MPEG - 2 Video Decoder for DVD

Nien-Tsu Wang
Computer Engr. Dept.
Santa Clara University
Santa Clara, CA 95053,
U.S.A.
nwang @ scudc.scu.edu

Chen-Wei Shih
Electrical Engr. Dept.
Santa Clara University
Santa Clara, CA 95053,
U.S.A.
cshih @ scudc.scu.edu

Duan Juat Wong-Ho
School of EEE
Nanyang Tech. Univ.
Singapore 639798
Singapore
ehdjwong @ ntu.edu.sg

Nam Ling
Computer Engr. Dept.
Santa Clara University
Santa Clara, CA 95053,
U.S.A.
nling @ scuacc.scu.edu

***Abstract* – A video decoder with an efficient controller scheme and a sub-picture decoder for DVD application is presented in this paper. Most of the reported architecture for MPEG2 video decoding uses a 64 bit bus and a complex bus arbitration scheme. Our design uses synchronous DRAMs instead of standard EDO DRAMs and involves a novel controller scheme that allocates bus space for DRAM access efficiently. This efficient allocation allows us to reduce bus width from 64 bits to 32 bits, without significantly increasing embedded buffer sizes, and still meeting the requirements for MPEG2 MP@ML decoding. The bus arbitration algorithm is also simple allowing for a less complex controller design. Our main strategy is to impose a certain order in the DRAM access by the various processes instead of allowing any process to request for bus access arbitrarily. We also take advantage of the restricted GOP(group of picture) sequence in the DVD format to allow a longer decoding time for B frames. The sub-picture pixel data are run-length compressed bitmaps that are overlayed on top of the MPEG reconstruction video. The architecture for sub-picture decoding is simple and easy to implement.**

1. Introduction

The improved storage capacity in DVD (Digital Versatile Disc) finds wide applications in both the computer as well as the consumer electronics industries. As DVD is mainly an application for the low cost consumer market it is important that its architecture be efficient and low cost. A limited version of MPEG-2 Main Profile at Main Level (MP @ML) [1] is used in the DVD format.

Our low cost MPEG2 decoding system includes a high performance single chip MPEG2 decoder and the associated DRAM buffer. Most of the reported architectures [2], [5], [6] use a 64 bit bus and a bus arbitration unit that uses priority assignment and polling to resolve conflicts on the bus. We propose the use of synchronous DRAMs (SDRAMs) and a novel controller scheme to reduce the I/O bus width from 64 bits to 32 bits. This controller scheme allocates DRAM accesses of deterministic processes according to a schedule. The hardware cost is reduced while maintaining the flexibility needed for accommodating the stochastic processes in the architecture. The clock frequency is chosen to be 27 MHz, which is a simple multiple of the video sampling rate.

2. Decoder Architecture

Figure 1 shows block diagram of our decoder architecture. A controller directs the flow of operations among the decoder functions as well as the flow of data to the DRAM. The decoder consists of three main processing units, the VLD (variable length decoder), the baseline unit, the Motion Compensator (MC), and the associated buffers. The baseline unit consists of two functional units, the IQ/IZZ (Inverse Quantization and Inverse Zigzag) and the IDCT (Inverse Discrete Cosine Transform). The decreasing cost of synchronous DRAMs coupled with their ease of control makes them attractive for use in our architecture. In previous work [3] [4], a bus arbitration scheme is used to allocate DRAM accesses using scheduling schemes like First Come First Served. The bus is acquired whenever a buffer overflows/underflows.

The sub-picture pixels data are run-length compressed bitmaps that are overlayed on top of the MPEG reconstruction video. The pixels are divided into four types: background, foreground, emphasis-1, and emphasis-2. The sub-picture buffer size is restricted to 62Kbytes. This means that a maximum of 62Kbytes per GOP/cell and the maximal pixel data are 30Kbytes. The decoding speed is not critical in a sub-picture decoder. Figure 3 shows our architecture for this run-length decoder. The compressed sub-picture data has variable input rate, so a buffer is needed to smooth the data. One most important process in decoding is to detect the number of zeros. After we know the number of zeros, we can identify the number of pixels followed and extract and bypass pixel data to the Zero/One signal Generator. The decoding process can then be completed. This architecture is a serial decoder which decodes compressed pixel data at a rate of 2-bit per cycle and the output rate is not constant. This is a straightforward technique and is easy to implement.

