# RD Optimized Bandwidth Efficient Motion Estimation and Its Hardware Design with On-Demand Data Access

Gwo-Long Li, *Student Member, IEEE,* and Tian-Sheuan Chang, *Senior Member, IEEE*

***Abstract*—Data bandwidth dominates the performance and power consumption in the video encoder design. In which a low bandwidth and bandwidth aware motion estimation design enables smooth and better video quality as well as lower power consumption on data accesses. This paper proposes a bandwidth efficient motion estimation and its hardware implementation to deal with the bandwidth issues. First, an on-demand data access mechanism is proposed to acquire the reference data according to the video content for motion estimation process and thus can avoid unnecessary reference data loading. Furthermore, the available bandwidth constraint is properly modeled into our proposed rate distortion optimization framework to efficiently use the data bandwidth. Simulation results show that our proposed algorithm not only allocates proper data bandwidth for motion estimation according to video content but also saves 79.15% data bandwidth demand with 0.03 dB PSNR drop and 2.50% bitrate increase in maximum for 4 CIF resolution sequences, when compared to the fully data reuse full search motion estimation which reuses the overlapped reference data to avoid unnecessary data reloading. In addition, under the available data bandwidth constraint, our proposed algorithm can achieve 2.43%, 0.08%, and 0.20% BD-bitrate saving with 0.17 dB, 0.01 dB, and 0.01 dB BD-PSNR increase on average for high, median, and low motion sequences when compared to the full search motion estimation algorithm. The resulted design only needs 75.27 K gate counts when running at 23 MHz operating frequency for 4 CIF at 30 frames/s with 90 nm CMOS process due to its search range independent buffer design.**

***Index Terms*—Bandwidth aware motion estimation, bandwidth efficient motion estimation, motion estimation.**

## I. Introduction

MOTION ESTIMATION (ME) not only contributes to most of the coding efficiency but also is the most computational intensive as well as data bandwidth intensive component in modern video encoding design [1]. ME finds out the best matching block by matching its current coding macroblock (MB) (a block with $16 \times 16$ pixels) with search candidates within the search range. A direct approach called full search, which searches all candidates, has been widely used in hardware implementations [2] due to its simplicity and regularity. With its regularity, it is easy to fully reuse the overlapped search range data between different searches [3] to reduce the required access and several hardware architectures were proposed in [4]–[6] to take the advantages of search range data overlapping. However, the bandwidth requirement is still high due to its content independent data loading. Other approaches like fast ME algorithms [7]–[13] can reduce the computational complexity but they do not help a lot to reduce the required bandwidth for irregular search pattern.

To deal with data bandwidth problem, works in [14]–[19] proposed various efficient ME architectures to reduce data bandwidth requirements. In [14], the authors proposed a modified MB processing order to further increase the data reuse of Level C data reuse scheme [3]. For multiple reference frames motion estimation in H.264, [15] proposed a single reference frame multiple current macroblocks (MBs) scheme to reuse the reference data. An alternative direction to reduce the data bandwidth requirement was introduced in [16] and [17] by decreasing the size of search range centered at the motion vector predictor (MVP). In [16], the authors efficiently selected the search area to reduce the data bandwidth requirement by tracing the motion vectors. [17] used a two-step windowing approach to dynamically decide whether to load more reference data for motion estimation. In hardware design, [18], [19] proposed the hardware architectures to reduce the memory bandwidth overhead by adopting the binary motion estimation mechanism which executed the matching process on the generated bit map instead of on the pixels. However, these works still require large and constant bandwidth support from a system because of the adopted full search algorithm. Large bandwidth requirement increases the cost as well as power consumption. Besides, assumption of constant bandwidth support is not practical for a modern complex system-on-a-chip due to high varieties of processing tasks. In addition, they do not consider how to efficiently use the available bandwidth or adapt the search process according to available bandwidth, which would be vital to portable video applications like video phones due to costly DRAM power consumption. As a result, the above bandwidth policy implies two consequences: either over design to meet the demands or design unchanged with insufficient bandwidth supply that results in quality degradation or coding time increase.

Manuscript received January 11, 2010; revised April 30, 2010 and July 19, 2010; accepted August 29, 2010. Date of publication October 14, 2010; date of current version November 5, 2010. This work was supported by the National Science Council of Taiwan, under Grant NSC98-2220-E-009-001. This paper was recommended by Associate Editor G. G. Lee.

The authors are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan (e-mail: glli@dragons.ee.nctu.edu.tw; tschang@twins.ee.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2010.2087450

1051-8215/$26.00 © 2010 IEEE

To solve the above problems without the side effects, this paper presents an on-demand data access efficient ME and its hardware realization under the rate distortion (RD) optimized framework. Based on the introduced bandwidth-rate-distortion model, the proposed approach can minimize and predict the data bandwidth requirement, and access required data on demand while maximizing the rate-distortion performance within available bandwidth constraint. The on-demand data access can reduce the unnecessary data access and thus minimize the search range buffer. The simulation results show that the proposed algorithm can effectively allocate and reduce the required data bandwidth with negligible quality loss. Such efficiency also brings the low cost benefits for the resulted hardware implementation for its smaller search range buffer than other designs.

The organization of this paper is as follows. Section II first briefly reviews the previous work and modeling the memory bandwidth of ME. Section III presents the proposed bandwidth aware ME algorithm and the simulation results are shown in Section IV. The hardware design and its implementation results are exhibited in Section V to demonstrate the efficiency of our proposal. Finally, a conclusion is given in Section VI.

## II. Memory Bandwidth Modeling for ME

### A. Search Algorithms and Their Memory Bandwidth Modeling

In ME search algorithms, the full search ME loads all pixels inside the search area from external memory to find out the best matching MB by exhaustively checking every candidate position. The data bandwidth per frame for full search ME ($DB_{FS}$) can be calculated as follows:

$$DB_{FS} = [(SR_V \times 2 + 16) \times (SR_H \times 2 + 16)] \times \#MBs_{frame} \quad (1)$$

where $SR_V$ and $SR_H$ indicate the search range in the vertical and horizontal directions, respectively, and $\#MBs_{frame}$ refers to the number of MBs per frame.

Due to the high computational complexity and data access of full search ME, several fast algorithms [7]–[10] tried to reduce the number of candidate positions instead of exhaustive checking to decrease the computational complexity of full search ME and thus lessen the bandwidth requirements with less data to be loaded. However, these fast algorithms suffer from the irregular data access and complex data access control and consequently result in the difficulty of hardware realization and ill data reuse.

In addition to check point reduction approach, search range data can be reused to decrease the bandwidth requirement since the adjacent MBs will share a large portion of overlapped search range as shown in Fig. 1. A systematic analysis for data reuse in full search algorithms has been proposed in [14]. In which the level C data reuse scheme as shown in Fig. 1 is widely used for ME design due to its high data reuse property. Hence, the data bandwidth per MB for full search with Level C reuse ($DB_{FS\_LevelC}$) can be computed as follows:

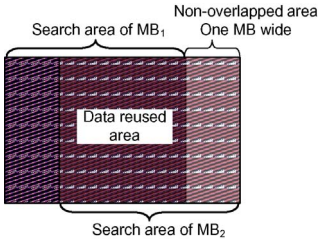


Fig. 1. Illustration of data reuse scheme for ME.

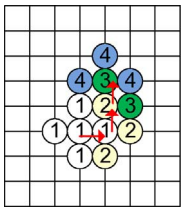


Fig. 2. Example of nearest neighbors search algorithm.

$$DB_{FS\_LevelC} \cong \begin{cases} (SR_V \times 2{+}16) \times (SR_H \times 2{+}16) \ldots \forall MB \in \textit{left most MB column} \\ (SR_V \times 2 + 16) \times 16 \ldots\ldots\ldots\ldots \textit{OtherMBs.} \end{cases} \quad (2)$$

For an example with QCIF image format and $\pm 8$ search range, the data bandwidth requirements are 33 kB and 99 kB per frame for full search with and without Level C data reuse, respectively. Although full search with data reuse scheme can greatly reduce the bandwidth requirements for ME, it still needs high bandwidth requirements in case of large search window, which could occur in large size video. Besides, the search range for full search in hardware implementation is usually larger than other algorithms to keep the quality since hardware regularity forces its search center at (0, 0) instead of motion vector predictor [20]. For an example of 480p frame size, the bandwidth will be 3849.19 kB per frame if the search range of $\pm 64$ is used.

### B. Nearest Neighbors Search Algorithm and Its Bandwidth Modeling

To gain the benefits both from data reuse scheme and search range size reduction with MVP, this paper adopts nearest neighbors (NN) ME [21] for its highly data reuse in consecutive search and small search range due to MVP.

