

# Implementation of Motion Estimation IP Core for MPEG Encoder

Jinku CHOI† Nozomu TOGAWA ‡ Masao YANAGISAWA † and Tatsuo OHTSUKI †

† Department of Electronic, Information and Communication Engineering  
Waseda University

‡ Department of Information and Media Sciences  
The University of Kitakyushu

3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan  
Tel: +81-3-5286-3396 Fax: +81-3-3203-9184  
E-mail: choezg@ohtsuki.comm.waseda.ac.jp

**Abstract** Motion estimation is the most computationally demanding part of MPEG applications. Hence, it usually requires hardware for a real-time implementation. However, motion estimation was implemented by hardware, it has a limitation on flexibility. If motion estimation can choose the most suitable algorithm according to the changing characteristics of input image signals, we can get benefits, which are improved quality and performance, reduced power consumption, and an optimized system. In this paper, we propose a reconfigurable approach to motion estimation algorithm. Our algorithm determines motion type and then selects adapted algorithm in order to improve quality and performance of images. We implemented the flexible and reconfigurable hardware architecture by hardware with address generator unit, delay unit, and parameters. We did not use FPGAs and programmable devices. Our architecture supports more than one block-matching algorithms and parameters providing to optimize system. We are implementing our architecture by using hardware description language (VHDL) and synthesis design tools. We analyze the performance of the architecture and present adaption to algorithm for a low cost real time application.

## 1. Introduction

The motion estimation is a technique to eliminate temporal redundancy of image sequences and it is a central part of the MPEGs (1/2/4) and video compression standards (H.261 /H.263). However, the motion estimation has a lot of computational complexity in the video encoders. Therefore it becomes the bottleneck in the real time applications, thus it seems that the motion estimation is implemented will determine the hardware characteristics. For mobile video applications, the requirement emerges to provide the necessary computational power under low power constraints. The requirement for applications differs significantly according to image encoding algorithms. Otherwise, the algorithms will not show the desired results.

It is necessary to discuss trade-off for algorithms and architectures of motion estimation and take into account design issues, chip area, speed, I/O bandwidth, power consumption, image quality, and some requirements for application.

Many motion estimations algorithms and architectures have been proposed. One of the main design goals has been the reduction of computational complexity and the power

consumption of the motion estimation while keeping quality of image. Some algorithms and architectures succeeded in reducing the power consumed and satisfying the required performance. However, some of them do not guarantee the optimal solution due to restricting the search space.

The reconfigurable algorithm, programmable algorithm, and adaptive algorithms and architectures have been proposed in literature [5]-[10]. Reconfigurable hardware based on FPGAs does allow parallelism pipelining, local memory and both function and specialization, but FPGAs suffer from several disadvantages for power and speed.

A programmable architecture proposed by literature [9] with macro-commands has been used to implement various search, which are sub-sampling algorithms. The programmable SIMD (Single Instruction Multiple Data) motion estimation processor was proposed by literature [10], which can support variable block size. The processor-based architecture offers a high flexibility and programmability, but compared to application-specified ASICs, it is paid for higher power consumption, higher area demand, and lower throughput. Application-specified ASICs usually have the disadvantage of limited flexibility.

The flexible motion estimation is capable of performing more than one algorithm. VLSI architectures for two block-matching algorithms were proposed by literature [6]. The variable block-size motion was investigated by several authors in literature [7] [8]. It provides better estimation of small and irregular motion, but supports different block sizes, which require more data bits in the bit stream for encoding motion vectors at smaller block sizes.

The flexibility and programmability of the VLSI architecture were required to support a wide range of motion estimation algorithms. However, increased flexibility has usually been paid for a loss of efficiency. Most of proposed architectures can be implemented in a smaller area, a reduced computational complexity, and little power consumption.

In this paper, we propose algorithm and flexible hardware architecture, which supports more than one algorithm. Our algorithm determines motion type whether large or small motion in searching and then selects adapted algorithm to the varying images characteristics in order to improve quality and performance of images. In addition, we are implementing our algorithm by VHDL.

Our work focuses on a design system of hardware architecture level, which has flexible architecture that is sup-

porting several motion estimation algorithms and optimizes an architecture associated with parameters.

