Algorithm and Architecture Design of Bandwidth-Oriented Motion Estimation for Real-Time Mobile Video Applications

Jui-Hung Hsieh, *Student Member, IEEE*, and Tian-Sheuan Chang, *Senior Member, IEEE*

***Abstract*—This paper proposes a data bandwidth-oriented motion estimation design for resource-limited mobile video applications using an integrated bandwidth rate distortion optimization framework. This framework predicts and allocates the appropriate data bandwidth for motion estimation under a limited bandwidth supply to fit a dynamically changing bandwidth supply. The simulation results show that our proposed algorithm can achieve 66% and 41% memory bandwidth savings while maintaining an equivalent rate-distortion performance and meeting real-time targets, when compared with conventional approaches for low-motion and high-motion D1 (704 $\times$ 576)-size video, respectively. The final implementation costs 122 K gate counts with TSMC 0.13-μm CMOS technology and consumes 74 mW of power for D1 resolution at 30 frames/s which is 40% of that achieved in previous designs.**

***Index Terms*—H.264/AVC, low power, memory bandwidth, motion estimation, video coding, VLSI architecture.**

I. Introduction

MOTION-COMPENSATED temporally predictive coding with motion estimation (ME) is an effective tool for removing temporal redundancy among adjacent frames in modern video encoder designs [1], which demands high computational complexity [2] and memory bandwidth. These design problems are greater for resource-limited mobile devices, which require a fine tuning of video coding algorithms to better utilize the available resources.

To address these design issues in mobile devices, various solutions have been proposed. The designs in [3]–[5] attempted to reduce computational complexity or hardware cost by adopting low-complexity fast algorithms. [3] proposed a complexity rate distortion optimization framework to reduce the complexity of ME operation in a rate-distortion (R-D) sense. Reference [4] introduced a new fast ME algorithm based on an adaptive search range adjustment scheme and a matching point decimation scheme to reduce computational costs. Reference [5] presented pixel truncation to reduce computation and memory access. These designs aim to reduce or consider computational complexity for limited-resource environments. Meanwhile, the algorithms and architectures in [6]–[10] focus on lower power consumption and higher hardware efficiency. Reference [6] proposed a new algorithm and architecture using the sum of the absolute difference (SAD) to achieve low-power design. Subsequently, [7] and [8] employed a fast algorithm and power-efficient hardware architecture to reduce power consumption. In addition, the power-aware designs in [9] and [10] have also been proposed for mobile devices to adjust power consumption in response to different conditions, such as user preferences and battery states [11]. However, all of the aforementioned designs, [3]–[10], assume limited computational resources in handheld devices, but sufficient memory bandwidth. This assumption does not always hold. For resource-limited systems such as mobile video applications, the available memory bandwidth is dynamically changing and limited under simultaneous coding and decoding cases or different operating modes with dynamic voltage frequency scaling [11]. With a lower than expected bandwidth supply, ME computations fail to meet real-time constraints or lead to significant R-D performance loss due to the macroblock (MB) skipping coding.

To solve the above problems, this paper proposes an optimization framework to allocate the bandwidth for ME computations. We first apply a heuristic approach to develop an analytical bandwidth-oriented model that allocates memory bandwidth according to the R-D gain and bandwidth constraints. Then, we propose a bandwidth-scalable ME algorithm that considers the video contents and bandwidth constraints to reduce memory access and maximize coding performance. In this algorithm, a greater bandwidth is allocated to high-motion MBs while bandwidth is significantly reduced for low-motion MBs. Consequently, the bandwidth is efficiently utilized and minimized to provide the optimal R-D gain. Finally, hardware implementation shows its effectiveness in achieving a higher throughput and lower power consumption than previous designs.

The remainder of this paper is organized as follows. Section II reviews related work. In Section III, we present a bandwidth-scalable algorithm for ME and show corresponding simulation results. The hardware architecture and implementation results of the proposed algorithm are shown in Section IV. Finally, Section V concludes this paper.

II. Review of Related Work

ME is the most bandwidth-consuming operation in video coding due to the requirement of large search windows to find the best matching block in the reference frames. For a device with a fixed video size and fixed frame rate video, the required bandwidth generally depends on the chosen search range (SR).

Manuscript received March 21, 2011; revised June 30, 2011, October 03, 2011; accepted November 10, 2011. Date of publication January 23, 2012; date of current version December 19, 2012.

The authors are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan (e-mail: jhhsieh.ee95g@nctu.edu.tw; tschang@twins.ee.nctu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2011.2178439

In general, the selection of SR has two effects on coding quality. First, a small SR could degrade the R-D performance due to a loss of optimal search points, but it requires less bandwidth, which in turn reduces memory access power consumption. Second, a large SR generally enhances the R-D performance but also consumes more bandwidth and power. To reduce the required bandwidth, conventional approaches usually exploit different ways of reusing data or using a dynamically changing SR [12], [13]. Many algorithms have been introduced to analyze and solve the problem of memory bandwidth in ME operation by exploiting data reuse [14] or cache design [15]–[17]. Among these, the Level C data reuse strategy in [14] can regularly reuse a region of $(2 \times \mathrm{SR}_V + 16) \times (2 \times \mathrm{SR}_H)$ pixels with a square search window at (0, 0), the search center, which is commonly used in most full-search ME hardware designs. In which, SR_V and SR_H respectively indicate the search range in the vertical and horizontal directions. Meanwhile, only a $(2 \times \mathrm{SR}_V + 16) \times 16$ rectangle is required to be loaded to on-chip memory (such as SRAM). This can significantly reduce the external memory bandwidth access and save SR memory writing power [15]. Instead of regular Level C data reuse, cache-based designs explore the possible data reuse probability through a complicated cache-controlled scheme. Thus, they are more suitable for designs with fast ME search algorithms. In addition to the aforementioned schemes, the bus bandwidth-effective ME design presented in [18] lowers the bandwidth requirement by reducing the pixel representation from 8 bits to a binary pattern. The design in [19] uses a selective search area reuse strategy to reduce the memory access by selecting a search center at the frame level. [20] proposed a memory-efficient ME design based on highly parallel architecture and a data overlapping scheme to decrease on-chip memory access.

However, these studies assume a constant and sufficient memory bandwidth, regardless of the limited and dynamic bandwidth resources available for power-limited mobile devices. Experimental studies have shown that, for D1 (704 × 576) picture sizes, ME consumes about 2/3 of the total bus bandwidth [17], [18] in a typical mobile video phone system, as shown in Fig. 1. In the above system, the available bandwidth changes dynamically due to simultaneous multi-source rather than single-source data access or a different power-oriented operating mode. Note that the constant and sufficient bandwidth assumption simplifies the design procedure, but this also has significant design implications in dynamic bandwidth environments. The actual bandwidth demand depends heavily on the video contents, but that is not easy to predict before coding. In summary, the key issue in the bandwidth allocation problem is how to avoid a bandwidth usage overflow and minimize bandwidth usage under bandwidth-limited constraints while keeping the R-D performance as constant as possible. Usage overflow will lead to encoding delays, which in turn imply a failure of real-time constraints or to quality losses due to frame dropping. In the following section, we show how to tackle this problem systematically.

