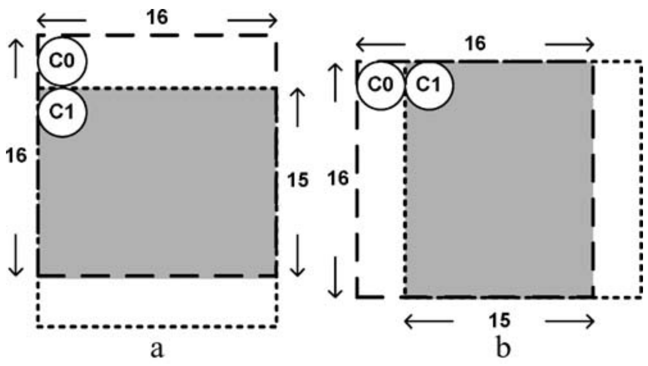


Figure 7. Inter-candidate data reuse for integer motion estimation. The overlapped (*grey*) region of reference pixels of search candidate C_0 and C_1 can be reused to reduce memory access. **a** Vertical data reuse; **b** horizontal data reuse.

are read from memory for a column of thirty-two search candidates, and thus AAP_{IME} is *23.5*.

The previous architecture can only reuse the reference data in the vertical direction (as shown in Fig. 7a). Further memory access reduction can be achieved if 2-D inter-candidate DR is supported. For this reason, a snake-scan search flow is adopted and illustrated in Fig. 9b. At the end of each column, a column of 16×1 reference pixels are read from memory and horizontally inter-candidate DR is achieved (as shown in Fig. 7b). Although the concept is simple, it is not straightforward for hardware implementation. It is because 2-D random access is required for memory access. To solve this problem, a technique called ladder-shaped SW data arrangement is proposed and will be introduced latter in Section 4.4. Here, we assume the SW SRAMs can provide 16×1 or 1×16 reference pixels random access. Besides, to support snake-scan search flow, the "16×16 ref-pel systolic array" in Fig. 8 is designed with three configurations—up-shift, down-shift, and right-shift by on pixels. Finally, with 2-D inter-candidate DR, AAP_{IME} becomes 16.23 for a 32×32 search range.

3. Results: Memory access requirement of different DR techniques is listed in Table 1. With the proposed architecture, inter-VBS and 2-D inter-candidate DR can be achieved. Memory access is reduced to 0.91% as compared to the designs without DR. By the way, the minimum memory access occurs if all the reference data in the SW SRAMs are accessed once. The value of minimal AAP_{IME} is 2.16 $(47^2/16^2)$, which is about 13.31% of the proposed design. However, much more degrees of parallelism are required to achieve the minimal memory access of IME. In that case, the area cost will be extremely high.

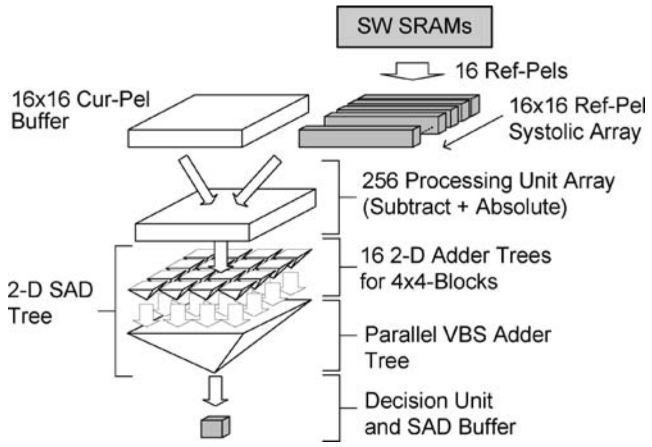

SW SRAMs
16 Ref-Pels
16x16 Cur-Pel Buffer
16x16 Ref-Pel Systolic Array
256 Processing Unit Array (Subtract + Absolute)
2-D SAD Tree
16 2-D Adder Trees for 4x4-Blocks
Parallel VBS Adder Tree
Decision Unit and SAD Buffer

Figure 8. 2-D adder tree architecture for the integer motion estimation engine.

