A Low-Power Motion Estimation Block for Low Bit-Rate Wireless Video

R. Steven Richmond II
Hewlett-Packard Company
3404 E. Harmony Road M/S 88
Fort Collins, CO 80525

E-mail: str@fc.hp.com

Dong Sam Ha
Virginia Tech VLSI for Telecommunications (VTVT) Lab
Bradley Dept. of Electrical and Computer Eng.
Virginia Tech, Blacksburg VA 24061

E-mail: ha@vt.edu   Web: www.ee.vt.edu/ha

ABSTRACT

This paper presents a low-power design of a motion estimation block targeting for a low-bit rate video codec H.263. The block is based on the Four-Step Search algorithm. The proposed design offers up to 38 % power reduction for logic blocks alone over a "baseline" implementation of the Four-Step Search (4SS) algorithm and up to 58 % power reduction over a baseline model of the Three-Step Search (TSS) algorithm. In addition, our design reduces power dissipation of an on-chip memory by up to 32% over the 4SS and 27% over the TSS.

1. INTRODUCTION

Motion estimation is one of the most important steps of any video encoding system. Motion estimation takes a reference block of pixels and attempts to find a suitable match for the candidate block, so re-encoding of the entire block can be eliminated. As result, the system can transmit only the *difference* across the channel, saving bandwidth.

For mobile video encoding applications, the motion estimation causes a couple of technical problems. First, with even a modestly sized search area, the number of computations can grow large very quickly. Second, the motion estimation consumes nearly 50 % of the power of a video encoding system [1]. Therefore, it is critical that a motion block be designed to dissipate a small amount of power for portable devices.

A motion estimation operation finds a motion vector, indicating the best direction of the motion, and a rating of the "fitness" of that motion vector. A variety of methods exist to compute the fitness of a motion vector. The method most widely used is the simple SAD (Sum-of-Absolute-Differences) method. SAD computes the sum of the differences in pixel values between the reference and the candidate blocks. The equation for a SAD rating for a given motion vector (x,y) is given as such:

$$SAD(x, y) = \sum_{j=0}^{N} \sum_{i=0}^{N} |curr(i, j) - prev(i + x, j + y)| \qquad (1)$$

where curr(*i,j*) is a pixel in the current (reference) frame and prev(*i+x,j+y*) is a pixel offset by (*x,y*) in the previous (candidate) frame.

A variety of algorithms exist to perform motion estimation. The most simple and direct is the full-search block matching (FSBM) algorithm. It calls for searching the entire set of all possible motion vectors and always yields the optimum motion vector with the lowest SAD value. However, searching the entire area is time consuming and may be unnecessary for low-quality video. A variety of hierarchical algorithms were proposed to alleviate the problem. These algorithms include the Three-Step Search (TSS) algorithm and the Four-Step Search (4SS) algorithm [2], [3]. These algorithms search a few points initially and refine the search over time until possibly an optimum vector is found. They increase the throughput by searching fewer points, and hence dissipate less power, compared to the FSBM, but the solutions are not necessarily optimum. However, since the quality of low bit-rate video such as the H.263 video codec is often degraded to meet the low bandwidth requirement, an optimal solution may not be always necessary. The circumstance was exploited for the proposed design to reduce power dissipation.

The rest of this paper is organized as follows. Section 2 describes the 4SS algorithm employed in the proposed design. Section 3 explains implementation of a baseline 4SS algorithm, on which our low-power design is based. The baseline model also served as a reference for power improvement for the proposed design. We present proposed low-power design techniques in Section 4 and. experimental results on power estimation in Section 5. Finally, Section 6 concludes the paper.

2. FOUR-STEP SEARCH ALGORITHM

The 4SS algorithm attempts to address some of the problems with the TSS algorithm such as a fixed number of search steps and center-biasing [3]. A salient point of the 4SS is that all the four steps of the algorithm are not necessary for each motion estimation operation. The algorithm aborts early for motion vectors close to the center (0,0) of the search area.