III. Proposed Framework

In this section, we introduce the proposed bandwidth-oriented ME using a bandwidth rate distortion framework. First,

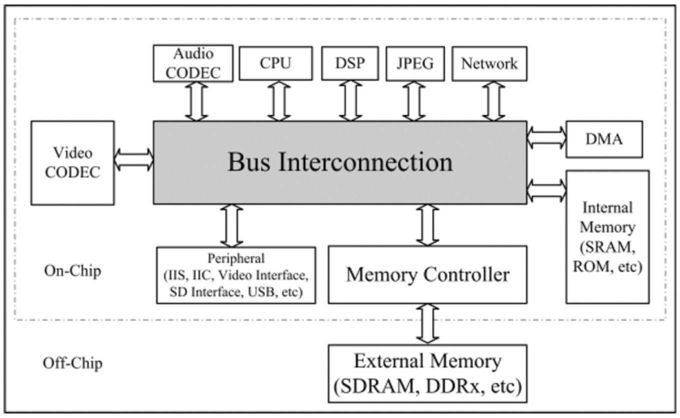
Audio CODEC
CPU
DSP
JPEG
Network
Video CODEC
Bus Interconnection
DMA
Internal Memory (SRAM, ROM, etc)
On-Chip
Peripheral (IIS, IIC, Video Interface, SD Interface, USB, etc)
Memory Controller
Off-Chip
External Memory (SDRAM, DDRx, etc)

Fig. 1. Block diagram of a typical mobile video phone.

the available bandwidth constraint is properly modeled in our R-D optimization framework to efficiently use the data bandwidth. Based on the model, a bandwidth-scalable ME algorithm is then established to maximize the R-D performance under constraints.

A. Proposed Optimized Bandwidth-Rate-Distortion Modeling

In the bandwidth-oriented ME, the coding distortion D is not only a function of the used bit rate R, but also a function of the available and consumed data bandwidth BW, which can be modeled as

$$D = D(\mathrm{R}, \mathrm{BW}). \tag{1}$$

However, the overall bandwidth usage of ME is linearly proportional to the SR. Thus, to control the SR in ME, we first introduce a set of bandwidth control parameters $\mathbf{B} = [\beta_1, \beta_2, \ldots, \beta_L]$. In which, the variable of L is a symbol to represent the maximum number of bandwidth control parameter for each system. Accordingly, the SR selection of ME is then a function of these control parameters, denoted by $\mathrm{SR}(\beta_1, \beta_2, \ldots, \beta_L)$. The bandwidth requirement of ME, denoted by BW, is a function of SR and is also a function of **B**, denoted by

$$\mathrm{BW} = \Phi(\mathrm{SR}) = \mathrm{BW}(\beta_1, \beta_2, \ldots, \beta_L) \tag{2}$$

where $\Phi(\cdot)$ is the SR selection model of the ME and $\{\beta_i; 1 \leq i \leq L\}$ are bandwidth control parameters. To optimize the bandwidth usage, the available data bandwidth β_i among the MBs should be dynamically allocated according to their motion characteristics, that is, R-D performance. Thus, for a given data bandwidth, the R-D behavior of ME can be well approximated by its R-D gain (*RDGain*, the Lagrange R-D cost difference)

$$\mathrm{RDGain}(\mathrm{BW}) = J_{\mathrm{MVP}}(\mathrm{BW}) - J_{\mathrm{BMA}}(\mathrm{BW}) \tag{3}$$

where $J_{\mathrm{MVP}}(\mathrm{BW})$ and $J_{\mathrm{BMA}}(\mathrm{BW})$, respectively, denote the R-D cost at the search center of the motion vector predictor

(MVP) and the final best position by the block-matching algorithm under a given BW. The Lagrange R-D cost function here is defined as

$$J(\mathrm{mv}, \lambda_{\mathrm{motion}}) = \mathrm{SAD}(p, c(\mathrm{mv})) + \lambda_{\mathrm{motion}} R(\mathrm{mv} - \mathrm{mvp}) \tag{4}$$

where mv is the calculated motion vector and $\lambda_{\mathrm{motion}}$ indicates the Lagrange multiplier. The distortion term $\mathrm{SAD}(p, c(\mathrm{mv}))$ is the sum of absolute differences between the original signal p and the coded video signal c. Meanwhile, the rate term $\lambda_{\mathrm{motion}} R(\mathrm{mv} - \mathrm{mvp})$ represents the motion information and the coded bit length of the motion vector difference (MVD) between the motion vector and the motion vector predictor. Thus, we can associate the R-D gain and the bandwidth control parameters with the coding efficiency to perform online bandwidth optimization for real-time video coding in a bandwidth update period (P_{update}) as

$$\text{coding efficiency} = \frac{\sum\limits_{\mathrm{MB}_{k \in P_{\mathrm{update}}}} (J_{\mathrm{MVP}}[k] - J_{\mathrm{BMA}}[k])}{\mathrm{BW}(\beta_1, \beta_2, \ldots, \beta_{\mathrm{L}})} \tag{5}$$

where $J_{\mathrm{MVP}}[k]$ denotes the R-D cost of the current coded kth MB (MB_k) at the position of the MVP, $J_{\mathrm{BMA}}[k]$ denotes the R-D cost of the current coded kth MB after the motion search of the block-matching algorithm. The bandwidth update period P_{update} denotes the period to update the available system bandwidth, which depends on the system conditions. In general, a shorter period can track the system bandwidth change well, which is more suitable for fast changed environments, but it will also need smart memory controllers [21] which can deliver information of fast bandwidth arbitration change. A longer period can fit into current ordinary memory controllers well with slow bandwidth change. However, with the proposed algorithm shown in the following subsection, we can still adapt to sudden change of system bandwidth and achieve similar performance. In this paper, the P_{update} is set from one frame to multiple frames. Note that P_{update} can be changed at any time to adapt to abrupt bandwidth change. With above definition, the operation mode with the greatest coding efficiency should be selected, because it also indicates a better R-D performance gain with equal bandwidth consumption.

B. *Proposed Bandwidth-Scalable ME Algorithm*

The concept of the proposed algorithm is to allocate the available bandwidth in a R-D-optimized sense within the given bandwidth budget. During this allocation, the R-D performance should be kept as smooth as possible for consecutive coding, while the R-D performance should also be maximized within the available bandwidth. However, the true consumed bandwidth is unknown before the real coding process occurs. A reasonable way to accomplish this task is based on using the statistics of past bandwidth usage and R-D performance, that is, the coding efficiency defined in (5), to predict the possible bandwidth usage of the next coding unit. With the above constraints and prediction, an appropriate bandwidth can be allocated to meet the requirements.

Thus, the whole algorithm can be divided into four steps as described below.

