IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
ISSN: 2278-2834, ISBN: 2278-8735. Volume 4, Issue 3 (Nov. - Dec. 2012), PP 01-14
www.iosrjournals.org

Low Power H.264 Video Compression Achitecture for Mobile Communication

[1]K. Venkataphani Raja, [2]S.Srilakshmi
[1]Department of Electronics and Communication Engineering, Mallareddy Institute of Technology and Sciences,
[2]Dhulapally, Hyderabad, Andhra Pradesh

Abstract: *This paper presents a method to reduce the computation and memory access for variable block size motion estimation (ME) using pixel truncation. Previous work has focused on implementing pixel truncation using a fixed-block size(16×16 pixels) ME. However, pixel truncation fails to give satisfactory results for smaller block partitions. In this paper, we analyze the effect of truncating pixels for smaller block partitions and propose a method to improve the frame prediction. Our method is able to reduce the total computation and memory access compared to conventional full-search method without significantly degrading picture quality. With unique data arrangement, the proposed architectures are able to saveup to 53% energy compared to the conventional full-search architecture. This makes such architectures attractive for H.264application in future mobile devices.*

Keywords- *Low-power design, motion estimation (ME), video coding, VLSI architecture.*

I. Introduction

VIDEO COMPRESSION plays an important role in today's wireless communications. It allows raw video data to be compressed before it is sent through a wireless channel. However, video compression is computation-intensive and dissipates a significant amount of power. This is a major limitation in today's portable devices. Existing multimedia devices can only play video applications for a few hours beforethe battery is depleted. The latest video compression standard MPEG-4 AVC/H.264 [1] gives 50% improvement in compression efficiency compared to previous standard. However, the coding gain comes at the expense of increased computational complexity at the encoder. Motion estimation (ME) has been identified as the main source of power consumption in video encoders. It consumes 50–90% of the total power used in video compression [2]. The introduction of variable block size partitions and multiple reference frames in the standard result in increased of computational load and memory bandwidth during motion prediction.

Block-based ME has been widely adopted by the industry due to its simplicity and ease of implementation. Each frame is partitioned into 16 × 16 pixels, known as macroblocks (MBs). Full-search ME predicts the current MB by finding the candidate that gives the minimum sum of absolute difference(SAD), as follows:

$$SAD(i,j) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} |C(k,l) - R(i+k, j+l)|$$

where *C (k, l)* is the current MB, and *R (i + k, j + l)* is the candidate MB located in the search window within the previously encoded frame. From (1), the power consumption in ME is affected by the number of candidates and the total computation to calculate the matching cost. Thus, the power can be reduced by minimizing these parameters.

Furthermore, to maximize the available battery energy, the computational power should be adapted to the supply power,picture characteristics, and available bandwidth. Because these parameters change over time, the ME computation should be adaptable to different scenarios without degrading the picture quality.
Pixel truncation can be used to reduce the computational load by allowing us to disable the hardware that processes the truncated bits. While previous studies focused on fixed-blocksize ME (16 × 16 pixels), very little work has been done to study the effect of pixel truncation for smaller block sizes. The latest MPEG-4 standard, MPEG-4 AVC/H.264, allows variable block size for motion estimation (VBSME). [1] defines 16×16, 16 × 8, 8 × 16, 8 × 8, 8× 4, 4 × 8, and 4 × 4 block sizes. At smaller block partitions, a better prediction is achieved for objects with complex motion.

Truncating pixels at a 16×16 block size results in acceptable performance as shown in the literature [3]. However, at smaller block sizes, the number of pixels involved during motion prediction is reduced. Due to the truncation error, there is a tendency for smaller blocks to yield matched candidates, which could lead to the wrong motion vector. Thus, truncating pixels using smaller blocks results in poor prediction.

In [4] and [5], we have proposed a low-power algorithm and architecture for ME using pixel truncation for smaller block sizes. The search is performed in two steps: 1) truncation mode and 2) refinement mode. This method reduces the computational cost and memory access without significantly degrading the prediction accuracy.

In this paper, we perform an in-depth analysis of this technique and extend the technique to a complete H.264 system. The rest of this paper is organised as follows. The existing techniques of low-resolution ME are reviewed in Section II. Section III investigates the effect of pixel truncation on VBSME. Section IV outlines the proposed two-step search for VBSME. Section V analyzes the proposed architecture. Our experimental results are discussed in Section VI. Finally, Section VII concludes this paper.

II. Low-Resolution Me

In low-resolution ME, the bit size and computational cost are normally tackled simultaneously. A one-bit transform(1BT) to reduce the computational cost was introduced in[6]. In this method, the original image is filtered using a band-pass filter. The output image is represented by one bit and the ME is carried out at this frame plane. To improve the frame prediction, [7] proposes a two-bit transform (2BT) where the original image is converted into two bits using the threshold value derived from the local image standard deviation. More than 0.2 dB improvement is achieved with this method as compared to 1BT. A low-resolution quantized ME (LRQME), where the pixel is transformed to two bits using an adaptive quantizer, is proposed in [8]. To produce the two-bit image, three quantization thresholds are calculated according to the current MB pixel mean.

In ME, pixel truncation has been used to reduce the computational load for the matching calculation unit [9]. In [3], an adaptive pixel truncation during ME is proposed. The pixel's least significant bits (LSBs) are adaptively truncated depending on the quantization parameter (QP). This direct quantization provides a tradeoff between peak signal to noise ratio (PSNR) and power. Truncating the pixel's most significant bits (MSB) was discussed in [10].

In low-resolution ME, most methods focus on reducing the power by minimizing the computational load. However, memory access is not taken into account. This is because the original pixel needs to be accessed before it is transformed into low-resolution. In some cases, where the PSNR is dropped due to truncation error, a second search is done at full-resolution. This increases the memory access and thus increases the power consumption. On the other hand, while direct pixel truncation has the potential to reduce memory access, it is often at the expense of a decrease in PSNR in some motion types.

III. Effect Of Pixel Truncation For Vbsme

For video applications, data is highly correlated, and the switching activity is distributed non uniformly [10]. Since the LSBs of a data word experience a higher switching activity,significant power reduction can be achieved by truncating these bits. In general, about 50% switching activity reduction is obtained if we truncate up to three LSBs. Further reduction can be achieved if the number of truncated bits (NTBs) is increased. For example, if the NTB is set to 6, the switching activity could be reduced by 80–90%. This makes pixel truncation attractive to minimize power in ME.

