IEICE TRANS. ELECTRON., VOL.E91–C, NO.4 APRIL 2008

PAPER Special Section on Advanced Technologies in Digital LSIs and Memories

# Reconfigurable Variable Block Size Motion Estimation Architecture for Search Range Reduction Algorithm

**Yibo FAN**[†a)], *Nonmember*, **Takeshi IKENAGA**[†], *Member*, *and* **Satoshi GOTO**[†], *Fellow*

**SUMMARY** Variable Block Size Motion Estimation (VBSME) costs a lot of computation during video coding. Search range reduction algorithm is widely used to reduce computational cost of motion estimation. Current VBSME designs are not suitable for this algorithm. This paper proposes a reconfigurable design of VBSME which can be efficiently used with search range reduction algorithm. While using proposed design, $n \times m$ reference MBs form an MB array which can be processed in parallel. $n$ and $m$ can be configured according to the new search range shape calculated by algorithm. In this way, the parallelism of proposed design is very flexible and can be adapted to any search range shape. The hardware resource is also fully used while performing VBSME. There are two primary reconfigurable modules in this design: PEGA (PE Group Array) and SAD comparator. By using TSMC 0.18 $\mu$m standard cell library, the implementation results show that the hardware cost of design which uses 16 PEGs (PE Groups) is about 179 K Gates, the clock frequency is 167 MHz.

***key words:*** *reconfigurable, VBSME, search range reduction, H.264*

## 1. Introduction

H.264 is the latest international video coding standard. It is also known as MPEG-4 part 10, or AVC (for Advanced Video Coding). It is jointly written by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving P icture Experts Group (MPEG) [1]. H.264 contains a number of new features that make it to compress video much more effectively than previous standards and to provide more flexibility for application to a wide variety of network environments.

Variable Block Size Motion Estimation (VBSME) is a key feature in H.264, which enables very precise segmentation of moving regions [2], [4]. As illustrated in Fig. 1, there are 7 block sizes in H.264, including $16 \times 16$, $16 \times 8$, $8 \times 16$, $8 \times 8$, $8 \times 4$, $4 \times 8$, and $4 \times 4$. Hence, the coding performance is greatly improved [3], [4].

VBSME achieves higher compression ratio. However, the computational cost is very huge. The procedure of VBSME is:

Firstly, all of the blocks listed in Fig. 1 are processed by using sum of absolute differences (SAD) algorithm:

$$SAD(m,n) = \sum_{j=1}^{N} \sum_{i=1}^{M} |cur(i,j) - ref(i+m, j+n)|, \quad -P \le m,\ n < P \tag{1}$$

Manuscript received August 14, 2007.
Manuscript revised November 12, 2007.

[†]The authors are with IPS, Waseda University, Kitakyushu-shi, 808-0135 Japan.
a) E-mail: fanyibo@ruri.waseda.jp

DOI: 10.1093/ietele/e91–c.4.440

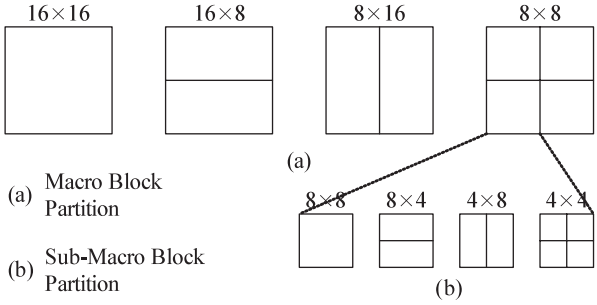



**Fig. 1** Variable block size in H.264.

where $cur(i, j)$ and $ref(i, j)$ are pixel value in current MB (Macro Block) and reference MB. P is search range, $(m, n)$ is one of search candidates in the search range. After processing all of the blocks, the SADs of 41 blocks in one MB are calculated. Secondly, after all of search candidates are processed, the search candidate which has the smallest SAD is selected as the motion vector of current MB, and the sub-blocks partition mode is selected as MB partition mode.

In order to accelerate the speed of VBSME, there are two approaches: 1) Reduce search candidates. 2) Parallelize computation of SAD algorithm. A lot of VBSME architectures have been proposed. However, almost all of these architectures focus on parallelize computation in full search range, and few of them consider about reduced search range.

In this paper, a reconfigurable VBSME architecture which is designed for search range reduction algorithm is proposed. The purpose of this design is to achieve low-power high-speed motion estimation through improving the hardware utilization and efficiency while using search range reduction algorithm. The procedure of this new VBSME engine is: Firstly, reducing original search range to a small range by using search range reduction algorithm. Secondly, configuring PEGA (PE Group Array) and SAD comparator based on the shape of new search range. Finally, performing VBSME by using new search range, configured PEGA and configured SAD comparator. Thus, there are two configurable modules in this architecture: 1) Reconfigurable PEGA. 2) Reconfigurable SAD comparator.

The rest of the paper is organized as follows: Search range reduction algorithm is introduced in Sect. 2. The existing Vertical-parallel and Horizontal-parallel VBSME architectures are introduced in Sect. 3. The proposed reconfigurable VBSME design is presented in Sect. 4. The experimental results and comparison are presented in Sect. 5.

Finally, conclusion is given in Sect. 6.

## 2. Search Range Reduction Algorithm

In order to reduce the computational cost of motion estimation, search range reduction algorithm is a popular way to reduce search candidates.