Step 1: First, the memory bandwidth budget, $\mathrm{BW}_{\mathrm{budget}}$, within P_{update} is initialized for bandwidth allocation in later coding processes and is calculated by

$$\mathrm{BW}_{\mathrm{budget}} = \frac{B_r}{F_r} \times G_p \tag{6}$$

where *Br* denotes the data rate (bytes/s) allocated by the memory controller [21], *Fr* denotes the coded frame number per second, and *Gp* denotes the frame numbers within P_{update}. Assume the bandwidth budget is evenly distributed over all coded MBs within P_{update}

$$\mathrm{BW}_{\mathrm{budget}} = (2 \times \mathrm{SR}_{\mathrm{sys}} + 16)^2 \times N_{\mathrm{MB}} \tag{7}$$

where N_{MB} is the total number of MBs within P_{update}. Thus, the global system SR $(\mathrm{SR}_{\mathrm{sys}})$ for the ME hardware is

$$\mathrm{SR}_{\mathrm{sys}} = \mathrm{Floor}\left\{\frac{1}{2}\left(\sqrt{\frac{\mathrm{BW}_{\mathrm{budget}}}{N_{\mathrm{MB}}}} - 16\right)\right\} \tag{8}$$

as the upper bound of the following SR adjustment. If P_{update} is changed when available bandwidth is changed abruptly, new value can be set as well to force lower bandwidth usage when necessary.

Step 2: To justify the coding efficiency under a given bandwidth, we define the actual bandwidth efficiency for coding the kth MBs. The bandwidth efficiency ($G[k]$, the R-D gain per unit bandwidth) for a current coded kth MB is calculated as (9), which implies how much R-D gain can be achieved for one unit of data bandwidth

$$G[k] = \frac{\sum\limits_{i=1}^{k-1} (J_{\mathrm{MVP}}[i] - J_{\mathrm{BMA}}[i])}{\mathrm{BW}_{\mathrm{used}}[k]} \tag{9}$$

where $\mathrm{BW}_{\mathrm{used}}[k]$ denotes the accumulated used data bandwidth up to the $(k-1)$th MB.

Step 3: With the average bandwidth efficiency, we can predict the required bandwidth of the current coded MB with the quality smoothness constraint. The quality smoothness constraint implies that the R-D gain should be slowly changed between neighboring coded MBs. Thus, we can expect the current coded MB to have an R-D gain similar to that of previous coded MBs. Using past statistics, we can predict the possible bandwidth usage (called past BW prediction, $\mathrm{BW}_{\mathrm{PP}}$) for the current coded MB. However, the remaining available bandwidth should be equally distributed for the remaining MBs (called future BW prediction $\mathrm{BW}_{\mathrm{FP}}$). Thus, we have

$$\mathrm{BW}_{\mathrm{PP}}[k] = \frac{J_{\mathrm{MVP}}[k] - \mathrm{avg}\left(\sum\limits_{i=1}^{k-1} J_{\mathrm{BMA}}[i]\right)}{G[k]} \tag{10}$$

$$\mathrm{BW}_{\mathrm{FP}}[k] = \frac{\mathrm{BW}_{\mathrm{budget}} - \mathrm{BW}_{\mathrm{used}}[k]}{N_{\mathrm{MB}} - (k-1)} \tag{11}$$

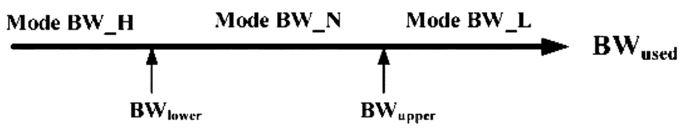

Fig. 2. Illustration of the bandwidth interval for search range decisions.

where avg(.) means the average R-D cost function up to the $(k-1)$th MB. Because we have no knowledge of the future R-D gain, the future BW prediction is set to the remaining bandwidth budget divided by the remaining MBs within P_{update} that are not yet coded.

To fully utilize the bandwidth budget in an R-D sense, as in (9), we use the future and past predicted bandwidth to determine an available bandwidth interval. In this case, the relationship and difference between the future BW prediction and the past BW prediction imply two possible bandwidth usage scenarios. If the future BW prediction is greater than the past BW prediction, this implies that less bandwidth had been allocated to the previous MBs and thus more bandwidth can be allocated to the next MB. In contrast, the other case implies that too much bandwidth had been allocated to the previous MBs, and hence, less bandwidth can be allocated to the next MB. Based on the above scenarios, we can define the lower bound $(\mathrm{BW}_{\mathrm{lower}})$ and upper bound $(\mathrm{BW}_{\mathrm{upper}})$ of the usage bandwidth for later bandwidth allocation, as in Fig. 2 and (12) and (13), shown at the bottom of the page, where the factors, 1/2 and 1/4 are selected empirically and are simple for hardware implementation due to their power-of-two nature.

Step 4: With the above available bandwidth bounds and R-D data, we can make a SR decision for the MB coding. The SR is determined according to three bandwidth modes shown in Fig. 2 and the bandwidth condition defined in Table I. In which, the mode BW_L denotes the low available bandwidth case because the average bandwidth usage of previous MBs falls outside the available bandwidth interval (BI) bounded by $\mathrm{BW}_{\mathrm{upper}}$. Thus, future SR should be reduced to avoid an overflow of bandwidth usage. However, if the average bandwidth usage of the previous MBs falls within the available bandwidth interval as denoted by mode BW_N or outside the available bandwidth interval bounded by $\mathrm{BW}_{\mathrm{lower}}$ as denoted by mode BW_H, we have a sufficient bandwidth supply and can thus optimize the R-D performance by increasing the SR.

TABLE I
BANDWIDTH MODE OF ME

Mode	EXPLANATION
BW_H	Bandwidth-high mode
BW_N	Bandwidth-normal mode
BW_L	Bandwidth-low mode

Meanwhile, the algorithm further uses the motion vector information of the neighboring coded blocks to refine the SR. First, we obtain the adjacent motion vectors (MVs) from neighboring blocks related to the current block, such as MV_{U}, $\mathrm{MV}_{\mathrm{UR}}$, MV_{L}, and $\mathrm{MV}_{\mathrm{UL}}$ (the motion vector of the neighboring upper MB, upper right MB, left MB and upper left MB of the current coded MB as shown in Fig. 3). $\mathrm{MV}_{\mathrm{UL}}$ is used when $\mathrm{MV}_{\mathrm{UR}}$ is not available. Then, we compare these first three MVs $(\mathrm{MV}_{\mathrm{U}}, \mathrm{MV}_{\mathrm{UR}}, \mathrm{MV}_{\mathrm{L}},),$) and the summation of the absolute values of these three MVs $(\mathrm{sum\_mv})$. Afterward, we set the predictive SR $(\mathrm{Pred\_SR})$ as follows in (14), shown at the bottom of the page, by referring to this $\mathrm{sum\_mv}$ and the above bandwidth interval consideration. where the $\mathrm{SR}_{\mathrm{sys}}$ indicates the system SR, the factors $\mathrm{SR}_{\mathrm{shift\_low}}$, $\mathrm{SR}_{\mathrm{shift\_middle}}$, and $\mathrm{SR}_{\mathrm{shift\_high}}$ are set as spatial-resolution-independent parameters as in Table II. Finally, the SR is determined as

$$\mathrm{Final\_SR} = \min\left(\mathrm{SR}_{\mathrm{sys}},\ \max\left(\mathrm{Pred\_SR},\ \frac{1}{4}\times\ \mathrm{sum\_mv}\right)\right). \quad (15)$$