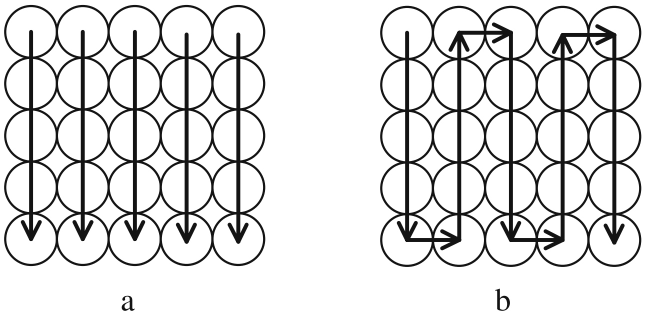


Figure 9. Scan flow of 5×5 search candidates for integer motion estimation. **a** Basic scan flow; **b** advanced snake scan flow.

The presented IME architecture is implemented with 0.18μm technology. The logic gate count is 65k. Full search algorithm is supported in a search range of H[−16, 15] and V[−16, 15] with one reference frame, and $1/2$ sub-sampling and 3-bit pixel truncation are adopted for reduction of SAD computation. Power consumption measured from the chip is 6.42mW with 1.3V supply voltage and 13.5 MHz operating frequency for CIF *30* fps format videos. In [20], Lin et al. proposes a 16 1-D adder tree architecture (256-PE) for the full search algorithm, and only 1-D inter-candidate DR is supported. The reported power consumption for 0.25μm process and 2.5V supply voltage is 46.52mW. Although the normalized power of this design for 0.18μm process and 1.3V supply voltage is similar to ours, the functionality of VBS-ME is not supported. In [21], a low-power ME engine based on 1-D adder tree architecture (16-PE) is proposed. In this design, a fast algorithm is adopted to reduce computation to $1/25$, and no DR can be achieved in this architecture. The power consumption is *6.56*mW for 0.18μ m process and 1.0 V supply voltage. Compared to the previous work, our design achieves the best DR efficiency and consumes the least power. It is shown that DR is critical for low-power design.

Table 1. Memory access requirement for integer motion estimation.

	(1)	(2)	(3)	(4)	Minimum
AAP_{IME}	1792	256	23.5	16.23	2.16
Ratio	100%	14.29%	1.31%	0.91%	0.12%

A 32×32 search range is assumed.
(1) No data reuse; *(2)* inter-VBS data reuse; *(3)* inter-VBS data reuse + 1-D inter-candidate data reuse; *(4)* Inter-VBS data reuse + 2-D inter-candidate data reuse

4.3. Fractional Motion Estimation

1. Loop analysis: The nested loop structure of FME algorithm in the reference software [13] is shown in Fig. 10. The DR techniques about the MB loop (*Loop 1*) and the reference frame loop (*Loop 3*) of FME are the same with those in IME and will be skipped here.

Loop 2	: the block index. For FME, different MVs should be refined for different blocks. There is no fixed inter-block data locality. Therefore, parallel processing cannot be applied here for DR.
Loop 5	: the pixel index in a candidate. Here, *Loop 5* is discussed before *Loop 4*. The 6-tap interpolation filter defined in H.264 increases the memory access requirement of FME engine. For example, a 6×6 window of reference data are required to generate an (H, H) interpolated pixel. DR can be explored from the neighboring interpolated pixels at the same search candidate. For two horizontally adjacent (H, H) interpolated pixels, 6×5 pixels in the overlapped region of the two interpolation windows could be reused as shown in Fig. 11a. For an (H, H) search candidate with the 4×4 block-size, a 9×9 window of reference pixels are enough to generate all the interpolated pixels as shown in Fig. 11b. In this case, the required memory access is reduced from

```
Loop 0 (current frame index)
    Loop 1 (MB index in current frame)
        Loop 2 (block index)
            Loop 3 (reference frame index)
                Loop 4 (candidate index)
                    Loop 5 (pixel index in a candidate)
                    {
                        Interpolation operations;
                        SATD operations;
                    }
```

Figure 10. Nested loops of fractional motion estimation.

576 ($4 \times 4 \times 6 \times 6$) to 81 ($9 \times 9$). This technique that the required reference data of neighboring interpolated pixels at the same search candidate are reused is called inter-pixel or intra-candidate DR of FME.