Figure 2 illustrates the search range reduction algorithm used with motion estimation in JM Software [7]. The detailed procedure is listed as follows:

*Step 1). Set MVP*

MVP means motion vector prediction. It is a motion vector between original position of current MB and the start search candidate in reference frame. The MVP can be calculated by the neighboring MBs which around current MB.

$$\text{MVP} = \text{median}(\text{MV\_A}, \text{MV\_B}, \text{MV\_C}) \quad (2)$$

For Hardware oriented motion estimation algorithm, all of blocks (from $16 \times 16$ to $4 \times 4$) share the same MVP.

*Step 2). Set Search Range*

The original search range is a square with fixed-size around the start search candidate. By using search range reduction algorithm, the new search range with rectangular shape can be calculated.

*Step 3). Perform VBSME within New Search Range*

After the start search candidate and new search range be decided, the VBSME can be performed by hardware processing elements (PEs).

There are a lot of search range reduction algorithms, software oriented algorithms update search range for every sub-blocks in MB. Hardware oriented algorithms only use one search range for all of sub-blocks. Most of the search range reduction algorithms are software oriented algorithm. For hardware oriented algorithm, two experimental algorithms are introduced as below:

***Algorithm:*** *Hardware oriented search range reduction algorithm.*

```
If (two of [MV_A, MV_B, MV_C] are unavailable)
    New_Search_Range = Original_Search_Range;
Else {
    mv_max= max(|mv_a|, |mv_b|, |mv_c|);
    mv_sum= |mv_a| + |mv_b| + |mv_c|;
    if (mv_sum == 0)
        small_ range = (Original_Search_Range + 4) ≫ 3;
    else if (mv_sum > 3)
        small_range = (Original_Search_Range + 2) ≫ 2;
    else
        small_range = (3*Original_Search_Range+8) ≫ 4;
    if (max(SAD_a, SAD_b, SAD_c) > Threshold)
        search_range1 = Original_Search_Range;
    else
        search_range1=min(Original_Search_Range,
            max(small_range, mv_max ≪ 1));
    search_range2=min(Original_Search_Range, mv_max);
}
```

***Algorithm 1****: New_Search_Range = search_range1*
***Algorithm 2****: New_Search_Range = search_range2*

There are two algorithms listed above. Algorithm 1 [6], [7] is original search range reduction algorithm in JM software. Algorithm 2 is a new algorithm proposed in this paper. It produces smaller search range than algorithm 1. All of the search ranges produced by these two algorithms are rectangular shape. More details about this algorithm can be found in Ref. [6], [7].

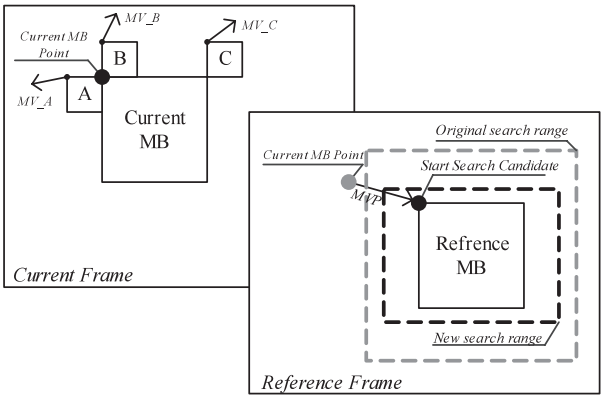



**Fig. 2** Search range reduction algorithm used with VBSME in H.264.

## 3. Parallel VBSME Architectures Overview

The up-to-date parallel VBSME architectures can be classified into two types: Horizontal-parallel architecture and Vertical-parallel architecture. The brief introduction of these two kinds of architectures is listed as follows.

### 3.1 Horizontal-Parallel Architecture

The horizontal-parallel architecture is shown in Fig. 3(a). It is proposed by Song and Liu in [8] and [9]. This architecture uses n PEGs (PE groups) in a horizontal line. Each PEG processes 1 reference MB. In this way, n reference MB in horizontal direction can be processed in parallel.

As shown in Fig. 3(a) and Fig. 3 d.2), *i*th row of current MB (Grey part in Fig. 3 d.1)) and *j*th row of reference MB (Grey part in Fig. 3 d.2)) are read from memory every clock cycle. One row of current MB memory includes 16 pixels. One row of reference frame memory contains $n + 15$ pixels, which includes n reference MB rows. For example, pixel [1:16] is a row of reference MB 1, pixel [2:17] is a row of MB 2. Each PEG loads one row of current MB and one row of reference MB. Totally, n PEGs process n reference MBs in parallel.

### 3.2 Vertical-Parallel Architecture

The vertical-parallel architecture is shown in Fig. 3(b). It is proposed by Chen in [10]. The PEGs are placed in a vertical line. The current MB data is propagated stage by stage along