3. Controller Scheme

Macroblock decoding follows a specific sequence. Our strategy is to take advantage of this sequence and impose a fixed schedule in the bus transactions to minimize buffer requests and waiting cycles. The processing sequence for a motion compensated macroblock is illustrated in Figure 2. Concurrency of operations is achieved with parallel processing and pipelining. Tasks have to be performed in a specific sequence. Accordingly, the required tasks in order are the Bitstream FIFO write, VLD buffer read, VLD decode, inverse quantized and zigzag and IDCT. If motion compensation is required, the MC task is also scheduled after VLD decode. The Controller synchronizes the MC and the IDCT unit on a block basis and also manages the synchronization of the tasks between blocks. The number under each process interval refers to the number of decoding cycle

0-8186-8409-7/98 $10.00 © 1998 IEEE

required [2]. For example, the first block of the IDCT process requires 120 cycles while the subsequent blocks require 64 cycles.

The bus schedule for memory transactions is shown in the last row of Figure 2. The number under each bus access represents the number of cycles required to transfer the designated data. The number of cycles includes the latency for address decoding. As the bus width is 32 bits, each transfer cycle represents 4 bytes of data. The transfers between the different processing units are scheduled as illustrated. First, paths 1, 2 and 3, which are stochastic in nature, are accessed. Then paths 4 and 5, which are deterministic in nature, are scheduled in that order within the processing of each of the six blocks. Buffer sizes are simulated by software and appropriate sizes are chosen, such that underflow or overflow is minimized. If an overflow or underflow occurs, the Controller arbitrates to either stop or feed the process accordingly before continuing the decoding.

There are five buffers that access the data in the DRAM. These are the Bitstream FIFO, the VLD buffer, the MC reference buffer, the Write Back buffer and the Display buffer. The Bitstream buffer is designed not to overflow even under the worst case conditions for DVD. That is,

FIFO buffer size (byte) =

$$(\textit{The longest macroblock decoding cycles}) \times \frac{9.8\ {}^{Mb}\!/_{s}}{8 \times 27\ MHz}$$

where 9.8 Mb/s is the input bitstream rate for DVD.

In our architecture only two of the five buffers, namely the VLD buffer and the Display buffer, are actively monitored by the Controller. The VLD buffer can underflow as the length of the encoded data for a block is not known ahead. Although the Display buffer is filled regularly every block it can still underflow if the decoding rate is slower than the display rate over some period of time. It will overflow if the converse occurs.

In our controller scheme, all processing units are synchronized on a block basis. Therefore, the VLD buffer will be refilled after finishing a block Display buffer request and before starting the next block decoding. However, if the VLD buffer does not hold the whole data of a block, that is the VLD unit does not encounter an EOB (end of block) symbol, it will request the Controller to refill the buffer. The Controller will do so after finishing a block display buffer request and the VLD unit will continue to decode until an EOB symbol is met. Only after this will the Controller start decoding the next block. If there are extra requests from the Display buffer, in the instance when the buffer would be underflow, the Controller will insert them after the normal DRAM accesses have been completed. Contrarily, if the Display buffer is going to overflow, the Controller would stop the whole decoding process to prevent this condition from occurring. The request from the Display buffer is given priority over the request from the VLD unit. The order of the processing between processing units is thus maintained. This eliminates the need for complex bus arbitration schemes.

4. Display Model

In the DVD format the most restrictive GOP sequence is an IBBPBBPBBP.... sequence. There is at most two B frames between either an I or P picture. Figure 4 illustrates the relationship between the decoding and display order. The first I picture, I1, is decoded followed by P1, B1 then B2 pictures. The display order is I1, B1, B2, P1, .. etc. After decoding the I1 picture, the decoding for the P1 picture is immediately started. After "a" interval of time the display for I1 is started. The display rate is a constant at 30 pictures per second so T = 33 ms. However, the decoding interval varies according to the picture type and characteristics. The decoding interval for a B frame is the longest followed by that for a P then an I frame. We exploit the DVD format to synchronize the decoding and display order to a set of three pictures. The real time decoding constraint is now $t_{P1} + t_{B1} + t_{B2} < 3T$ instead of $t_{P1} < T$, $t_{B1} < T$ and $t_{B2} < T$ where t_{P1}, t_{B1} and t_{B2} refers to the decoding time for the P1, B1 and B2 frame respectively. This means that a macroblock can sometimes be processed in more than 667 cycles. This gives a good safety margin for overheads like process requests for DRAM access due to buffer underflow/overflow conditions, start of a sequence Header processing, and for the variable nature of stochastic processes.

