PERFORMANCE ANALYSIS OF AN H.263 VIDEO ENCODER FOR VIRAM

Thinh PQ Nguyen, Avideh Zakhor, and Kathy Yelick*
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA 94720
e-mail: { *thinhq, avz}@eecs.berkeley.edu*, *yelick@cs.berkeley.edu*

ABSTRACT

VIRAM (Vector Intelligent Random Access Memory) is a vector architecture processor with embedded memory, designed for portable multimedia processing devices. Its vector processing capability results in high performance multimedia processing, while embedded DRAM technology provides high memory bandwidth at low energy consumption. In this paper, we evaluate and compare performance of VIRAM to other digital signal processors (DSPs) and conventional SIMD (Single Instruction Multiple Data) media extensions in the context of video coding. In particular, we will examine motion estimation (ME) and discrete cosine transform (DCT) which have been shown to dominate typical video encoders such as H.263. We will show that VIRAM outperforms other architectures by 4.6x to 8.7x in computing ME and by 1.2x to 5.9x in computing DCT.

1. Introduction

Traditionally, video processing is performed by high-end workstations or specialized hardware such as ASICS. Recent popularity of portable and hand-held devices such as digital cameras and wireless videophones has created a need for hardware architectures designed for mobile, portable video processing applications [7]. Conventional microprocessors are not well suited to video processing because they are optimized for traditional applications with complex control flow. On other hand, the kernels of multimedia applications are often characterized by large amounts of data parallelism and high memory bandwidth [2]. For instance, standardized video codecs such as MPEG-4 and H.263 consist of motion estimation (ME) and discrete cosine transform (DCT), both requiring high memory bandwidth and involving large amounts of data parallelism. In this paper, we analyze the speed performance of an H.263 encoder on VIRAM (Vector Intelligent Random Access Memory), the vector microprocessor being designed at U.C. Berkeley. By integrating vector processing with embedded DRAM technology, VIRAM eliminates off-chip memory accesses, and therefore achieves high memory bandwidth at low power consumption. In addition, vector architecture uses less instruction decode overhead, and as such, VIRAM requires less area and power.

The H.263 is the ITU (International Telecommunication Union) recommended standard for very low bit rate video compression We choose to analyze the performance of H.263 video codec on VIRAM because it has lower computational and power requirements compared to other algorithms, and therefore is well suited for portable devices. Section 2 gives an overview of

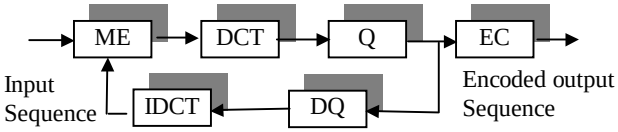

Figure 1. *Block diagram of H.263 encoder*

VIRAM architecture. Next, in Section 3 we characterize the time distribution of individual components of the H.263 encoder. In Section 4, we discuss the optimization of ME and DCT algorithms on VIRAM and the corresponding speed performance results. Finally, we conclude in Section 5.

2. Overview of VIRAM Architecture

VIRAM is a vector microprocessor designed for media processing with on-chip main memory [8]. Figure 2 shows the block diagram of VIRAM architecture. It contains a MIPS core scalar unit and a loosely coupled vector unit (VU). It is being designed using 0.18 µm embedded DRAM technology with target clock rate of 200MHz and 1.2V power supply. Since the processor and main memory are placed on the same chip, VIRAM can potentially increase memory bandwidth by 100 times as compared to conventional microprocessor systems [10]. The target power consumption for the vector unit and memory is 2 watts [8,3] which is suitable for some portable devices. We expect that an industry implementation of this chip would have

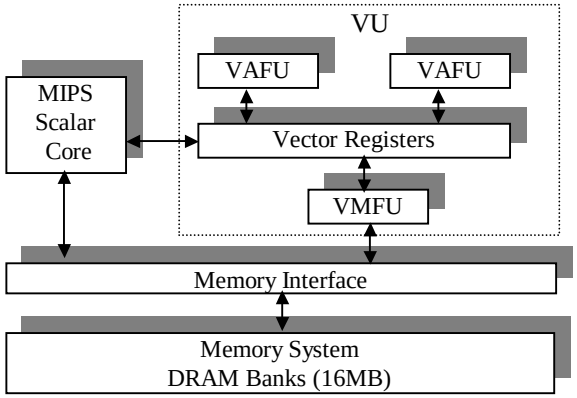

Figure 2. *Block diagram of VIRAM architecture*

* This work was supported by DARPA (DABT 63-96-C-0056), California State Micro Program. NSF (MIP-90-57466), NSF (CCR-9979442)

even lower power.