Fig. 2 shows the concept of a nearest neighbors algorithm. It first calculates the MVP for search process and the sum of absolute differences (SAD) of five positions centered at MVP (labeled by 1). If the minimum SAD is located at center position, the search operation is finished and the coordinate of center position is set as motion vector of current coding MB. Otherwise, the position with minimum SAD is set as the search center and another three positions labeled by 2 are checked. This operation is repeated until the position with minimum SAD is located at center or the search boundary is reached.

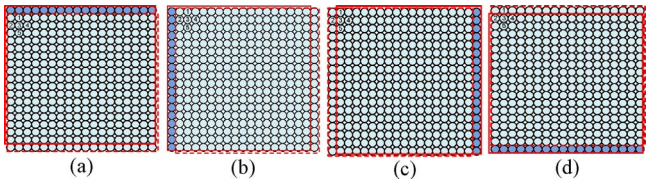


Fig. 3. Data overlapping for different nearest neighbors search results. (a) Min. SAD at 1. (b) Min. SAD at 2. (c) Min. SAD at 4. (d) Min. SAD at 5.

With above operations, Fig. 3 shows the data overlapping cases for different nearest neighbors search results. In this figure, the lighter pixels indicate the overlapped area and the dark pixel row or column is referred to the additional pixels to be loaded. From this figure, we can observe that there is a large portion of data overlapping between any two adjacent search positions, which leads to highly regular data reuse for hardware implementation. Therefore, by adopting nearest neighbors search pattern, only one extra column or row of pixels have to be loaded for search process.

The bandwidth requirements of nearest neighbors pattern are analyzed as follows. In the first step, $18 \times 18$ reference pixels for first five search points should be loaded from external memory to evaluate the SADs. If the minimum SAD is located at the center position, no more pixels are needed from external memory. Otherwise, 18 additional pixels are needed to be loaded from external memory for evaluation, if the position with minimum SAD is located at any one of four corner positions. Therefore, the data bandwidth per MB of nearest neighbors search pattern ($DB_{NN}$) can be formulated as follows:

$$DB_{NN} = (18 \times 18) + n \times 18 \tag{3}$$

where $18 \times 18$ indicates the required pixels for computing the SADs for first five positions and the $n$ is the remaining steps to search the best result. With this, we can model the data requirements of nearest neighbors search pattern with step $n$. With step $n$, the data requirement for ME can be adjusted freely for different quality.

## III. Proposed Framework

### A. RD Optimized Bandwidth Modeling for Video Contents and Search Steps

The SAD is commonly used as a similarity measurement in ME process due to its computational simplicity, but failed to consider the coding rate brought by motion vector encoding. Therefore, the commonly used rate-distortion cost (*RDCost*) is adopted in this paper and can be calculated as follows:

$$RDCost(MV, \lambda_{MOTION}) = SAD(s, c(MV)) + \lambda_{MOTION} R(MV - MVP) \tag{4}$$

where $\lambda_{Motion}$ indicates the Lagrange multiplier and the term $R(MV - MVP)$ represents the number of bits for coding the motion vector difference between the motion vector ($MV$) and the predicted motion vector. Through the adoption of *RDCost*, the best rate-distortion performance can be achieved.

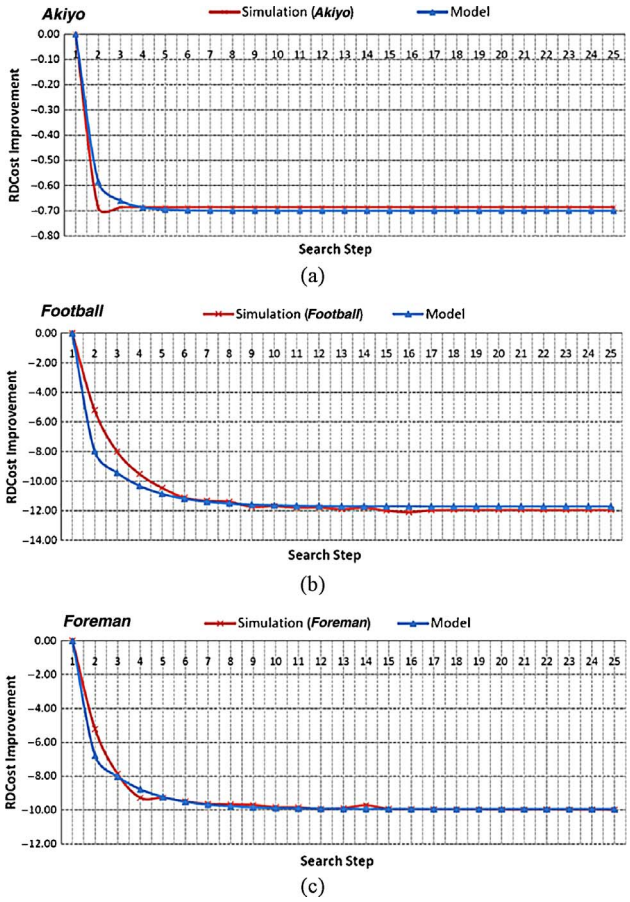


Fig. 4. Relationship between the steps $n$ of nearest neighbors search pattern and *RDCost* and modeled results for sequences of (a) *Akiyo*, (b) *Football*, and (c) *Foreman*.

Fig. 4 shows the relationship between the rate distortion performance improvement and search steps of the nearest neighbors search. In which the vertical axis is the percentage of *RDCost* improvement and the horizontal axis is the steps of $n$ in nearest neighbors pattern ME. This simulation uses JM11 reference software [22], quantization parameter (QP) with 28, $\pm 16$ search range and only $16 \times 16$ block size for simplicity. The mechanism of variable block size is not included for simplicity due to the SADs of other small block size can be obtained from the data of $16 \times 16$ block size in SAD tree based hardware realization, and it will not affect the data bandwidth. Thus, the *RDCost* improvement can be derived as follows:

$$\Delta RDCost_n = \left( \frac{RDCost_n - RDCost_0}{RDCost_0} \right) \times 100 \tag{5}$$

where $RDCost_0$ and $RDCost_n$ indicate the *RDCosts* after the first and $n+1$ steps search of nearest neighbors search, respectively. From these figures we can observe that the $\Delta RDCost$ is increased significantly in the first few steps. However, the $\Delta RDCost$ is increased slightly after more steps. For example, for the *Akiyo* sequence in Fig. 4(a), the $\Delta RDCost$ would be stable after three steps. For the *Football* sequence in Fig. 4(b), sixteen steps are required for the $\Delta RDCost$ stabilization. Therefore, it is unnecessary to search too many steps since the improvement would be negligible after checking certain

number of steps. Meanwhile, this also helps to save the data bandwidth requirement if the required steps can be properly predicted.

From Fig. 4, we can obtain three properties. First, the convergence speed of the sequence is content dependent. For example, the high motion sequence such as *Football* has slower convergence speed than the slow motion sequence such as *Akiyo*. This property mainly comes from that the high motion sequence needs more search steps to find out the best result. The second property is that the magnitude of $\Delta RDCost$ is also content dependent. For instance, the $\Delta RDCost$ of *Akiyo* sequence is smaller than that of the *Football* sequence since most MVs in low motion sequence have their best MV highly around the MVP. The last property is that the magnitude of *RDCost* significant influences the $\Delta RDCost$ convergence speed. That is, the sequences with smaller *RDCost* like *Akiyo* would have faster $\Delta RDCost$ convergence speed than the sequences with larger *RDCost* like *Football*.

By combining the three above properties, we can use the *RDCost* and $\Delta RDCost$ of the first few steps to predict the data bandwidth requirement of ME process while maintain best rate distortion performance. Therefore, the convergence speed influenced by $\Delta RDCost$ can be modeled by the following equation:

$$\Delta RDCost_{improved} = \alpha \times e^{(\Delta RDCost_2 - \Delta RDCost_1)\times\beta} \tag{6}$$

where $\alpha$ is *RDCost* controlling factor which is related to initial *RDCost* and defined by (7) and the $\beta$ stands for the bandwidth adjustment factor to achieve trade-off between rate distortion performance and data bandwidth requirements. That is, the larger the $\beta$ is, the less the data bandwidth is necessary. Oppositely, the larger $\beta$ also results in more rate distortion performance degradation. The $\beta$ is set to 0.1 in this paper empirically to achieve the best tradeoff.