Table I shows the cumulative distribution function (CDF)for SAD that is obtained during ME using five *Foreman*sequences. The SAD is grouped into five categories: 0%represents the percentage for $SAD = 0$, 5% represents the percentage of $SAD < 5\% SADmax$, and so on. For 16 × 16block size with $NTB = 4$, the percentage of $SAD = 0$ is close to the untruncated bit ($NTB = 0$). This shows that for 16×16 block size, the truncated pixel is more likely to have the same matched candidate as in the untruncated pixel. However, for 4×4 block with $NTB = 4$, the percentage of $SAD = 0$ is 12% compared to 0% for $NTB = 0$. This shows that there are more matched candidates using truncated pixel for 4×4 block size, which could lead to incorrect motion vectors.

To illustrate the effect of pixel truncation on VBSME, we computed the average PSNR for 50 predicted frames of *Foreman* sequence (QCIF@30frames/s) as shown in Table II. The frames are predicted using full-search algorithm at different block sizes and NTB. From Table II, for full pixel resolution($NTB = 0$), the prediction accuracy improves as the block size decreases. This is reflected by a higher PSNR for predictions using a 4 × 4 block compared to a 16 × 16 block.

For $NTB = 4$, a small PSNR drop is observed for a block size of 16×16 (0.08 dB) compared to untruncated pixels. The PSNR drop for predictions using smaller block sizes is higher with 0.54 dB and 2.16 dB drops for frames with block sizes 8 × 8 and 4 × 4, respectively.

TABLE I
CDF OF CALCULATED SAD DURING MOTION ESTIMATION USING *Foreman* SEQUENCE (SEARCH RANGE, $p = \pm 8$)

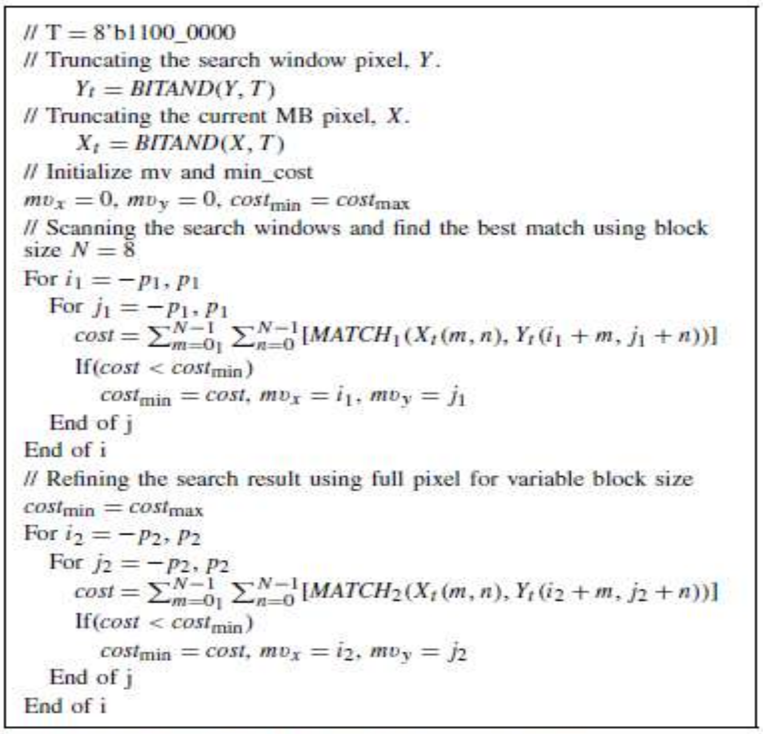
Block size	NTB	0%	<5%	<10%	<20%	<40%
16 × 16	0	0	25	60	94	100
	4	0.2	25	60	94	100
8 × 8	0	0	35	58	87	98
	4	5	35	58	87	98
4 × 4	0	0	58	58	81	99
	4	12	58	58	81	99

TABLE II
AVERAGE FULL-SEARCH PSNR FOR VARIOUS NTB USING SAD AS MATCHING CRITERIA (SEARCH RANGE, $p = \pm 8$)

NTB	Block size					
	16 × 16	Diff.	8 × 8	Diff.	4 × 4	Diff.
0	33.11	—	34.89	—	36.82	—
2	33.12	0.01	34.85	−0.03	36.75	−0.07
4	33.03	−0.08	34.35	−0.54	34.66	−2.16
6	31.79	−1.33	30.29	−4.60	27.46	−9.36

IV. Two-Step Algorithm

In this paper, we propose a method of pixel truncation for VBSME. This method is based on the following observations.



```
// T = 8'b1100_0000
// Truncating the search window pixel, Y.
    $Y_t = BITAND(Y, T)$
// Truncating the current MB pixel, X.
    $X_t = BITAND(X, T)$
// Initialize mv and min_cost
$mv_x = 0, mv_y = 0, cost_{\min} = cost_{\max}$
// Scanning the search windows and find the best match using block size $N = 8$
For $i_1 = -p_1, p_1$
  For $j_1 = -p_1, p_1$
    $cost = \sum_{m=0_1}^{N-1} \sum_{n=0}^{N-1} [MATCH_1(X_t(m,n), Y_t(i_1+m, j_1+n))]$
    If($cost < cost_{\min}$)
      $cost_{\min} = cost, mv_x = i_1, mv_y = j_1$
  End of j
End of i
// Refining the search result using full pixel for variable block size
$cost_{\min} = cost_{\max}$
For $i_2 = -p_2, p_2$
  For $j_2 = -p_2, p_2$
    $cost = \sum_{m=0_1}^{N-1} \sum_{n=0}^{N-1} [MATCH_2(X_t(m,n), Y_t(i_2+m, j_2+n))]$
    If($cost < cost_{\min}$)
      $cost_{\min} = cost, mv_x = i_2, mv_y = j_2$
  End of j
End of i
```

Fig. 1. Pixel truncation algorithm using two-step approach.

1) Truncating pixels for larger block sizes can result in better motion prediction compared to smaller block sizes.
2) At higher pixel resolutions, smaller block sizes can result in better prediction compared to the larger block sizes.

To avoid having large motion vector errors with smaller blocks ,we have implemented motion prediction in two steps. In the first search, the prediction is performed using pixels with *NTB* = 6 at 8 × 8 block size. Then, the result of the first search is refined using full pixel resolution (8-bit) in a smaller search area. The algorithm is summarized in Fig. 1.

Fig. 2 shows the simulation results using truncated pixels with several matching criteria. Two error-based matching criteria and two boolean-based matching criteria are compared against SAD, namely MinMax [11], mean removed MAD(MRMAD) [12], binary XOR (BXOR) [13], and difference pixel count (DPC) [8], respectively. From the figure, at high NTB, error-based matching criteria gives a poor result compared to the boolean-based matching criteria. The combination of *NTB* = 6 and DPC gives a good tradeoff between PSNR and the computational load.