VIRAM has two vector arithmetic functional units (VAFU) and one vector memory functional unit (VMFU). Both VAFU and VMFU have four 64-bit vector data paths that can be used to perform eight 32-bit or sixteen 16-bit operations per cycle. Both VAFUs support integer, fixed point, logical operations but only one supports floating-point operations. VIRAM's peak performance is 6.4 GOPS for 16-bit data type, 3.2 GOPS for 32-bit data type, and 1.6 GOPS for 64-bit data type. Besides vector arithmetic and logical instructions, VIRAM also supports a wide range of scalar–vector instructions which have a scalar and a vector as operands. Most instructions are fully pipelined. The VMFU can load and store up to 256 bits per cycle. There are three types of vector memory accesses: *unit stride* which accesses contiguous memory locations, *strided* which accesses memory locations by a fixed offset, and *indexed* which accesses memory locations referenced by elements in a vector register. VIRAM's register file contains 32 vector registers and 32 scalar registers. Each vector register is 2048 bits long. VIRAM's memory system has 16 Mbytes of DRAM organized into eight banks. Finally, there is an I/O interface with 100MB/s parallel lines. Since VIRAM is not yet available, the performance results in this paper are based on a near cycle-accurate simulator that was developed at U.C. Berkeley [4].

3. H.263 Performance Characterization

To characterize the performance of the encoder, we use the H.263 version 2 written by Telenor, a popular public-domain implementation of H.263. Our test environment is a SGI machine running at 180MHz. We first optimize the H.263 encoder by changing many common functions into macros. We also replace the slow IDCT with the fast IDCT provided with the Telenor source code. During the measurements, we minimize the computational load of the machine by not running any other program. Our tests include four QCIF (176x144) standard H.263 test sequences: Akyio, Mom, Hall, and Foreman. Each contains 300 frames. We use quantization level Q = 10, and target frame rate of 10fps for all the test sequences. To measure the time spent on memory and arithmetic operations only, each test sequence is run from 3 to 5 times until the total time converges to a stable value. This method avoids the influence of disk accesses since all the data are already cached in the memory from the previous runs.

As shown in Table 1, ME and DCT dominate around 81% and 9.5% of the total encoding time, respectively. Since they are also highly vectorizable, we will optimize the H.263 encoder for VIRAM, based entirely on ME and DCT.

Sequence	ME	DCT + IDCT	Other	Total
Akiyo (12.95 kbit/s)	18765 (80.9%)	2306 (9.9%)	2130 (9.1%)	23201
Mom (16.25 kbit/s)	22446 (82.3%)	2508 (9.2%)	2310 (8.4%)	27264
Hall (20.47 kbit/s)	17745 (79.5%)	2282 (10.2%)	2300 (10.3%)	22327
Foreman (65.52 kbit/s)	27367 (82.8%)	2967 (9.0%)	2706 (8.2%)	33040

Table 1 *Time in ms for components of H.263*

4. Performance of H.263 on VIRAM

4.1 Motion Estimation

Motion estimation is used to exploit the inherent temporal redundancy of a video. In a typical motion estimation process, each frame of the video is divided into 16x16 macro-blocks. Given a macro-block in the current frame, the goal is to find the 16x16 region from the previously reconstructed frame that is most similar to it. In general, the similarity is defined based on the minimum value of Sum of Absolute Value (SAD) between the luminance pixel values of the two macroblocks:

$$SAD(x,y,k,l) = \sum_{j=0}^{15}\sum_{i=0}^{15} | F1(x+i, y+j) - F0(k+i, l+j) |$$

where (x,y) and (k,l) are the lower left-corner positions of the current macro-block and the 16x16 region from the previously reconstructed frame, respectively, and F_1, F_0 are the pixel luminance values from current frame and previously reconstructed frame.

