Journal of Signal Processing Systems 50, 69–80, 2008

DOI: 10.1007/s11265-007-0115-0

Design of an H.264/AVC Decoder with Memory Hierarchy and Line-Pixel-Lookahead

TSU-MING LIU AND CHEN-YI LEE
Department of Electronics Engineering and SoC Research Center, National Chiao-Tung University, 1001, Ta Hsueh Road, Hsinchu, 300 Taiwan, Republic of China

Received: 27 March 2007; Revised: 13 May 2007; Accepted: 13 June 2007

Abstract. This paper describes a novel memory hierarchy and line-pixel-lookahead (LPL) for an H.264/AVC video decoder. The memory system is the bottleneck of most video processors, particularly in the newly announced H.264/AVC. This is because it utilizes the neighboring pixels to create a reliable predictor, leading to a dependency on a long past history of data. This problem can be resolved by allocating memory space but inducing large silicon area and power consumption as well. We first review the existing solutions and propose a three-level memory hierarchy with line-pixel-lookahead to improve access efficiency. Three-level memory hierarchy includes registers, content/slice SRAM and external frame DRAM. We emphasize the need to consider the secondary hierarchy, content/slice SRAM, during the design of an H.264/AVC decoder. Specifically, we introduce a slice SRAM and line-pixel-lookahead to lower the memory capacity and external bandwidth. This SRAM stores neighboring pixels and prevents the data re-access from DRAM. Line-pixel-lookahead exploits multi-dimensional pixel locality so as to averagely improve prediction performance by 6.54% compared to conventional vertical prediction. Simulation results also reveal that the proposal makes a better trade-off between memory allocation and external bandwidth as well as power, leading to 50% of memory power reduction compared to the design without exploiting the secondary slice SRAM hierarchy.

Keywords: H.264/AVC, memory hierarchy, lookahead, prediction

1. Introduction

While there has been much work studying memory performance for scientific and general-purpose applications, this paper focuses on the need of H.264/AVC [1] video applications. H.264/AVC achieves high compression ratio since it adequately utilizes the neighboring pixels to obtain a reliable predictor and reduces the prediction errors. Compared to prevalent MPEG-x and H.26x video standards, H.264/AVC [1] decodes present pixel from a long history of pixel data and therefore requires much intermediate storage for VLSI implementation. Therefore, this high data correlation or dependency leads to a great challenge of memory subsystem in designing multimedia systems [2–4].

Many memory-hierarchy-based designs of H.264/AVC have been reported of the time [5–10]. However, they usually developed a bandwidth and/or memory capacity-starved design approaches without taking into account memory allocation and data locality issues. In this paper, we first review existing memory hierarchies in video processing and microprocessor systems. We found that three-level memory hierarchy

This work was supported by the National Science Council of Taiwan, R.O.C. under Grant NSC94-2215-E-009-046, and by the NCTU-MTK Research Program.

is generally accepted in existing H.264/AVC systems and composed of registers, content SRAM and frame DRAM. Since the disparity between registers and DRAM hierarchy, SRAM hierarchy design plays an increasing role and will dominate the system area or power requirements. Hence, this paper pays more attention on this level of the hierarchy. The emphasis on SRAM is also apparent from Fig. 1 which shows the die photo of the MPEG-2/H.264 video decoding system in [4]. Most of the usable area is dedicated to the on-chip SRAM and data buffers which occupy 40% and 70% of system area and power dissipation, respectively. As a result, this observation leads us to extend the topic of memory hierarchy in the H.264/AVC video decoding system.

To improve the memory hierarchy in a video decoding system, we exploit three-level memory hierarchy with line-pixel-lookahead (LPL) schemes. In addition to the content SRAM in the secondary hierarchy, we additionally introduce a slice SRAM to pre-store the neighboring pixel to improve the access efficiency. On the other hand, H.264/AVC video standard [1] is characterized by a peculiar access locality since a high probability exists to access logically adjacent pixel in vertical direction. Because of this predictable data access pattern, a hypothesis-based lookahead scheme has been proposed to predict what data will be necessary well in advance, and thereby improve predictive miss rates [4]. In this paper, we further improve the prediction scheme by utilizing multi-dimensional features and incorporating a 4×3 TAG template. Therefore, the proposal can averagely reduce miss rates by 6.54% compared to the vertical prediction in [4].

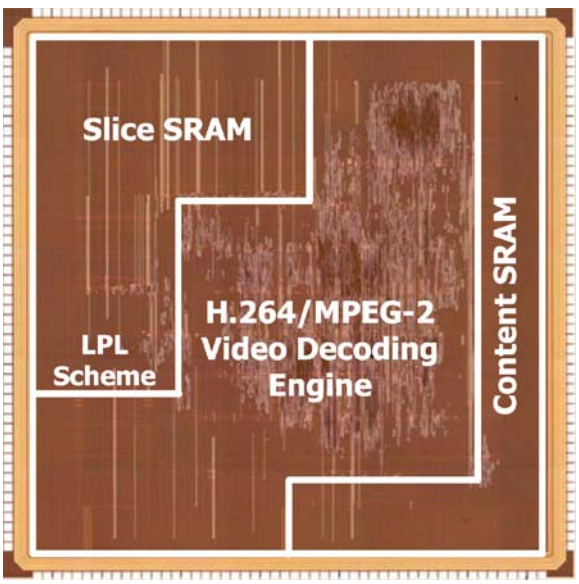


Figure 1. Memory area utilization of an MPEG-2/H.264 video decoder chip [4].

We proceed to integrate the proposed memory hierarchy with LPL scheme into H.264/AVC video systems. In general, the performance of prediction unit mainly relies on the prediction miss/hit. Moreover, the performance also impacts the memory size and external bandwidth. To make a better trade-off in different performance indices, we analyze and optimize the memory capacity and bandwidth in SRAM as well as DRAM hierarchies. Therefore, the optimized memory hierarchy with line-pixel-lookahead achieves 50% of memory power reduction compared to the design without exploiting the slice SRAM hierarchy. The remainder of this paper is structured as follows. Section 2 outlines a review of related works in the memory hierarchy of H.264/AVC. Sections 3 and 4 describe how data reuse can be exploited in the three-level memory hierarchy and line-pixel-lookahead (LPL) scheme, respectively. Simulation results are summarized in Section 5 and conclusions are made in Section 6.