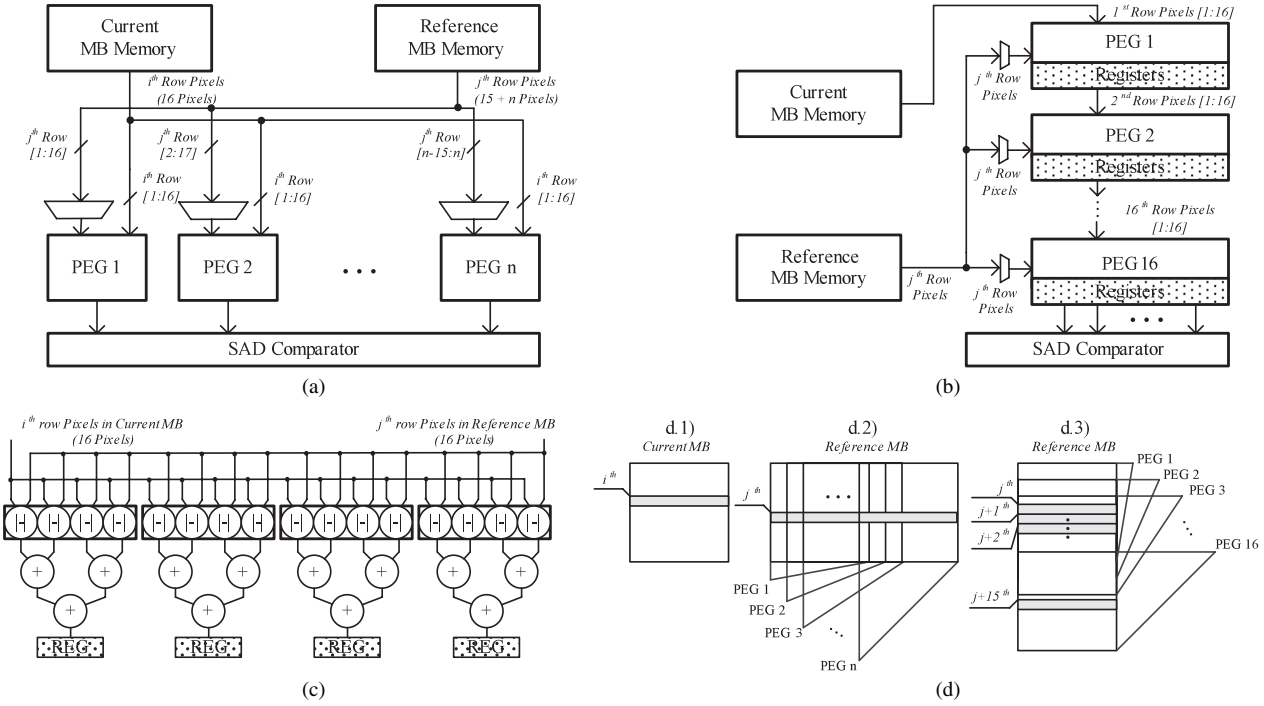



**Fig. 3** Existing VBSME architectures. (a) Horizontal-parallel architecture. (b) Vertical-parallel architecture. (c) PEG architecture. (d) Parallelism of two architectures. d.1) current MB. d.2) Horizontal MBs computed in parallel. d.3) Vertical MBs computed in parallel.

with PEGs in vertical direction. The reference MB's data is broadcasted to all of the PEGs simultaneously. In this way, 16 reference MBs in vertical direction can be processed in parallel.

Figure 3 d.3) illustrates the vertical MBs parallel computing. As current MB is propagated along with PEGs in vertical direction, each PEG has different row of current MB. For example, PEG 1 has the first row of current MB. PEG 2 has the second row of current MB. While *j*th row of reference MB is loaded in each PEG, the computation is done between 16 rows of current MB with *j*th row of reference MB. It is equal to *j*th row of current MB computed with 16 rows of reference MBs. As shown in Fig. 3 d.3), 16 rows of reference MBs can be processed simultaneously.

### 3.3 Discussion of Search Range Reduction Algorithm Used with Current VBSME Architectures

The VBSME architectures discussed in Sects. 3.1 and 3.2 are fixed data path architecture. All of the PEGs are connected through hard wire. As a result, the parallelism of motion estimation is also fixed. For example, the horizontal-parallel architecture only supports parallel computation of n horizontal MBs, and the vertical-parallel architecture only supports parallel computation of n vertical MBs. This fixed parallelism causes problems while using search range reduction algorithm.

As introduced in Sect. 2. Search range reduction algorithm reduces the original square large search range to a rectangular small range. This new rectangular search

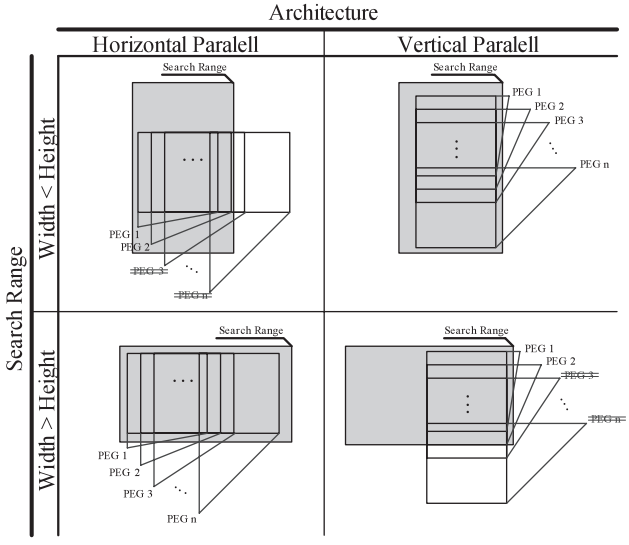



**Fig. 4** Current VBSME architectures used with search range reduction algorithm.

range has two different basic shapes: 1) Width<Height. 2) Width>Height. While using different architectures with these two shapes, it comes out different effect.

As illustrated in Fig. 4, different architectures used with different search range shapes are listed. There are two conclusions can be made:

1) For horizontal-parallel architecture: Because the parallelism of this architecture is horizontal, the hardware uti-

lization for search range whose Width>Height is much higher than Width<Height. As illustrated in the figure, while search range is Width<Height, parts of PEGs are out of search range. They can't be used while performing VBSME.

2) For vertical-parallel architecture: This architecture is much more suitable for Width<Height search range. While search range is Width>Height, parts of PEGs are out of search range.