Our architecture is applicable to object coding in which foreground or background objects and segment coding which is to classify the image into changed segments or unchanged segments. For example, motion estimation algorithm can be optimized for low complexity for the background and unchanged segments video objects, while for higher complexity or high quality for the foreground and segments video objects.

## 2. Block-Matching Motion Estimation

Motion estimation removes temporal redundancy between successive frames in digital video. The most popular technique for motion estimation is the block-matching algorithm.

Among the block-matching algorithms and methods, the full-search algorithm (FS) provides an optimal solution by matching all the blocks in a search area, but the computational complexity is too high. An abundant number of fast motion estimation algorithms were proposed to reduce the computational complexity in literature [1-4]. Those algorithms are composed of one or more of these basic strategies; search strategy, block and feature matching, temporal, and spatial prediction, distance criterion, pixel decimation etc. Fig.1 describes the basic principle of the block-matching algorithm. In block matching, the current frame is divided into non-overlapping blocks, and then search for best-matched block between a current block and a reference block.

Block  $I_n(k,l)$  is defined as the pixel intensity at location  $(k,l)$  in the  $n$ -th current frame, Block  $I_{n-1}(k+u,l+v)$  is the pixel intensity at location  $(k+u,l+v)$  at the  $(n-1)$ -th reference frame. The current block size is defined as  $M \times N$  pixels. The motion of each block is represented by motion vector. This is the displacement between the current block ( $I_n$ ) and its best match block within search windows in a reference frame ( $I_{n-1}$ ).

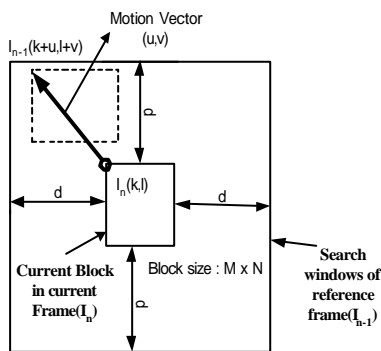


Fig.1 Block matching diagram.

This matching is usually decided by using distance criteria between the current blocks in current frame and search area blocks in reference frame. In general, SAD(Sum of Absolute Differences) is widely used as matching criterion because it is simple, can offer some advantages for VLSI implementation, and show not bad results in predicted image quality. We also use SAD, and reduce the computation. the SAD is defined as:

$$SAD(k,l;u,v) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |I_n(k+i,l+j) - I_{n-1}(k+i+v,l+j+u)| \quad (1)$$

Where  $M$  and  $N$  represents block size  $M \times N$  and  $(u,v)$  is the location in search windows, block  $I_n(k+i,l+j)$  is a pixel at  $(k+i,l+j)$  in the current frame and block  $I_{n-1}(k+i,jl)$  is a pixel in the reference frame.

The motion estimation process determines the best displacement motion vector(MV) by trying to minimize the energy within a search window. When the value of  $SAD(u,v)$  is minimum, $(u,v)$  stands for the motion vector of block in search area. The motion vector  $MV(u,v)$  is defined as:

$$MV(u,v) = \underset{(u,v)}{\operatorname{argmin}} SAD(u,v) \quad (2)$$

### 2-1 Fast Block-Matching Algorithms

We describe general block-matching algorithms in this section. Our algorithm is implemented with different kinds of block-matching algorithms. One is TSS(Three Step Search) and the other is BBDS(Block Based gradient Decent Search). The TSS performs better in searching large motion while BBDS performs better in searching small motion. In addition, we perform pixel sub-sampling algorithm according to external parameter.

#### (1) Three Step Search (TSS)

The three-step search algorithm (TSS) is proposed by literature [1]. This algorithm is based on a coarse-to-fine approach with logarithmic decreasing in step size. The initial step size is half of the maximum motion displacement. The center is moved to the minimum SAD. For each step, nine checking points are matched and the minimum SAD point of that step is chosen as the starting center of the next step.

#### (2) Block-Based Gradient Decent Search (BBDS)

The block-based gradient descent search algorithm (BBDS) was proposed by literature [4], It relies on the assumption of center-based motion. This algorithm uses a very center-biased search pattern of nine checking points in each step with step size of one.

The search is stopped, when the minimum checking point of the current step is the center one, or it reaches by the search window boundary. We suppose that search procedure stops after three steps.