2. Reviews of Memory Hierarchy in H.264/AVC

2.1. H.264/AVC Video Standards

We start with a brief overview of H.264/AVC video standard and illustrate why memory storage is required in practical VLSI implementation. In general, the intent of the H.264/AVC project was to create a standard that would be capable of providing good video quality at bit rates that are substantially lower than what previous standards would need (e.g., relative to MPEG-2, H.263, or MPEG-4 Part 2), and to do so without so much of an increase in complexity. The reduced bit rates come from some new techniques such as spatial prediction in intra-coding, variable block-size motion compensation, 4×4 integer transformation, context-adaptive entropy coding, and adaptive deblocking filter. Although those coding tools improve compressed performance, they suffer from dependencies on a long past history of pixel data. That is, present data will reference the previously decoded neighboring pixels or syntax elements. Consequently, previously decoded results should be stored into a certain amount of storage for

easily fetching in a short while. The storage will become voluminous especially in H.264/AVC video standard [1] and dominate the system area as well as power consumption.

To clarify aforementioned dependencies, Fig. 2 depicts an example of deblocking filter and spatial intra prediction in H.264/AVC. The input sequence is coded and thereby decoded in column, row and frame dimensions. A raster-scan order is performed by a 16×16 macroblock-wise manner in each image frame. In H.264/AVC, deblocking filter is exploited to efficiently compensate annoying blocking artifacts which is introduced by block-based prediction, transformation, and quantization. In the 16×16 macroblock boundaries, deblocking filter requires upper four pixels of each column to decide filtering modes and perform pixel-wise interpolations. Those upper pixels have been previously decoded and thereby required when decoding the current row of macroblocks. On the other hand, spatial intra prediction is a well-known method to predict the pixel value based on values previously coded in one frame. It needs upper neighboring pixel in each column to reconstruct the pixel results when vertical-like predictive modes (e.g. vertical, diagonal down–left, vertical–right, etc.) are applied. In addition to the deblocking filter and intra prediction, entropy coding, motion compensation and other coding tools feature long dependencies of historical data in either current or previous frames. In summary, a great deal of storage is required to solve the dependency problem in H.264/AVC and usually implemented via on-chip or off-chip memory.

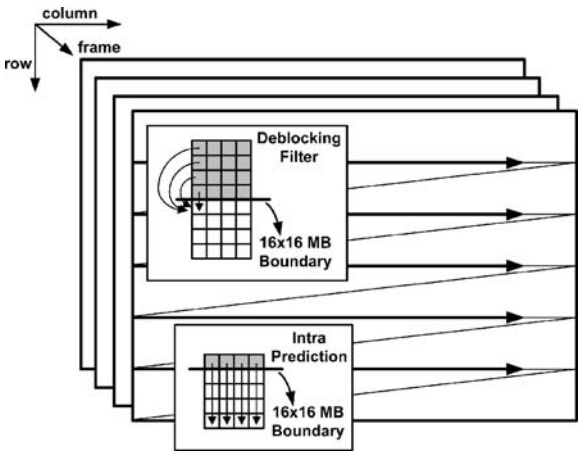


Figure 2. Correlation within one row of pixels over the input frame.

2.2. *Memory Hierarchy*

An architectural choice advocated for dealing with long past history of data is a memory hierarchy. The idea is that due to often huge volume of pixel data they cannot be permanently kept even on the disks but more inexpensive volatile storage capacity is satisfied. The current memory technology offers DRAM and SRAM memories to be used at lower levels of the hierarchy. In a video processing space, DRAM is widely adopted for storing whole frame data while SRAM stores a small amount of neighboring or decoded pixel data. DRAM acts as a higher level of the hierarchy since it is of lower speed and cost, and is of smaller size, than lower levels. Another reason for adopting memory hierarchy is data reuse exploration. Data reuse means that a data item, previously written to memory, is later read until it is finally consumed. It takes a local copy of pixel data to a smaller memory or register at the first time it is obtained. After that, the pixel data is read from the smaller memory instead of larger memory, and speed or power performance can then be improved [13].

A wide variety of memory hierarchies have been designed and implemented to deal with the high performance memory requirements. This design concept originates in general-purpose microprocessor but can be applied to application-specific H.264/AVC video processors. Figure 3a shows a five-level memory hierarchy in microprocessor systems. They mainly include registers, SRAM, DRAM and disk hardware blocks for instruction and data storage. Meanwhile, many H.264/AVC video processors [5–10], have been reported of the time and adopt three-level memory hierarchy to keep the pixel data in registers, SRAM and frame DRAM. As the disparity between registers and DRAM, SRAM hierarchy design plays an increasing role in memory performance [3]. Note that content and slice SRAMs in Fig. 3b are defined by Liu et al. [11] and will be thoroughly discussed in Section 3. Basically, SRAM hierarchies of existing solutions can be partitioned into two groups: w/t and w/o slice SRAM. Most designs [7–10] develops a bandwidth-hungry approach and only use content SRAM to construct the second-level of memory hierarchy. On the other hand, Liu [5] and Hu [6] additionally include slice SRAM (or row-store buffer in [6]) to facilitate the memory accesses. Those SRAMs cache the decoded or neighboring pixels that are frequently used in next