Loop 4 : the candidate index. DR can also be explored between the neighboring search candidates for FME. Take the half-pel refinement of a 4×4 block for example. There are nine candidates, comprising one (I, I), two (H, I), two (I, H), and four (H, H) candidates. With intra-candidate DR, the (I, I), (H, I), (I, H), and (H, H) candidates require 4×4, 4×9, 9×4, and 9×9 interpolation windows of reference data, respectively. However, many parts of those interpolation windows are overlapped as shown in Fig. 12. If the nine search candidate are processed in parallel, the reference data read from memory can be shared with each other. Finally, a 10×10 window of reference data are enough to compute the matching costs of all the nine candidates for half-pel refinement. In this case, the required memory access is reduced from 484 ($4 \times 4 + 2 \times 4 \times 9 + 2 \times 9 \times 4 + 4 \times 9 \times 9$) to 100 ($10 \times 10$). The technique that the reference data of neighboring search candidates are reused is called inter-candidate DR of FME.

2. Architecture mapping: Based on the analysis of [22], the basic hardware architecture of FME engine is shown in Fig. 13. The 4×4 block-size is the smallest element of VBS, and the SATD computation is also based on 4×4 blocks. In addition, all other larger block-sizes can be decomposed into several 4×4 -elements with the same MV. Therefore, a 4×4-processing unit (PU) is designed and reused for larger blocks by use of folding. For the low-power consideration, intra-candidate and inter-candidate DR are both applied in this architecture. For horizontal filtering, six reference pixels are required to interpolate one half pixel while nine pixels are needed for four adjacently half-interpolated pixels in a row of a search candidate. Furthermore, ten reference pixels are enough to generate a row of the required data for three horizontally adjacent search candidates as shown in Fig. 12. Therefore, a row of ten reference pixels are read from SW SRAMs and input to the row-parallel interpolation engine in a cycle. Then, a rows of interpolated pixels are generated at the same time. For intra-candidate DR, each 4×4-element PU is designed

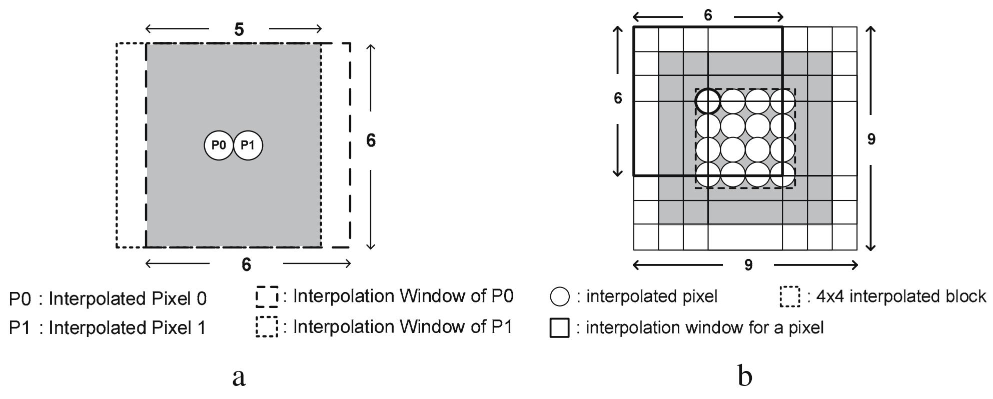
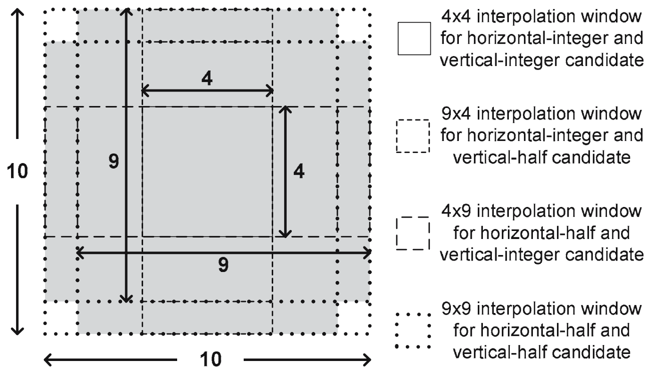

5
6
6
P0
P1
P0 : Interpolated Pixel 0
P1 : Interpolated Pixel 1
: Interpolation Window of P0
: Interpolation Window of P1
6
6
9
9
: interpolated pixel
: interpolation window for a pixel
: 4x4 interpolated block
a
b

Figure 11. Intra-candidate data reuse for fractional motion estimation. **a** Reference pixels in the overlapped (*grey*) interpolation windows for two horizontally adjacent interpolated pixels *P0* and *P1* can be reused; **b** overlapped (*grey*) interpolation windows data reuse for a 4×4 interpolated block. Totally, 9×9 reference pixels are enough with the technique of intra-candidate data reuse.