Since almost all of the current VBSME architectures are fixed architecture. The parallelism of these architectures is either horizontal or vertical. However, the search range reduction algorithm produces flexible search range shape. The current VBSME architecture is not suitable for flexible search range. A new flexible VBSME architecture which can provide parallelism in both of vertical and horizontal direction is needed for search range reduction algorithm.

## 4. Proposed Reconfigurable Architecture of VBSME

In order to improve the flexibility of VBSME hardware implementation and provides dual-directional parallelism, a reconfigurable design of VBSME is proposed in this paper.

### 4.1 Reconfigurable Architecture for VBSME

As shown in Fig. 5(a), $n \times m$ PEGs (PE Groups) make up a $n \times m$ PEGA (PE Groups Array). REG (Register) in left part is used to support vertical parallelism like Fig. 3(b). A unique SAD comparator is used to process SADs from all of PEGs. The PEGA can parallel process multiple reference MBs both in horizontal direction and vertical direction. As shown in Fig. 5(b), while using $n \times m$ PEGA, $i$th row of current MB can be parallel computed with n rows of reference search area (from $j$th row to $j + n - 1$th row), and every rows in reference search area contains m reference MB rows. Thus, totally $n \times m$ reference MBs can be processed in parallel.

Different from all of existing VBSME design, the PEGA in this paper is not fixed. It can be configured to different structures by adjusting n and m. The SAD comparator module also can be configured to adapt to PEGA structures. While using different n and m to configure the PEGA, the various size of parallelized reference MBs array can be achieved. For example, while $n = 2$, $m = 8$, the PEGA structure is $2 \times 8$. Thus, $2 \times 8$ reference MBs can be processed in parallel. In this way, the reconfigurable architecture can be very efficiently used with search range reduction algorithm.

As shown in Fig. 6, the reconfigurable architecture supports different search range shapes very efficiently. For example, while the search range is Width<Height, the PEGA can be configured to $n > m$. All of the parallel computed reference MBs locates within the search range. In this way, all of the PEGs can be used simultaneously while performing VBSME. The hardware utilization is much higher than other architectures.

### 4.2 New Procedure of VBSME

While using proposed reconfigurable VBSME design and search range reduction algorithm, the procedure of VBSME should be modified as follows:

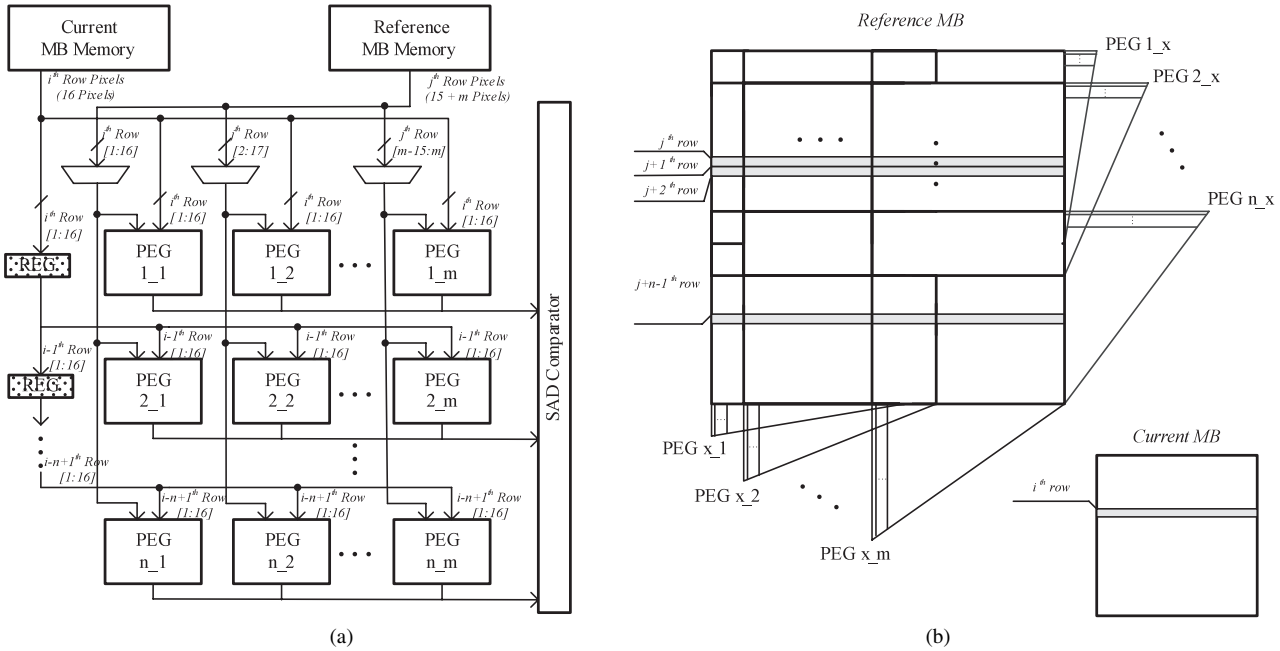



**Fig. 5** Proposed reconfigurable design of VBSME. (a) Reconfigurable architecture for VBSME. (b) Parallelism in horizontal and vertical direction.

*Step 1).* Set MVP,
*Step 2).* Set New Search Range,
*Step 3).* Configure PEGA based on new search range,
*Step 4).* Perform VBSME by using new search range and specific PEGA structure.