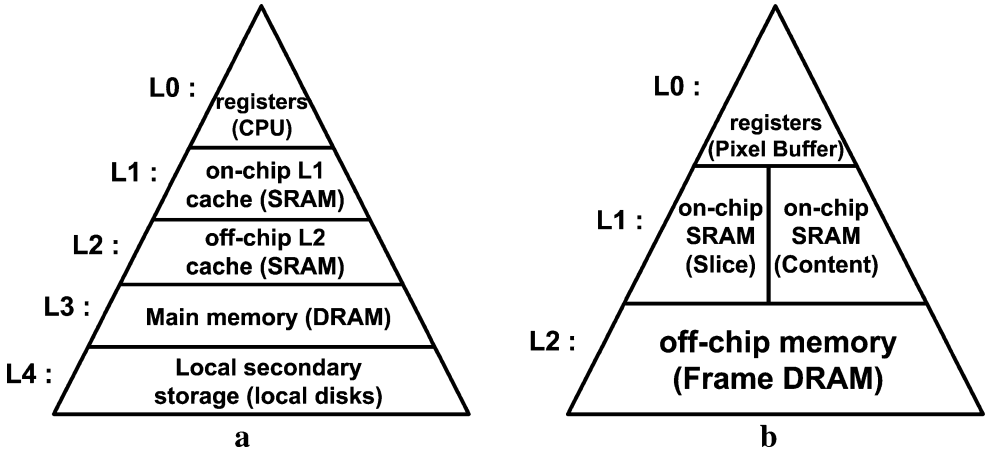


Figure 3. Memory hierarchy in **a** general-purpose microprocessors and **b** application-specific H.264/AVC.

time for increasing the data reuse probability and improving memory performance.

3. Three-Level Memory Hierarchy

Improving the memory hierarchy or reducing the embedded SRAM size is very effective for lowering the external bandwidth and achieving low power dissipation. To improve the memory performance, we aim at a memory hierarchy where copies of data from larger memories that exhibit high data-correlation are stored to additional layers of smaller memories. In this way, the great part of data accesses is moved to smaller memories and the significant bandwidth reduction and power savings can be achieved since accesses to smaller level of memory hierarchy are less power consumed [13]. Therefore, we propose a three-level memory hierarchy by leveraging registers, content/slice memory and frame DRAM as depicted in Fig. 3b. To clarify how it connects and completes data transaction, Fig. 4 shows detailed block diagrams and its bus structure. System and memory buses are a collection of parallel wires that carry address, data, and control signals. Data accesses in frame DRAM are accomplished via memory/system bus and I/O bridges. In an H.264/AVC decoder chip, the transactions between registers and ALUs (e.g. deblocking filter, motion compensation, etc.) frequently take place. As for the middle level of transactions, however, implementing content/slice SRAM is a more challenging task since they may impact the performance in not only registers but also frame DRAM. To limit the scope on memory hierarchy and focus on this upmost factor, we emphasize on allocations and arrangements of the SRAM hierarchy in the following sub-sections.

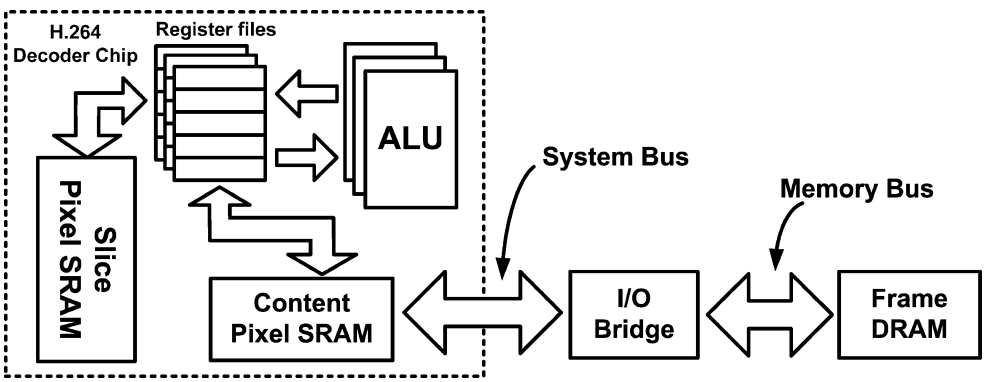


Figure 4. Memory hierarchy and bus structure.

3.1. Content SRAM

The goal of content SRAM is to allow data reading and writing simultaneously and accomplish data transactions without introducing bubbles or waiting cycles. However, the use of content SRAM may incur extra overhead of external bandwidth when storing neighboring pixels. While it has been extensively used in motion compensation, deblocking filter, intra prediction, etc., we illustrate this observation by using deblocking filter of H.264/AVC herein. It acts as either a single-port SRAM with ping-pong structure [11] or a dual-port SRAM with interleaving nature [12] to resolve structural hazard. Figure 5 depicts the memory organization with single-port ping-pong SRAM in deblocking filter. Particularly, the ping-pong SRAM coordinates the transaction between deblocking filter and other ALUs. It stores two macro-blocks (MBs) to resolve the structural hazard when reading and writing processes occur simultaneously. On the other hand, upper neighboring data labeled with gray region are previously decoded. If we exploit content SRAM to store neighboring pixels, additional transactions between external DRAM and content SRAM are required. Content SRAM have to be updated periodically from DRAM, and external bandwidth will increase definitely. To alleviate the aforementioned problem, slice SRAM replaces content SRAM to facilitate the access efficiency and will be addressed in the next sub-section.

3.2. Slice SRAM

Slice SRAM [5] or row store buffer [6] is designed to take advantage of neighboring pixel locality and acts as an add-on and/or secondary memory hierarchy. Considering a frame size of W×H in Fig. 6, each square represents the 16×16 macroblock (MB). Each MB contains 16 points and 4×4 pixels within each point. When following decoding procedures are performed from the MB index B to $B+1$, the pixel data in upper neighbors will be updated as the arrow indicates. The shaded region should be kept in a certain amount of memory storage when the decoding index is $B+1$. The reason has been addressed in Fig. 2 and we name those kept data as slice SRAM thereafter.