10
9
4
4
9
10
4x4 interpolation window for horizontal-integer and vertical-integer candidate
9x4 interpolation window for horizontal-integer and vertical-half candidate
4x9 interpolation window for horizontal-half and vertical-integer candidate
9x9 interpolation window for horizontal-half and vertical-half candidate

Figure 12. Inter-candidate data reuse for half-pel refinement of fractional motion estimation. The overlapped (*grey*) region of interpolation windows can be reused to reduce memory access.

with four degrees of parallelism to process four horizontally adjacent pixels of a candidate at the same time. For inter-candidate DR, nine 4×4-element PUs is arranged to process the nine search candidates in parallel. Totally, 36 degrees of parallelism are provided. In this way, the interpolated fractional pixels are reused, and the redundant computation of interpolation operation is saved. Besides, the on-chip memory BW of SW SRAMs for reading reference pixels is also be decreased. In summary, not only logic power but also memory access power is reduced.

Because the proposed architecture is based on a 4×4 PU, VBS should be decomposed into several 4×4 blocks which are processed sequentially. The basic processing flow in [22] is shown in Fig. 14a. Because only 1-D random access of SW SRAMs is supported in [22], the 4×4 blocks are strung up in the vertical direction. Therefore, vertically inter-block DR can be obtained as depicted in Fig. 15a. For the next column of 4×4 blocks, reference pixels should be reloaded, and no DR can be achieved horizontally. With the basic flow of FME, DR of 4×8 and 8×16

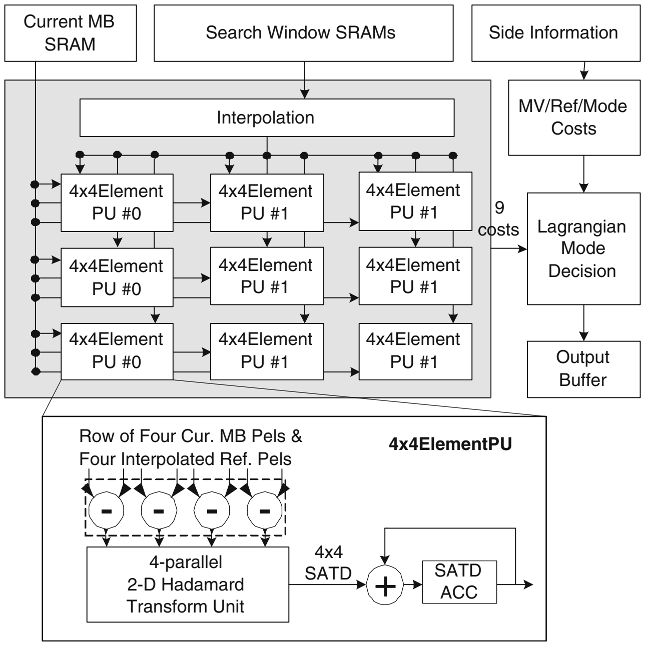


Figure 13. Hardware architecture for fractional motion estimation engine.