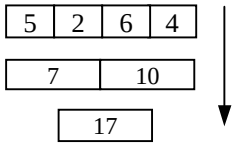

Figure 3. *Reduction instruction*

Since the calculation of SAD is a dominant operation during motion estimation, most DSPs and multimedia extensions vectorize the SAD routine. Vectorizing SAD requires the *reduction* operation which takes a vector and reduces to a scalar value by summing all the elements of the vector, as shown in Figure 3. Generally, *reduction* operation is expensive as it breaks the flow of the vector pipeline. Figure 4 shows a simple example of vectorized ME, using a 2x2 macro-block and 3x3 search area. If one places the macroblock on top of the search area as shown in Figure 4(a), and starts sliding from left to right and top to bottom, there are 4 possible positions for the macroblock. In each position, subtracting the corresponding elements of the macroblock and the search area

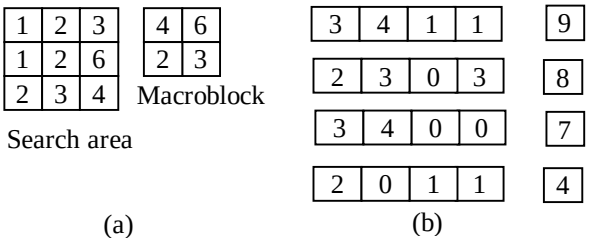

Figure 4. *(a) Search area and macroblock, (b) Vector registers containing the absolute differences*

block, and taking the absolute value of the subtraction, results in four 4-dimensional vectors as shown in Figure 4(b). In order to find out the minimum, we need to use four *reduction* operations to obtain four scalar values and choose the minimum value of the four, which in this case happens to be 4. Alternatively, we propose a different approach to implement exhaustive search algorithm on VIRAM that uses only one *min reduction* operation when applied to the same example. *Min reduction* operation

takes a vector and returns the smallest element in the vector. We first describe the algorithm then present the VIRAM architecture features that enable the algorithm.

The method is as follows. *Step one*: Subtract each entry in the macroblock from all possible overlapped entries of the search area, and take the absolute value of the subtraction. For example, the first entry (0,0) of the macroblock will overlap with entries (0,0), (0,1), (1,0), (1,1) of the search area, and the resulting vector is (3,2,3,2). Since the macroblock has 4 elements in this example, we will have 4 resulting vectors as shown in Figure 5. *Step two:* Add all resulting vectors to obtain vector (9,8,7,4). Each element in this vector is the SAD at a particular position in the search area. *Step three:* Use only one *min reduction* on the final vector to obtain the minimum SAD.

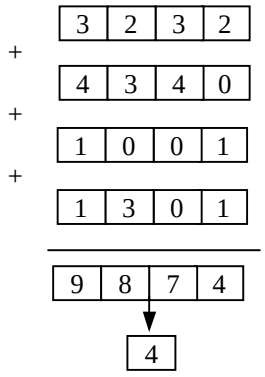

Figure 5. *Improved method of ME*

To implement the above algorithm efficiently, the scalar and vector subtraction instruction is needed to perform the *step one* in the example. Unlike other architectures such as MMX, VIRAM has a wide range of arithmetic scalar and vector instructions which take a scalar operand and a vector operand to produce a vector result. Next, to store many possible overlapped entries of the search area in a few vector registers to perform the subtraction in *step one*, the vector registers have to be long. While most existing architectures have 64 to 128 bits vector registers, and hence can only store 4 to 8 16-bit values, VIRAM's registers are 2048-bit long that can store up to 128 16-bit values. As such, two VIRAM vector registers are enough to hold 16x16 overlapped search area. In addition, VIRAM provides the *indexed load* instruction with autoincrement of the base register to read the overlapped search area which is usually a block in the image, into a vector register. Other DSPs such as the TriMedia [11] do not have the ability to load a block of the image into a vector register in one instruction. In *step 3*, VIRAM provides an efficient way to do *min reduction* on a N-dimensional vector with log(N) complexity.