The overall data organizations of slice SRAM are tabulated in Table 1. It consists of pixel data or syntax elements per unit of one pixel, 4×4 sub-block and 16×16 MB. In the pixel data, the size of slice SRAM depends on the frame width W as the shaded region of Fig. 6 indicates. Moreover, the size of pixels relates to the chrominance format (C.F.[1]) and pixel depth. Here, a 4:2:0 format and 8bits/pixel is assumed for simplicity. Note that MB-level adaptation of frame/field (MBAFF) coding tools for H.264/AVC main/high profile additionally requires the

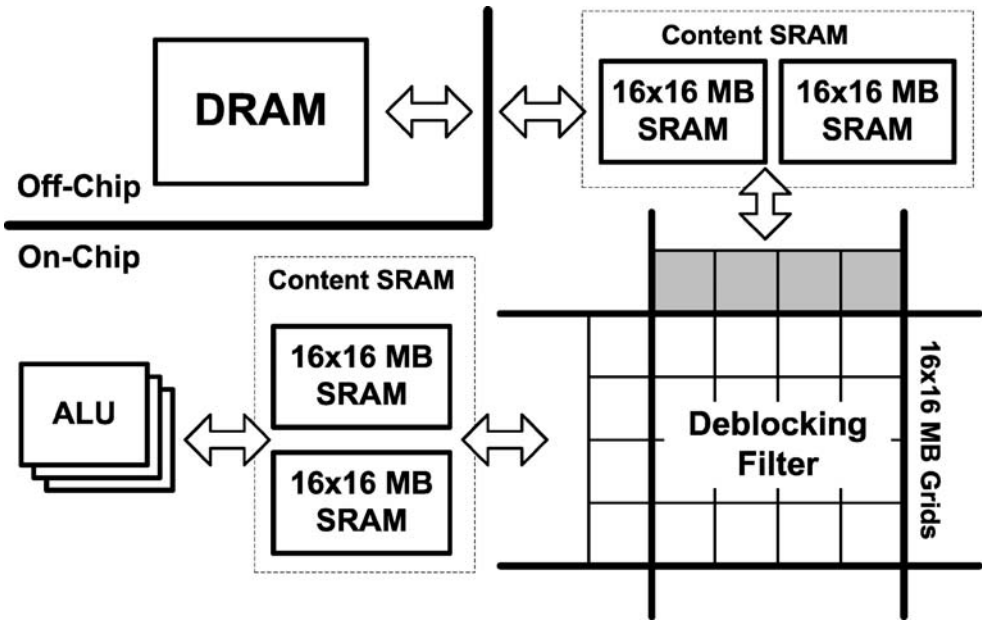


Figure 5. Memory organization of deblocking filter in H.264/AVC.

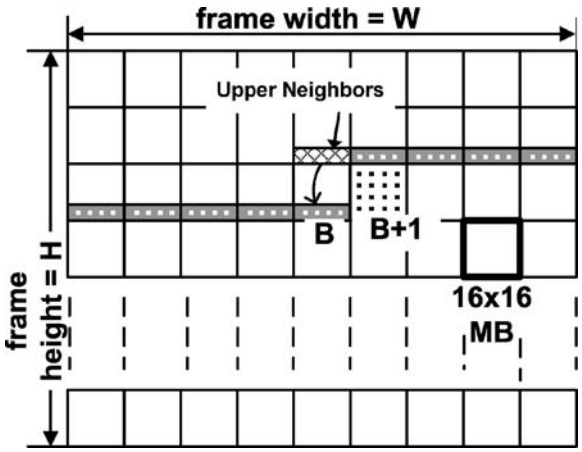


Figure 6. Data correlation of upper neighboring pixels in H.264/AVC.

storage space for the same field parity issue [17]. As for syntax elements, it is required keeping the flags since the decoding procedures of H.264/AVC will reference the previously decoded neighboring flags. For instance, motion vector will be generated from the previously decoded one. Besides, H.264/AVC develops adaptive entropy coding methods to achieve high compression ratio. In particular, CAVLC adaptively selects coding table by *nC* calculated from the *Coeff_Tokens* of neighboring blocks. CABAC adaptively estimates a large number of conditional probabilities through the neighboring syntax flags such as *MB_Type*, *CBP*, etc. These adaptive schemes lead to the high dependency and extra data storage for hardware implementation. In summary, the total memory size achieves 338.2 Kb under the H.264/AVC video decoding with 1080 HD resolution and 4:2:0 chrominance formats.

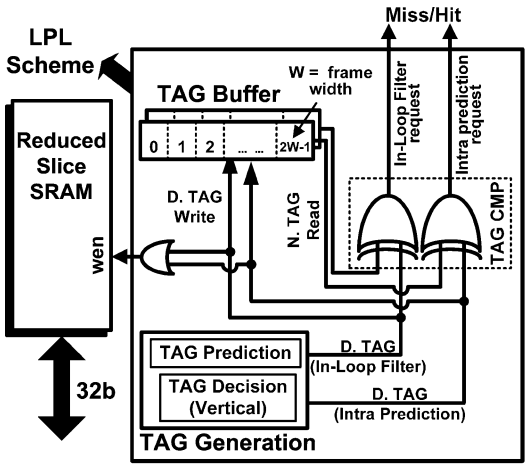
As slice SRAM stores one row of upper neighboring pixels, data transaction between DRAM and SRAM can be eliminated and therefore bandwidth requirements can be lowered. Compared to content SRAM, however, the introduced slice memory space is proportional to decoded frame width *W* and is of great challenge in memory testing, power and area issues. It may degrade the overall system performance, especially in high resolution video coding/decoding. To deal with large slice SRAM, we find that storing all pixels in rows of upper pixels is unnecessary when the following decoding process is unrelated to the upper neighboring pixels. Therefore, we further investigate line-pixel-lookahead to reduce the memory capacity in the next section.

4. Line-Pixel-Lookahead

Line-Pixel-Lookahead (LPL) scheme exploits spatial pixel locality in vertical direction and looks ahead before decoding the next line of pixels in order to reduce the slice memory capacity and improve the access efficiency. Figure 7 depicts the LPL scheme and the related pseudo codes [14]. In particular, a reduced slice SRAM caches the pixels of upper neighbors, and an LPL scheme predicts whether the follow-up pixel data should be kept or not. In the scope of this LPL scheme, we only focus on the pixel-level of storage content since it occupies a great portion of overall slice memory space in Table 1.

Table 1. Data organizations in slice SRAM.