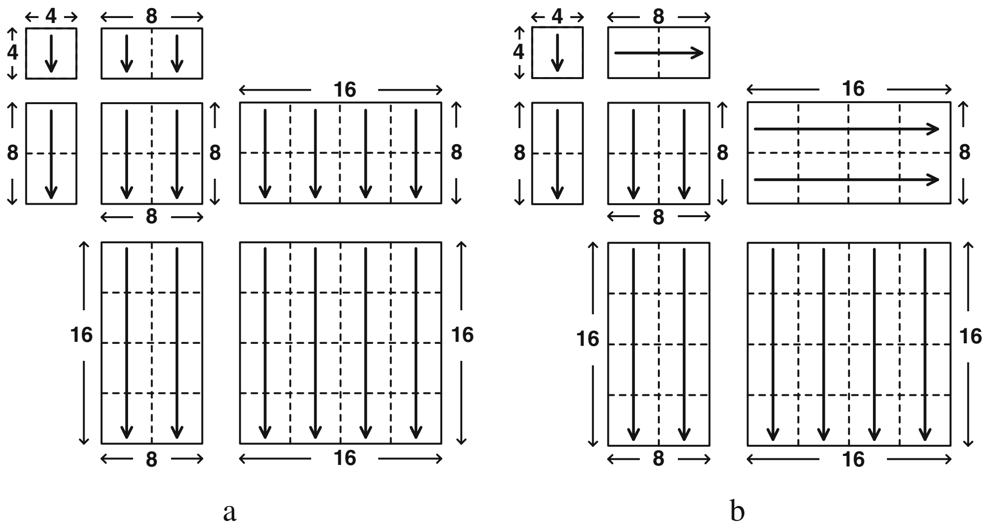


Figure 14. Hardware processing flow of variable-block size fractional motion estimation. **a** Basic flow; **b** advanced flow.

block-sizes is not efficient. However, the proposed ladder-shaped SW SRAMs data arrangement in Section 4.4 can provide 2-D random access, and an advanced flow can be applied as shown in Fig. 14b. For 4 × 8 and 8 × 16 block-sizes FME, the 4 × 4 sub-blocks can be strung up in the horizontal direction. Reference pixels are read column-by-column from SW SRAMs and input to the interpolation engine. In this case, horizontally inter-block DR (as shown in Fig. 15b) can be achieved. Therefore, memory access can be further reduced.

3. Results: Here, we define a factor AAP_{FME}, which stands for Average Accessed Pixels for one FME half-pel search candidate, to indicate the DR efficiency of a FME engine. AAP_{FME} can be computed as

$$AAP_{FME} = \frac{Total\ numbers\ of\ accessed\ pixels}{9\ (\#.\ of\ candidate) \times 7\ (\#.\ of\ block\ size)} \tag{3}$$

The amount of memory access for different block-sizes and AAP_{FME} for different DR tech-

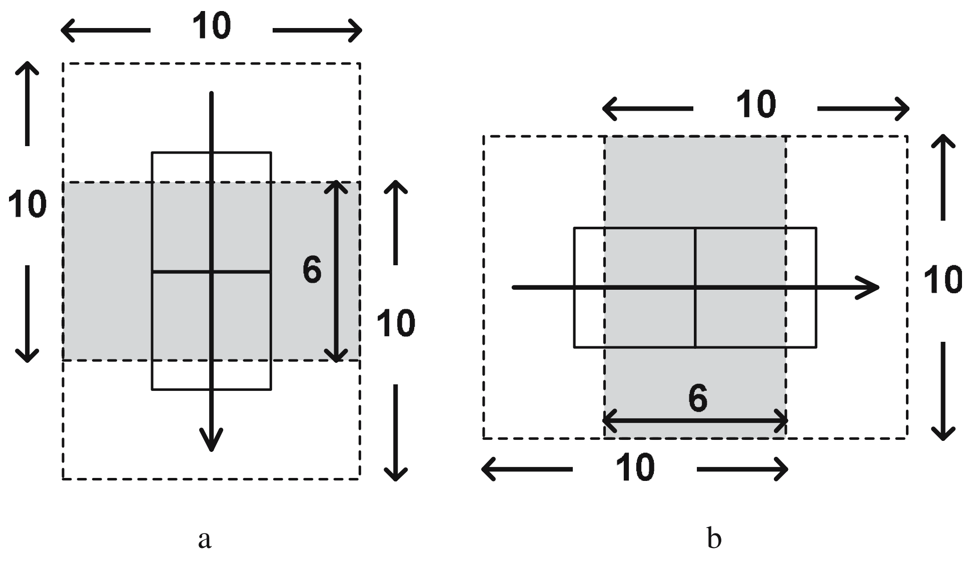


Figure 15. Inter-*4* × *4*-block interpolation window data reuse **a** Vertical data reuse; **b** horizontal data reuse.

Table 2. Memory access requirement for half-pel refinement of fractional motion estimation.