### 4.3 Sub-Modules Design

*A). PEG*

The design of PEG is shown in Fig. 7(a). The inputs of this module are one row of current MB (16 pixels) and one row of reference MB (16 pixels). The outputs are sub-blocks' SAD values. As shown in the figure, there are three levels of register banks: $4 \times 4$ SAD registers, $4 \times 8$ SAD registers and $8 \times 16$ SAD registers. The other SADs can be generated by these three register banks. For example, $8 \times 4$ SAD can be calculated by adding two $4 \times 4$ SAD registers. The updating of these three level register banks is different: $4 \times 4$ SAD registers are updated every 4 clock cycles, $4 \times 8$ SAD registers are updated every 8 clock cycles and $8 \times 16$ registers are updated every 16 clock cycles.

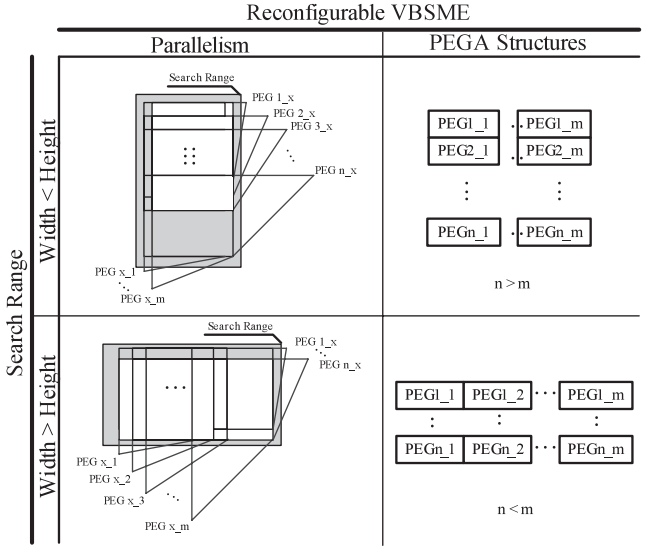



**Fig. 6** Reconfigurable VBSME architecture used with search range reduction algorithm.

*B). SAD Comparator*

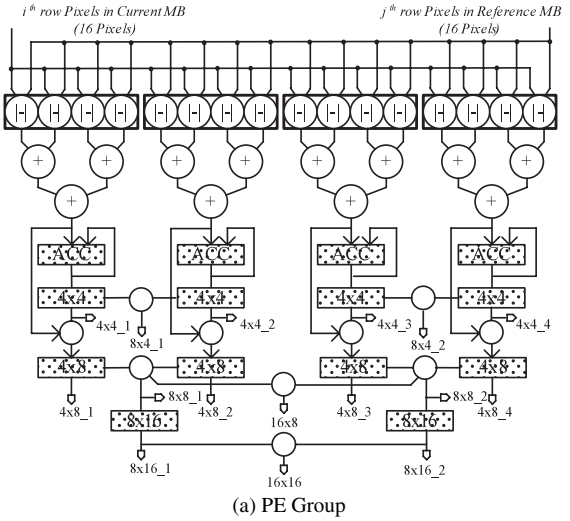
In order to reuse SAD comparator module by different PEGA structures, the SAD comparator is also configurable. The design and configuration of SAD comparator depend on the number of PEGs and structure of PEGA.

As shown in Fig. 7(b), there are three kinds of SAD comparators: 4-in SAD comparator, 2-in SAD comparator and 1-in SAD comparator. 4-in SAD comparator has three configurations: one 4-SADs comparator (*config.a*), two 2-SADs comparators (*config.b*) and four 1-SAD comparators (*config.c*). 2-in SAD comparator has two configurations: one 2-SADs comparator (*config.a*), two 1-SAD comparators (*config.b*). The configurations of these comparators depend on the data flows of VBSME.

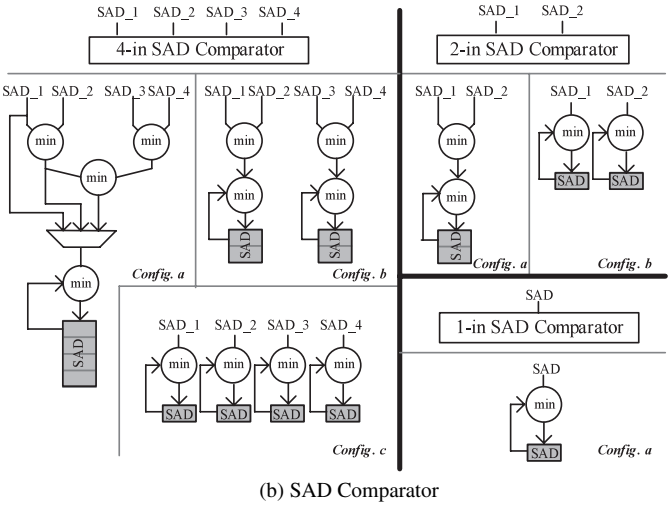
### 4.4 Dataflow

Figure 8 shows the data flow of reconfigurable VBSME. The circles in this figure represent the SADs with same types. Each circle (or SADs) belongs to different search point, and it is produced by different PEG in PEGA. The vertical ordinate represents clock cycles consumed in VBSME. The horizontal ordinate counters the number of SADs produced simultaneously by PEGA.