The BBDS searches for motion vector along the direction where the minimum SAD is expected to be. The BBDS algorithm performs better in searching small motions, and the computational complexity is significantly less than that of TSS [4].

#### (3) Pel Sub-Sampling Techniques

The pel sub-sampling reduces the number of pixels taken into account for the distance calculation at every position. This method reduces the computation complexity for the distance criterion and the memory bandwidth, among the possibilities to reduce the number of pixels per block, sub-sampling by ratio of 2:1 or 4:1.

### 3. Proposed Algorithm

Usually, the images have various contents and characteristics. We can get better performance, if the algorithm of motion estimation is adaptively selected according to image characteristics.

We propose algorithm that selects better algorithm according to motion type. In order to select better algorithm, we are the first to determine motion type whether small motion or large motion. The motion type was determined by checking motion vector and SAD.

#### 3-1 Selection Algorithm

We propose algorithm for adaptively selection algorithm to the varying motion content. Selection algorithm procedures are as follows:

Step1. Calculate the SAD at origin point.

Step2. Compare the  $SAD_n$  with a threshold ( $TH_n$ ).

If the  $SAD_n < TH_n$ , we select BBDS algorithm, we classify the motion characteristics as small motion and go to step3. Else, we select TSS algorithm, we also classify it as large motion and go to step3.

Step3. We search the MV using each algorithm.

Step4. We update the threshold ( $TH_n$ ) based on the output of motion vector and SAD.

We defined small motion, which means that the motion vectors are inside the 3x3 windows and large motion vectors outside the 3x3 windows.

#### 3-2 Update Threshold

The motion type is updated by checking motion vector. If the algorithms can search their statistically favorite motion types, matching accuracy will increase. We assume that block distortion measure of the origin checking point increases with the motion vector magnitude. This assumption not always holds. The threshold  $TH_n$  is updated by using the following linear equation:

$$TH_n = TH_{n-1} + (dSAD/dMV) \quad (3)$$

Where , are constants and factors to adjust threshold.

$dSAD = SAD_n - SAD_{n-1}$ , and  $dMV$  means change of motion vector magnitude,  $dMV = MV_n - MV_{n-1}$ .

### 4. Proposed Architecture

We propose hardware architecture for implementing our proposed algorithm. Our hardware architecture employs multi-block matching algorithms, which are TSS, BBDS, and Pel Sub-sampling. In order to select adaptive algorithm, the motion type is the first determined to large motion or small motion.

If these algorithms can search their statistically favorite motion type, the matching accuracy and performance will be improved. Our proposed architecture shows in Fig.2. This supports multi-motion estimation modes and has control unit, and PE unit. The control unit has some modules; parameter control, memory-addressing control, PE control unit and selection motion.

The parameter control unit has inputs, and external constraints are system requirement and constraints for power

and performance. MS0 and MS1 select architecture and algorithm mode, for example, block-matching algorithms, which are pel sub-sampling, 8x8, or 16x16, block size.

Those parameters are used to control algorithm and optimize architecture. The controller unit controls address generator, PEs, delays, and motion selection. The unit calculates the start and next address of data in the memory and interface with external memory. If address generator is programmable, the hardware architecture will have flexibility.

The address decoder consists of an adder and a multiplier. Two algorithms have similar search procedure, but have different starting address.

Thus, we can implement simply. Two algorithms show a similar scanning of the search but they have a different starting point. Therefore, we simply implemented without a lot overhead. To select algorithm, we compared the  $SAD_n$  with a threshold ( $TH_n$ ), and then the result is input address generator unit. PE control can control PEs for power management, and enable or disable PE.

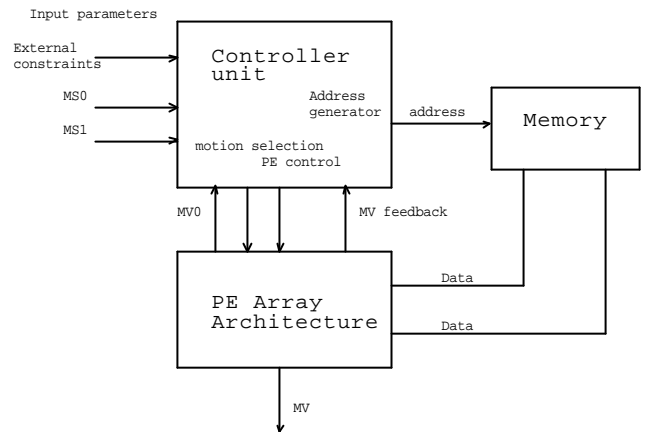


Fig.2: block diagram of flexible motion estimation.