Block-size	(1)	(2)	(3)	(4)	(5)
16×16	24832	2482	484	880	880
16×8	24832	2852	616	1120	880
8×16	24832	2852	704	880	880
8×8	24832	3272	784	1120	1120
8×4	24832	4112	1120	1600	1120
4×8	24832	4112	1120	1120	1120
4×4	24832	5152	1600	1600	1600
Total	173824	24834	6428	8320	7600
AAP_{FME}	2759.11	394.19	102.03	132.06	120.63
Ratio	100%	14.29%	3.70%	4.79%	4.37%

(1) No data reuse; *(2)* only intra-candidate data reuse; *(3)* full intra-candidate and inter-candidate data reuse; *(4)* proposed architecture with the basic processing flow; *(5)* proposed architecture with the advanced processing flow

niques are listed in Table 2. The presented architecture with the advanced processing flow can achieve efficient DR and reduce the memory BW to 4.37% which is very close to the lower bound of 3.70% with fully inter- and intra-candidate DR.

The proposed FME design is also implemented with 0.18μm technology. The gate count is 127k. For real-time CIF 30fps video coding with the full search FME algorithm, power consumption reported from gate-level simulation is 22.58mW at 1.8V and 27 MHz. Because our design is based on the 4×4 block engine, full reuse of accessed reference pixels cannot be achieved. To further reduce memory access power, a FME engine with the 16×16 PU can be designed [23]. In this case, the area is increased to 189k gates [23], and about 15% memory access power can be further reduced.

4.4. Ladder-Shaped Search Window Data Arrangement

In order to support 2-D DR for both the presented IME and FME designs, memory access capability of consecutively horizontal or vertical 16 pixels is required. Here comes a simple example. Physical location of pixels in the SW is shown in Fig. 16a. The conventional data arrangement is shown in Fig. 16b. Horizontally adjacent pixels are arranged in different banks of SW SRAMs. The first column of reference pixels, $A1$–$A8$, are placed in the bank $M1$. The second column of pixels, $B1$–$B8$, are placed in the bank M2, and so on. If there are eight banks of SRAM, the ninth column of pixels are placed in the first bank $M1$. In this way, a row of reference pixels, like $A5$–$H5$, can be read in parallel. However, a column of reference pixels, like $C1$–$C8$, cannot be accessed

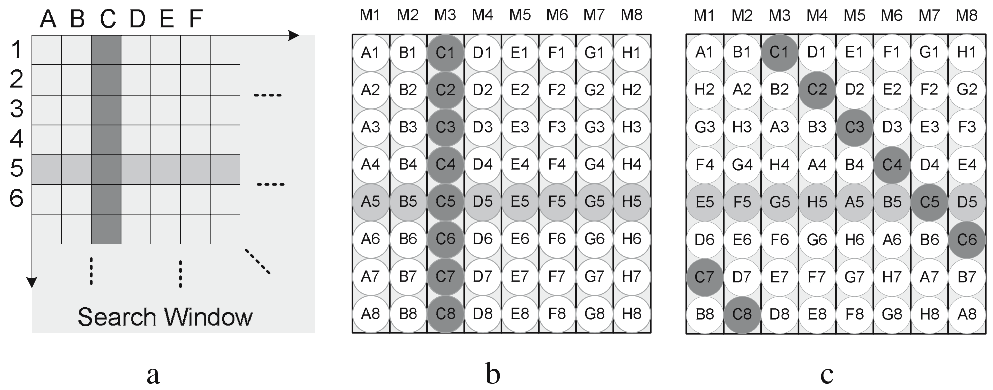


Figure 16. Search Window SRAMs data arrange. **a** Physical location of reference pixels in the search window; **b** traditional data arrangement with 1-D random access; **c** proposed ladder-shaped data arrangement with 2-D random access.

at the same time because they are located in the same bank *M*3. This is called 1-D random access.

The ladder-shaped SW data arrangement technique is depicted in Fig. 16c. The second and third rows are rotated rightward by one and two pixels. The other rows are also rotated in the same manner. In this way, the reference pixels of *A*5–*H*5 and *C*1–*C*8 are both put in different banks of SRAMs and can be accessed in one cycle. Therefore, a row or a column of pixels can both be accessed in parallel, and 2-D random access is achieved. As a result, the proposed data arrangement can enhance random access capability of SW SRAMs, and improve DR efficiency of the IME and FME designs.