Furthermore, since different *RDCost* magnitude might result in different convergence speed as mentioned in the last property, the $RDCost_0$ is used to derive the factor of $\alpha$ by the following equation:

$$\alpha = RDCost_0 \times \gamma. \tag{7}$$

The $\gamma$ is set to 0.001 empirically. Therefore, through (6), the relationship between *RDCost* improvement and search steps can be described and its fitting results are shown as the curve labeled with "*Model*" in Fig. 4. Finally, the steps needed for executing nearest neighbors search while keeping acceptable rate distortion performance degradation can be decided by the following equation by considering the maximum allowed search range size:

$$n = \Delta RDCost_{improved} \times SR_{Max} \tag{8}$$

where $SR_{Max}$ indicates the maximum size of search range. After the step number $n$ has been estimated by (8), the allocated data bandwidth for current MB ($DB_{Allocated\_MB\_i}$) can be obtained as follows:

$$DB_{Allocated\_MB\_i} = n \times 18. \tag{9}$$

In summary, we can model the relationship of rate distortion performance and bandwidth by the search steps, initial *RDCost* and subsequent *RDCost* improvement according to (6). These two *RDCost* terms faithfully reflect the characteristics of video content. By this modeling, we can further determine the optimal steps to maximize the rate distortion performance under bandwidth constraints.

However, it is worth mentioning that any other fast algorithm can also be used in the optimization framework of this paper if it satisfies the property of highly data reuse in its consecutive search, such as full search, logarithmic search [23], and one-at-a-time search [24]. To fit different kinds of search algorithms, (3) and (6)–(9) should be really remodeled. For example, the one-at-a-time search algorithm checks three candidates in the first step so that $16 \times 18$ pixels are loaded from external memory. For each search candidate, 16 pixels in average are demanded to be loaded for cost evaluation. Therefore, (3) can be rewritten as $(16 \times 18) + n \times 16$ and (6)–(9) should be remodeled. The modeling of (6)–(9) can be done by a curve fitting tool once the relationship between rate distortion cost and search step is derived.

### B. Proposed Bandwidth Aware ME Algorithm

Fig. 5 shows the flow chart of proposed bandwidth aware ME algorithm. It operates as follows. First the available bandwidth for current frame is initialized. Afterward, the *step0, step1,* and *step2* of nearest neighbors search are executed for obtaining $\Delta RDCost_1$ and $\Delta RDCost_2$. If the minimum *RDCost* is located at center position, the search process is finished. Otherwise, the $\Delta RDCost_1$ and $\Delta RDCost_2$ from previous steps are used to calculate the corresponding data bandwidth requirement of current MB. Afterwards, more steps are applied to search the best result according to the allocated data bandwidth. After finding the best result, the available data bandwidth pool is updated for further usage. The details of proposed algorithm are described as follows.

1) *Bandwidth Initialization:* In the first step, the total data bandwidth ($DB_{Total}$) for a certain encoding period is calculated by considering the available system bandwidth as follows:

$$DB_{Total} = \frac{BW_{Bus}}{FR} \times GOP \tag{10}$$

$$DB_{Used} = 0 \tag{11}$$

where $BW_{Bus}$ is the data transmission rate (bytes/s) of bus, *FR* indicates the frame rate, and the $DB_{Used}$ indicates the used data bandwidth. In this paper the certain encoding period is defined as the group of pictures (GOP) and *GOP* refers to the number of frames per coding group. This total data bandwidth, $DB_{Total}$, stands for the available bandwidth per GOP. However, the available bandwidth for ME should subtract the bandwidth requirements for basic quality coding. Thus, the available data bandwidth ($DB_{Available}$) should be as follows:

$$DB_{Available} = DB_{Total} - DB_{Basic} \tag{12}$$

where $DB_{Basic}$ is the necessary data bandwidth requirements for *step0, 1,* and *2* in nearest neighbors search algorithm and

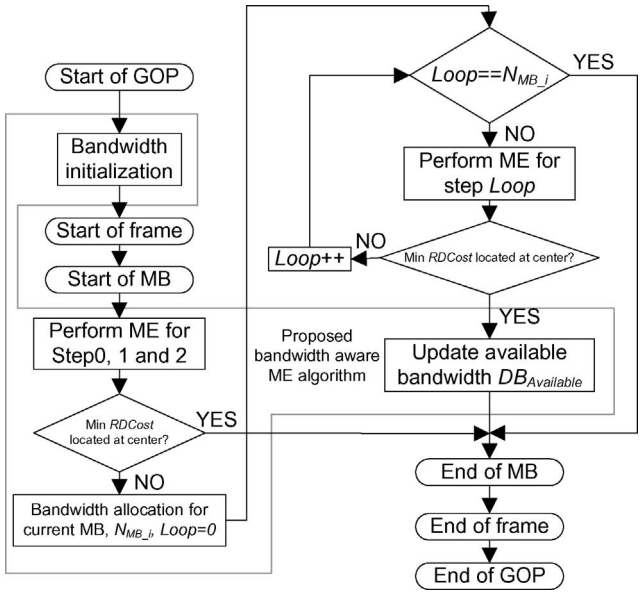


Fig. 5. Flowchart of proposed bandwidth aware ME algorithm.

can be calculated by (3) so that $DB_{Basic} = (18 \times 18) + 2 \times 18 = 360$ bytes.

2) *Bandwidth Allocation for Current MB:* After the initialization step, the data bandwidth usage, $DB_{MB\_i}$, for current MB $i$ is allocated as follows:

$$DB_{MB\_i} = \begin{cases} DB_{Allocated\_MB\_i} \cdots\cdots \text{if } DB_{Allocated\_MB\_i} \\ \qquad\qquad \leq \frac{DB_{Available} - DB_{Used}}{MB_{Remained}} \\ \frac{DB_{Available} - Used}{MB_{Remained}} \cdots\cdots \text{otherwise} \end{cases} \quad (13)$$

where $DB_{Allocated\_MB\_i}$ stands for the allocated data bandwidth for current $i$th MB by (6)–(9) and the $MB_{Remained}$ is the number of un-encoded MBs. In this allocation, the allocated bandwidth by (9) will be adopted only if the allocated bandwidth is smaller than average remaining data bandwidth per MB. Otherwise, the data bandwidth usage is restricted to average remaining data bandwidth per MB. By this restriction, the problem of over allocation can be avoided. After bandwidth allocation for current MB, the allocated bandwidth should be converted to the corresponding steps in nearest neighbors ME algorithm. The allocated steps can be calculated as follows:

$$N_{MB\_i} = \frac{DB_{MB\_i}}{18}. \quad (14)$$

For the following step, the nearest neighbors search method is applied according to the allocated steps.

3) *Data Bandwidth Update:* The allocated data bandwidth might not be fully reused due to early termination condition of the search algorithm. Thus, the unused bandwidth can be recycled for further use. Therefore, an additional data bandwidth update stage should be added into the whole system. The data bandwidth update operations are as follows:

$$DB_{Used} = \sum_{j=0}^{i-1} DB_{Used\_MB\_j} \quad (15)$$

$$DB_{Remained\_MB\_i} = DB_{MB\_i} - DB_{Used\_MB\_i} \quad (16)$$

$$DB_{Available} = DB_{Available} + DB_{Remained\_MB\_i} \quad (17)$$

TABLE I
ENVIRONMENT SETTINGS FOR SIMULATION

| | |
|---|---|
| Platform | JM11 |
| Frame structure | IPPPP…. |
| Frames to be encoded | 300 for CIF and 4CIF, 100 for 1080p |
| Frame rate | 30fps |
| Frame resolution | CIF, 4CIF, and 1080p |
| Quantization Parameter (QP) | 8, 18, 28, 32, 38 |
| Group of Picture (GOP) | 16 |
| Entropy coding | CABAC |
| Rate Distortion Optimization (RDO) | Simple |

TABLE II
BRIEF CONTENT DESCRIPTION FOR TEST VIDEO SEQUENCES

| | Size | Sequence | Property |
|---|---|---|---|
| Slow motion | 1080p | Sunflower | Complex texture in background and slow moving in object |
| | | Rush hour | Quiescent motion in background and slow motion in moving objects |
| | CIF & 4CIF | Mother Daughter (MD), Hall, City, Akiyo, News | |
| Median motion | 1080p | Tractor | Median motion in background and high motion in moving object |
| | | Station2 | Zoom out motion in all scene |
| | | Riverbed | Monotonic texture in all scene |
| | CIF | Table tennis | Median motion in contents with scene changes |
| | CIF & 4CIF | Mobile | Complex texture and different motion direction |
| | | Foreman | Zoom in and zoom out motion |
| | | Coastguard | Median motion in background and moving objects |
| High motion | 1080p | Blue sky | High motion in all scene |
| | | Pedestrian, | High motion in moving objects |
| | CIF & 4CIF | Soccer, Stefan Football, | |

where $DB_{Remained\_MB\_i}$ refers to the remained data bandwidth for current MB $i$ and $DB_{Used\_MB\_i}$ is used data bandwidth after executing nearest neighbors search.