From this figure, we can see that SADs are produced block by block, and the block size is $n \times m$, same as the size of PEGA. The delay cycles between two SADs block depend on the SAD types. As discussed in Sect. 4.3A), $4 \times 4$ SAD and $8 \times 4$ SAD are output every 4 clock cycles. Thus, the delay cycles for $4 \times 4$ SAD and $8 \times 8$ SAD in data flow are 4 clock cycles. In the same way, the delay cycles for $4 \times 8$,




(a) PE Group




(b) SAD Comparator

**Fig. 7** Sub-module's design. (a) PEG structure. (b) Reconfigurable SAD comparator.

**Table 1** Experimental results. (H.264 baseline profile, 1 reference frame, JM11,0, QCIF@IPPP...).

| Sequence | PSNR | | | BitRate | | | Computational Cost | | Hardware Utilization | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fast Full Search (JM) | Search Range Reduction | Drop | Fast Full Search (JM) | Search Range Reduction | Increase | Average Search Range | Reduced Computat-ional Cost | Horizontal (16 PEGs) | Vertical (16 PEGs) | Proposed (16 PEGs) |
| **Search Range Reduction Algorithm 2 in Section 2.** | | | | | | | | | | | |
| container | 36.678 | 36.677 | +0.001 | 23411 | 23367 | -0.19% | 2.02x1.82 | 98.6% | 14.1% | 11.7% | 15.0% |
| bus | 34.157 | 34.169 | -0.012 | 315600 | 324240 | +2.7% | 13.74x3.30 | 82.3% | 98.2% | 29.9% | 98.5% |
| foreman | 36.006 | 35.979 | +0.027 | 87665 | 86095 | -1.8% | 6.04x4.99 | 88.2% | 62.3% | 49.8% | 92.9% |
| football | 34.003 | 33.961 | +0.042 | 515084 | 550931 | +7.0% | 6.33x6.38 | 84.2% | 54.2% | 57.0% | 76.2% |
| **Search Range Reduction Algorithm 1 in Section 2.** | | | | | | | | | | | |
| container | 36.678 | 36.678 | 0 | 23411 | 23389 | -0.09% | 3.80x3.64 | 94.6% | 36.4% | 34.2% | 100% |
| bus | 34.157 | 34.166 | -0.009 | 315600 | 317651 | +0.65% | 15.80x5.52 | 65.9% | 99.2% | 55.1% | 100% |
| foreman | 36.006 | 35.969 | +0.037 | 87665 | 86204 | -1.7% | 10.22x8.31 | 66.8% | 86.4% | 81.6% | 100% |
| football | 34.003 | 33.999 | +0.004 | 515084 | 520560 | +1.1% | 9.88x10.26 | 60.4% | 76.2% | 77.8% | 100% |

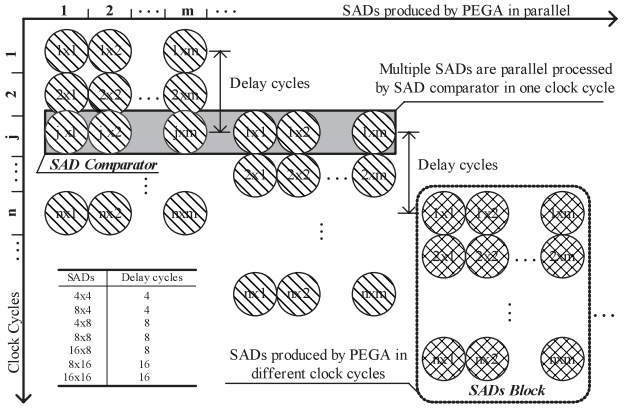



**Fig. 8** Data flow.

**Table 2** PEGA structures and SAD comparator configurations.

| PEG config. | SAD Comparators configurations | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4×4 SADs | 8×4 SADs | 4×8 SADs | 8×8 SADs | 16×8 SADs | 8×16 SADs | 16×16 SADs |
| 1×16 | Config a | Config a | Config a | Config a | Config a | Config a | Config a |
| 2×8 | Config a | Config a | Config a | Config a | Config a | Config a | Config a |
| 4×4 | Config a | Config a | Config a | Config a | Config a | Config a | Config a |
| 8×2 | Config b | Config b | Config a | Config a | Config a | Config a | Config a |
| 16×1 | Config c | Config c | Config b | Config b | Config b | Config a | Config a |

$8\times8$ and $16\times8$ SADs are 8 clock cycles, and for $8\times16$ and $16\times16$ SADs are 16 cycles.

Normally, m SADs (one row) with same type are output in the same clock cycle. However, for some special cases, multiple SADs blocks will overlap. Thus, multiple rows of SADs are output from PEGA in these overlapped cycles. For example, at $j$th clock cycle in Fig. 8, two SADs block overlapped, so $2\times m$ SADs are output from PEGA.

## 5. Experimental Results and Analysis

### 5.1 Experimental Implementation of Reconfigurable VBSME

In experimental VBSME implementation, we use 16 PEGs to implement PEGA ($n\times m$ equals 16). There are 5 structures of PEGA: $1\times16$, $2\times8$, $4\times4$, $8\times2$ and $16\times1$. As discussed in Sect. 4.4, there are 5 data flows corresponding to these 5 structures. Each data flow has different parallelism, and the SADs block also inherits 5 shapes: $1\times16$, $2\times8$, $4\times4$, $8\times2$ and $16\times1$.

The SAD comparator highly depends on PEGA structures. In this experiment, we use four 4-in SAD comparators for $4\times4$ SADs, two 4-in SAD comparators for $4\times8$ SADs, four 2-in SAD comparators for $4\times8$ SADs, two 2-in SAD comparators for $8\times8$ SADs, one 2-in SAD comparator for $16\times8$ SADs, two 1-in SAD comparators for $8\times16$ SADs and one 1-in SAD comparator for $16\times16$ SADs. The relationship between PEGA structures and SAD comparator configurations are listed in Table 2.

### 5.2 Search Range Reduction Algorithm Evaluation

The performance of search range reduction algorithms is listed in Table 1. There are two algorithms listed in this table. The search range is reduced much smaller by using algorithm 2 than Algorithm 1. The total computational cost of algorithm 2 is lower than algorithm 1. However, The PSNR (Peak signal-to-noise) drop and Bit-rate increment of algorithm 2 is worse than algorithm 1.