The system SR is used here as the upper bound of the allocated SR to avoid bandwidth usage overflow. This also brings one

$$\mathrm{BW}_{\mathrm{lower}}[k] = \begin{cases} \frac{1}{2}\times(\mathrm{BW}_{\mathrm{PP}}[k]+\mathrm{BW}_{\mathrm{FP}}[k]), & \text{if}\,\mathrm{BW}_{\mathrm{FP}}[k] > \mathrm{BW}_{\mathrm{PP}}[k] \\ (\mathrm{BW}_{\mathrm{FP}}[k]-\frac{1}{2}\times\mathrm{BW}_{\mathrm{PP}}[k]), & \text{otherwise} \end{cases} \quad (12)$$

$$\mathrm{BW}_{\mathrm{upper}}[k] = \begin{cases} (\mathrm{BW}_{\mathrm{FP}}[k]-\frac{1}{4}\times\mathrm{BW}_{\mathrm{PP}}[k]), & \text{if}\,\mathrm{BW}_{\mathrm{FP}}[k] > \mathrm{BW}_{\mathrm{PP}}[k] \\ \mathrm{BW}_{\mathrm{FP}}[k], & \text{otherwise} \end{cases} \quad (13)$$

$$\mathrm{Pred\_SR} = \begin{cases} \frac{(\frac{1}{2}\times\mathrm{SR}_{\mathrm{sys}})+2}{\mathrm{SR}_{\mathrm{shift\_low}}}, & \text{if}\,\mathrm{sum\_mv}=0 \\ \frac{(2\times\mathrm{SR}_{\mathrm{sys}})+4}{\mathrm{SR}_{\mathrm{shift\_middle}}}, & \text{if}\,0<\mathrm{sum\_mv}<3 \\ \frac{\mathrm{SR}_{\mathrm{sys}}+2}{\mathrm{SR}_{\mathrm{shift\_high}}}, & \text{if}\,(\mathrm{sum\_mv}\geq 3 \text{ and BI} \neq \text{Mode BW\_H}) \\ \mathrm{SR}_{\mathrm{sys}}+2, & \text{if}\,(\mathrm{sum\_mv}\geq 3 \text{ and BI} = \text{Mode BW\_H}) \end{cases} \quad (14)$$

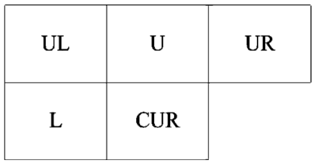

Fig. 3. Current block and its neighboring blocks.

TABLE II
FACTOR SETTINGS FOR PREDICTIVE SR

Bandwidth Interval	Experimental Factor		
	SR_{shift_low}	SR_{shift_middle}	SR_{shift_high}
Mode BW_H	2	4	0
Mode BW_N	2	4	2
Mode BW_L	4	8	4

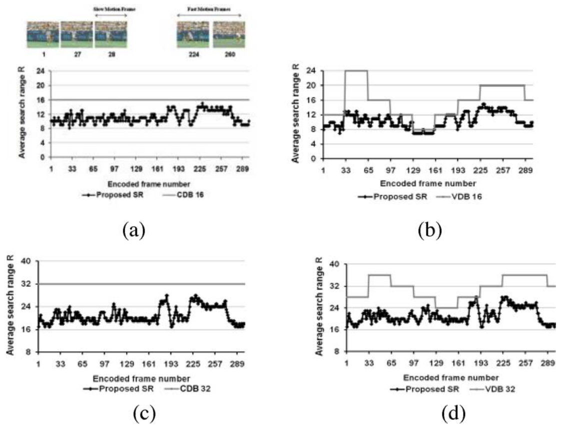

Fig. 4. Average SR comparison of the proposed algorithm and the full search algorithm for different cases: (a) CDB 16 (CIF size); (b) VDB 16 (CIF size); (c) CDB 32 (D1 size); (d) VDB 32 (D1 size).

additional advantage that adapts to sudden bandwidth change by setting new system SR.

C. Simulation Results

To evaluate the performance, we integrated our proposed algorithm into JM 12.2 [22] for simulation. The simulation conditions are CIF (352×288)-size and D1 (704×576)-size test sequences with the main profile, no R-D optimization, one reference frame, variable block size ME, full-search ME algorithm, IPPP sequences, 30 frames/s (fps), CABAC entropy coder, quantization parameters (QPs) of 18, 23, 28, 33, and P_{update} at 16. Note, the variable block size ME design adopts a tree-based computation, that is, the R-D cost of the larger block size is summed from the cost of the small block size. Thus, no extra memory bandwidth is required for the variable block size ME computation. Therefore, we will not include the memory bandwidth of different block sizes in the following results.

In the following simulations, we classify the corresponding bandwidth conditions into two patterns: constant data bandwidth (CDB) and variable data bandwidth (VDB). Both patterns provide the same amount of reference block data for the same search range $\pm R$. However, the CDB pattern will assume the available bandwidth is constant and fixed during ME operations, while the VDB pattern will assume the available bandwidth is variable during ME operations. Note that all the available data bandwidth will be consumed for the conventional full search algorithm in JM, but not in our approach. The CDB pattern is the scenario used in the traditional ME design without considering other components, while the VDB pattern is to simulate the bandwidth changing scenario due to situations like simultaneous coding and decoding in a video phone or different power modes for mobile applications. In the following simulations, we assume the SR for the full search in JM is $\pm\mathrm{R}$ for CDB R or VDB R case. For simulation purpose, we assume the bandwidth variation for encoding is due to simultaneously decoding a 300-frame video constructed by concatenating groups of 32 frames from several video sequences (in the order of Stefan, Akiyo, Stefan, Football, Stefan, triple Football, and double Akiyo). That is, the available bandwidth for ME is the overall system bandwidth (a constant value) subtracted by the bandwidth for decoding above video sequence. This video is to simulate fast scene changes for a memory bandwidth variation. To ease understanding the bandwidth size, the available bandwidth case will be termed as CDB R or VDB R for the search range $\pm\mathrm{R}$ case. For example, CDB 8 means that the available system bandwidth is constant during ME searches, and its total fetched data amount is equal to fetch the ± 8 search range data. VDB 8 means that the available system bandwidth is variable during ME searches, but its total fetched data amount is also equal to fetch the ± 8 search range data. Besides, the bandwidth used in the following simulation results will be converted as the average search range $\pm\mathrm{R}$ in a frame, which means the equivalent data amount to fetch the $\pm\mathrm{R}$ search range data in a frame.

Fig. 4 shows the average SR of our proposed algorithm for the Stefan sequence with CIF and D1 resolution. The results show that our algorithm can adapt to the available system bandwidth and better utilize the bandwidth. As shown in Fig. 4, the SR of our proposed algorithm becomes small in slow motion frames (frame 28) and large in fast motion frames (frame 224 to 260). Besides, when the available bandwidth becomes lower, the SR of our proposed algorithm will also follow the change. The SR selection is always lower than the system SR under limited-bandwidth constraints and thus avoids an overflow of bandwidth. In summary, the SR of our algorithm can adapt to the contents and available bandwidth.