5. Simulation Results

A software simulator to simulate and monitor the decoding process in the architecture is developed. The controller function is implemented according to the scheme described above. 83 MHz SDRAM is adopted in our simulation. VLD size is specified by a parameter. The input bitstream is simulated at 9.8Mbits/s, the worst case condition. The Bitstream buffer is fixed at 45 byte. This figure is based on the total decoding cycles of 1000 needed for the first macroblock (including sequence header, GOP header,.. , etc.) in the first frame. The MC and write back buffers are fixed at 182 byte and 64 byte respectively. The tested movie, Mobile video at MP@ML, has 150 frames and each frame has 1320 macroblocks. The movie has 11 I frames, 40 P frames and 99 B frames. The performances are evaluated with various sizes of VLD buffer. The Display buffer is fixed at 1 Kbyte. The results are shown in Table 1. From the results, we can see that VLD buffer size of 8 byte and a Display buffer size of 1Kbyte is adequate for our Controller scheme. This compares well with the result of [3] which suggests a VLD buffer of 16 byte and a Display buffer of 1Kbyte.

The bus utilization factor is defined as the number of active bus cycles over the total number of decoding cycles. Table 1 shows the bus utilization factors compare well with the reported in [3], which use a 64 bit bus architecture. The results also show that the Bitstream buffer will not overflow at 45 byte. The results satisfy real time MP@ ML decoding.

By using the display model we proposed, the macroblocks that require more than 667 decoding cycles in B frame can be absorbed by I or P frames and so will not cause display delay. Furthermore, if we can send or receive 32 bit data on both positive- and negative-going edges of the 27 MHz clock internally, then the total number of macroblocks in B frames that exceed 667 decoding cycles can be reduced to 0.04% while VLD buffer size is 8 byte.

6. Conclusion

In this paper we have presented an efficient controller scheme for video decoder in DVD application. We incorporate SDRAM in our Controller scheme to enable a reduction of bus width from 64 bit to 32 bit. Memory access is scheduled according to the processing sequence resulting in much fewer bus requests and contentions. The Controller is simpler to implement and the bus utilization is high, which means an efficient use of bus resource. The DVD format is also exploited to allow a more relaxed constraint for decoding interval which means a cheaper hardware design.

7. References

[1] ISO/ICE 13818, " Generic Coding of Moving Pictures and Associated", (MPEG2).

[2] Jui-Hua Li and Nam Ling, " An Efficient Video Decoder Design for MPEG-2 MP@ML ", *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 509 – 518, July 14-16, 1997.

[3] Nam Ling and Jui-Hua Li, " A Bus-Monitoring Model for MPEG Video Decoder Design ", *IEEE Trans. on Consumer Electronics*, Vol. 43, No. 3, pp. 526 – 530, Aug. 1997.

[4] C.H. Liu, C.M. Chen & C.W. Jen, " Low Power Design for MPEG-2 Video Decoder ", *IEEE Trans. on Consumer Electronics*, Vol. 42, No. 3, pp. 513 – 521, Aug. 1996.

[5] T. Demura, et al., " A Single-Chip MPEG2 Video Decoder LSI ", *IEEE ISSCC Digest of Tech. Papers*, pp. 72 – 73, Feb. 1994.

[6] M. Toyokura et al., " A Video DSP with a Macroblock-Level_Pipeline and a SIMD Type Vector-Pipeline Architecture for MPEG2 CODEC ", *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 12, pp.1474 – 1481, Dec. 1994.