Level	Content	Size Equation (bits)	1080HD@4:2:0 (Kb)
Pixel	Upper neighboring pixels for deblocking filter	4×W×C.F.×8	122.8
	Upper neighboring pixels of the same parity for deblocking filter in MBAFF	4×W×C.F.×8	122.8
	Upper neighboring pixels for intra prediction	W×C.F.×8	30.7
	Upper neighboring pixels of the same parity for intra prediction in MBAFF	W×C.F.×8	30.7 Kb
4×4 sub-block	Upper neighboring flags for motion vectors	2×(W/4)×10	9.6
	Upper neighboring *nCs* for CAVLC	(W/4)×5	2.4
16×16 macroblock	Upper neighboring *CBP*, *MB_Type* for CABAC	(W/16)×10×16	19.2
Total			338.2




```
for(i=0;i<W;i++)                                   0
{ Step1: Generate decoding tag and buffered.       1
  D. TAG(i) = F(pre_mode, bS);                     2
  Step2: write the D. TAG into Buffer              3
         and Slice memory.    }                    4
  Step3: when encountering the next line of        5
         pixel.                                    6
for(j=0;j<W;j++)                                   7
{ Step4: reading the N. TAG data from TAG          8
         buffer.                                   9
  N. TAG(j) = (if requiring upper                  10
              neighbor) ? 1:0;                     11
  Step5: comparing the TAG value.                  12
  if(N. TAG(j) == D. TAG(j)) req. = 0; (hit)       13
  else  req. = 1; (i.e. miss)                      14
}                                                  15
```

b

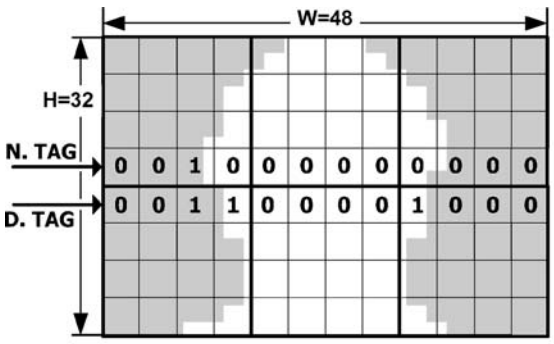
Figure 7. The **a** data flow and **b** related pseudo-code in the LPL scheme.

Therefore, only deblocking filter and intra prediction will be considered herein. In the LPL scheme, the TAG generation unit issues a decoding TAG (*D. TAG*) that contains a pair of signals for the purpose of deblocking filter and intra prediction units, and the *D. TAG* equals the neighboring TAG (*N. TAG*) after buffering one row of TAGs. Two 2W-bit TAG buffers record each *D. TAG*, where *W* means frame width. A TAG CMP (compare) unit perceives the contrast between *N. TAG* and *D. TAG*. A prediction miss will be noticed via a request signal from the output of TAG CMP when current *D. TAG* differs from *N. TAG*.

Figure 8a describes the 4×4 intra prediction behavior of the LPL scheme through an example with a frame size of 48×32. Each square represents a 4×4 sub-block labeled by a 1-bit TAG signal. In the *N. TAG* field, we tag the 4×4 pixel data decided by TAG generation unit and the tagged data will be pre-stored in slice memory for follow-up decoding procedures. Furthermore, the un-tagged pixel will be discarded via *wen* [see the reduced slice SRAM in Fig. 7a], resulting in reducing memory size. Hence, according to results in *D. TAGs* and *N. TAGs*, Fig. 8b lists the prediction miss/hit analysis. An additional cycle penalty is introduced when prediction miss occurs and *N. TAG* equals zero. We have to fetch upper neighboring pixels from frame DRAM via system and memory buses. Other case in prediction miss is of cycle-penalty free thanks to the un-used upper pixels in *D. TAG*.

4.1. TAG Generation

A crucial factor to making a success in the LPL scheme is TAG generation since it adversely impacts



N. TAG	D. TAG	miss /hit	Cycle penalty
0	0	hit	N
0	1	miss	Y
1	0	miss	N
1	1	hit	N

b

Figure 8. **a** TAG prediction and **b** miss/hit analysis.

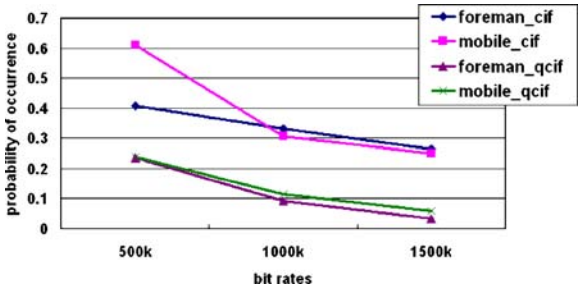
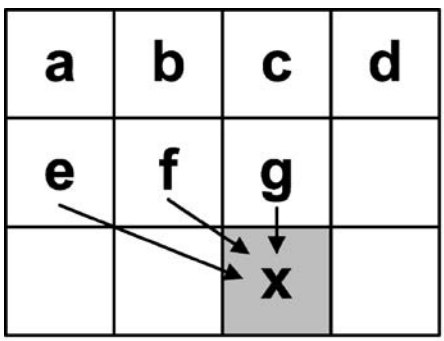


Figure 9. The block probability of occurrence without exploiting upper neighbors.

the prediction miss/hit, leading to the extra cycle penalty or external bandwidth. Although [14] presents a 1-tap vertical prediction to realize the LPL scheme as Fig. 7 illustrates, the introduced miss rates are still considerable. To further investigate the prediction method, we propose several TAG prediction modes to improve the pixel data locality.

To illustrate how the TAG generation method works, it can be partitioned into TAG prediction and decision steps. The prediction step produces the TAG signal which forecasts data accesses needed in advance, so a specific piece of data is pre-stored in the SRAM before it is actually desired by the follow-up decoding processes. A key observation is that not all upper neighboring pixels need to be pre-stored for the decoded blocks when they are determined as a "near-horizontal prediction mode" in intra prediction or a "SKIP mode" in deblocking filter [14]. To clarify aforementioned blocks occurring in a typical video sequence, Fig. 9 shows the average probability of block occurrence in different sequences and bit rates. The probability means the number of block events without exploiting upper neighbors over one row of blocks. Hence, there are higher probabilities to eliminate the upper pixels of slice SRAM in both high resolution and low bit rate. On the other hand, after generating the TAG signal, a decision step decides the prediction mode by incorporating a 4×3 TAG template in Fig. 10a. Each square represents the TAG signal as Fig. 8a indicates. The decoding orders are based on raster scan labeled from a to g. At the decoding index g, we have to predict the TAG value of index x based on previously decided TAGs (i.e., $a\sim g$). Therefore, we develop a decision table in Fig. 10b. In addition to a vertical TAG prediction, several diagonal predictors are involved as well. *Maj.* indicates the majority of TAG indices e, f, and g. Compared to original 1-tap vertical prediction [14], the proposal realizes 2-tap multi-dimensional prediction to improve pixel locality but is two times larger than TAG storage capacity of the 1-tap one.