For fairly comparing, we use Fast Full search algorithm in JM11, because fast full search algorithm uses one search range for all sub-blocks in one MB, same as hardware-oriented search range reduction algorithm. From the table, the video quality is not reduced very much. Especially for algorithm 1, the video quality is almost as same as full search.

The average search range is presented as P × Q. The

**Table 3** Implementation and comparison.

| Ref | Tech | Area PE+SAD Comp. (Gates) | Freq. | PEG Number | Year |
|---|---|---|---|---|---|
| [8] | 0.18$\mu$m | 168K | 228MHz | 8 | 2006 |
| [9] | 0.18$\mu$m | 152K | 261MHz | 16 | 2006 |
| [10] | 0.18$\mu$m | 82K (Not include SAD Comp.) | 110MHz | 16 | 2006 |
| This paper | 0.18$\mu$m | 179K | 167MHz | 16 | 2007 |

meaning is the x-ordinate of search range is [−P, P], and the y-ordinate of search range is [−Q, Q]. The reduced computational cost can be calculated by Eq. (3).

$$\text{Computational Cost}_{\text{Reduced}} = \left(1 - \frac{\text{P} \times \text{Q}}{\text{Original search range}}\right) \times 100\% \quad (3)$$

The original search range is typically set to $16 \times 16$ for QCIF sequence. From Table 1, we can see that the computational cost is greatly reduced by search range reduction algorithm.

As a conclusion, Search range reduction algorithm can greatly reduce the computational cost with little PSNR drop. It is very useful for fast motion estimation. Different algorithms have different ability to reduce search range, and also produce different PSNR drop. Algorithm 1 and 2 are examples to show the effectiveness of search range reduction algorithm. Some new algorithms can be proposed to balance the computational cost and PSNR drop.

### 5.3 Proposed Reconfigurable VBSME Evaluation and Comparison

The hardware utilizations of horizontal-parallel architecture, vertical-parallel architecture and proposed reconfigurable architecture are listed in Table 1. The definition of hardware utilization is:

$$\text{Hardware Utilization} = \frac{\text{Used PEGs}}{\text{Total PEGs}} \times 100\% \quad (4)$$

The used PEGs are decided by reduced search range. As discussed in Sect. 3.3, while the parallel processed reference MBs are out of search range, part of PEGs can't be used while doing VBSME. For fairly comparing, we use 16 PEGs for all of architectures.

We use four video sequences with different motion content to do comparison. Container is an almost still video sequence. Bus has intensive motion in horizontal direction. Forman is an irregular moving sequence. Football is sequence with violent motion. From the table, the hardware utilization of horizontal-parallel and vertical-parallel architectures are much lower than reconfigurable architecture, especially for some sequences with special motion or irregular motion. Because such kinds of motion result in very narrow search range, lots of PEGs can't be used in fixed VBSME architecture.

### 5.4 ASIC Implementation and Comparison

Using TSMC 0.18$\mu$m CMOS standard cell library and Sysnopsys Design Compiler Tools, the ASIC implementation results and comparison are shown in Table 3. The total hardware cost for our design is 179K Gates, and the frequency is 167 MHz. Song and Liu's design in [8] and [9] are horizontal-parallel VBSME architecture which uses 8 and 16 PEGs. Chen's design in [10] is vertical-parallel VBSME architecture which uses 16 PEGs. The hardware cost listed in the table includes PEGs and SAD comparator cost, and doesn't include Memory cost. Chen's design in [10] didn't include SAD comparator.

Compared with others' design, our design achieves high hardware utilization while using search range reduction algorithm. Since the search range reduction algorithm greatly reduces the computational cost. Our design needn't very high frequency.

### 5.5 Processing Capability

The processing capability of proposed design is shown in Eqs. (5) and (6).

$$NUM_{CLK} = \frac{S \times R}{N \times U} \times 16 \quad \text{(clk/MB)} \quad (5)$$

Equation (5) calculates the number of clock cycles consumed for one current MB's motion estimation. $S$ is search range size. $R$ is average search range reduced rate which depends on search range reduction algorithm. $N$ is number of PEGs. $U$ is hardware utilization of PEGA.

$$\begin{aligned} \textit{Processing Capability} &= \frac{F}{NUM_{CLK}} \\ &= \frac{F \times N \times U}{S \times R \times 16} \quad \text{(MBs/s)} \quad (6) \end{aligned}$$

Equation (6) calculates the number of processed Current MBs in one second. $F$ is working frequency for motion estimation. From Eq. (6), it can be seen that:

1) Under the same working condition (Same $F$, $N$, $S$, $R$), because the reconfigurable design achieves high U, the processing capability is higher than fixed architecture designs.
2) Under the same processing capability, the reconfigurable design can decrease the working frequency to achieve low-power computation.

### 5.6 Video Resolution Vs. Hardware Utilization

In order to evaluate the effectiveness of our architecture for different video resolutions, the search range reduction algorithm 2 is performed to CIF sequence in this experiment.