The four step algorithm proceeds as follows [3]:

1. Using a search box of 5x5, search the area centered around (0,0).
2. If the previous step's winner is NOT the zero vector centered at (0,0), search again with a 5x5 box centered at step 1's winner, otherwise proceed to Step 4.
3. Repeat Step 2 centered at Step 2's winner.
4. Search using a 3x3 box around the previous step's winner.

The following Figure 1 illustrates two example searches finding motion vectors at (+3, -7) and (-7, +7).

The 4SS algorithm needs 17 searches in the best case. The worst case for the 4SS algorithm is 27 search points compared to 25 for

60

the TSS. If most motion vectors are around at the center of the search area, the 4SS performs better than the TSS.
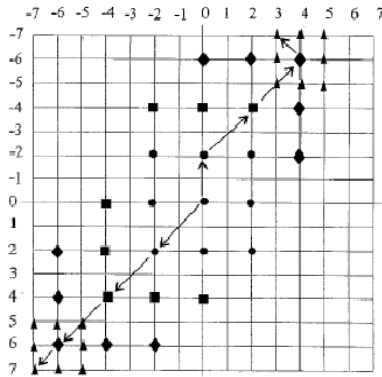

Figure 1. Example Search Patterns for 4SS [3]

3. BASELINE MODEL IMPLEMENTATIONS

The ITU-T H.263 video codec standard is for low bit-rate video systems. It provides acceptable picture quality and frame rate for video streams at a multiple of 64 Kbps or less than 64 Kbps.

Our baseline system design is based on the implementation (which is efficient in hardware) presented in [4] adapted for the 4SS algorithm. Since our system is targeted for H.263, the macroblock size is set to 16x16 pixels conforming to the H.263 standard. The search area is [-7,+7] by [-7,+7]. Figure 2 shows a block diagram of our motion estimation block. Several important blocks of the block diagram are explained in the following:
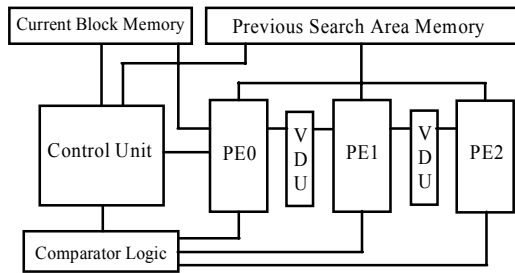

Figure 2. Block Diagram of the Motion Estimation Block [4]

3.1 Processing Element (PE)

The three PEs operates on a set of three motion vectors at a time. (Readers may refer to the block diagram of the low-power version of a PE shown in Figure 3.) A subtractor and an absolute value unit compute the difference between a pair of pixels under consideration. The difference is accumulated by an adder and a register, whose final output is the computed SAD value.

3.2 Variable Delay Unit (VDU)

The implementation of the VDUs is a simplified model of the one presented in [4]. The delay unit can delay an 8 bit value (which is a pixel value) for either 1 or 2 clocks. Note that the 2 delay cycle mode effectively implements a 5x5 search box, or steps 1-3 of the 4SS algorithm. The 1 cycle delay mode implements a 3x3 search box, or step 4 of the 4SS algorithm.

3.3 Control Unit

The control unit consists of counters to address the memory, a comparator logic to select which SAD value to use, and a finite state machine that controls the rest of the system.

3.4 Memory

The current pixel memory consists of 256 bytes to store all pixel values. The previous frame search area consists of 900 pixel values, to enable searching the entire search space. For the purposes of this implementation, a 1kB memory is considered for the previous frame search area memory.

4. PROPOSED LOW-POWER DESIGN TECHNIQUES

This section describes low-power techniques incorporated in our motion estimation block. The first two techniques are the elimination of redundant searches and an early termination of the SAD calculation. These are architectural enhancements that reduce the number of necessary computations without compromising the quality of the solution. The last two methods, zero-bias and the reduced range arithmetic, can possibly degrade the quality of the solution while reducing computation.