The generally architecture of the motion estimation consists of a processing element, a memory (search area and current block), memory controller, and MV detector module.

Once motion type is determined and then one of the two algorithms is employed for search according to the predicted motion type. We combined with off-chip memory in this work. Our architecture offers flexibility but inquired less overhead compared to other architectures.

#### 4-1 PE(Processor Element) Unit

A single PE unit shows in Fig3. The PE has the reference and current data input ports, clock, reset, control signals.

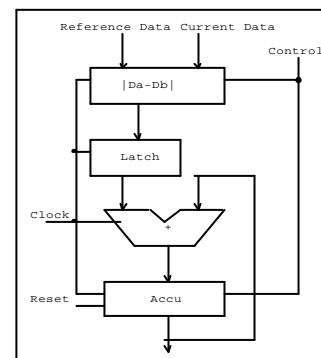


Fig3: SAD PE element

Each PE unit has clock with the appropriate data from memory. Each PE unit calculates the sum of all absolute difference until an external signal resets the accumulator of the SAD summation.

#### 4-2 PE Array

Motion estimation can be parallelized at the PE level with each PE unit, calculating the PE of a single motion vector candidate. This PE array can be of 1-Dimensional or two Dimensional types.

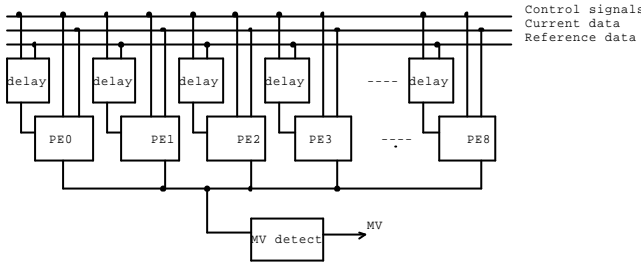


Fig4: PE array with 9 PEs

We select 9x1 1-Dimension PEs array (Fig.4). We can parallelize by using 9 PEs, each PE calculating the SAD of one of the nine search positions. Control line can be programmable delay unit and enables or disables PE units. MV detect unit is used to find the minimum SAD and can determine the stopping of a search procedure.

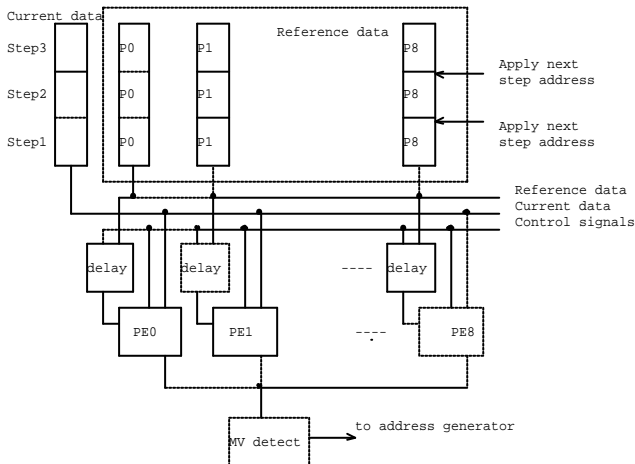


Fig.5: PE array block diagram.

Fig.5 shows basic PE array structure and its input data flow. The current pixels sequentially input and input to all PEs. The reference pixels are buffering in each delay unit and then being used for calculation.

To map our algorithm, we select 9x1 PE array. The TSS and BBDS algorithms calculate nine candidates check points in each search step.

The TSS and BBDS algorithms consist of three loops. The loop1 is looping for each of the search steps, the loop2 is looping for each of nine candidate position of motion vectors, and the loop3 is looping for calculating the absolute pixel difference. We can parallelize loop2 by using

9PEs with a single PE calculating the SAD of one of the nine search positions and sequentially execute operation in each of step looping and calculate pixels.