## IV. SIMULATION RESULTS

Tables I and II show the simulation environment settings and test sequences. This simulation uses two scenarios to demonstrate the efficiency of our proposed algorithm. One scenario is to show the data bandwidth savings without data bandwidth constraint. In this scenario, we use the search range to represent $BW_{Bus}$ for simplicity and compare with the full search with Level C data reuse scheme since it can achieve the highest data reuse [13], [25], [26]. Another scenario is to demonstrate the rate distortion performance under the data bandwidth constraint. This scenario compares three algorithms including full search (FS), NN [21], and block-based gradient descent search algorithm (BBGS) [27] under the data bandwidth constraints of 20 kB and 38 kB for CIF sequences and 385 kB and 765 kB for 1080p sequences.

Fig. 6 demonstrates the efficiency of bandwidth allocation for our method. For low motion *Akiyo* sequence in Fig. 6(a), the variation of allocated bandwidth is small, only 0.5 kB, for each frame. However, the variation of allocated bandwidth becomes larger for medium and high motion sequences. For example, in *Table Tennis* sequence, more bandwidth has been

TABLE III
COMPARISON OF AVERAGE PSNR WITHOUT BANDWIDTH CONSTRAINT

| Sequences | CIF | | | | | | Sequences | 4CIF | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SR: ±16 | | | SR: ±32 | | | | SR: ±64 | | |
| | FS | Pro. | Δ (dB) | FS | Pro. | Δ (dB) | | FS | Pro. | Δ (dB) |
| Akiyo | 41.44 | 41.42 | −0.02 | 41.44 | 41.42 | −0.02 | Akiyo | 44.02 | 43.99 | −0.03 |
| Coastguard | 37.54 | 37.54 | −0.00 | 37.54 | 37.54 | −0.00 | City | 38.20 | 38.19 | −0.01 |
| Football | 38.75 | 38.78 | +0.03 | 38.74 | 38.76 | +0.03 | Coastguard | 40.04 | 40.03 | −0.01 |
| Foreman | 38.76 | 38.74 | −0.02 | 38.76 | 38.74 | −0.02 | Football | 41.33 | 41.38 | +0.05 |
| Mobile | 38.31 | 38.19 | −0.12 | 38.19 | 38.19 | −0.00 | Foreman | 41.43 | 41.41 | −0.02 |
| MD | 42.01 | 41.99 | −0.02 | 42.00 | 41.99 | −0.01 | Hall | 42.39 | 42.38 | −0.02 |
| News | 41.21 | 41.20 | −0.01 | 41.21 | 41.20 | −0.01 | Mobile | 38.90 | 38.89 | −0.01 |
| Table tennis | 38.33 | 38.32 | −0.01 | 38.34 | 38.32 | −0.01 | News | 42.51 | 42.54 | +0.03 |

allocated to the scene change frame (frame index 131) to satisfy the demands of motion search.

Tables III–V show the comparisons of peak signal-to-noise ratio (PSNR), bitrate and consumed data bandwidth with search range ±16 and ±32 for CIF resolution, ±64 for 4 CIF resolution and ±128 for 1080p resolution. The MVP is adopted as the prediction center for simulations of all algorithms. The data bandwidth here only considers the data amount accessed from external memory. The effect of DRAM latency will be discussed in the next section. The result shows that the proposed algorithm consumes 18.00% less data bandwidth for the search range ±16 case. The saving becomes larger, 61.58% and 78.82%, for larger search range (search range ±32 for CIF) and larger frame size (search range ±64 for 4 CIF), respectively. The bandwidth saving is also content dependent: more saving in low motion sequences and less saving in high motion sequences due to different allocated search steps. Above results show that the proper data bandwidth requirement can be dynamically allocated for different video content through our proposed algorithm. With less consumed data bandwidth, the PSNR degradation of our proposed algorithm is only 0.12 dB for all search range size in maximum, with only 0.37% bitrate increase on average. As a result, the proposed algorithm can efficiently predict the suitable data bandwidth requirements for ME and thus result in less rate distortion performance degradation.

Tables VI–VIII show the BD-PSNR and BD-bitrate comparisons for low, median, and high motion sequences under different data bandwidth constraints. All these results are relative to the results of the FS algorithm. In our simulation, the total data bandwidth is evenly distributed to each MB in FS, NN, and BBGS motion estimation algorithms and is dynamically allocated to each MB in our proposed algorithm. For low motion sequences, the rate distortion performance of our proposed algorithm is near the same as FS since the best MV can be easily found near MVP from a limited data bandwidth support. For median motion sequences, the BD-PSNRs of our proposed algorithm become better than FS algorithm and the BD-bitrates of some sequences are decreased. In Table VII, we can observe that the BD-bitrate saving of *Station2* sequence is much higher than other sequences since this sequence has zoom out motion over entire frame. As a result, the data bandwidth requirement of MBs is different from the others due to the MBs in the outer having higher motion than the MBs in the center. Our proposed algorithm can detect such variation and allocate suitable amounts of data bandwidth to each MB. For high motion sequences in Table VIII, the BD-PSNR and BD-bitrate of our proposed algorithm are much better than FS algorithm due to better detection of data bandwidth requirement of each MB. The BD-bitrate saving of our proposed algorithm can achieve up to 7.143% for *Stefan* sequence in maximum and 2.44% in average for all high motion sequences. In summary, our proposed algorithm has higher performance in high motion sequences which are also the hardest sequences to deal with.

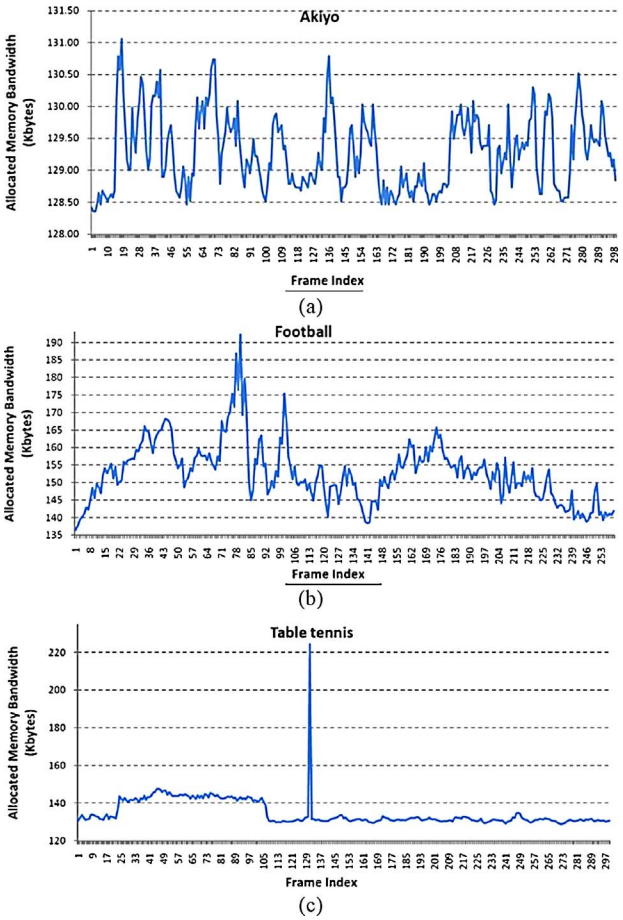


Fig. 6. Allocated bandwidth for difference video sequences. (a) *Akiyo*. (b) *Football*. (c) *Table Tennis*.