5. Conclusion

A systematic method of DR exploration for low-power design is proposed and applied to the IME and FME algorithms of H.264 in this paper. With the loop analysis, data locality in the algorithms is first explored. A large number of required reference pixels are overlapped and can be reused. Therefore, the corresponding DR techniques for each loop of the ME algorithms are presented. With the DR techniques, not only memory access but also computation can be saved. Then, suitable parallel architectures and memory hierarchy design are mapped into the loops in the algorithm for DR. Finally, a low-power design with efficient DR is realized. The amount of memory access of IME and FME designs is reduced to 0.91 and 4.37%, respectively. The design method of DR exploration for low-power architecture design can also be extended to other signal processing systems with a low-power consideration.

Acknowledgements

This work was supported in part by the National Science Council, Taiwan, R.O.C. under the grant number NSC95-2752-E-002-008-PAE.

References

1. T. Mudge, "Power: A First-class Architectural Design Constraint," *IEEE Comput.*, vol. 34, no. 4, pp. 52–58, Apr. 2001.
2. K. Danckaert, K. Masselos, F. Catthoor, H. J. D. Man, and C. Goutis, "Strategy for Power-Efficient Design of Parallel Systems," *IEEE Trans. VLSI Syst.*, vol. 7, no. 2, 1999, pp. 258–265, June.
3. S. Wuytack, F. Catthoor, L. Nachtergaele, and H. D. Man, "Power Exploration for Data Dominated Video Applications," in *Proc. IEEE Int. Conf. on Low Power Electronics and Design (ISLPED)*, 1996.
4. S. Wuytack, J.-P. Diguet, F. V. M. Catthoor, and H. J. D. Man, "Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings," *IEEE Trans. VLSI Syst.*, vol. 6, no. 4, 1998, pp. 529–536, Dec.
5. C.-P. Lin, P.-C. Tseng, Y.-T. Chiu, S.-S. Lin, C.-C. Cheng, H.-C. Fang, W.-M. Chao, and L.-G. Chen, "A 5mW MPEG4 SP Encoder with 2D Bandwidth-sharing Motion Estimation for Mobile Applications," in *ISSCC Digest of Technical Papers*, 2006.
6. H.-J. Stolberg, S. Moch, L. Friebe, A. Dehnhardt, M. B. Kulaczewski, M. Berekovic, and P. Pirsch, "An SoC with Two Multimedia DSPs and a RISC Core for Video Compression Applications," in *ISSCC Digest of Technical Papers*, 2004.
7. *Information Technology—Coding of Audio-Visual Objects—Part 2: Visual*. ISO/IEC 14496-2, 1999.
8. Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification," Mar. 2003.
9. A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS Digital Design," *IEEE J. Solid State Circuits*, vol. 27, no. 4, pp. 473–483, Apr. 1992.
10. W. M. Elgharbawy and M. A. Bayoumi, "Leakage Sources and Possible Solutions in Nanometer CMOS Technologies," *IEEE Circuits and Syst. Mag.*, vol. 5, no. 4, 2005, pp. 6–17.
11. Y. Su and M.-T. Sun, "Fast Multiple Reference Frame Motion Estimation for H.264," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2004.
12. Y.-H. Chen, T.-C. Chen, and L.-G. Chen, "Hardware Oriented Content-adaptive Fast Algorithm for Variable Block-size Integer May 25, 2007 DRAFT Motion Estimation in H.264," in *Proc. IEEE Int. Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, 2005.
13. *Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, H.264/AVC Reference Software JM8.2*. http://bs.hhi.de/ suehring/tml/download/, May 2004.
14. J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, 2002, pp. 61–72, Jan.
15. Y.-W. Huang, T.-C. Wang, B.-Y. Hsieh, and L.-G. Chen, "Hardware Architecture Design for Variable Block Size Motion Estimation in MPEG-4 AVC/JVT/ITU-T H.264," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2003.
16. S. Y. Yap and J. V. McCanny, "A VLSI Architecture for Variable Block Size Video Motion Estimation," *IEEE Trans. Circuits Syst. II*, vol. 51, no. 7, pp. 384–389, July 2004.
17. H. F. Ates and Y. Altunbasak, "SAD Reuse in Hierarchical Motion Estimation for the H.264 Encoder," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2005.
18. C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and Architecture Design of Variable block size Motion Estimation for H.264/AVC," *IEEE Trans. Circuits Syst. 1, Fundam. Theory Appl.*, vol. 53, no. 3, 2006, pp. 578–593.
19. T.-C. Chen, Y.-W. Huang, C.-Y. Tsai, C.-T. Huang, and L.-G. Chen, "Single Reference Frame Multiple Current Macroblocks