The goal of improving TAG generation is to lower miss rate, resulting in the reduction of external bandwidth. The miss rate stands for a probability of missing events and is equal to the number of miss over one row of TAGs. Existing designs [7–10] only exploited content SRAM to store neighboring pixel in one MB and therefore update pixel data frequently via system/memory bus. Hence, those designs contribute 100% of miss rates. On the other hand, traditional slice memory [5] and row store buffer [6] without exploiting LPL scheme keep whole pixels in one row of MB, and the miss rate can be eliminated (i.e. 0%). Thanks to the LPL scheme, the proposal provides different design space with variable miss rates and can be further optimized through multi-dimensional prediction. To clarify the performance between vertical-based [4] and the proposed prediction, Table 2 exhibits the miss rate reduction over different kinds of video bit-rates, resolutions and test sequences. Note that slice SRAM size is proportional



a

condition	prediction
a == f	x == f
a == g	x == e
c == g	x == g
misc.	x == maj.(e,f,g)

b

Figure 10. **a** TAG prediction template and **b** decision table.

Table 2. Miss-rate reduction of proposed multi-dimensional prediction

		Miss Rate Analysis					
		QCIF			CIF		
Sequence	Bit Rate (kbps)	Vertical	Multi-dimensional	Reduction (%)	Vertical	Multi-dimensional	Reduction (%)
Mobile calendar	500	0.4102	0.4047	1.32	0.3321	0.2952	11.12
	1,000	0.2075	0.188	9.37	0.3858	0.3452	10.54
	1500	0.1089	0.0938	13.87	0.4834	0.4917	−1.72
Suzie	500	0.1837	0.1747	4.86	0.1544	0.1584	−2.63
	1,000	0.1138	0.1069	6.09	0.1537	0.1537	0.00
	1,500	0.0948	0.0926	2.39	0.1567	0.154	1.66
Foreman	500	0.358	0.353	1.30	0.4597	0.4638	−1.28
	1,000	0.215	0.192	10.88	0.416	0.4223	−1.52
	1,500	0.1172	0.0963	17.8	0.3624	0.3643	−0.53
Table tennis	500	0.2087	0.2102	−0.73	0.3498	0.3554	−1.60
	1,000	0.1197	0.1132	5.40	0.3182	0.322	−1.17
	1500	0.0935	0.088	5.92	0.271	0.27	0.17
Average			6.54			1.08	

to width W when the scaling factor f is fixed at one in [4]. Moreover, this proposal achieves better performance in lower resolution case. As a result, 6.54 and 1.08% of miss rate reduction can be achieved averagely under QCIF and CIF resolutions, respectively.

4.2. *Slice SRAM with LPL vs. Cache Memory*

After introducing the slice SRAM with line-pixel-lookahead, we review and summarize the differences between slice and cache memory in Table 3. Specifically, memory hierarchies between Fig. 3a and Fig. 3b are a little bit different in the secondary level. A major difference is the transaction with neighboring levels of hierarchies. For instance, cache memory accomplishes data transaction with not only lower-level but higher-level of hierarchies. The proposed slice SRAM with LPL is an add-on to content SRAM and dedicated to lower-level of the register hierarchy. Consequently, the cache memory supports different write policies (e.g., write back/through) while slice SRAM has been written via *wen*. Moreover, the content of cache memory comes from higher-level of the external main memory. The data in slice SRAM are produced from raw bitstream in the lower-level of the register hierarchy. As for the data locality, the proposed SRAM with LPL stores pixel data and therefore features high correlation for natural scenes. In summary, we present a novel hierarchy which is somewhat different from preliminary hierarchy in microprocessor and discuss the features of slice SRAM with LPL from a memory-hierarchy's perspective.

Table 3. Differences in middle-level of memory hierarchies

Items	Cache Memory	Slice SRAM with LPL
Applications	General-purpose microprocessor	Application-specific H.264/AVC
Position	Acts as a buffer between main memory and registers	Dedicated to registers
Write policy	Write back/through	Written by *wen* from TAG generation
Data type	Stores a copy of some part of the main memory	Stores a copy of pixel data generated from bit-streams of the register hierarchy.
Data locality	Spatial/temporal locality	High locality for natural scenes

5. Performance Evaluation on H.264/AVC Decoding Systems

To clarify and analyze the memory power performance, we describe the power modeling where the

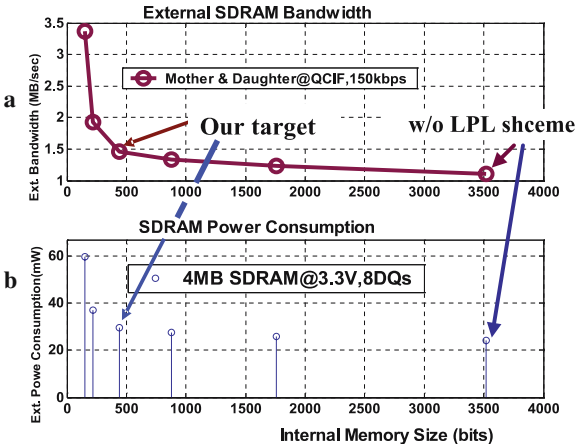


Figure 11. Analysis of **a** external bandwidth and **b** power consumption.