4.1 Elimination of Redundant Searches

As the 4SS algorithm proceeds to the 2nd and 3rd steps of the algorithm, substantial SAD calculations are unnecessary. Those SADs have already been computed or are found to be useless. If those calculations can be removed in subsequent steps, a power savings can be achieved. Generally speaking, six searches are redundant when a side point of a search box wins, and four searches are redundant when selecting a corner point of the search box [5]. Note that this method is more effective for video with large motion that usually requires the 2nd and 3rd steps of the algorithm to complete the motion vector calculation.

In our low-power design, logic has been added to detect this case and remove the redundant SAD calculations from the subsequent step. A special bit is flagged if the SAD value of a PE is the lowest found so far. During the next search step, the bit is checked. If it is flagged, ENABLE signal to the AND gates in Figure 3 becomes "0," to gates off all inputs to the PE to "0." It also disables its accumulator register, effectively stopping all switching activity for that SAD calculation.

4.2 Early Termination of SAD Calculation

The SAD calculation requires accumulation of differences of all individual pairs of pixels. During the accumulation, if the intermediate value of the SAD becomes larger than the best SAD (i.e., the smallest value) found so far, the SAD computation can be terminated immediately. This scheme is most effective for smaller motion video, in which an optimum SAD is usually found quickly around the center area, and, hence, subsequent SAD calculations can be terminated early.

This method is implemented with a simple comparator built-in to each PE. The "best" SAD found so far is supplied to each PE (represented as signal SAD_IN in Figure 3). If the calculated SAD for that PE becomes greater than the best SAD found so far, the comparator shuts off the inputs to the PE and disables the accumulator register as well, effectively shutting off all the switching activity associated with the SAD calculation.
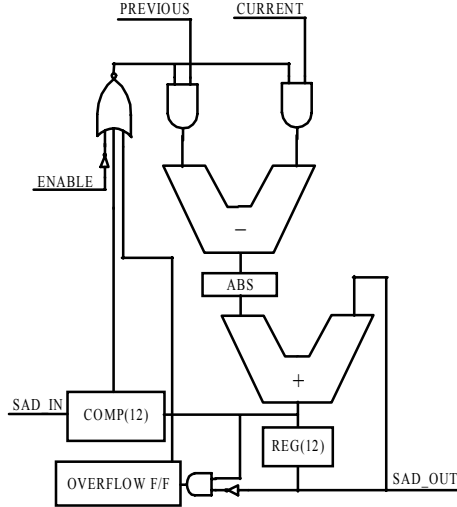
Figure 3. PE with Power-Saving Features

4.3 Zero-Bias Method

The zero-bias method artificially reduces the SAD value of the center point in the search box during the 1st through the 3rd steps. The artificial reduction of the value by a certain bias value may result in a faster termination of the search, while hopefully coming to an acceptable solution.

Since PE1 is responsible for the SAD value of the center point, the SAD value of the center point is subtracted by a bias value before being applied to the comparator logic in Figure 1. One necessary design consideration is that the early termination of SAD calculation method explained in the above should not be applied for the center point. Note that early cutoff of the SAD calculation results in an incorrect SAD value. The bias value was set to 100 based on our experiment.

4.4 Reduced Range Arithmetic

Our experiment on SAD values reveals that the best SAD values rarely exceed 12 bits for the 4SS algorithm and are often of much smaller bit-widths. In addition, H.263 does not use the motion estimation if the SAD value is too large (called INTRA mode). The observation offers a possibility to reduce the bit width of PEs for SAD computation, which is 16 for the baseline model to accommodate all possible SAD values.

To handle the case where the SAD may "overflow." PEs employ an overflow detection scheme, which simply compares the MSB of the previous sum with the newly computed sum. If the previous sum has an MSB of '1' and the current sum has an MSB of '0', then an overflow condition has occurred. A flip-flop is flagged to gate off any further inputs and to disregard the current SAD value during the output evaluation phase. The proposed PE with both comparator and reduced precision arithmetic logic is shown in Figure 3.