Tables III and IV show the BD-PSNR, BD-Rate [23] and BW-saving for three CIF- and D1-size test sequences under different bandwidth conditions when compared with the full search algorithm. Our proposed algorithm can attain similar quality performance (BD-PSNR: -0.08 to 0.038 dB for CIF size video and -0.014 to 0.017 dB for D1 size video, BD-Rate: 1.86% to -0.562% for CIF size video and 0.33% to -0.374% for D1 size video) under different bandwidth supplies in the low-motion sequence (Akiyo sequence), the medium-motion sequence (Foreman sequence) and the high-motion sequence (Stefan sequence) but with less bandwidth. In the case of the low-motion sequence, the proposed algorithm can save 40% to 58% of the bandwidth under different SRs. For the medium-motion sequence, our algorithm can reduce the bandwidth by 26% to

TABLE III
PERFORMANCE COMPARISON OF BD-PSNR, BD-RATE, AND BW-SAVING FOR CIF RESOLUTION VIDEOS

BW Pattern	BD-PSNR (dB)			BD-Rate (%)			BW-Saving (%)		
	Akiyo	Foreman	Stefan	Akiyo	Foreman	Stefan	Akiyo	Foreman	Stefan
CDB 8	-0.001	-0.080	+0.038	+0.026	+1.860	-0.562	40.4	26.3	21.4
CDB 16	-0.011	-0.017	-0.013	+0.302	+0.402	+0.196	53	38.5	30.3
CDB 24	+0.003	-0.002	-0.011	-0.067	+0.043	+0.299	58.3	44.8	37.2
VDB 8	-0.003	-0.065	-0.023	+0.082	+1.551	+0.318	41.7	29.2	25.1
VDB 16	-0.007	-0.019	-0.024	+0.206	+0.456	+0.383	54	40.5	33.9
VDB 24	-0.001	-0.006	-0.080	+0.042	+0.138	+1.233	58.6	46.1	38.9

TABLE IV
PERFORMANCE COMPARISON OF BD-PSNR, BD-RATE, AND BW-SAVING FOR D1 RESOLUTION VIDEOS

BW Pattern	BD-PSNR (dB)			BD-Rate (%)			BW-Saving (%)		
	Akiyo	Foreman	Stefan	Akiyo	Foreman	Stefan	Akiyo	Foreman	Stefan
CDB 32	-0.003	-0.012	-0.006	+0.054	+0.333	+0.120	66.7	50.4	41.1
CDB 48	-0.007	-0.003	+0.012	+0.271	+0.073	-0.263	70	55.7	47.9
CDB 64	+0.005	-0.008	+0.014	-0.171	+0.210	-0.282	71.2	58.2	52.2
VDB 32	-0.002	-0.014	-0.002	-0.001	+0.357	+0.031	66.7	50.1	41.6
VDB 48	-0.002	-0.005	+0.017	+0.037	+0.127	-0.374	70.2	55.8	48.2
VDB 64	+0.007	-0.008	+0.014	-0.212	+0.211	-0.296	71.8	58.3	52.3

TABLE V
EFFECT OF UPDATE PERIOD UNDER THE VDB 16 CASE FOR CIF RESOLUTION VIDEOS

Update Period	BD-PSNR (dB)			BD-Rate (%)			BW-Saving (%)		
	Akiyo	Foreman	Stefan	Akiyo	Foreman	Stefan	Akiyo	Foreman	Stefan
1 frame	-0.007	-0.013	0.009	0.202	0.329	-0.114	51.2	42.2	33.5
16 frames	-0.007	-0.019	-0.024	0.206	0.456	0.383	54	40.5	33.9

46%. For the high-motion sequence (Stefan sequence), our algorithm can reduce the bandwidth by 21% to 38%. Similar observations were also found for the D1 resolution video, as shown in Table IV. Table V shows the effect of different P_{update}. The performance and savings are similar for P_{update} at one frame and 16 frames. Similar observations can also be found for other size videos. Thus, we choose P_{update} at 16 in this paper for the simulations. Meanwhile, both JM and our approach use motion vector predictor (MVP) as the search center. Thus, the overlapped region of search area between successive MBs is not regular and the overlapped region is accessed again in the simulations. Note that the bandwidth saving with reuse of overlapped region will be 0.3% more than that without reuse according to our simulation. Thus, for simplicity, we will not consider this reuse. In summary, these results show that our algorithm can better utilize the bandwidth for ME computation.

Another aim for real-time ME design is to complete data access and computation within a specified time slot. Without a loss of generality, we can assume that the computation time can be fully overlapped with the data access time in the bandwidth-limited case. For a practical scenario, we can assume the following case. Assume the available bandwidth is not enough to support the search range $\pm\mathrm{R}$ due to bus load went up when other devices access the bus frequently. In our case, we can adapt to the bandwidth change and meet the real time constraint. However, for the traditional algorithm, if X % bandwidth is not satisfied in above scenario, the access of search range data will need extra

$$\frac{1}{\left(1-\frac{\mathrm{X}}{100}\right)}-1\cong\frac{\mathrm{X}}{100}$$

time than the original planned constraint (approximated by Taylor series). For example, if 10% bandwidth is not satisfied, the data access will need extra 10% time, which delays the completion time of ME process by extra 10%. Larger variation will cause worse coding delay or frame drop, which could significantly degrade the video quality.

IV. HARDWARE ARCHITECTURE AND IMPLEMENTATION RESULTS

A. Proposed Bandwidth-Scalable ME Architecture

Fig. 5 shows the proposed ME architecture design that integrates the proposed bandwidth-scalable algorithm with a full-search ME engine as in Fig. 6. Fig. 7 shows the timing diagram. The whole operation basically follows the proposed algorithm. First, the SR of the current MB is decided by the *bandwidth scalable ME controller*. The data request of SR data and current MB are issued by the *pre-retrieval control unit* and these data are loaded from external memory by the *memory controller* and are respectively stored in the *reference pixel buffer* and *current*

TABLE VI
PERFORMANCE COMPARISON WITH STATE-OF-THE-ART LOW-POWER ME DESIGNS

Design	FS[1]+PT[2] [5]	FSS[3][16]	GDS[4] [7]	FS [6]	FFS[5] [8]	FS [Proposed]	
Resolution	QCIF@30fps	CIF@30fps	CIF@30fps	480p@30fps	D1@30fps	CIF@30fps	D1@30fps
Search Range	[-8, +7]	V[-16, +15],H[-32,+31]	[-16, +15.5]	[-16, +15]	[-16, +15]	[-24, +23]	[-64, +63]
Block Size	16x16 to 4x4	16x16 to 4x4	16x16 to 8x8	16x16 to 4x4	16x16 to 4x4	16x16 to 4x4	
Gate Count of ME (kilogates)	339.5	131.2	250	370	350	122[6]	
On-Chip Memory (kilobytes)	n.a.[7]	8	5	n.a.[7]	2.5	4.6	21.5
Average PSNR Loss (db)	0.4	n.a.[7]	0.1	0.002	n.a.[7]	0.003	
Operating Frequency (MHz)	1.4	13.5	13.5	233	80	7.5	29
Voltage (V)	1.2	1.8	1.2	1.2	1.8	1.2	
Process (μm)	0.13	0.13	0.13	0.13	0.18	0.13	
Bandwidth- Scalable Capability	No	No	No	No	No	Yes	
Power (mW)	1.33	16.72	12	95	55.2	4.28[8]	28.7[9]

[1]Full Search.
[2]Pixel Truncation.
[3]Four Step Search.
[4]Gradient Descent Search.
[5]Fast Full Search.
[6]122 = 98(gate count of ME) + 24 (gate count of bandwidth-scalable ME controller).
[7]Not available.
[8]4.28 = 4.13 (power dissipation of ME) + 0.15 (power dissipation of bandwidth-scalable ME controller).
[9]28.7 = 28.1 (power dissipation of ME) + 0.61 (power dissipation of bandwidth-scalable ME controller).