The TSS and BBDS are require to 265 cycle at each search step for calculate SAD. Each macro-block can be processed in 795 cycles.

#### 4-3 Address Generation Units

The address generation unit became an important part of the hardware design for multi-motion estimation algorithms. This unit consists of multiplier, adders, counters, and registers. The address generator can provide base address of next step when each PE is calculated at the 9 check points, and then a correct address can be picked out immediately after the minimum SAD is decided. This unit is achieved by a selection of different parameters.

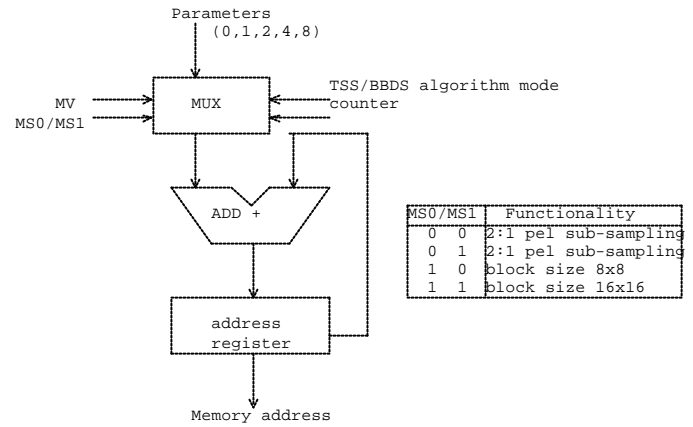


Fig.6: Address generation unit

## 5. Implementation Results

At first, we select motion estimation algorithm and implementing by C language, we evaluate the motion estimation algorithm. In the design space of hardware architecture, we designed first single PE, PE array and control unit with VHDL description, synthesized, and measured clock, and chip area.

#### 5-1 Quality Comparisons

The PSNR (peak signal to noise ratio) is used to evaluate the quality of a reconstructed image sequence of an image quantitatively. It is defined as follows:

$$PSNR [dB] = 10 \log_{10} \left[ \frac{255^2}{MSE} \right] \quad (4)$$

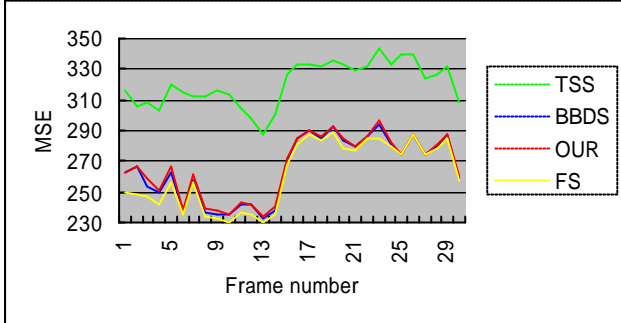
Where MSE (Means Square Error) is defined as:

$$MSE(k, l; u, v) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I_n(k+i, l+j) - I_{n-1}(k+i+u, l+j+v)]^2 \quad (5)$$

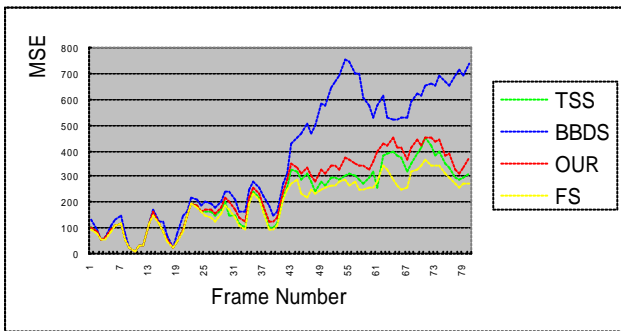
We simulated it by using test image sequences “Football (SIF size; 352x240 pixels)”, “Tennis (QCIF size; 176x144 pixels)” and “Mobile (SIF size; 352x240 pixels)”.