At highly truncated bits, 16×16 block size is more reliable since it has more data compared to the smaller block size. However, for complex motion, the motion vector for a smaller block size, especially a 4×4 block, is not necessarily close to that of a 16 × 16 block. Since the block with smaller size difference tends to move in a similar direction, the 8 × 8 block is used in the first search. This allows us to get better predictions for either the smaller block (8×4, 4×8, and 4×4) or the larger block (16 × 8, 8 × 16, 16 × 16) from the 8 × 8 motion vector.

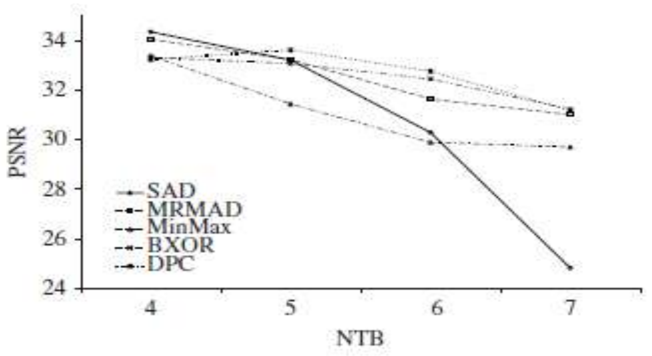


Fig. 2. Average PSNR versus NTB for different block matching criteria using 8 × 8 block size (50 *Foreman* frame sequences, QCIF@30frames/s, search range, $p = [-8, 7]$).

TABLE III
MEMORY BANDWIDTH AND COMPUTATIONAL COST COMPARISON
(p REPRESENTS THE SEARCH RANGE, q AND r ARE THE MEMORY ACCESS AND COMPUTATIONAL LOAD PER CANDIDATE, RESPECTIVELY)

Memory bandwidth

	Conv. full-search	Proposed two-step
First search	$(2p)^2 \times q \times 8$-bit	$(2p)^2 \times q \times 2$-bit
Second search	—	$(2\frac{p}{2})^2 \times q \times 8$-bit
Total	$32p^2q$	$16p^2q$

Computational cost

	Full-search	Proposed two-step
First search	$(2p)^2 \times r \times 1$	$(2p)^2 \times r \times 0.25$
Second search	-	$(2\frac{p}{2})^2 \times r \times 1$
Total	$4p^2r$	$2p^2r$

In the second step, we perform full-pixel resolution to refine the result obtained from the first search. To ensure that the overall computation cost does not exceed the conventional fullsearch computation, the second search is done at a quarter of the size of the first search area. Increasing the refinement area will not only increase the total computation, but also increase the memory access, as shown in Table III.

Fig. 3 illustrates the method used to determine the second search area where blocks A, B, C, and D are the 8 partitions for an MB. After the first search, each partition A, B, C, and D has its own motion vector, *mv A*, *mvB*, *mvC*, and *mvD* respectively. Let *mvx* and *mvy* represent their horizontal \and vertical motion vector components respectively. Thus, *mvxA* and *mvyA* represent the horizontal and vertical components, respectively, of the motion vector for block A; *mvxB* and *mvyB* represent the horizontal and vertical components, respectively, of the motion vector for block B, and so on. The minimum and maximum motion vector of each component is represented by

$$mvx\text{min} = min\{mvxA, mvxB, mvxC, mvxD\}$$

$$mvx\text{max} = max\{mvxA, mvxB, mvxC, mvxD\}$$

$$mvy\text{min} = min\{mvy\ A, mvyB, mvyC, mvyD\}$$

$$mvy\text{max} = max\{mvyA, mvyB, mvyC, mvyD\}\ .$$

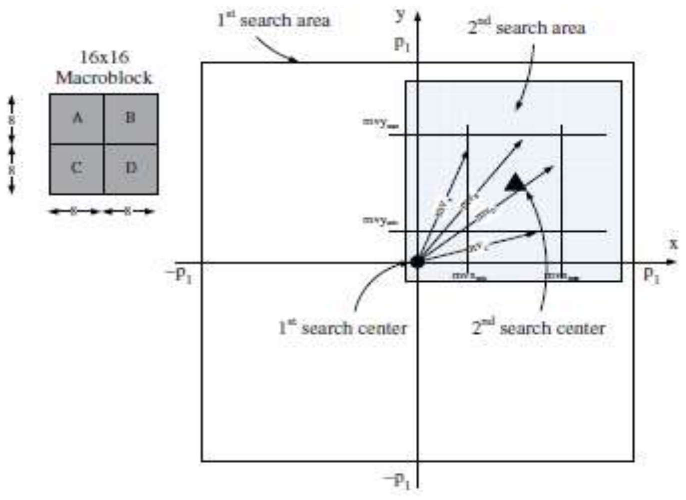


Fig. 3. Defining the second search area for the proposed two-step algorithm.

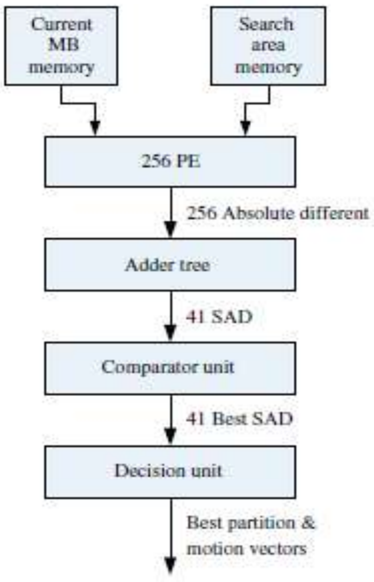


Fig. 4. Me_sad block diagram.

The second search center is defined as

$$\left(\frac{mvx_{\min} + mvx_{\max}}{2}, \frac{mvy_{\min} + mvy_{\max}}{2}\right)$$

with search range, $p_2 = (1/2)p_1$.

V. Hardware Implementation

This section discusses the proposed architectures to implement the two-step algorithm. First, the conventional ME architecture that is used in our analysis is reviewed. Next, we discuss the architectures needed to support the two-step method as proposed in Section IV. The area and power overhead for the computation and memory unit are also investigated. Based on these analyzes, we propose three low-power ME architectures with different area and power efficiencies.

In this paper, we implement the ME architecture based on 2-D ME as discussed in [2]. We choose 2-D ME because it can cope with the high computational needs of the real-time requirement of H.264 using a lower clock frequency than 1-D architecture.