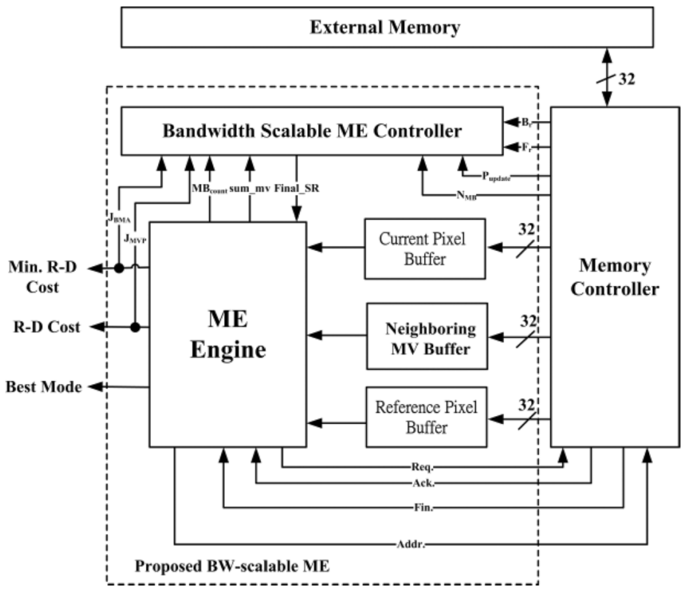

Fig. 5. Block diagram of the proposed bandwidth-scalable ME.

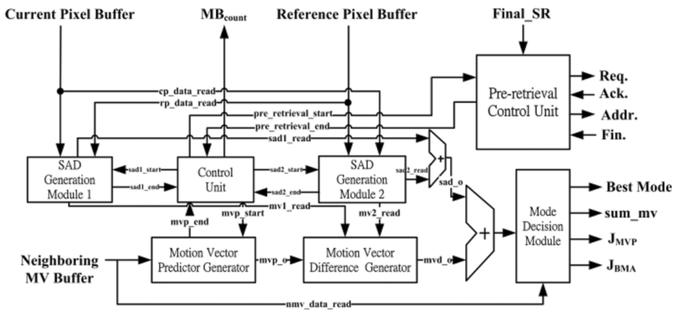

Fig. 6. Block diagram of our ME engine.

pixel buffer. Then these data are used to calculate SADs through the two *SAD generation modules* so that two search points can be done at each cycle. The final best mode is selected by the *mode decision module* and sent as output along with its motion vector difference. The corresponding R-D cost data like $\mathrm{J_{BMA}}$ and $\mathrm{J_{MVP}}$, the best MB mode, and the $\mathrm{sum\_mv}$ are sent back to the *bandwidth scalable ME controller* for SR decision of the next MB.

The details of the ME engine is described below. In Fig. 6, the control unit simultaneously produces control signals to the two *SAD generation module*, the *motion vector predictor generator*, and the *pre-retrieval control unit*. In addition, it needs to count the number of MBs $(\mathrm{MB_{count}})$ up to the current coded kth MB. Here, the handshaking policy between the *pre-retrieval control unit* and the *memory controller* is in the order of request-acknowledge-address-finish. The *pre-retrieval control unit* must send a *request* signal to the memory controller for the sake of

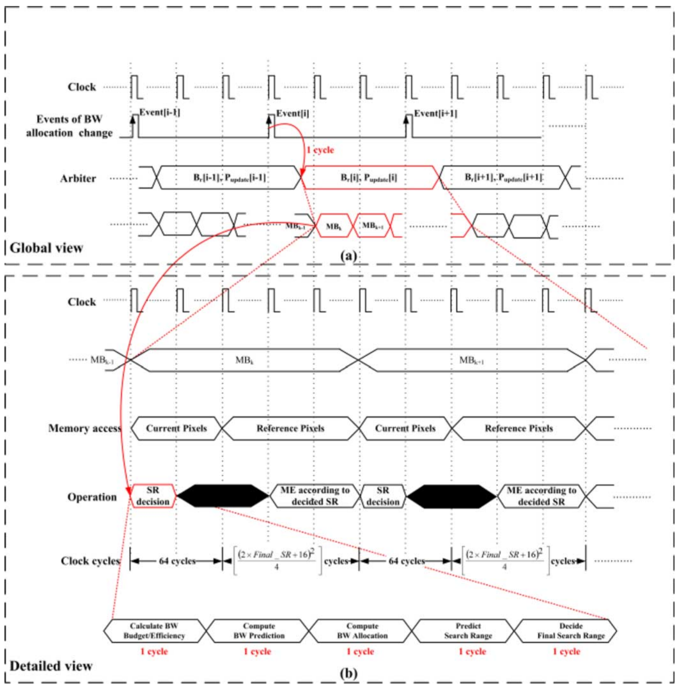

Fig. 7. (a) Timing constraints between bus load changes and bandwidth update period. (b) Timing diagram of the proposed bandwidth-scalable ME design.

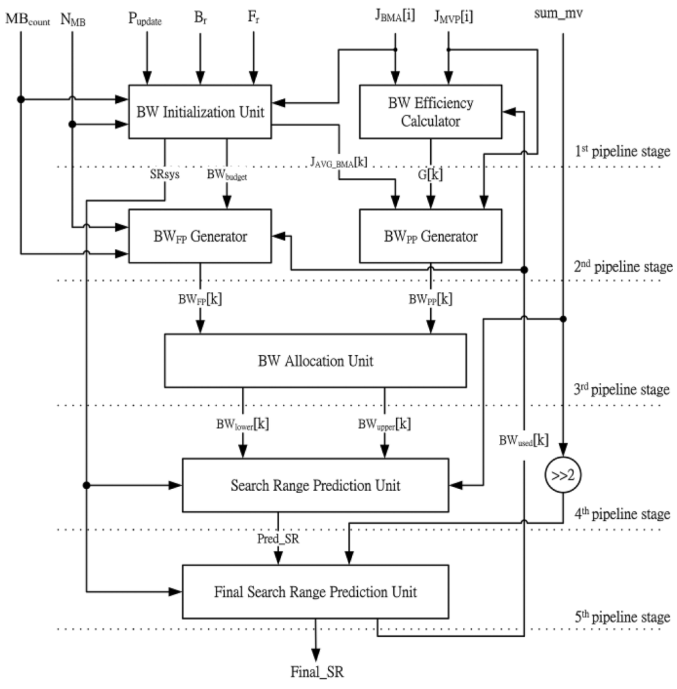

Fig. 8. Block diagram of the proposed bandwidth-scalable ME controller.