TABLE IV
COMPARISON OF AVERAGE BITRATE WITHOUT BANDWIDTH CONSTRAINT

| Sequences | CIF | | | | | | Sequences | 4CIF | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SR: ±16 | | | SR: ±32 | | | | SR: ±64 | | |
| | FS | Pro. | Δ (%) | FS | Pro. | Δ (%) | | FS | Pro. | Δ (%) |
| Akiyo | 550.67 | 550.53 | -0.03 | 550.67 | 550.53 | -0.03 | Akiyo | 1380.74 | 1379.57 | -0.08 |
| Coastguard | 2988.49 | 2985.74 | -0.09 | 2987.58 | 2985.75 | -0.06 | City | 10840.42 | 10741.38 | -0.91 |
| Football | 3349.39 | 3410.22 | +1.82 | 3330.66 | 3389.97 | +1.78 | Coastguard | 6915.81 | 6919.85 | +0.06 |
| Foreman | 2311.85 | 2324.39 | +0.54 | 2311.89 | 2323.88 | +0.52 | Football | 7106.68 | 7284.16 | +2.50 |
| Mobile | 5091.61 | 5072.55 | -0.37 | 5099.20 | 5073.05 | -0.51 | Foreman | 5380.74 | 5459.90 | +1.47 |
| MD | 1355.34 | 1356.12 | +0.06 | 1355.10 | 1355.71 | +0.04 | Hall | 5460.56 | 5453.84 | -0.12 |
| News | 1125.50 | 1126.60 | +0.10 | 1125.55 | 1126.65 | +0.10 | Mobile | 9431.67 | 9424.50 | -0.08 |
| Table tennis | 2006.60 | 2025.80 | +0.96 | 2007.39 | 2026.84 | +0.97 | News | 2319.75 | 2326.26 | +0.28 |
| **Average** | **2347.43** | **2356.49** | **+0.37** | **2346.01** | **2354.05** | **+0.35** | **Average** | **6104.55** | **6123.68** | **+0.39** |

TABLE V
COMPARISON OF DATA BANDWIDTH SAVING

| Sequences | CIF | | Sequences | 4CIF |
|---|---|---|---|---|
| | SR: ±16 | SR: ±32 | | SR: ±64 |
| Akiyo | 22.19% | 63.55% | Akiyo | 80.24% |
| Coastguard | 17.69% | 61.44% | City | 79.05% |
| Football | 10.18% | 57.94% | Coastguard | 78.91% |
| Foreman | 15.92% | 60.55% | Football | 75.48% |
| Mobile | 16.26% | 60.77% | Foreman | 77.99% |
| MD | 20.61% | 62.81% | Hall | 79.79% |
| News | 21.39% | 63.16% | Mobile | 79.17% |
| Table tennis | 19.78% | 62.41% | News | 79.96% |
| **Average** | **18.00%** | **61.58%** | **Average** | **78.82%** |

TABLE VI
COMPARISON OF BD-PSNR AND BD-BITRATE FOR LOW MOTION SEQUENCES

| Image size | $DB_{Total}$ | Sequence | BD-PSNR (dB) | | | BD-Bitrate (%) | | |
|---|---|---|---|---|---|---|---|---|
| | | | NN | BBGS | Proposed | NN | BBGS | Proposed |
| 1080p | 385KB | Sunflower | –0.032 | –0.025 | +0.097 | +0.550 | +0.386 | –0.832 |
| | | Rush hour | –0.049 | –0.033 | +0.034 | +0.916 | +0.615 | –1.467 |
| | 765KB | Sunflower | –0.001 | –0.001 | +0.016 | +0.020 | +0.026 | –0.294 |
| | | Rush hour | –0.006 | –0.006 | +0.021 | +0.046 | +0.046 | –0.669 |
| CIF | 20KB | Akiyo | –0.013 | –0.044 | –0.002 | +0.276 | +1.046 | +0.021 |
| | | News | –0.031 | –0.051 | –0.014 | +0.825 | +1.421 | +0.743 |
| | | MD | –0.030 | –0.051 | –0.027 | +0.626 | +1.030 | +0.287 |
| | 38KB | Akiyo | –0.011 | –0.035 | –0.001 | +0.298 | +0.877 | +0.022 |
| | | News | –0.023 | –0.056 | +0.003 | +0.452 | +1.129 | –0.057 |
| | | MD | –0.040 | –0.063 | –0.010 | +1.109 | +1.739 | +0.259 |

TABLE VII
COMPARISON OF BD-PSNR AND BD-BITRATE FOR MEDIAN MOTION SEQUENCES

| Image size | $DB_{Total}$ | Sequence | BD-PSNR (dB) | | | BD-Bitrate (%) | | |
|---|---|---|---|---|---|---|---|---|
| | | | NN | BBGS | Proposed | NN | BBGS | Proposed |
| 1080p | 385KB | Tractor | –0.029 | –0.016 | +0.007 | +0.439 | +0.244 | –0.144 |
| | | Station2 | –0.243 | –0.145 | +0.069 | +4.586 | +2.902 | –1.467 |
| | | Riverbed | –0.002 | –0.003 | –0.002 | +0.009 | +0.020 | +0.021 |
| | 765KB | Tractor | –0.001 | –0.001 | +0.004 | +0.006 | +0.006 | –0.068 |
| | | Station2 | –0.031 | –0.031 | +0.129 | +0.454 | +0.454 | –2.048 |
| | | Riverbed | –0.000 | –0.000 | +0.001 | +0.005 | +0.005 | –0.017 |
| CIF | 20KB | Table tennis | –0.128 | –0.164 | –0.100 | +2.680 | +3.337 | +2.053 |
| | | Mobile | –0.000 | –0.039 | +0.019 | +0.024 | +0.605 | –0.248 |
| | | Foreman | –0.182 | –0.106 | +0.043 | +3.670 | +2.222 | +0.839 |
| | | Coastguard | –0.005 | –0.038 | +0.010 | –0.027 | +0.631 | –0.156 |
| | 35KB | Table tennis | –0.113 | –0.139 | –0.051 | +2.287 | +2.817 | +1.046 |
| | | Mobile | +0.019 | –0.018 | +0.000 | –0.247 | +0.318 | +0.009 |
| | | Foreman | –0.293 | –0.159 | +0.021 | +6.304 | +3.268 | –0.514 |
| | | Coastguard | +0.004 | –0.029 | +0.024 | –0.052 | +0.595 | –0.494 |

Figs. 7–9 show the frame by frame bitrate usage of different algorithms. This simulation condition is the same as previous one but with unevenly distributed data bandwidth to each MB. This condition is to simulate the traditional ME design with fixed and pre-allocated bandwidth. In that condition, the ME design has fixed search range and expects the search range data arrival in time for computation. However, if the real allocated bandwidth is lower than expected due to bandwidth competition from other devices, the operations of motion estimation will be skipped and the intra mode will be selected as best prediction mode when the allocated data bandwidth has run out. For the bandwidth usage, the simulation shows that both of FS and our proposed algorithm would use up all the available bandwidth but the BBGS and NN algorithm wouldn't. The PSNR results are similar to the previous one but with quite different bit rate. The result shows that the bitrate usage of the proposed algorithm is much less than those in other algorithms due to fewer intra block coding. This proves that our algorithm can properly allocate bandwidth to MBs according to their content and use up the available bandwidth but never exceed the budget to support ME operations.

## V. PROPOSED ARCHITECTURE

Fig. 10 shows the proposed architecture and its operation is described as follows. First, the current pixels and $22 \times 22$ reference pixels will be, respectively, loaded into *Current buffer* and *Reference buffer* through the help of *Memory controller*. Here, the scheduling policy of *Memory controller* is first-in-first-serve so that the requests are processed in the request order. Afterwards, the pixels stored in *Current buffer* and *Reference buffer* will be used to calculate SADs through the *SAD calculation module* which will combine the motion vector costs generated from *MV cost generator* to obtain rate distortion costs of $RDCost_0$, $RDCost_1$, and $RDCost_2$. In our design, the *SAD calculation module* is composed of 16 $PE_4 \times 4$s (Fig. 11), and each $PE\_4 \times 4$ [Fig. 12(a)] computes the SAD value of a $4 \times 4$ block so that the SAD of one candidate position can be generated per cycle. The rate distortion costs generated from *SAD calculation module*

TABLE VIII
COMPARISON OF BD-PSNR AND BD-BITRATE FOR HIGH MOTION SEQUENCES

| Image size | $DB_{Total}$ | Sequence | BD-PSNR (dB) | | | BD-Bitrate (%) | | |
|---|---|---|---|---|---|---|---|---|
| | | | NN | BBGS | Proposed | NN | BBGS | Proposed |
| 1080p | 385KB | Blue sky | −0.449 | −0.151 | +0.035 | +6.669 | +2.235 | −0.439 |
| | | Pedestrian | −0.017 | −0.016 | +0.146 | +0.316 | +0.273 | −2.707 |
| | 765KB | Blue sky | −0.027 | -0.027 | +0.079 | +0.361 | +0.361 | −1.509 |
| | | Pedestrian | −0.004 | −0.004 | +0.033 | +0.055 | +0.055 | −0.645 |
| CIF | 20KB | Football | −0.203 | −0.126 | +0.038 | +3.011 | +1.853 | −0.631 |
| | | Soccer | −0.215 | −0.150 | +0.105 | +3.929 | +2.689 | −1.880 |
| | | Stefan | −0.152 | −0.153 | +0.544 | +2.161 | +2.143 | −7.143 |
| | 38KB | Football | −0.149 | −0.061 | +0.085 | +2.220 | +0.927 | −1.295 |
| | | Soccer | −0.209 | −0.112 | +0.086 | +3.760 | +2.090 | −1.517 |
| | | Stefan | −0.085 | −0.067 | +0.516 | +1.158 | +0.876 | −6.599 |