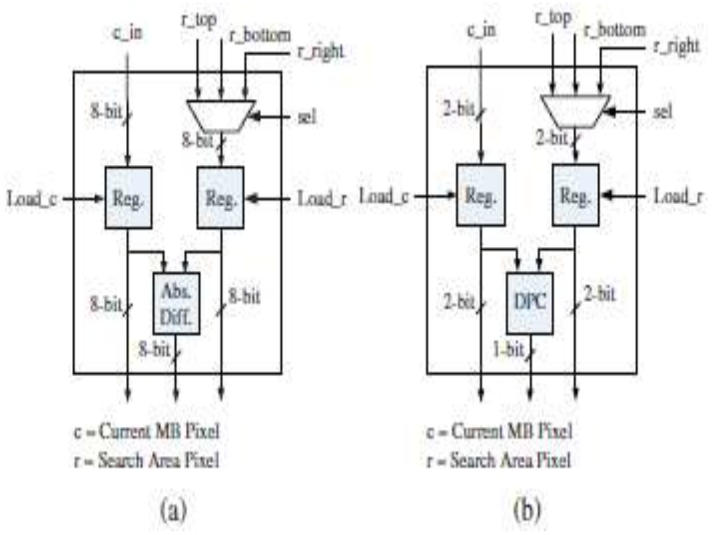


Fig. 5. Processing element (PE) to support (a) SAD and (b) DPC.

A. Computation Unit

Fig. 4 shows the functional units in the conventional 2-D ME (me_sad) [2]. The ME consists of search area (SA) memory, a processing array which contains 256 processing elements (PEs), an adder tree, a comparator, and a decision unit. The search area memory consists of 16 memory banks where each bank stores 8-bit pixels in a $H \times W/N$ total word, where H and W are the search area window's height and width respectively, and N is the MBs width. During motion prediction, 16 pixels are read from the 16 memory banks simultaneously. The data in the memory are stored in a ladder like manner to avoid delay during the scanning [14].

At each initial search, the current and the first candidate MB are loaded into the processing array's registers. Then, it calculates the matching cost for one candidate per clock cycle. The 256 absolute differences from the PEs are summed by the adder tree, and outputs the SAD for 41 block partitions. The adder tree reuses the SAD for 4×4 blocks to calculate a larger block partition. In total, the adder tree calculates 41 partitions per clock cycle.

Throughout the scanning process, the comparator updates the minimum SAD and the respective candidate location for each 41 block partition. Once the scanning is complete, the decision unit outputs the best MB partition and its motion vectors. The ME requires 256 clock cycles to scan all candidates. For me_sad, the input and output for each of the PE are 8-bits wide as shown in Fig. 5(a). The input for the adder tree is 8-bits wide, and the SAD output is 12 to 16-bits wide, depending on the partition size. These data are then input into the comparator, together with the current search e as in me_sad, DPC-based ME (me_dpc) requires two bits for the current and reference pixel inputs as shown in Fig. 5(b). Furthermore, the matching cost is calculated using boolean logic (XOR and OR) rather than arithmetic operation as in SAD-based PE. These make the overall area for the 256 PEs in me_dpc much smaller than me_sad. The reduction in output bit width in DPC-based PE also reduces the bit width required for adder tree and comparator unit. The input and output for the adder tree is 1-bit and 5 to 9 bit widths, respectively. A similar bit width is applied to the comparator's input. Table IV compares the area mm2 , the total equivalent gates (based on 2-input NAND gate), and power consumption

TABLE IV
ME_SAD AND ME_DPC AREA (mm^2), TOTAL EQUIVALENT GATES (BASED ON 2-INPUT NAND GATE) AND POWER (mW)

Modules	me_sad			me_dpc		
	Area	Gates	Power	Area	Gates	Power
256 PE	0.90	173 611	28.67	0.32	61 728	2.31
Adder_tree	0.13	25 077	5.53	0.04	7716	0.99
Comparator Unit	0.11	21 219	1.25	0.09	17 361	0.86
Decision Unit	0.10	19 290	0.54	0.07	13 503	0.50
Total	1.24	239 198	36.00	0.52	100 309	4.66

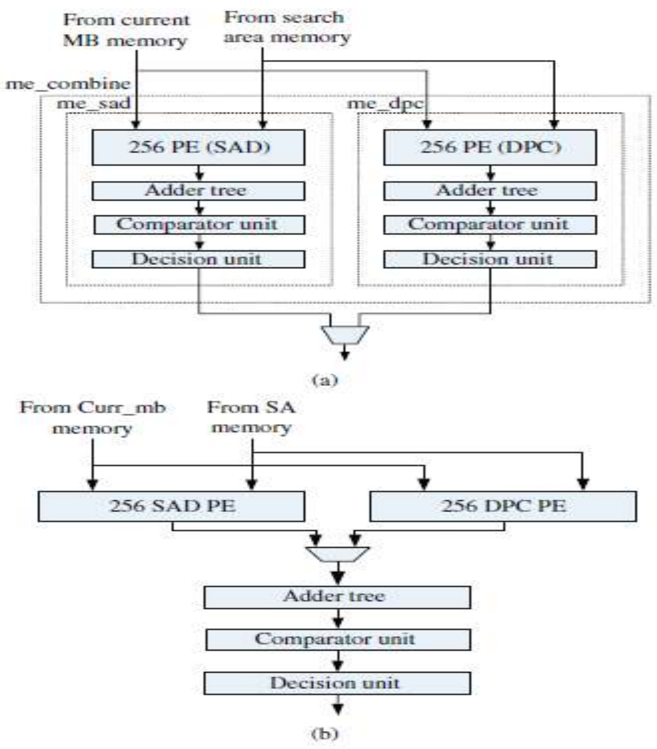


Fig. 6. Computational unit: (a) me_split and (b) me_combined.

(mW*)* for me_sad and me_dpc computational units. The comparisons are based on synthesis results using 0.13μm CMOS UMC technology. The table shows that me_sad's area is dominated by the 256 PE (73%). Thus, with the significantly smaller area for 256 PE, the me_dpc will require less area than the me_sad. The overall me_dpc requires 42% of the me_sad area. Based on the above analysis, we propose two types of architectures for the ME computation unit that can perform both low-resolution and full-resolution searches. These are me_split and me_combine as shown in Fig. 6. Me_split implements both me_sad and me_dpc as two separate modules, as shown in Fig. 6(a). During low-resolution

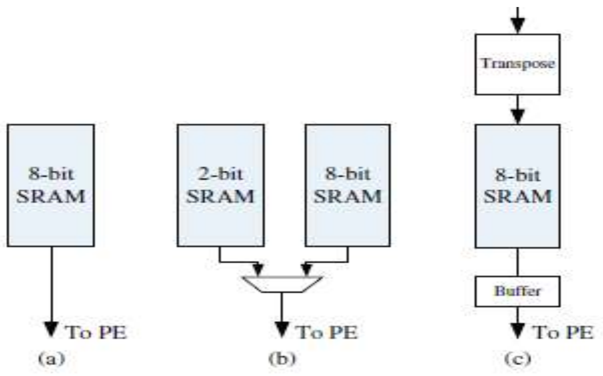


Fig. 7. SA memory arrangement (a) mem8 (b) mem28, and (c) mem8pre.

search, me_sad is switched off while the me_dpc is used to perform the search. The second step uses the me_sad, while the me_dpc is switched off. This architecture allows only the necessary bit size to be used during different search modes. While potential power savings is possible, this architecture requires additional area for the adder tree, comparator and decision unit to support the low-resolution search.Due to the functions of the adder tree, the comparator and the decision units are similar for both me_sad and me_dpc, and me_combine shares these units during low-resolution search and full pixel resolution [Fig. 6(b)]. This architecture results in a much smaller area compared to me_split. However, higher power consumption is expected during the low-resolution search because the adder tree, comparator, and decision unit operate at higher bit size than needed.