5. EXPERIMENTAL RESULTS

This section reports experimental results of three baseline (FSBM, TSS, and 4SS) motion estimation systems and the proposed one in terms of video quality, gate count, and power consumption. To measure the video quality and to collect other statistical data, an H.263 software encoder/decoder system was developed at the

VTVT laboratory of Virginia Tech. Peak signal-to-noise ratio (PSNR) was used as a quantitative measure of video quality. The baseline and proposed systems were coded in VHDL and synthesized with a 0.35 μm CMOS standard cell library operating at 3.3V. The power estimation was performed at the gate level using a commercial tool. Three QCIF (176x144 pixels) video clips, "Suzie," "Carphone," and "Foreman," were used for our experiment. Suzie and Carphone are low-motion video clips, while Foreman contains some high motion scenes.

5.1 Video Quality

Table 1 shows PSNR of four different motion estimation systems. The four systems achieve similar PSNRs for all the three video. The largest difference between the proposed system and the FSBM system is for Foreman, which is about 0.5 dB. It should be noted that degradation of the video quality, if any, for the last three systems, was unnoticeable compared with the FSBM system.

Table 1. PSNR of Various Systems (dB)

	Suzie	Carphone	Foreman
FSBM	32.575	28.780	27.722
TSS	32.476	28.618	27.199
4SS	32.458	28.744	27.346
Prop	**32.451**	**28.866**	**27.213**

5.2 Gate Count

The gate counts of the four different systems are reported in Table 2. The gate count in the table denotes the number of equivalent NAND2 gates *without* considering the memory block. The last row in the table indicates the relative gate count (in percentile) of each system compared with that of the FSBM system.

Table 2. Gate Counts of Various Systems

FSBM	TSS	4SS	Prop.
9583	3301	2828	3331
100 %	50.1 %	30.0 %	34.8 %

As expected the FSBM system has the highest circuit complexity, but it achieves the highest throughput. The 4SS requires less hardware than the TSS (mostly due to the fact that the VDUs shrink in size). The proposed system requires extra logic for power savings features and hence the circuit complexity is slightly higher than that of the 4SS system. We noticed that the PEs takes about 49% of the total gate count for the proposed system, VDUs about 10 %, and the controller about 39%.

5.3 Power Estimation

We considered power dissipation of logic blocks and memory blocks separately. The power for logic blocks was estimated through gate level simulation. The power consumed by the memory was estimated based on SRAM power statistics (such as power dissipation associated with read and write operations) in [6] and from RTL simulations of the systems.

Power dissipation of logic blocks (without memory blocks) of the three baseline systems is shown in Table 3. The table includes the power consumed by major blocks of the circuit for later comparison with the proposed system. Notice that there are miscellaneous logic blocks not reported in the table.

Table 3. Power Estimation for Baseline Models (mW)

Baseline Systems		Suzie	Carphone	Foreman
FSBM	PEs	23.25	22.69	21.15
	Controller	1.45	1.44	1.44
	Total	**25.93**	**25.33**	**23.88**
TSS	PEs	3.445	3.252	3.009
	VDUs	0.987	0.967	0.966
	Controller	0.785	0.786	0.785
	Total	**6.633**	**6.428**	**6.202**
4SS	PEs	2.335	2.231	3.394
	VDUs	0.396	0.387	0.656
	Controller	0.506	0.510	0.801
	Total	**3.858**	**3.758**	**5.853**

As expected the TSS and the 4SS systems dissipate much less power than the FSBM system, and PEs are the major source of power consumption for all the three systems. The table illustrates power dissipation for the 4SS system depends heavily on the video processed, while the power dissipation of the TSS system remains roughly the same. The 4SS system dissipates over 40% less power than the TSS model for low motion video, Suzie and Carphone. However, the difference is small for large motion video Foreman.