Size	VIRAM-1	VIRAM-2	MMX
QCIF	$7.1x10^6$ (4.6x)	$3.9x10^6$ (8.4x)	$3.3x10^7$
CIF	$2.8x10^7$ (5.0x)	$1.6x10^7$ (8.7x)	$1.4x10^8$

Table 2. *Cycles/frame for the exhaustive search*

Tables 2 shows the performance of two versions of VIRAM vs. Pentium II MMX. The number in the parentheses represents the speed-up factor of VIRAM over MMX. On the Pentium MMX, the measured time is multiplied by the clock rate to obtain the number of cycles, and the measurement is done using SAD routine provided by Intel Corp [5]. On VIRAM, we use the cycle-accurate performance simulator. VIRAM-1 is the current design that can generate four addresses/cycle while VIRAM-2 design can generate eight addresses/cycle for *indexed loads*. Note that VIRAM-1 performance is much worse than VIRAM-2 because of the stalls in address generation units by *indexed loads*. Address generation stalls happen when address computations are not fast enough for the vector unit. Still, VIRAM-1 outperforms MMX by a factor of 4.6.

4.2 Discrete Cosine Transform

The discrete cosine transform maps the pixel values from spatial domain into the frequency domain for energy compression. A two-dimensional forward DCT of NxN pixels *f(x,y)* is given by:

$$F(u,v) = \frac{2}{N} k_u k_v \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

where $k_i = 1/\sqrt{2}$ for i = 0 and $k_i = 1$ otherwise.

There are several fast methods for computing the DCT. In this paper, we consider a fast DCT algorithm called LLM [9]. While the original LLM algorithm uses 11 multiplies and 29 additions, we implement the alternate LLM [9] which uses 12 multiplies and 32 additions. The advantage of this method is that no data path contains more than one multiplication. This allows a simple and accurate implementation in scaled fixed-point arithmetic, with a minimal number of shifts. We also use 32-bit data to comply with the MPEG standard. To compute 2-dimensional DCT, we first take a 1-dimensional DCT along the column, then take another 1-dimensional DCT along the row. To do the DCT along the column, we use *unit strided load* to read the first 8 rows of the image into 8 vector registers. We then apply the normal DCT operations across the vector registers as though they are scalar values, and use *unit strided store* to write back the results of column DCT. This process is repeated for all the rows of the image. Next, to take the DCT along the row, we apply the same method as above except using *strided load and store* to work with the columns of the image. Unlike other architectures, VIRAM has high memory bandwidth due to its embedded DRAM technology that allows efficient *strided load and stores* [8].

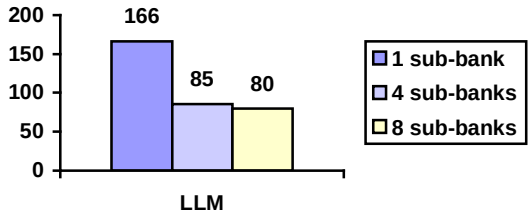

Figure 6. *Cycles/block with different numbers of sub-banks*

Our simulations show that the DCT performance of VIRAM degrades due to the stalls from address conflicts. Address conflicts happen when there is an access to the same memory bank while the previous access has not been completed. To avoid address conflicts, VIRAM memory banks is divided into many sub-banks. Figure 6 shows the average number of cycles for computing 2-dimensional DCT of an 8x8 block using QCIF image for LLM with different number of sub-banks. As

seen, the number of cycles is reduced by nearly a factor of 2, from 1 sub-bank design to 4 sub-banks design. However, we gain only 5 cycles going from 4 sub-banks design to 8 sub-banks design due to the fact that address conflict is no longer the bottleneck but the computation.

Table 3 shows number of cycles to compute the DCT of a 8x8 block on different architectures. The number in parentheses represents the speed-up factor of VIRAM versus other architectures. As seen, VIRAM outperforms both DSPs and SIMD multimedia extensions architectures by 1.2x to 5.88x in computing the DCT. VIRAM and V830 numbers comply with MPEG accuracy standard. For other numbers, compliance is not claimed, and hence computationally cheaper algorithms could have been used. We also obtain 59 cycles per 8x8 block for a less accurate DCT algorithm called AAN [1] using 16-bit data.