Scheme for Multi-Frame Motion Estimation in H.264/AVC," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2005.
20. S.-S. Lin, P.-C. Tseng, and L.-G. Chen, "Low-power Parallel Tree Architecture for Full Search Block-matching Motion Estimation," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2004.
21. J. Miyakoshi, Y. Kuroda, M. Miyama, K. Imamura, H. Hashimoto, and M. Yoshimoto, "A Sub-mW MPEG-4 Motion Estimation Processor Core for Mobile Video Application," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2003.
22. T.-C. Chen, Y.-W. Huang, and L.-G. Chen, "Fully Utilized and Reusable Architecture for Fractional Motion Estimation of H.264/AVC," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2004.
23. C. Yang, S. Goto, and T. Ikenaga, "High Performance VLSI Architecture of Fractional Motion Estimation in H.264 for HDTV," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2006.

Tung-Chien Chen was born in Taipei, Taiwan, R.O.C. in 1979. He received the B.S. degree in Electrical Engineering and the M.S. degree in Electronic Engineering from National Taiwan University, Taipei, Taiwan, R.O.C. in 2002 and 2004, respectively, where he is working toward the Ph.D. degree in Electronics Engineering. His major research interests include motion estimation, algorithm and architecture design of MPEG-4 and H.264/AVC video coding, and low power video coding architectures.

Yu-Han Chen was born in Taipei, Taiwan, R.O.C. in 1981. He received the B.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. in 2003. He is currently pursuing the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University. His research interests include image/video signal processing, motion estimation, algorithm and architecture design of H.264 video coder, and low-power and power-aware video coding system.

Chuan-Yung Tsai was born in Kaohsiung, Taiwan in 1982. He received his B.S. degree from the Department of Electrical Engineering, National Taiwan University in 2004. He is currently pursuing his Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University. His research interests include algorithm and architecture design of H.264 video encoder/decoder, low-power video coding system, and intelligent architecture for video processing.

Sung-Fang Tsai was born in Hsinchu, Taiwan, R.O.C. in 1983. He received the B.S. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, R.O.C. in 2005. Now he is working toward the M.S. degree in the Graduate Institute of Electronics Engineering, National Taiwan University. His major research interests include motion estimation, algorithm and architecture design of H.264/AVC video coding standard.

Liang-Gee Chen received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan, R.O.C. in 1979, 1981, and 1986, respectively. In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. From 1993 to 1994, he was a Visiting Consultant in the DSP Research Department, AT&T Bell Labs, Murray Hill, NJ. In 1997, he was a Visiting Scholar of the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor in National Taiwan University.

Dr. Chen has served as an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON VLSI SYSTEMS, and IEEE TRANSACTIONS CIRCUITS AND SYSTEMS II since 1996, 1999, and 2000, respectively. He has been the Associate Editor of the Journal of Circuits, Systems, and Signal Processing since 1999, and a Guest Editor for the Journal of Video Signal Processing Systems. He is also the Associate Editor of the PROCEEDINGS OF THE IEEE. He was the General Chairman of the Seventh VLSI Design/CAD Symposium in 1995 and of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He is the Past-Chair of Taipei Chapter of IEEE Circuits and Systems (CAS) Society and is a member of the IEEE CAS Technical Committee of VLSI Systems and Applications, the Technical Committee of Visual Signal Processing and Communications, and the IEEE Signal Processing Technical Committee of Design and Implementation of SP Systems. He is the Chair-Elect of the IEEE CAS Technical Committee on Multimedia Systems and Applications. From 2001 to 2002, he served as a Distinguished Lecturer of the IEEE CAS Society. He received the Best Paper Award from the R.O.C. Computer Society in 1990 and 1994. Annually from 1991 to 1999, he received Long-Term (Acer) Paper Awards. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on circuits and systems in the VLSI design track. In 1993, he received the Annual Paper Award of the Chinese Engineer Society. In 1996 and 2000, he received the Outstanding Research Award from the National Science Council, and in 2000, the Dragon Excellence Award from Acer. He is a member of Phi Tau Phi.

His current research interests are DSP architecture design, video processor design, and video coding systems.