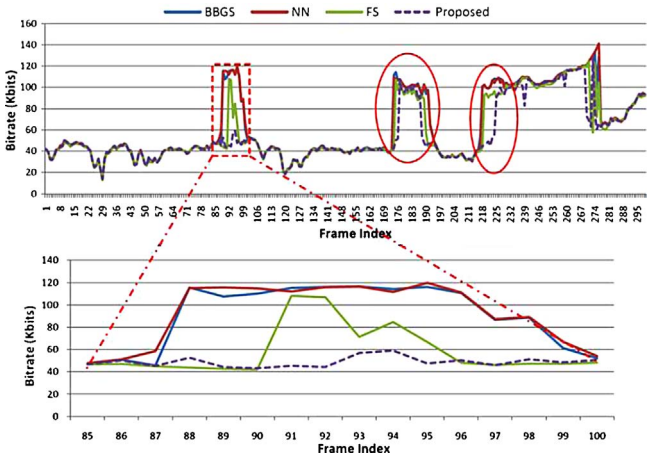


Fig. 7. Comparison of bitrate for *Stefan* sequence under 38 kB data bandwidth constraint.

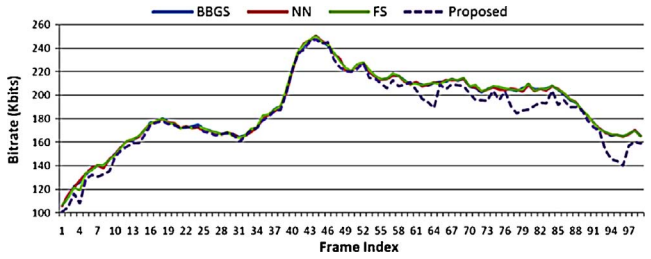


Fig. 8. Comparison of bitrate for *Pedestrian* sequence under 385 kB data bandwidth constraint.

are stored in *RDCost buffer* for the following usage. Once the $RDCost_0$, $RDCost_1$, and $RDCost_2$ have been calculated, the modeled information of search steps will be computed as (5)–(9) by the *Bandwidth modeling module* and sent to *Bandwidth aware ME controller*. Afterward, the *Bandwidth aware ME controller* will execute the computations as (10)–(17) with the calculated search steps, and generate the memory address to *Memory Controller* to load the required reference pixels for the following search steps. Once the allocated bandwidth has been run out or the termination criterion of nearest neighbours search has been met, the motion estimation search will be finished.

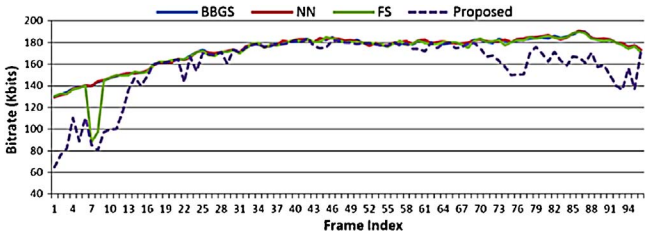


Fig. 9. Comparison of bitrate for *Station2* sequence under 385 kB data bandwidth constraint.

### A. Bandwidth Modeling Module

The required search steps of our bandwidth aware ME is mainly computed by (5)–(9). However, these equations need five multipliers, three dividers, three subtractions and a look-up table to generate the corresponding search steps by a direct implementation approach, which is too costly. Therefore, some approximations are applied to reduce the hardware cost. First, the operation of $\Delta RDCost_2 - \Delta RDCost_1$ is computed as in (18) so that the subtractions can be reduced from three to two with an additional left-shift operation

$$\begin{aligned}\Delta RDCost &= \Delta RDCost_2 - \Delta RDCost_1 \\ &= \left(\frac{RDCost_2 - RDCost_1 - 2RDCost_0}{RDCost_0}\right) \times 100. \end{aligned} \quad (18)$$

In addition, since the $\beta$ is set to 0.1, the $\Delta RDCost_{improved}$ can be rewritten as follows:

$$\Delta RDCost_{improved} = \alpha \times e^{(\Delta RDCost)\times 0.1}. \quad (19)$$

From (7), we observed that $\gamma$ is set to 0.001 empirically. However, we can rewrite (7) as follows:

$$\alpha = RDCost_0 \times \gamma | \gamma = 0.001 = \frac{1}{1000} \rightarrow \alpha = \frac{RDCost_0}{1000}. \quad (20)$$

To reduce the hardware cost of division operation, the division operation of 1/1000 is simply approximated by 1/1024 in our design. The benefit of this approximation is that experimental results show that there is no rate distortion performance difference between the operations of 1/1000 and 1/1024 but the division operation of 1/1000 can be simply replaced by a right-shit operator to reduce the hardware cost. (The results in Section IV already include this approximation into the simulation.) Finally, the $\Delta RDCost_{improved}$ can be rewritten as follows:

$$\Delta RDCost_{improved} = \frac{RDCost_0}{1024} \times e^{\left(\frac{RDCost_2 - RDCost_1 - 2RDCost_0}{RDCost_0}\right)\times 10}. \quad (21)$$

Fig. 13 shows the architecture of our proposed *Bandwidth modeling module*. In which the *Exp. LUT* module is a look-up table to implement the exponential operation, and it is organized by 46 entries with 12 bits per each. With the above approximation, the hardware cost can be reduced significantly due to our proposed architecture only needs two multipliers, two subtractions, one divider, three shifters, and a look-up table when compared to direct implementation approach.

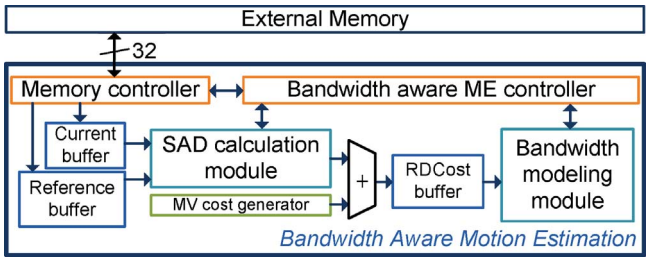


Fig. 10. Architecture of proposed bandwidth aware ME.

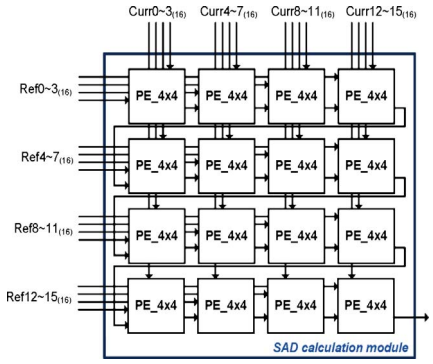


Fig. 11. Architecture of SAD calculation module.

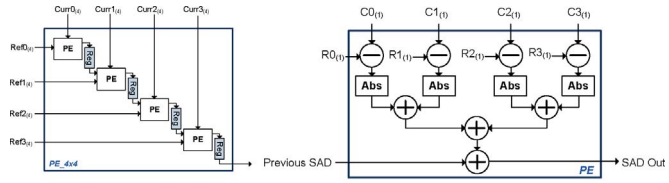


Fig. 12. (a) PE_4 × 4. (b) PE.

## B. Design of the On-Demand Reference Buffer

Fig. 14 shows the design of the *Reference buffer*. In our reference buffer design, the *rc_sel* signal is used to determine where a row of 22 pixels or a column of 22 pixels should be replaced in 22 × 22 Pixels buffer. In addition, the *addr_x* and *addr_y* signals play the role of determining which 256 pixels should be selected to output for SAD calculation according to ME search algorithm. Note that the size of *Reference buffer* is only 22 × 22 pixels which can cover the first three steps and help the data reuse in the remaining steps. In addition, another benefit of 22 × 22 buffer size is that the reference data loading request can be issued simultaneously to memory controller to load future necessary reference data once the search direction has been decided and SAD calculation module starts to compute SAD. As a result, the stalled cycles can be eliminated.