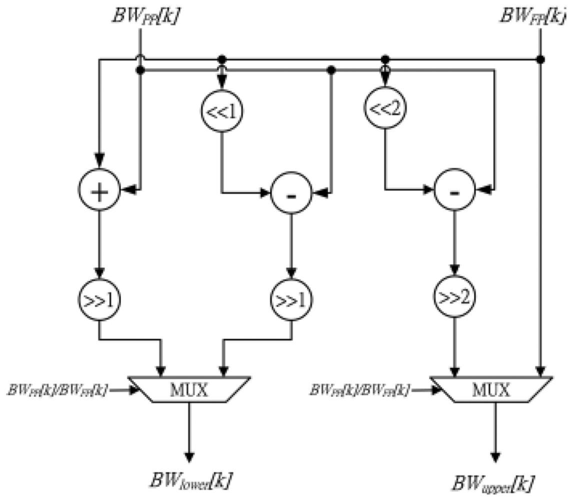

Fig. 9. Architecture of the bandwidth allocation unit.

data processing. If the memory controller is idle, it will be activated and respond with an *acknowledge* signal to the *pre-retrieval control unit*. When the *pre-retrieval control unit* receives the *acknowledge* signal, it generates a memory address for the *memory controller* to load the required reference pixels for the following step. After the task is finished, the *memory controller* sends a *finish* signal and waits for the next request.

The variable-block-size motion estimation (VBSME) is implemented by the SAD tree as other designs [24] but with four pipeline stages. The VBSME calculate all ME modes of a search position in a search area within a clock cycle. For a searching candidate, SAD costs of 4×4 blocks are first computed with the current block pixels and the reference area pixels to establish one set of SAD trees. Subsequently, all other SAD costs of large block sizes are calculated by immediately summing up the corresponding 4×4 costs. As a result, the matching costs of the smaller block sizes are reused by the larger block sizes.

The relation in bus load changes and bandwidth update period is as follows. First, we would like to point out that not every bus load change will affect the bandwidth update period. In the practical case such as mobile video device, the constraint is to meet the required encoding frame number per second. Therefore, the real constraint is the encoding time for one frame. Thus, if the bandwidth arbiter can guarantee to allocate the required bandwidth for one frame (not necessarily the bandwidth for default system search range), the arbiter can ignore the short term variation and keep the allocation to ME unchanged. This kind of arbiter can be found in modern memory controller or bus controller [21].

However, if the bandwidth variation is out of the guaranteed allocation, the arbiter shall issue the required change of bandwidth update period as illustrated in Fig. 7(a). When the event of BW allocation change occurs, the arbiter sends the new allocated data transmission rate (B_r) and BW update period (P_{update}) to the bandwidth-scalable ME controller. Then, the controller will calculate the new search range based on the proposed method after reference and current block pixels of current MB have been accessed.

The overall cycle for one MB ME execution is listed in Fig. 7(b) by assuming 32-bit memory access bus. The listed cycle count does not consider the DRAM memory latency since the operating frequency is quite low and these latency could be hidden under such case [25]. In this timing diagram, the most critical part is the memory access cycle, which depends on the selected SR. Note that we will start the ME computation as soon as the required reference data is available to hide ME computation cycles within the memory access cycles.

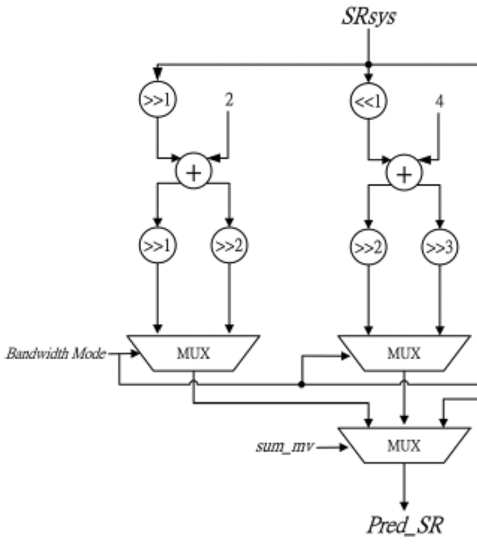

Fig. 10. Architecture of the search range prediction unit.

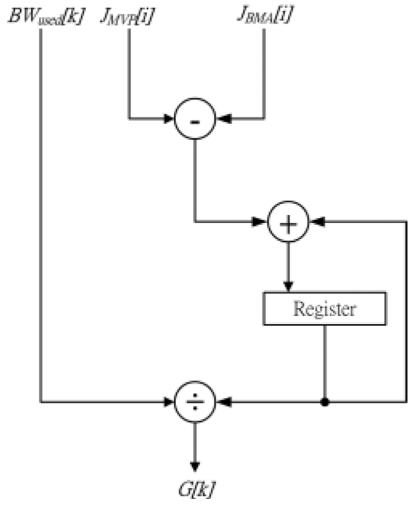

Fig. 11. Architecture of the BW efficiency calculator.

Fig. 8 depicts the detailed *bandwidth-scalable ME controller* to dynamically determine the SR as defined by (6), (8)–(15), and generate the memory address to the *pre-retrieval control unit* to load the required reference pixels for the following ME search. This architecture accepts R-D cost data of $\mathrm{J_{BMA}}$ and $\mathrm{J_{MVP}}$ from ME engine and system information from the memory controller to generate the appropriate SR for ME coding. The proposed system architecture has six major tasks, BW initialization as (6) and (8), BW efficiency calculation as (9), BW prediction as (10)–(11), BW allocation as (12)–(13), SR prediction as (14), and Final SR prediction as (15). These tasks are ordered as shown in Fig. 7 and take five cycles to complete. In the above algorithm, the most timing critical part is the two division operations, $J_{\mathrm{AVG\_BMA}}[k]$ and $\mathrm{BW_{PP}}[k]$, in (10) calculation. To avoid long delay due to two successive divisions, we move the $J_{\mathrm{AVG\_BMA}}[k]$ calculation to the first pipeline stage since its data is already available at that time slot. All the six major tasks are pipelined into five stages to reduce the critical path delay. Figs. 9–11 show the details of some modules, a *bandwidth allocation unit*, a *SR prediction unit*, and a *BW efficiency calculator*, respectively. These three modules are implemented with only shifters, subtracters, adders, comparators, and one divider for low hardware cost.

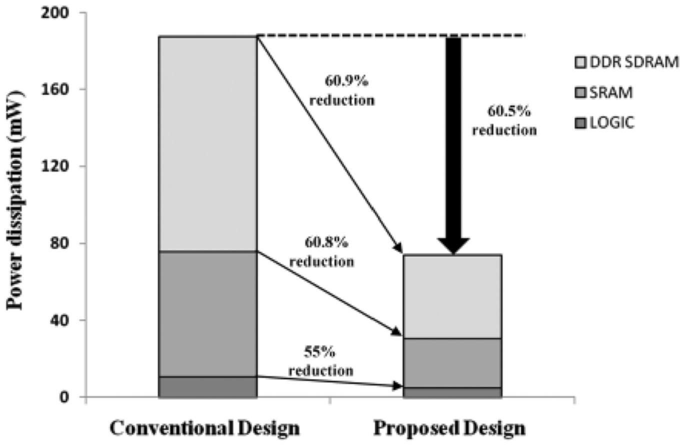

Fig. 12. Average power savings of the proposed design for a D1-size format video sequence @ 30 fps.