power consumption of memory hierarchy can be viewed as a summation of on-chip and off-chip memories in Eq. (1). Further, memory accesses are the most power consuming operation [15]. The access power dominates the overall power budget in the on-chip memory. The power consumed on memory accesses is a function of on-chip memory size, the access frequency and technology etc. In this paper, we assume that the on-chip power is closely related to the SRAM capacity (i.e. *word-length* × *# of words*) [16]. Therefore, the simplified power described by Eq. (2) suffices for the purpose of evaluating the power effect on memory hierarchy. As for off-chip memory, the DRAM power modeling becomes more complicated. Not only data access but also I/O and background power (e.g. pre-charge, active etc) should be concerned in the power calculation of off-chip memory. We just choose a system-power calculator [19] as the off-chip power model in a preliminary design phase.

$$P_{\text{total}} = P_{\text{on-chip}} + P_{\text{off-chip}} = P_{\text{access}} + \left(P_{\text{access}} + P_{\text{I/O}} + P_{\text{BG}}\right) \quad (1)$$

$$P_{\text{on-chip}} = P_{\text{access}} = f_{\text{access}} \times F(\text{Length}_{\text{word}}, \#\text{ of word}, V_{\text{dd}}) \quad (2)$$

Slice SRAM size can be reduced since we eliminate un-used neighboring data if LPL prediction hits. However, because an error of prediction may occur, an additional penalty is introduced to re-fetch the missed data from external memory, leading to the increment of external memory bandwidth as well as power consumption. Therefore, it is more imperative to decide a suitable memory size to be survived since inappropriate memory size will be harmful to memory performance. From the experiments, we choose CAS latency=2, BL=4 and ^{t}CK=7ns as our DRAM model configuration [18]. We use "Mother & Daughter" (QCIF) as our test sequence and encode it at 150 kbps and 15 fps for mobile applications. An observation is that the curve between internal SRAM size and external memory bandwidth/power is shown in Fig. 11. The external bandwidth can be derived from miss rates [14] while power consumption on DRAM is calculated by system-power calculator [19]. Compared to a design point without exploiting LPL scheme, we optimize the memory hierarchy which has smaller SRAM capacity and lower external DRAM bandwidth/power. Moreover, we also optimize the on-chip power consumption where SRAM power is related to the memory size in Eq. (2).

As compared to the existing designs, we improve the access efficiency by taking into accounts both internal SRAM and external DRAM. Figure 12 describes the power comparison from a memory hierarchy's perspective. For overall system power indices, they can be found on [4] in more details. The overall memory power consumption consumes only 55.5% of original power when the proposed three-level memory hierarchy is applied. Moreover, we consider the access efficiency by exploiting the spatial locality of neighboring pixel via the LPL scheme. The power consumption can be further reduced to 49.3% of original design. This success

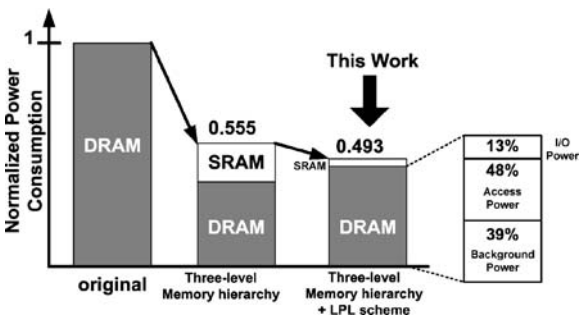


Figure 12. Power reduction via three-level memory hierarchy and line-pixel-lookahead scheme.

to power reduction is that we choose a better design point from the observation in Fig. 11. Hence, the SRAM power can be greatly reduced with a penalty of slight power increment on DRAM. However, access power occupies a great portion of total DRAM power. The reason is that motion compensation required a great deal of accesses from external memory. In the future, it can be optimized by the same way in intra prediction and deblocking filter. Note that the metric of DRAM's power consumption can be further optimized through well-known low-power techniques. In our implementation, we just choose a preliminary DRAM module to verify the power performance in a system point of view.

On the other hand, we also verify the proposed memory hierarchy with LPL on a test chip which has been published in [4] and re-shown in Fig. 1. This test chip is fabricated using Artisan 0.18 μm standard CMOS cell library with UMC 0.18 μm 1P6M technology. The maximum working frequency is 100 MHz and the die size is 3.9×3.9 mm^2. The slice SRAM is positioned on the left-top corner of this chip. The LPL scheme is interfaced to the embedded SRAM for improving access efficiency. Its gate count approximately reaches 22.3 K and only occupies 4% of overall system area.

6. Conclusion

In this paper, a review and design of memory hierarchy on H.264/AVC video decoding systems is presented. Three-level memory hierarchy, registers/content SRAM/DRAM, without keeping neighboring pixel is widely used but incurs large external DRAM bandwidth as well as power dissipation. To alleviate this problem, a new slice SRAM is introduced to store upper neighboring pixel for reducing the access frequency on external DRAM. To efficiently manipulate the slice SRAM, a line-pixel-lookahead (LPL) unit is proposed to improve the stored pixel locality in SRAM. Compared to conventional vertical prediction [14], we develop a multi-dimensional prediction by incorporating a 4×3 TAG prediction template, leading to 6.54% of miss rate improvement in QCIF resolution. Overall, the proposed three-level memory hierarchy with LPL greatly enhances and optimizes the size and data in slice SRAM hierarchies. Therefore, 50% of internal/external memory power can be saved compared to existing designs [7–10] without exploiting the slice SRAM hierarchy. And it is believed that this improved memory hierarchy is a key to making memory-rich video system feasible for low memory area/power mobile requirements.

Acknowledgment

The authors would like to thank the anonymous reviewers for many valuable comments to improve this paper's quality. They also acknowledge several fruitful discussions with Wen-Ping Lee in National Chiao-Tung University.