B. Memory Architecture

Conventional ME architecture implements the SA memory using single-port static random access memory (SRAM) with one pixel (8-bits) per word. To implement the two-step search, we need to access the first two MSBs for each pixel during the first search and 8-bits in the second stage. Thus, the pixels need to be stored to allow two reading modes. For this, three types of memory architecture are proposed. These are (a) 8-bit memory (mem8), (b) 2-bit and 8-bit memory (mem28), and (c) 8-bit memory with prearranged data and transposed register (mem8pre) as shown in Fig. 7.

Mem8 stores the data in the same way as in the conventional ME. We access 8-bit data during both low-resolution and the refinement stage. However, during the low-resolution search, the lower six bits are not used by the PE. Because the memory is accessed during both low-resolution and the refinement stage, it results in higher memory bandwidth than the conventional ME architecture.

To overcome the problem in mem8, mem28 uses two types of memory: 2-bit and 8-bit. The 2-bit memory stores the first two MSBs of each datum, and the 8-bit memory stores the complete full pixel bitwidth. During the low-resolution search, the data from the 2-bit memory are accessed. This allows only the required bits to be accessed without wasting any power during low-resolution. In the refinement stage, the 8-bit memory is read into the PEs. Although this architecture can potentially reduce memory bandwidth and power consumption,it needs an additional area for the 2-bit memory. In mem8pre, the data is prearranged before storing themmin 8-bit memory. Four pixels are grouped together, and then transposed according to their bit position, as shown in Fig. 8. During the lresolution search, we read only the memory locations that store the first two MSBs of the original pixels.Thus, the total memory accessed during the low-resolution isone-fourth of the conventional full pixel access.

In full-resolution search, we read four memory locations that contain the first up to eighth bits in four clock cycles. Delay buffers, as shown in Fig. 7(c), realigns these words to match the original 8-bit pixel. By prearranging the pixels this way, we can use the same memory size as in the conventionalfull-search while retaining the ability to access the first two MSBs, as well as the full bit resolution. The drawback of this approach is that it needs additional circuitry to transpose and realign the pixels during the motion prediction. The estimated bandwidth for the above three memory architectures are shown in Table V.

TABLE V
MEMORY BANDWIDTH FOR DIFFERENT ARCHITECTURES

	Low-resolution	High-resolution
mem8	$NWH \times$ 8-bit	$N\frac{WH}{4} \times$ 8-bit
mem28	$NWH \times$ 2-bit	$N\frac{WH}{4} \times$ 8-bit
mem8pre	$NWH \times$ 2-bit	$N\frac{WH}{4} \times$ 8-bit

TABLE VI
PSNR DROP AGAINST CONVENTIONAL FULL-SEARCH SAD
QCIF@30FRAMES/S, $p_1 = [-8, 7]$

	16 × 16			
	Akiyo	*Mobile*	*Foreman*	*Stefan*
fs_p4	0	0	0.12	0.22
2step16	0	0.01	0.08	0.03
2step8	0	0	0.07	0.03

	8 × 8			
	Akiyo	*Mobile*	*Foreman*	*Stefan*
fs_p4	0	0.02	0.3	0.41
2step16	0	0.03	0.23	0.13
2step8	0	0.02	0.19	0.11

	4 × 4			
	Akiyo	*Mobile*	*Foreman*	*Stefan*
fs_p4	0.06	0.16	0.54	0.58
2step16	0.06	0.17	0.47	0.31
2step8	0.05	0.14	0.44	0.27

C. Overall Architecture

From the above discussion, we propose three different architectures that can perform both low-resolution and ful l resolutionsearches. By combining different computation and memory units, we propose the following architectures:

TABLE VII
PSNR Drop Against Conventional Full-Search Using SAD
CIF@30frames/s, $p_1 = [-16, 15]$

	16 × 16		
	Mobile	*Foreman*	*Stefan*
fs_p8	0.04	0.14	0.29
2step16	0.04	0.25	0.04
2step8	0.11	0.12	0.04

	8 × 8		
	Mobile	*Foreman*	*Stefan*
fs_p8	0.12	0.24	0.44
2step16	0.12	0.39	0.21
2step8	0.20	0.21	0.09

	4 × 4		
	Mobile	*Foreman*	*Stefan*
fs_p8	0.38	0.44	0.58
2step16	0.37	0.59	0.44
2step8	0.40	0.39	0.32

1) me_split+mem28 (ms_m8);
2) me_combine+mem8 (mc_m8);
3) me_combine+mem8pre (mc_m8p).

In these architectures, both low-resolution and full resolution search can be performed. With proper configuration, the conventional full-search algorithm can be used during normal conditions to ensure a high-quality picture at the output. In condition where energy consumption is the main concern, the two-step method is used. This allows us to reduce the energy consumption without significantly degrading the output picture quality.

VI. Simulation And Implementation Results

A. Performance of the Proposed Two-Step Algorithm

Tables VI and VII show the PSNR difference using the proposed method against the conventional full-search ME (FS). The comparison is done for the frames predicted using 16 × 16, 8 × 8, and 4 × 4 partitions. Other block sizes are not included for simplicity. The difference is calculated on the basis of the average PSNR of 85 frames. Different frame sequences that represent various types of motion from low to high are used in this experiment: *Akiyo*, *Mobile*, *Foreman*, and *Stefan*. Both QCIF and CIF frame resolutions are considered, which represent the typical frame size for mobile devices. The search range, $p1 = [-8, 7]$ and $p1 = [-16, 15]$ is defined for QCIF and CIF, respectively.