Fig. 15 shows the mechanism of reference data updating for our *Reference buffer*. In our *Reference buffer* design, we adopt the circular addressing mode so that the new loaded reference data will be stored in most-top-row [Fig. 15(a)], most-bottom-row [Fig. 15(b)], most-left-column [Fig 15(c)], or most-right-column [Fig. 15(d)] registers depending on the required search direction. Through this updating mechanism,

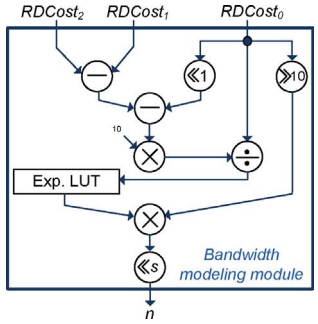


Fig. 13. Architecture of proposed bandwidth modeling module.

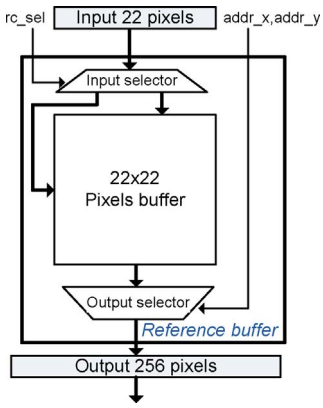


Fig. 14. Architecture of reference buffer.

only one row or column reference data should be updated instead of overall reference data and consequently reducing the power consumption for updating *Reference buffer* contents.

With the above scheme, the required buffer size is independent of required search range. This is quite different from other designs for fast algorithms [4]–[6], [18], [19] that load all search range data into on-chip memory. However, a common issue for above scheme is the DRAM access latency if the accessed data pattern is a column access as Fig. 3(b) and (c). If these access data patterns are required, the DRAM access latencies will be increased due to the required column of pixels are not stored continuously in DRAM. Therefore, the required cycle counts to load the data will be analyzed in the following subsection with the consideration of DRAM access latency.

## C. Timing Analysis with DRAM Access Latency

Fig 16 shows the timing diagram of the proposed design. This timing diagram accurately includes the necessary memory latency caused by DRAM access into the simulation framework by adopting the Micron SDRAM model [28] in our design. In general, the DRAM latency is mainly from the row miss and page miss due to the cross page or cross row access in a data access. Such misses in our algorithm could occur in the same step access or the different step access if the required data are located in different pages or rows. Therefore, the video data mapping in DRAM uses the nearly optimal data mapping proposed in [29] to gain the benefit of high spatial locality of neighboring MBs to reduce the access penalty by row miss or page miss. Fig. 17 shows the data mapping of

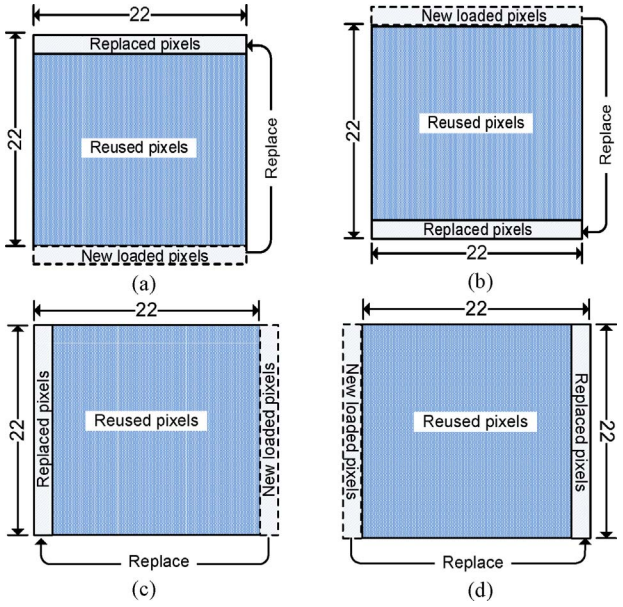


Fig. 15. Mechanism of *Reference buffer* updating in which reference data are loaded for (a) down-toward, (b) up-toward, (c) right-toward, and (d) left-toward search process.

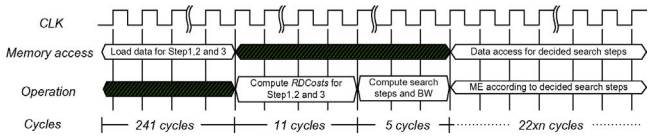


Fig. 16. Timing diagram of proposed bandwidth aware ME.

our design. In which the luminance, chrominance, and motion vector components of four adjacent MBs are grouped and stored at the same memory row. With above data mapping, the external DRAM is interconnected to the proposed ME module via a 32 bits data bus so that four pixels can be loaded from external memory per cycle. Thus, initial $22 \times 22$ reference data loading needs 241 cycles in average according to the simulations. With these data, 11 cycles are needed to calculate the *RDCosts* for five candidates at the first step and three candidates at the second and third step. Once the *RDCosts* of *step 0*, *1*, and *2* have been calculated, 5 cycles are used to derive necessary search steps and bandwidth for the following ME search. Finally, the nearest neighbours search is executed to obtain the best results according to the decided search steps $n$. It is worth mentioning that the required cycles for each step are only the external memory access cycles, 22 cycles in maximum under our operating frequency, since the SAD computation cycles are less than external memory access cycles and thus these operations can be parallel executed.

Table IX shows the memory bandwidth saving comparison per MB for our proposed algorithm and Level C data reuse full search ME scheme. Both of memory access latencies and data amounts are considered to derive the real memory bandwidth saving in this simulation results. From this table, we observed that the memory bandwidth savings for search range $\pm 16$, $\pm 32$, and $\pm 64$ are 13.51%, 59.47%, and 77.43%, respectively. In contrast to memory bandwidth saving listed in

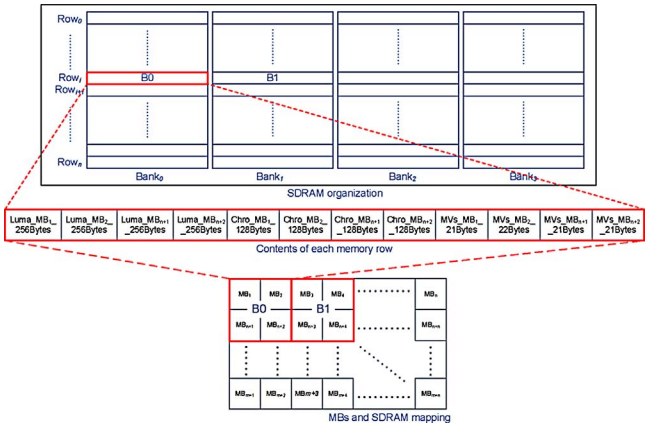


Fig. 17. SDRAM data mapping of our proposal.

TABLE IX
COMPARISON OF MEMORY BANDWIDTH SAVING PER MB WITH DRAM LATENCY

| Sequences | CIF | | Sequences | 4CIF |
|---|---|---|---|---|
| | SR: ±16 | SR: ±32 | | SR: ±64 |
| Akiyo | 17.59% | 61.40% | Akiyo | 79.15% |
| Coastguard | 14.52% | 59.96% | City | 78.29% |
| Football | 2.08% | 54.13% | Coastguard | 78.07% |
| Foreman | 11.72% | 58.52% | Football | 71.73% |
| Mobile | 13.60% | 59.53% | Foreman | 76.38% |
| MD | 16.39% | 60.83% | Hall | 78.71% |
| News | 16.79% | 61.00% | Mobile | 78.37% |
| Table tennis | 15.41% | 60.35% | News | 78.77% |
| **Average** | **13.51%** | **59.47%** | **Average** | **77.43%** |

Table V, we can observe 3% data bandwidth saving difference on average caused by DRAM latency. For the high motion sequence, the memory bandwidth saving of *Football* sequence is very different from the data bandwidth saving analyzed in Table V. The reason is that the most of data accesses in the *Football* sequence are tended to load a column of pixels for SAD calculation from external DRAM since it has higher motion behavior in objects and moving in horizontal direction so that many search steps are allocated in this direction. Unfortunately, the memory access cycles for a column of pixels are much higher than a row of pixels in case of the same data access amounts, when constrained by the external DRAM data mapping mechanism. Therefore, the memory bandwidth saving of *Football* sequence is lower than other sequences and different from the data bandwidth saving listed in Table V. However, it still has 2.08% memory bandwidth saving. For the median and low motion sequences, the difference between these two tables is not noticeable. As a result, the impact on the bandwidth usage depends on the motion of the sequences instead of the frame resolution, said the higher impact on higher motion sequences.