Table 4 shows the PSNR and Bit-rate comparison of search range reduction algorithm with Fast Full search algorithm. The result is good for CIF sequence to use search

**Table 4** Experimental results, CIF@IPPP...

| Sequence | PSNR | | | BitRate | | |
|---|---|---|---|---|---|---|
| | Fast Full Search | Algorithm 2 | Drop | Fast Full Search | Algorithm 2 | Increase |
| container | 36.298 | 36.288 | +0.01 | 0.132e6 | 0. 132e6 | -0.4% |
| bus | 34.939 | 34.936 | +0.00 | 1.010e6 | 1.073e6 | +6% |
| foreman | 37.060 | 37.062 | -0.00 | 0.343e6 | 0.349e6 | +1% |
| football | 34.891 | 34.867 | +0.02 | 1.477e6 | 1.660e6 | +10% |

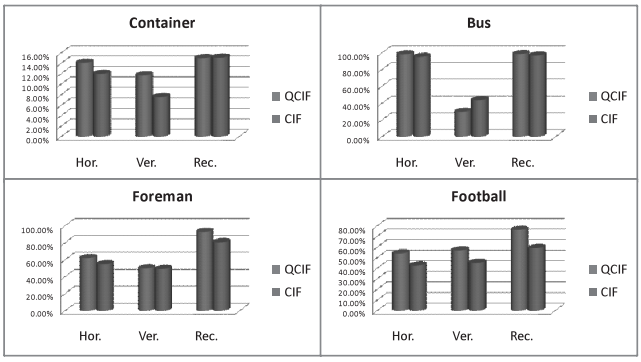



**Fig. 9** Hardware utilization for QCIF and CIF. (Search range reduction algorithm 2 is used).

range reduction algorithm.

The comparison of hardware utilization of QCIF and CIF sequences are shown in Fig. 9. For QCIF sequence, the original search range is set to $16 \times 16$. For CIF sequence, the original search range is set to $32 \times 32$. From this figure, our reconfigurable design also outperforms Horizontal and Vertical design.

There is an interesting thing in this figure. For CIF sequence, while using search range reduction algorithm 2, the hardware utilization is reduced. The reason is that CIF is larger than QCIF, every MB in CIF becomes flat compared to QCIF. As a result, much smaller MVDs are produced. While using algorithm 2, these small MVDs produce small search range, and finally reduce the hardware utilization. If using different research range reduction algorithms, the things will be different.

As a conclusion, hardware utilization highly depends on search range reduction algorithm. Under the same algorithm, hardware utilization is also high related to video resolution. Unless the utilization for Horizontal and Vertical design becomes 100%, the performance of reconfigurable design will outperform both of Horizontal and Vertical design.

## 6. Conclusion

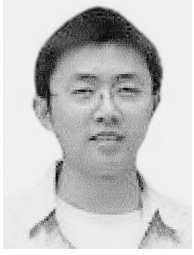
In this paper, we proposed a reconfigurable design for VBSME. This design provides flexible parallelism of motion estimation both in horizontal and vertical direction. The experimental results show that this design can be very efficiently used with search range reduction algorithm. The hardware utilization of this design is much higher than current VBSME designs. The proposed reconfigurable VBSME design is very suitable to be used in low-power video coding system.

## Acknowledgement

This research is supported by CREST, JST.

### References

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity," IEEE Circuits Syst. Mag., vol.4, no.1, pp.7–28, First Quarter 2004.
[2] T. Wiegand, G.J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.7, pp.560–576, July 2003.
[3] I.E.G. Richardson, H.264 and MPEG-4 video compression, video coding for next-generation multimedia, John Wiley & Sons, 2003.
[4] ITU-T, "Advanced video coding for generic audiovisual services," http://www.itu.int/rec/T-REC-H.264, 2003.
[5] Y.H. Chen, T.C. Chen, and L.G. Chen, "Hardware oriented content-adaptive fast algorithm for variable block-size integer motion estimation in H.264," International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp.341–344, Dec. 2005.
[6] JVT-Q089, "Comments on motion estimation algorithms in current JM software," http://ftp3.itu.ch/av-arch/jvt-site/2005_10_Nice/JVT-Q089-L.doc
[7] JM 11.0 source code, http://iphome.hhi.de/suehring/tml/
[8] Y. Song, Z.Y. Liu, S. Goto, and T. Ikenaga, "Scalable VLSI architecture for variable block size integer motion estimation in H.264/AVC," IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.979–998, April 2006.
[9] Z.Y. Liu, Y. Song, T. Ikenaga, and S. Goto, "A fine-grain scalable and low memory cost variable block size motion estimation architecture for H.264/AVC," IEICE Trans. Electron., vol.E89-C, no.12, pp.1928–1936, Dec. 2006.
[10] 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. I, vol.53, no.3, pp.578–593, March 2006.



**Yibo Fan** received the B.E. degree in electronics and engineering from Zhejiang University, China in 2003 and M.S. degree in Microelectronics from Fudan University, China in 2006. Currently, he is a Ph.D. candidate in Graduate School of Information, Production and Systems, Waseda University, Japan. His research interesting includes information security, video coding and associated VLSI architecture.

**Takeshi Ikenaga** received his B.E. and M.E. degrees in electrical en- gineering and the Ph.D. degree in infor-mation & computer science from Waseda University, Tokyo, Japan, in 1988, 1990, and 2002, respectively. He joined LSI Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 1990, where he has been undertaking research on the design and test methodologies for high-performance ASICs, a real-time MPEG2 encoder chip set, and a highly parallel LSI & System design for image-understanding processing. He is presently an associate professor in the system LSI field of the Graduate School of Information, Production and Systems, Waseda University. His current interests are application SOC for image, security and network processing. Dr. Ikenaga is a member of the IPSJ and the IEEE. He received the IEICE Research Encouragement Award in 1992.

**Satoshi Goto** was born on January 3rd, 1945 in Hiroshima, Japan. He received the B.E. and M.E. degree in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He also received the Dr. of Engineering from the same university in 1981. He is IEEE fellow, member of Academy Engineering Society of Japan and professor of Waseda University. His research interests include LSI system and Multimedia System.