Next, we examined the effectiveness of each low-power design technique described earlier and the overall power saving of proposed system which combines all the presented methods. Those results are shown in Table 4.

Table 4. Power Estimation for Low-Power Techniques (mW)

Model		Suzie	Carphone	Foreman
Elimination of Redundant Search	PEs	1.856	1.862	2.321
	VDUs	0.339	0.343	0.544
	Controller	0.550	0.556	0.863
	Total	**3.467**	**3.491**	**4.875**
Early Termination of SAD Calculation	PEs	1.947	1.656	3.114
	VDUs	0.375	0.367	0.624
	Controller	0.504	0.508	0.798
	Total	**3.520**	**3.232**	**5.638**
Zero-Bias	PEs	2.137	2.065	3.165
	VDUs	0.354	0.350	0.604
	Controller	0.483	0.493	0.787
	Total	**3.523**	**3.472**	**5.466**
Reduced Range Arithmetic	PEs	2.091	1.854	2.471
	VDUs	0.372	0.360	0.523
	Controller	0.485	0.485	0.671
	Total	**3.565**	**3.320**	**4.525**
Proposed System (Combinted)	PEs	1.399	1.213	1.690
	VDUs	0.310	0.324	0.429
	Controller	0.479	0.524	0.670
	Total	**2.786**	**2.714**	**3.634**

From the table, elimination of the redundant search is the most effective, especially for the large-motion video. It is notable that the early termination of SAD calculation works best on Carphone. The zero-bias method resulted in an average power savings.

Table 5 compares power consumption of both logic and memory blocks of the three baseline systems and the proposed system.

Table 5. Power Estimation of the Four Systems (mW)

	Suzie		Carphone		Foreman	
	Logic	Memory	Logic	Memory	Logic	Memory
FSBM	25.93	8.99	25.33	8.99	23.88	8.99
TSS	6.63	6.65	6.43	6.65	6.20	6.65
4SS	3.86	5.65	3.76	5.74	5.85	8.41
Prop.	**2.79**	**4.86**	**2.71**	**4.91**	**3.63**	**5.69**

The proposed system offers considerable power savings over the other three systems. Power reduction for logic blocks ranges from 28 % to 40 % over the 4SS system and 41 % to 58% over the TSS system. Memory power saving of the proposed block is sensitive to the activity of video. The power saving on a memory block is 14% over the 4SS block for Suzie and 32% for Foreman. Note that Foreman contains several high motion scenes. Details on power estimation on memory blocks as well as logic blocks are available in [5].

6. CONCLUSION

This paper presents methods to presents low-power design techniques for a motion estimation block targeted for H.263 video codec. The proposed techniques were implemented based on the 4SS algorithm, and power savings were estimated at the gate level.

7. REFERENCES

[1] Vasily G. Moshnyaga, "A New Architecture for Computationally Adaptive Full-Search Block-Matching Motion Estimation." Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI, vol.4, pp.219-22, 1999.

[2] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-Compensated Interframe Coding for Video Conferencing." IEEE 1981 National Telecommunications Conference. Innovative Telecommunications - Key to the Future, vol.4, pp.G5.3/1-5, 1981.

[3] Lai-Man Po and Wing-Chung Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation." IEEE Transactions on Circuits & Systems for Video Technology, vol.6, no.3, pp.313-17, June 1996.

[4] Alessandra Costa, Alessandro De Gloria, Paolo Faraboschi, and Filippo Passaggio, "A VLSI Architecture for Hierarchical Motion Estimation." IEEE Transactions on Consumer Electronics, vol.41, no.2, pp.248-57, May 1995.

[5] R. S. Richmond, "A Low-Power Design of Motion Estimation Blocks for Low Bit-Rate Wireless Video Communications," M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, March 2001.

[6] Meenatchi Jagasivamani, "Development of a Low-Power SRAM Compiler." M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, August 2000.