In summary, the simulation results demonstrate that our proposed algorithm can still work well and achieve efficient bandwidth usage even considering the DRAM latency.

### D. Implementation Results

The proposed design is implemented and synthesized by the UMC 90 nm technology. Table X shows the design comparison

TABLE X
COMPARISON OF HARDWARE IMPLEMENTATION RESULTS

| Algorithm | Supporting Resolution | Max. Search Range | Gate Counts (K) | Average PSNR loss | On-chip SRAM (Bytes) | Frequency (MHz) | Process |
|---|---|---|---|---|---|---|---|
| [4] | D1@25fps | 16X16 | 165.9 | 0.00 | 1.66K | 180 | 0.25um |
| [5] | CIF@30fps | 16X16 | 89.39 | 0.08 | 3.01K | 27.8 | 0.35um |
| [6] | CIF@30fps | 16X16 | 250 | 0.10 | 5K | 13.5 | 0.13um |
| [18] | CIF@30fps | 16X16 | 68.5 | 0.19 | 1.23K | 1.67 | 0.18um |
| [19] | CIF@30fps | 16X16 | 62.6 | 0.23 | 1.08K | 1.67 | 0.18um |
| Proposed | CIF @30fps | 32X32 | 75.27 | 0.003 | 0 | 5.6 | 90nm |
| | 4CIF@30fps | 64X64 | | 0.005 | | 23 | 90nm |

with other architectures. As shown in the table, the proposed design can process CIF and 4 CIF sized video at 30 frames/s when running at 5.6 MHz and 23 MHz operating frequency, respectively. It is worth to mention that the on-chip SRAM size is usually increased with the growth of search range size for most of ME hardware due to noticeable reference pixels have to be stored temporally in on-chip SRAM for SAD computation process. However, the main difference and contribution of our proposed architecture is that the demanded on-chip SRAM size is irrelevant to the search range size due to the reference pixels are loaded from external memory on real demand. When compared to other designs, our proposal can support large search range with very slight hardware overhead, with 75.27 K gate counts.

## VI. CONCLUSION

In this paper, an on-demand data bandwidth efficient ME with rate distortion optimization algorithm was proposed. Three properties were revealed in our proposal. First, the on-demand reference data acquiring mechanism led our proposal only accessing reference data on-demand and could avoid unnecessary reference data loading. Second, through modeling the relationship between the data bandwidth and rate distortion performance, the data bandwidth could be efficiently allocated to the ME process according to the characteristics of video content under the bandwidth constraint. Finally, via the efficient prediction of proposed method, the low data bandwidth requirement ME could also be achieved. Simulation results show that our proposed method can save 79.15% of data bandwidth requirements in maximum with only 0.03 dB PSNR drop and 2.50% of bitrate increase when compared with the full search method for search range $\pm 64$ in 4 CIF resolution. Furthermore, our proposed algorithm can achieve 2.43%, 0.08%, and 0.20% BD-bitrate saving with 0.17 dB, 0.01 dB, and 0.01 dB BD-PSNR increase on average for high, median, and low motion sequences under the available data bandwidth constraint when compared to the full search motion estimation algorithm. The resulted hardware implementation with the simplified bandwidth control scheme reveals that our proposed ME design can process 30 frames in 4 CIF resolution per second when running at 23 MHz operating frequency with only 75.27 K gate counts and search range independent buffer.

## REFERENCES


[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, 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, Jul. 2003.

[2] C.-H. Hsieh and T.-P. Lin, "VLSI architecture for block-matching motion estimation algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 2, pp. 169–175, Jun. 1992.

[3] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[4] W.-F. He, Y.-L. Bi, and Z.-G. Mao, "Efficient frame-level pipelined array architecture for full-search block-matching motion estimation," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3. May 2005, pp. 2887–2890.

[5] Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, "Global elimination algorithm and architecture design for fast block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 898–907, Jun. 2004.

[6] M. Miyama, J. Miyakoshi, Y. Kuroda, K. Imamura, H. Hashimoto, and M. Yoshimoto, "A sub-mW MPEG-4 motion estimation processor core for *Mobile* video application," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1562–1570, Sep. 2004.

[7] S. Kappagantula and K. R. Rao, "Motion compensated interframe image prediction," *IEEE Trans. Commun.*, vol. 33, no. 9, pp. 1011–1015, Sep. 1985.

[8] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[9] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[10] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 349–355, May 2002.

[11] P.-L. Tai, S.-Y. Huang, C.-T. Liu, and J.-S. Wang, "Computation-aware scheme for software-based block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 9, pp. 901–912, Sep. 2003.

[12] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 645–658, May 2005.

[13] W.-M. Chao, C.-W. Hsu, Y.-C. Chang, and L.-G. Chen, "A novel hybrid motion estimator supporting diamond search and fast full search," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2. May 2002, pp. 492–495.

[14] C.-Y. Chen, C.-T. Huang, Y.-H. Chen, and L.-G. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, Apr. 2006.

[15] T.-C. Chen, C.-Y. Tsai, Y.-W. Huang, and L.-G. Chen, "Single reference frame multiple current macroblocks scheme for multiple reference frame motion estimation in H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 2, pp. 242–247, Feb. 2007.

[16] H. Shim, K. Kang, and C.-M. Kyung, "Search area selective reuse algorithm in motion estimation," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2007, pp. 1611–1614.

[17] M.-C. Lin and L.-R. Dung, "Two-step windowing technique for wide range motion estimation," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Nov. 2008, pp. 1478–1481.

[18] S.-H. Wang, W.-L. Tao, C.-N. Wang, W.-H. Peng, and T. Chiang, "Platform-based design of all binary motion estimation with bus interleaved architecture," in *Proc. IEEE Int. Symp. VLSI Des., Automat. Test.*, Apr. 2005, pp. 241–244.

[19] S.-H. Wang, S.-H. Tai, and T. Chiang, "A low-power and bandwidth-efficient motion estimation IP core design using binary search," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 760–765, May 2009.

[20] T.-C. Wang, Y.-W. Huang, H.-C. Fang, and L.-G. Chen, "Performance analysis of hardware oriented algorithm modification in H.264," in *Proc. IEEE Int. Conf. Multimedia Exp*, vol. 3. Jul. 2003, pp. 601–604.

[21] M. Gallant, G. Gote, and F. Kossentini, "An efficient computation-constrained block-based motion estimation algorithm for low bit rate vide coding," *IEEE Trans. Image Process.*, vol. 8, no. 12, pp. 1816–1823, Dec. 1999.

[22] JM11 [Online]. Available: http://iphome.hhi.de/suehring/tml/download

[23] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799–1808, Dec. 1981.
[24] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. 33, no. 8, pp. 888–896, Aug. 1985.
[25] Y.-H. Chen, T.-D. Chuang, Y.-J. Chen, and L.-G. Chen, "Bandwidth-efficient encoder framework for H.264/AVC scalable extension," in *Proc. IEEE Int. Symp. Multimedia*, Dec. 2007, pp. 401–406.
[26] T.-C. Chen, Y.-H. Chen, S.-F. Tsai, S.-Y. Chien, and L.-G. Chen, "Fast algorithm and architecture design of low-power integer motion estimation for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 5, pp. 568–577, May 2007.
[27] L.-K. Liu and B. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 419–422, Aug. 1996.
[28] Micron. (2010). *DDR SDRAM* [Online]. Available: http://www.micron.com/products/dram/ddr/partlist.aspx
[29] G.-S. Yu and T.-S. Chang, "Optimal data mapping for motion compensation in H.264 video decoding," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2007, pp. 505–508.

**Gwo-Long Li** (S'09) received the B.S. degree from the Department of Computer Science and Information Engineering, Shu-Te University, Kaohsiung, Taiwan, in 2004, and the M.S. degree from the Department of Electrical Engineering, National Dong-Hwa University, Hualien, Taiwan, in 2006. He is currently pursuing the Ph.D. degree from the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan.

His current research interests include video signal processing and its very large scale integration architecture design.

Mr. Li received the Excellent Master Thesis Award from the Institute of Information and Computer Machinery in 2006.

**Tian-Sheuan Chang** (S'93–M'06–SM'07) received the B.S., M.S., and Ph.D. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1993, 1995, and 1999, respectively.

He is currently an Associate Professor with the Department of Electronics Engineering, NCTU. From 2000 to 2004, he was a Deputy Manager with Global Unichip Corporation, Hsinchu. His current research interests include (silicon) intellectual property and system-on-a-chip design, very large scale integration signal processing, and computer architecture.