B. Implementation Results

The proposed bandwidth-scalable ME was implemented in Verilog and synthesized by TSMC 0.13-μm CMOS technology. Various video test sequences were adopted to verify the hardware functionality and obtain the estimated power consumption. For 30-fps CIF video with GOP of IPPP, the working frequency is 7.5 MHz and the power consumption is 4.28 mW, as measured, respectively, by Synopsys PrimeTime and PrimePower. Meanwhile, for 30-fps D1 video with GOP of IPPP, the working frequency is 29 MHz and the power consumption is 28.7 mW.

Table VI shows a comparison of the proposed ME with state-of-the-art designs [5], [6]–[8], [16]. Compared with other designs, the proposed design shows the advantages of reduced power consumption and bandwidth-scalable capability with similar or lower hardware cost.

Note that, with a lower external bandwidth, the power consumption by the external DRAM is also significantly reduced, which is usually a significant portion of the power consumption (50%–80%) in a video system [26]. Fig. 12 shows the power distribution in our ME design and in conventional work, adopting the mobile low-power DDR SDRAM model [27], [28] as an example (i.e., MT46H8M32LF with $\mathrm{VDD} = 1.8$ V, $\mathrm{I_{DD}4R} = 140$ mA, $\mathrm{I_{DD}3N} = 16$ mA, and ${}^{t}\mathrm{CK} = 10$ ns). Our design can reduce the power by 60.5% when compared to the conventional design with the same 29 MHz working frequency, same full search algorithm, same search range constraint ± 64, and same MB-based DRAM data mapping [25]. In summary, the power reduction is attributed to the bandwidth-scalable engine, which uses lower data access to the external memory (lower DRAM power), less computation and lower reference data access with dynamic search range (lower logic power and SRAM power).

V. Conclusion

This paper has presented an efficient bandwidth-scalable ME design for bandwidth-limited mobile video coding. The bandwidth efficiency was co-optimized at the algorithm and architecture levels. Experimental results showed that the proposed algorithm can utilize a bandwidth very close to the target system bandwidth, while its coding efficiency is similar to that in JM. As a result, the final design can provide superior power efficiency under CIF and D1 30-frames/s video coding as compared to the previous state-of-the-art designs. The presented approach can be combined with other low-power approaches to further reduce the overall power. Besides, further extending this approach to multiple reference frame ME is easy by considering the R-D gain for each reference frame into the bandwidth efficiency formula, which will be studied in our future work.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaad, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–575, Jul. 2003.

[2] 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.

[3] H. F. Ates and Y. Altunbasak, "Rate-distortion and complexity optimized motion estimation for H.264 video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, pp. 159–171, Feb. 2008.

[4] S. Lee, "Fast motion estimation based on search range adjustment and matching point decimation," *IET Image Process.*, vol. 4, no. 1, pp. 1–10, 2010.

[5] A. Bahari, T. Arslan, and A. T. Erdogan, "Low-power H.264 video compression architectures for mobile communication," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 9, pp. 1251–1261, Sep. 2009.

[6] M. G. Koziri, A. N. Dadaliaris, G. I. Stamoulis, and I. X. Katsavounidis, "A novel low-power motion estimation design for H.264," in *Proc. IEEE Int. Conf. Appl. Specific Syst. Arch. Processors*, 2007, pp. 247–252.

[7] 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.

[8] P. Li and H. Tang, "A low-power VLSI implementation for variable block size motion estimation in H.264/AVC," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 2972–2975.

[9] T. C. Chen, Y. H. Chen, C. Y. Tsai, S. F. Tsai, S. Y. Chien, and L. G. Chen, "2.8 to 67.2 mw low-power and power-aware H.264 encoder for mobile applications," in *Proc. IEEE Symp. VLSI Circuits*, 2007, pp. 222–223.

[10] Y. H. Chen, T. C. Chen, C. Y. Tsai, S. F. Tsai, and L. G. Chen, "Algorithm and architecture design of power-oriented H.264/AVC baseline profile encoder for portable devices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 8, pp. 1118–1128, Aug. 2009.

[11] C. J. Lian, S. Y. Chien, C. P. Lin, P. C. Tseng, and L. G. Chen, "Power-aware multimedia: Concepts and design perspectives," *IEEE Circuits Syst. Mag.*, vol. 7, no. 2, pp. 26–34, 2007.

[12] X. Z. Xu and Y. He, "Modification of dynamic search range for JVT," presented at the JVT-Q088, Nice, France, 2005.

[13] Z. Liu, J. Zhou, S. Goto, and T. Ikenaga, "Motion estimation optimization for H.264/AVC using source image edge features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 8, pp. 1095–1107, Aug. 2009.

[14] 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.

[15] C. Y. Tsai, C. H. Chung, Y. H. Chen, T. C. Chen, and L. G. Chen, "Low power cache algorithm and architecture design for fast motion estimation in H.264/AVC encoder system," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2007, pp. II-97–II-100.

[16] 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.

[17] W. Y. Chen, L. F. Ding, P. K. Tsung, and L. G. Chen, "Algorithm and architecture design of cache system for motion estimation in high definition H.264/AVC," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2008, pp. 2193–2196.

[18] S. H. Wang, S. H. Tai, and T. H. 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.

[19] H. Shim and C. M. Kyung, "Selective search area reuse algorithm for low external memory access motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1044–1050, Jul. 2009.

[20] C. Y. Kao and Y. L. Lin, "A memory-efficient and highly parallel architecture for variable block size integer motion estimation in H.264/AVC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 6, pp. 866–874, Jun. 2010.

[21] K. B. Lee, T. C. Lin, and C. W. Jen, "An efficient quality-aware memory controller for multimedia platform SOC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 620–633, May 2005.

[22] Fraunhofer Institute for Telecommunications, Germany, "Joint Video Team Reference Software JM12.2," ITU-T, 2007. [Online]. Available: http://iphome.hhi.de/suehring/tml/download/

[23] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," presented at the 13th Meet. VCEG-M33, Austin, TX, 2001.

[24] Y. K. Lin, C. C. Lin, T. Y. Kuo, and T. S. Chang, "A hardware-efficient H.264/AVC motion-estimation design for high-definition video," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 6, pp. 1526–1535, Jul. 2008.

[25] 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.*, 2007, pp. 505–508.

[26] K. Danckaert, K. Masselos, F. Catthoor, H. J. D. Man, and C. Goutis, "Strategy for power-efficient design of parallel systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 2, pp. 258–265, Jun. 1999.

[27] "Calculating memory system power for DDR SDRAM," *DesignLine J. Micron Technol. Inc.*, vol. 10, no. 2, pp. 1–12, 2001.

[28] Micron Technology Inc., USA, "Mobile LPDRAM," 2011. [Online]. Available: http://www.micron.com/parts/dram/mobile-ddr-sdram/mt46h8m32lfb5-6?pc={77C6853B-2F4D-4E62-A2F8-7B42387D1533}

Jui-Hung Hsieh received the B.S. degree from the Department of Electronics Engineering, I-Shou University, Kaohsiung, Taiwan, in 2000, and the M.S. degree from the Department of Electrical Engineering, National Chung-Cheng University, Chiayi, Taiwan, in 2002. 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 the video signal processing and VLSI architecture design.

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 Processor 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.