VIRAM (4 sub-banks, LLM)	85
TriMedia TM-1000 from Philips	160 (1.88x)
TI TMS320C62	230 (2.71x)
PowerPC with Alitvec	102 (1.20x)
HP PA-8000 with MAX2	147 (1.73x)
Intel Pentium II + MMX	500 (5.88x) [6]
NEC V 830/A	201 (2.36x)

Table 3. *DCT performance comparison of VIRAM vs. others*

4.3 Overall Performance

Currently, our cycle-accurate simulator only emulates vector unit of VIRAM. We do not yet have a scalar performance simulator. To measure the overall speed performance of the H.263 encoder which has both scalar and vector codes, we measure the time spent on the scalar code separately on a SGI machine, and the time spent on the vector code on the simulator. We then combine the results to get the overall performance of H.263. This method is reasonably accurate since VIRAM has the same scalar core as the SGI machine. Applying this technique to VIRAM with 4 sub-banks, we obtain the average achievable encoding frame rate for the test sequences in Table 4 using exhaustive search for motion estimation, and LLM for DCT. As we optimize the motion estimation and the DCT, the time spent on variable length coding (VLC) becomes significant. For example, the DCT and the motion estimation of the Foreman sequence take about 42 percent of the total time while VLC and other miscellaneous operations take the other 58 percent. At present time, we have not optimized the VLC for VIRAM. As vector processing speed increases, we anticipate VLC to become the bottleneck in a typical video encoder.

Akiyo (12.95 kbit/s)	Mom (16.25 kbit/s)	Hall (20.47 kbit/s)	Foreman (65.52 kbit/s)
23.5 fps	22.7fps	22.7fps	20.9fps

Table 4. *Average encoding speed for H.263 on VIRAM.*

As seen, the achievable encoding rates are high enough to result in acceptable quality for most multimedia and communication applications.

5. Summary

In this paper we presented an overview of VIRAM architecture, a vector microprocessor with embedded memory, optimized for multimedia applications. Although VIRAM is a general-purpose processor, its performance exceeds other DSP and multimedia extension architectures consistently by a factor of 4.6 to 8.7 in computing motion estimation and by 1.2 to 5.88 in computing discrete cosine transform. The improvement in the H.263 encoder performance is due to VIRAM's high memory bandwidth based on the embedded DRAM technology. With high memory bandwidth, VIRAM architecture is able to provide efficient *indexed* and *strided* memory operations that are well suited for memory access patterns of the discrete cosine transform and motion estimation. The long vector registers of VIRAM allow an efficient vectorized algorithm for the exhaustive search motion estimation. In addition, unlike existing multi-chip solutions, VIRAM is a one-chip solution, and as such, it results in lower power consumption, and smaller area.

6. References

[1] Arai, Agui, and Nakajima. Trans. IEICE E-71(11):1095

[2] K. Asanovic. *Vector Microprocessors*. PhD thesis, Computer Science Division, UCB, 1998.

[3] R. Fromm, et al. "The energy efficiency of IRAM architectures." ISCA, pages 327-337, June 1997.

[4] R. Fromm, et al. "Vector IRAM Memory Performance For Image Access Patterns". Technical Report UCB//CSD-99-1067. University of California – Berkeley-Oct 1999

[5] Intel Corp. "Using MMX TM Instructions to Compute the Absolute Difference in Motion Estimation", http://developer.intel.com/drg/mmx/AppNotes/ap530.htm

[6] Intel Corp. "JPEG Inverse DCT and Dequantization Optimized for Pentium II Processor." Online Document: http://developer.intel.com/drg/pentiumII/appnotes/886.htm

[7] C.E Kozyrakis and D.A. Patterson. "A New Direction in Computer Architecture Research. " IEEE Computer, 31(11):24-32, November 1998

[8] C.E Kozyrakis. "A media-enhanced vector architecture for embedded memory systems." Technical Report UCB//CSD-99-1059, UCB, July 1999

[9] C. Loeffler, et al. "Practical Fast 1-D DCT Algorithms with 11 Multiplications", ICASSP '89, pp. 988-991.

[10] D. Patterson, et al. "A Case for Intelligent DRAM". IEEE Micro, April 1997

[11] TM1000 data book, Philips Inc.