2step8 represents the proposed two-step search using the 8×8 block partition. For comparison, we include the result for the two-step search where the first search is done using 16×16 partitions (2step16). The result of the first search is used as the center for the second search. fs_p4 and fs_p8 represent the conventional full-search ME with a search range equivalent to $(1/2)p1$ for QCIF and CIF, respectively

From the table, our method is able to achieve a good prediction with a smaller PSNR drop compared to the other method. For a low-motion sequence such as *Akiyo*, the PSNR drop for QCIF is below 0.05 dB. The PSNR drop increases slightly for a high-motion sequence such as *Stefan*. This is due to the prediction error and search range limitation during the first and second searches, respectively

The smaller PSNR drop for 2step8 compared to 2step16 shows that the first search using 8 × 8 partition gives a good approximation compared to 16 × 16 block size. In the 8 ×8 partitions, we have more information for the MB motion,which is important when determining the second search range for the high-motion sequence. In addition, the PSNR drop varies depending on the level of detail of the frame. For a frame sequence with high detail, such as *Mobile*, the first search with $NTB = 6$ contains more information for the MB feature. Thus, it can obtain a better match, which is reflected by the lower PSNR drop. For a frame with less object detail, the PSNR drop is slightly higher. The same explanation is applicable for the higher PSNR drop for CIF compared to QCIF frame resolution. This is because the MB content for the CIF frame is more sparse compared to the QCIF MB.

TABLE VIII
AVERAGE PSNR DROP (dB) FOR SEVERAL MOTION ESTIMATION TECHNIQUES

	FS	2BT	LRQME	Proposed
Low-resolution	No	Yes	Yes	Yes
High-resolution	Yes	No	Yes	Yes
Block size: 16 × 16	0.0	0.7	0.8	0.1
Block size: 8 × 8	0.0	1.3	—	0.2
Block size: 4 × 4	0.0	3.0	—	0.4

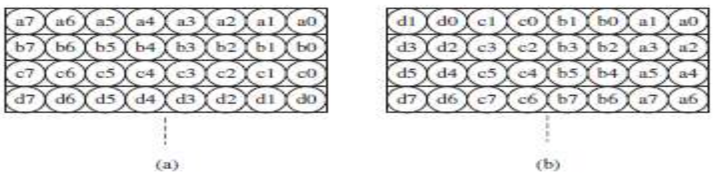


Fig. 8. Storing 8-bit pixel in 8-bit memory: (a) conventional arrangement and (b) mem8pre.

The reduction in the search range in the refinement stage does affect the prediction for the 4 × 4 block partition. This is expected, since in 2step8, the full pixel resolution is done at one-fourth of the conventional full-search area. Thus, the reduction in computation comes at the expense of decreasing the prediction accuracy. However, compared to the direct reduction of the search range, our method gives a higher PSNR as opposed to the fs_p4 and fs_p8 for the QCIF and CIF frames, respectively.
In all frame sequences tested, our method gives good PSNR compared to the conventional full-search with PSNR drop < 0.5dB. Furthermore, our method could yield a more uniform motion vector for the 4 × 4 partitions which is required to reduce the overall bitrates.
Table VIII shows the average PSNR drop for several existing ME techniques discussed in Section II. In 2BT, frames

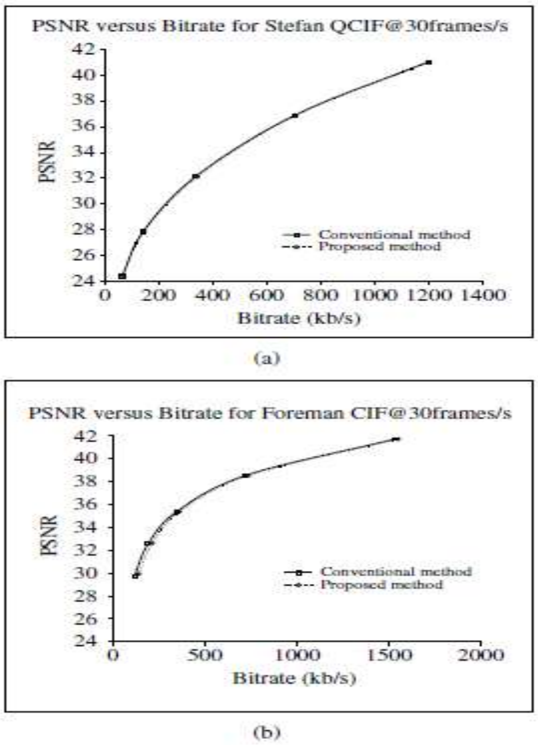


Fig. 9. PSNR versus bitrate using the proposed two-step method for (a) QCIF@ 30 frames/s, $p = [-8, 7]$ and (b) CIF@30 frames/s, $p = [-16, 15]$.

are converted from 8-bits to 2-bits using the threshold value derived from the local image standard deviation. The ME is performed based on the transformed 2-bit image. LRQME transforms the 8-bit pixel to two bits using an adaptive quantizer. The result of the ME obtained using the low-resolution image is refined at higher pixel resolution using 16×16 block size. As shown in Table VIII, our method shows superior performance compared to other techniques for 16×16 to 4×4 block sizes.

To evaluate the effectiveness of our method within H.264 software toward generating the final bitrates, we modified the existing H.264 reference software (version JM8.6 for baseline profile) to include the proposed algorithm. We replaced the xisting full-search ME algorithm with the two-step method.
The simulations were done with rate control off, QP = {20, 25, 30, 35, 40}, 85 frames, QCIF and CIF frame format at 30 frames/s. Both the conventional and the proposed method were compared. Fig. 9 shows the PSNR versus bitrates graphs simulated using the modified H.264 reference software. Two video sequences (*Foreman* and *Stefan*), which represent medium and high motion sequence, are shown. From the graphs, it can be seen that the proposed method could achieve good performance, close to the conventional method, without significantly degrading the picture quality. For typical application of QCIF@30 frames/s at 146 kb/s and CIF@30 frames/s at 736 kb/s, only 0.02 dB difference is observed compared to the conventional method.

TABLE IX
COMPUTATIONAL UNIT: AREA (mm^2), TOTAL EQUIVALENT GATES (BASED ON 2-INPUT NAND GATE) AND POWER (mW) COMPARISON

Modules	me_split				me_combine			
	Area	Gates	Power		Area	Gates	Power	
			Low-res.	High-res.			Low-res.	High-res.
256 PE	1.17	225694	3.62	24.44	1.06	204475	3.222	24.39
Adder_tree	0.18	34722	1.54	6.63	0.13	25077	1.871	6.48
Comparator unit	0.20	38580	0.80	1.26	0.11	21219	1.034	1.25
Decision unit	0.16	30864	0.50	0.54	0.10	19290	0.543	0.54
Others	0.04	7716	0.01	1.29	0.07	13503	0.17	1.36
Total	1.75	337577	6.46	34.16	1.47	283565	6.84	34.02

B.**Performance of the Proposed Architectures**

This section presents the synthesized results of our analysis. First, we present the results for the computation unit. Next, the area and power consumption for the proposed memory architectures are analyzed. Finally, we present the results for the overall ME architectures that can provide efficient area and power consumption.