The search range is -15 to 16 pixels, and the block size is fixed at 16 x16 pixels for both horizontal and the vertical direction. The proposed our algorithm is simulated with TSS, BBDS, and FS(full search algorithm) algorithm. We

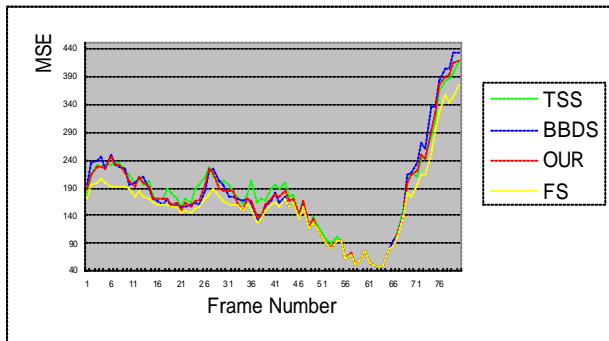
compare the MSE performances of different algorithms. Fig.7 shows an MSE comparison for four algorithms for test sequence images. These results show that the MSE of our architecture are a little higher than that of one of two algorithms. The MSE and average search times proposed are always very close to either the segment of TSS or BBDS.



(a) Mobile(352x240 pixels)



(b) Tennis(176x144 pixels)



(c) Football(352x240 pixels)

Fig.7: MSE comparison for three algorithms

## 5-2 Implementation Data

Table.1 shows LSI architecture has been designed by using the Synosys design tools and 0.35 $\mu$  Hitachi CMOS technology. The clock rate is constrained to 20[nsec].

Table.1 : Area of LSI Architecture

Module	Area[ $mm^2$ ]
PE Array (9x1PEs)	0.63786
Controller	0.31262
Memory (Off-chip)	0.0
Total	0.95048

Our architecture has less chip area and power consumption than the FS architecture and implemented two algorithms; TSS, BBDS.

## 6. Conclusion

In this paper, we described a flexible architecture that supports two motion estimation algorithms and optimizes system with parameter. As simulation results, we improved performance by means of selection of better algorithm. We proposed flexible architecture, which is capable of performing multi-algorithms. Our flexible architecture was advantageous in terms of performance, power consumption, and compared to in terms of chip size, computation time and memory bandwidth.

In addition, remaining work will be improving motion selection algorithm updating threshold method algorithm and hardware architecture, and expanding architecture with more than three algorithms and power consumption.

Finally, We would like to optimize each module and overall system.

## References

- [1] T. Koga, K. Linuma, A. Iilima and T. Ishiguro, "Motion compensated inter-frame coding for video conferencing," in *Proc. NTC81*, pp.C9.6.1-9.6.5, New Orleans, L.A., 1981.
- [2] L.M.PO and W.C Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits and System for Video Tech.*, vol.6, no.3, pp.313-317, 1996.
- [3] J.Lu and M.L.Liou, "A simple and efficient search algorithm for block matching motion estimation," *IEEE Trans. Circuits and System for Video Tech.*, vol.7, no.2, pp.429-433, 1996.
- [4] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits System. and Video Technol.*, vol. 6, pp. 419-422, Aug. 1996.
- [5] M. Bierling, "Displacement estimation by hierarchical block-matching," in *Proc. SPIE Conf. on Vis. Commun. and Image Proc.*, vol. 1001, pp. 942-951, Jan. 1988.
- [6] Santanu Dutta and Wayne Wolf, "A Flexible Parallel Architecture Adapted to Block-Matching Motion Estimation Algorithms," *IEEE Trans. Circuit and System for Video tech.*, vol.6, no.1, pp74-86, Feb. 1996.
- [7] J.LEE, "Optimal Quadtree for variable Block Size Motion Estimation," *ICIP-95*, Washington DC, Oct. 1995.
- [8] J.P.Berns and T.G.Noll, "A flexible motion Estimation Chip for Variable Size Block Matching," *ASAP96*, International Conference on Application-Specific System, Architecture and Processor, Chicago, 1996.
- [9] H.D.Lin and A.Anesko, B.Petryna, "A 14GOPS programmable Motion Estimator for H.26X Video Coding," *IEEE Journal of Solid State Circuits*, vol31, no.11, pp. 1096-1075, Nov.1996.
- [10] E.Hanssens and J.D.Legat, "A Parallel Processor for motion Estimation," *SPIE Vol.2727*, pp1006-1016, 1996.
- [11] A. Takagi,S. Muramatsu, and H. Kiya, "Motion estimation with power scalability and its VHDL model," in *IEEE Int Conf. Image Proc.*, Sep.2000.