Note

1. C.F. means chrominance format: {4:4:4, 4:2:2, 4:2:0}→ C.F. = {3, 2, 2}

References

1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), May. 2003.
2. S. Dutta, W. Wolf and A. Wolfe, "A Methodology to Evaluate Memory Architecture Design Tradeoffs for Video Signal Processors," *IEEE Trans. Circuits Syst. Video Technol*, vol. 8, no. 1, 1998, pp. 36–53, Feb.
3. S.T. Fu, D.F. Zucker and M.J. Flynn, "Memory Hierarchy Synthesis of a Multimedia Embedded Processor," *IEEE International Conference on Computer Design: VLSI in Computers and Processors, ICCD '96*, pp. 176–184, Oct. 1996.
4. T.-M. Liu et al., "A 125 μW, Fully Scalable MPEG-2 and H.264/AVC Video Decoder for Mobile Applications," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, 2007, pp. 161–169, Jan.
5. T.-M. Liu et al., "An 865-μW H.264/AVC Video Decoder for Mobile Applications," *IEEE Asian Solid-State Circuits Conference*, 2005, pp. 301–304, Nov.
6. Y. Hu, A. Simpson, K. McAdoo and J. Cush, "A high definition H.264/AVC hardware video decoder core for multimedia SoC's," *IEEE International Symposium on Consumer Electronics*, 2004, pp. 385–289, Sept.
7. Y.-W. Huang et al., "A 1.3TOPS H.264/AVC Single-Chip Encoder for HDTV Applications," *ISSCC Digest of Technical Papers*, 2005, pp. 128–129, Feb.
8. H.-Y. Kang et al., "MPEG4 AVC/H.264 Decoder with Scalable Bus Architecture and Dual Memory Controller," *IEEE International Symposium on Circuits and Systems*, 2004, pp. II-145–II-148, May.
9. C.-C. Lin et al., "A 160kGate 4.5 kB SRAM H.264 Video Decoder for HDTV Applications," *ISSCC Digest of Technical Papers*, 2006, pp. 406–407, Feb.
10. S.-H. Wang et al. "A platform-based MPEG-4 advanced video coding (AVC) decoder with block level pipelining," *IEEE International Conference on Joint Conference*, 2003, pp. 15–18, Dec.

11. T.-M. Liu, W.-P. Lee, T.-A. Lin and C.-Y. Lee, "A Memory-Efficient-Deblocking Filter for H.264/AVC Video Coding," *IEEE International Symposium on Circuit and System (ISCAS'05)*, 2005, pp. 2140–2143, May.
12. Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-C. Wang, T.-H. Chang and L.-G. Chen, "Architecture Design for De-blocking Filter in H.264/JVT/AVC," *Proc. IEEE Intl. Conf. on Multimedia and Expo*., vol.1, 2003, pp. 693–696, July.
13. S. Wuytack, J.-P. Diguet, V.M. Francky Catthoor and H.J. De Man, "Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 6, no 4, 1998 pp. 529–537, Dec.
14. T.-M. Liu and C.-Y. Lee, "Memory-Hierarchy-Based Power Reduction for H.264/AVC Video Decoder," *IEEE International Symposium on VLSI International Symposium on Design, Automation and Test (VLSI-DAT'06)*, 2006, pp. 247–250, Apr.
15. J. M. Rabaey and M. Pedram, "Low Power Design Methodologies," *Kluwer Academic Publishers*, 1995.
16. N.D. Zervas, K. Masselo, O.G. Koufopavlou and C.E. Goutis, "Power Exploration of Multimedia Applications Realization on Embedded Cores," *IEEE Int. Symp. Circuits Syst*., vol. 4, 1999, pp. 378–381, June.
17. L. Wang, K. Panusopone, R. Gandhi, Y. Yu and A. Luthra, "Interlace coding tools for H.26L video coding", *VCEG-O37*, Pattaya, 2001, Dec.
18. Micron® Technology Inc. MT48LC2M32B2 64Mb SDRAM. [Online Available]: http://www.micron.com/products/dram/
19. Micron® Technology Inc. The Micron® System-Power Calculator: SDRAM. [Online Available]: http://www.micron.com/products/dram/syscalc.html

Tsu-Ming Liu was born in I-Lan, Taiwan, R.O.C. in 1980. He received the B.S. degree in Electronics Engineering from National Chiao-Tung University, Taiwan, in 2002, and the M.S. and Ph.D. degrees from National Chiao-Tung University, Taiwan, in 2004 and 2007, respectively. From July to October 2004, he was an intern in Sunplus Technology Company, Ltd, Hsinchu, Taiwan. From 2004 to 2006, he served as a Lecturer in the Tze-Chiang Foundation of Science and Technology (TCFST). In October 2007, he will join the MediaTek Inc., Hsinchu, Taiwan, R.O.C. In the past three years, he has authored and coauthored over 30 papers in academic journals and national/international conference proceedings. His major research interests include binary shape coding, joint source and channel design, H.264/AVC video decoding, and associated VLSI architectures. Dr. Liu received the Best Impact Award from IEEE Taipei Section. He received a Best Paper Candidate and Top 10% papers of the 2004 Asia-Pacific conference on Circuits and Systems and the 2005 International conference on Image Processing, respectively. He was a recipient of the internal Ph.D. candidate scholarship from MediaTek Inc., HsinChu, Taiwan, and he is an honorary member of Phi-Tau-Phi.

Chen-Yi Lee received the B.S. degree from National Chiao Tung University, Hsinchu Taiwan in 1982, and the M.S. and Ph.D. degrees from Katholieke University Leuven (KUL), Belgium in 1986 and 1990 respectively, all in Electrical Engineering. From 1986 to 1990, he was with IMEC/VSDM, working in the area of architecture synthesis for DSP. In February 1991, he joined the faculty of the Electronics Engineering Department, National Chiao Tung University, Hsinchu Taiwan, where he is currently a Professor. His research interests mainly include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of system-on-chip design technology, very low power designs, multimedia signal processing, and wireless communications. He served as the Director of Chip Implementation Center (CIC) from July 2000 to December 2003, an organization for IC design promotion in Taiwan. He was the former IEEE CAS Taipei Chapter Chair January 2000 to August 2002, the SIP task leader of National SoC Research Program January 2003 to December 2005, and the microelectronics program coordinator of Engineering Division under National Science Council of Taiwan December 2002 to December 2005. He also served as the Department Chair of Electronics Engineering, National Chiao Tung University August 2003 to July 2006. He is a Member of IEEE.