We have synthesized our design using the UMC 0.13μm CMOS library. We have used Verilog-XL for functional simulation and Power Compiler to perform power analysis at20 MHz. Actual video data is used to verify the hardware and to obtain the estimated power consumption. Table IX compares the synthesis results for the proposedcomputation units me split and me_combine. The power consumption during low-resolution and full-resolution searches are shown in detail. During the motion prediction, the search range [−8, 7] and [−4, 3] is used during low-resolution and full pixel resolution search, respectively. From the table, the me_split consumes 6% less power during the low-resolution search than the me_combine. However, this comes at the cost of an additional 41% of area on top of the existing me_sad. On the other hand, because me_combine shares some modules, it requires only 19% additional area compared to me_sad. This shows that, combining the adder tree, comparator and decision unit as in me_combine is preferred over me_split.

Table X shows the area and power comparison for the proposed memory unit. The reported power includes the power consumed by the additional circuit needed by the respective memory architectures. The SRAM model is generated using a UMC 0.13 memory compiler with the estimated area and power provided by the datasheet. From Table X, it is clear that the mem8pre provided the lowest bandwidth and power compared to the other memory configurations during the low-resolution search. This is expected since only one-fourth of the memory is accessed during the low-resolution search. However, it requires extra area for the additional circuits needed to arrange the pixels.

On the other hand, mem8 gives the minimum area compare to the other configurations. Because the full pixel bitwidth is accessed in both low and full-resolution, it requires higher memory bandwidth and uses more power than the others during the low-resolution search.

TABLE X
MEMORY UNIT: AREA (mm^2) AND POWER (mW) COMPARISON

Modules	Area	Power	
		Low-res.	High-res.
mem8	0.23	4.7	4.7
mem28	0.39	3.3	4.7
mem8pre	0.42	2.2	7.5

Table XI shows the total area and power consumption results based on extracted layout for the proposed overall ME architectures ms_m28, mc_m8, and mc_m8p. In order to make a fair comparison between these architectures, we calculate the total energy needed during each motion prediction. For the two-step method, this includes the energy consumed duringboth low-and full-resolution search. The energy is calculated as the power consumption multiplied by the total time taken to complete the motion prediction. The normalized energy and area is shown in Table XII. From the table, the architecture mc_m8p consumes the least energy compared to the others, and saves 53% compared to conventional ME (me_sad). However, because it requires additional area for the DPC-based PEs and buffer to arrange the pixels, 28% area overhead is required to implement this method.

Compared to other architectures, mc_m8 gives the best tradeoff in terms of both area and energy efficiency. By using the two-step method, this architecture can save 48% of the energy compared to conventional ME with 16% additional area required to implement a low-resolution search.

Table XIII shows the comparison of various motion estimation architectures. Since our architecture requires less than 500 clock cycles to process one MB, the architecture can achieve real-time operation for processing QCIF@30 frames/s at 1.4 MHz. At this clock frequency, our method shows that it consumes lower power per MB compared to other architectures.

C. Energy Saving on H.264 System

In order to evaluate the low-power techniques, an H.264 system was built, as shown in Fig. 10, which will be referred to as the conventional system in this paper. The architecture of each module in the system has been carefully selected basedon existing literature to achieve real time implementation [2], [16]–[19].

TABLE XI
ME ARCHITECTURE AREA (mm^2), TOTAL EQUIVALENT GATES (BASED ON 2-INPUT NAND GATE), AND POWER (mW) COMPARISON

Modules	Area	Gates	Power					
			Low-res.			Full-res.		
			Logic	Mem	Total	Logic	Mem	Total
me_sad	1.77	341435	NA	NA	NA	45.00	4.7	49.70
ms_m28	2.56	493827	8.32	3.3	11.62	44.02	4.7	48.72
mc_m8	2.05	395448	8.79	4.7	13.49	43.72	4.7	48.42
mc_m8p	2.26	435957	9.80	1.2	10.98	45.74	4.7	50.44

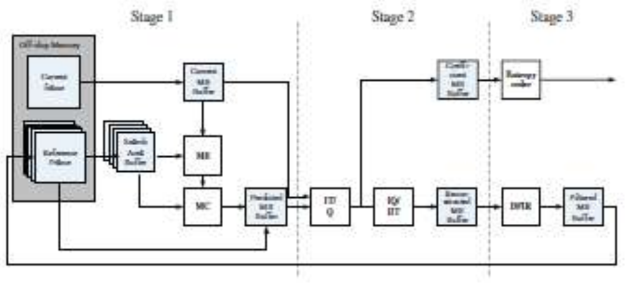


Fig. 10. MPEG system three stage pipeline.

TABLE XII
NORMALIZED ME ARCHITECTURE AREA AND POWER COMPARISON

	Area	Energy	Area*Energy
me_sad	1.00	1.00	1.00
ms_m28	1.45	0.48	0.70
mc_m8	1.16	0.52	0.60
mc_m8p	1.28	0.47	0.61

This system consists of a ME, a motion compensator (MC), a transform coder, a deblocking filter (DFIR), and an entropy coder (EC). The transform coder is comprised of an integer transform (IT), an inverse integer transform (IIT), a quantizer (Q), and an inverse quantizer (IQ). Modules dealing with intra prediction and fractional pel ME were not included due to time constraints. Furthermore, since the proposed low-powertechniques mainly affect the integer ME and transform coding loop, intra prediction and fractional pel motion estimation are not directly affected by these techniques. The architecture aimed to process QCIF resolution (YUV420) at 30 frames/s with a search range of $p = [-8, 7]$.

The basic processing unit in a video encoder is an MB. To encode the MB in serial order, starting from predicting the current MB to bitstream generation would result in slow throughput and low hardware utilization. To overcome this problem, MB pipeline processing is typically adopted in MPEG hardware architecture [20].

In this paper, the system is divided into three pipelinestages. The first stage performs motion prediction. This stage includes loading the search area from external memory into on-chip memory and performing ME and motion compensation. The second stage performs transform coding including integer transform, quantizer, inverse quantizer, and inverse integer transform. Since the entropy coder and deblocking filter are operated based on the output from the transform coder, these operations are executed in the third pipeline stage. This arrangement allows the hardware of each stage to be ready for processing the next MB once the output is stored into the pipeline buffers.

In this design, a maximum of four reference frames are used, which allows the throughput for the pipeline to be set to 2000 clock cycle per MB. To achieve real-time operation, the clock has to operate at 6 MHz for QCIF@30 frames/s. In total, the conventional architecture requires 4.85 mm2 of chip area for the chip core.

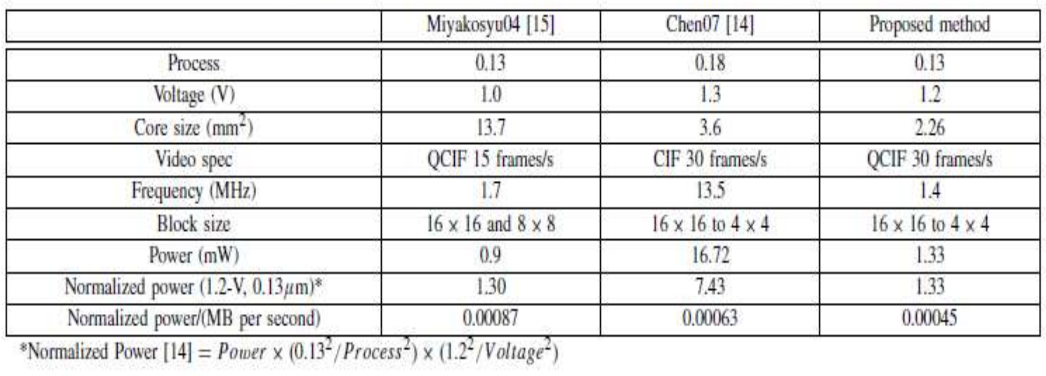
The second column of Table XIV shows the energy consumed by the conventional H.264 architecture. Since different modules require different clock cycles to process one MB, the energy consumption is more accurate than the power values in representing the actual amount of work required to process MBs. In total, using one reference frame during motion prediction, the system consumes 695.43 nJ to process one MB within 2000 clock cycles (6 MHz). The ME dominatesenergy consumption, taking 77% of the total power. This is followed by the transform coder, deblocking filter, and entropy coder at 11, 8, and 4%, respectively.

As discussed in Section VI-B, since the mc_m8p architecture gives the largest energy saving compared to other architectures, this architecture is used to implement the twostep algorithm. By applying the proposed architecture, additional area is introduced to allow low-resolution searches to be performed. As discussed in previous section, this increases the ME area by 0.49 mm2. At the system level, the additional area increases to 10% of the total area. The introduction of the proposed two-step hardware results in reduced energy consumption in the ME as shown in the third column of Table XIV. The total energy consumed by the ME using the proposed two-step method is 265.74 nJ. The proposed architecture has saved 50% of ME power compared to the conventional ME architecture. The implementation of the low-resolution, which requires a smaller computational load, has contributed to this saving.

Due to a slight decrease in the motion prediction accuracy during the two-step search, the residue generated from the twostep technique is slightly higher than that of the conventional ME. This results in a small increase in energy consumption (<2%) in the transform coder. Since most of the coefficient is quantized to zero, the small increase in residue is masked by the quantizer. Thus, the increase in the residue does not propagate to other modules, so there is no change in energy consumption in the deblocking filter and entropy coder. In total, for one reference frame, the two-step method reduces the total energy consumption for the H.264 system by 40%, as shown in Table XIV.

TABLE XIII

POWER COMPARISON FOR DIFFERENT MOTION ESTIMATION ARCHITECTURES



	Miyakosyu04 [15]	Chen07 [14]	Proposed method
Process	0.13	0.18	0.13
Voltage (V)	1.0	1.3	1.2
Core size (mm^2)	13.7	3.6	2.26
Video spec	QCIF 15 frames/s	CIF 30 frames/s	QCIF 30 frames/s
Frequency (MHz)	1.7	13.5	1.4
Block size	16 x 16 and 8 x 8	16 x 16 to 4 x 4	16 x 16 to 4 x 4
Power (mW)	0.9	16.72	1.33
Normalized power (1.2-V, 0.13μm)*	1.30	7.43	1.33
Normalized power/(MB per second)	0.00087	0.00063	0.00045

*Normalized Power [14] = $Power \times (0.13^2/Process^2) \times (1.2^2/Voltage^2)$

TABLE XIV
ENERGY CONSUMPTION (nJ) COMPARISON BETWEEN THE CONVENTIONAL H.264 ARCHITECTURE (CONV.) AND THE PROPOSED H.264 LOW-POWER ARCHITECTURE (LP) AT 6 MHZ FOR ONE REFERENCE FRAME

Modules	Energy	
	Conv.	LP
ME	536.27	265.74
MC	4.62	4.62
IT/IIT, Q/IQ	75.34	76.41
DFIR	54.02	54.02
EC	25.18	25.18
Total	695.43	425.97

VII. Conclusion

This paper has presented a method to reduce the computational cost and memory access for VBSME using pixel truncation. Previous work has shown that pixel truncation provides an acceptable performance for motion prediction using a 16 × 16 block size. However, for motion prediction using smaller block sizes, pixel truncation reduces the motion prediction accuracy. In this paper, we have proposed a two-step search to improve the frame prediction using pixel truncation. Our method reduces the total computation and memory access compared to the conventional method without significantly degrading the picture quality. The results show that the proposed architectures are able to save up to 53% energy compared to the conventional full-search ME architecture, which is equivalent to 40% energy saving over the conventional H.264 system. This makes such architecture attractive for H.264 application in future mobile devices.

References

[1] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Recommendation H.264 & ISO/IEC 14496-10 (MPEG-4) AVC, 2005.

[2] 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 estimationfor H.264/AVC," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 53, no. 3, pp. 578–593, Mar. 2006.

[3] Z.-L. He, C.-Y. Tsui, K.-K. Chan, and M. L. Liou, "Low-power VLSI design for motion estimation using adaptive pixel truncation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, pp. 669–678, Aug. 2000.

[4] A. Bahari, T. Arslan, and A. T. Erdogan, "Low computation and memory access for variable block size motion estimation using pixel truncation," in *Proc. IEEE Workshop Signal Process. Syst. 2007 (SiPS '07)*, pp. 681–685.

[5] A. Bahari, T. Arslan, and A. Erdogan, "Low-power hardware architecture for vbsme using pixel truncation," in *Proc. 21st Int. Conf. VLSI Design*, Hyderabad, Andhra Pradesh, 2008, pp. 389–394.

[6] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 702–706, Aug. 1997.

[7] A. Erturk and S. Erturk, "Two-bit transform for binary block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7,pp. 938–946, Jul. 2005.

[8] S. Lee, J.-M. Kim, and S.-I. Chae, "New motion estimation algorithm using adaptively quantized low bit-resolutionimage and its VLSI architecture for MPEG2 video encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 6, pp. 734–744, Oct. 1998.

[9] Y. Chan and S. Kung, "Multi-level pixel difference classification methods," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3. Washington D.C., 1995, pp. 252–255.

[10] V. G. Moshnyaga, "Msb truncation scheme for low-power video processors," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 4. Orlando, FL,1999, pp. 291–294.