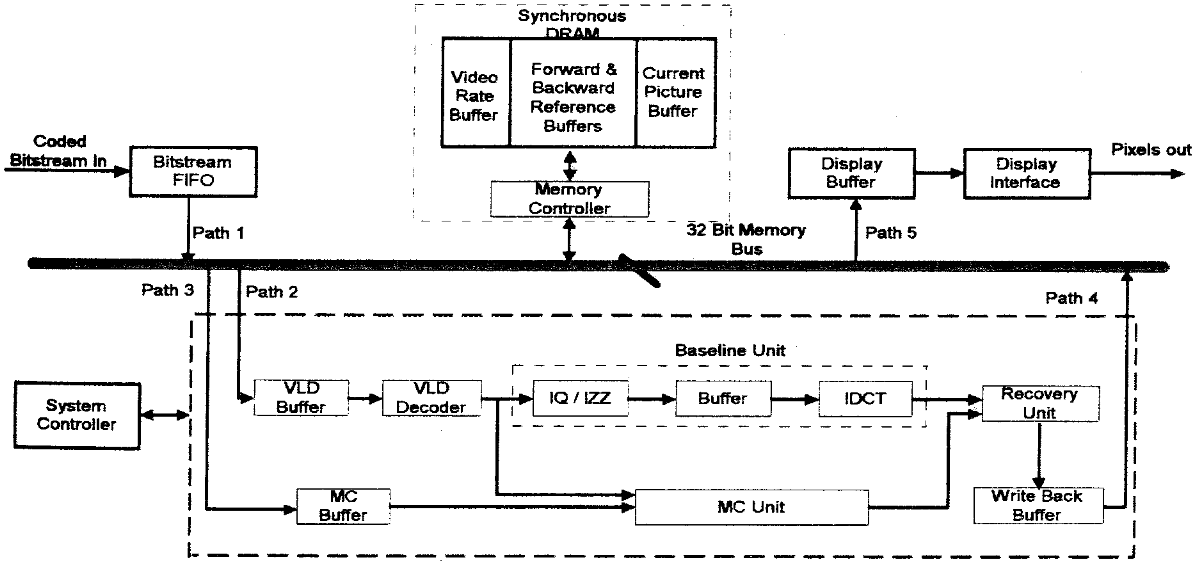
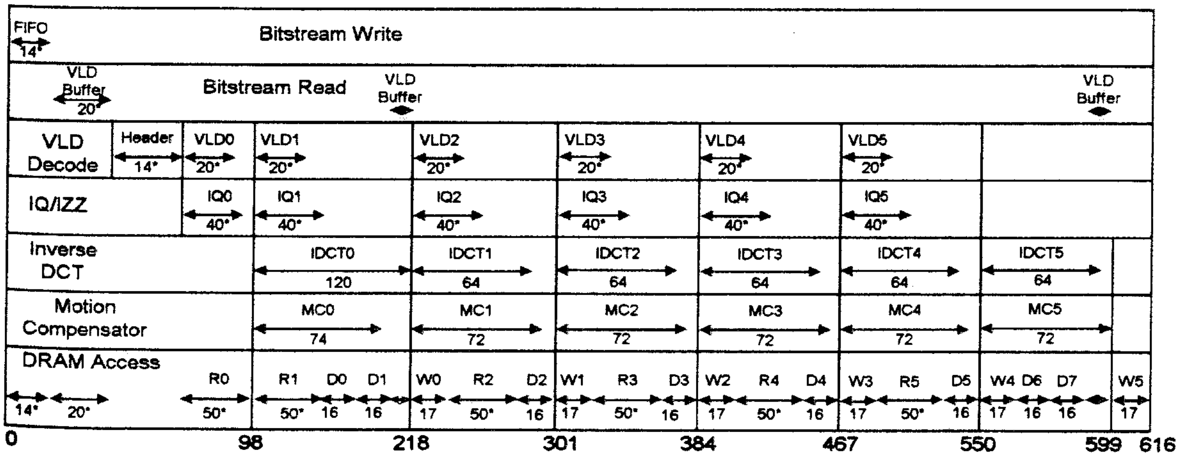


Figure 1 Block Diagram of the Video Decoder



* : Stochastic
Rn : Reference Block Read
Wn : Current Picture Write
Dn : Display Buffer Read

Figure 2 Processing sequence for decoding a predictive coded macroblock

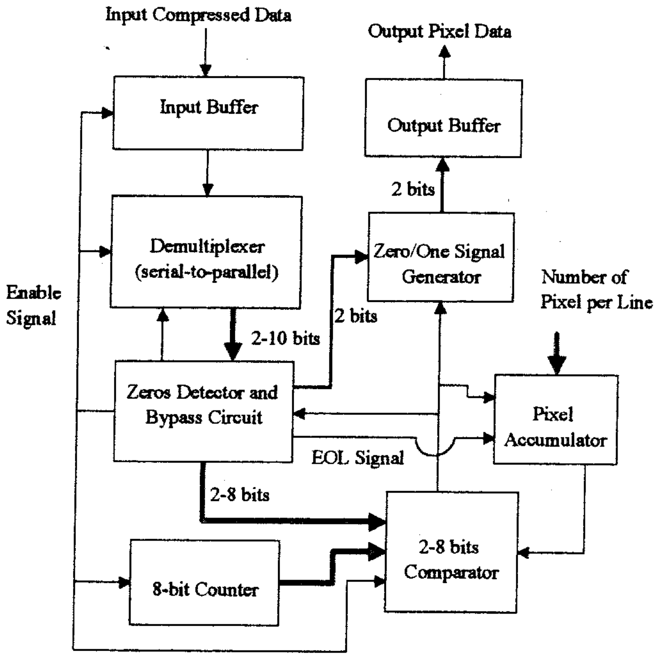


Figure 3 Sub-picture Decoder Architecture

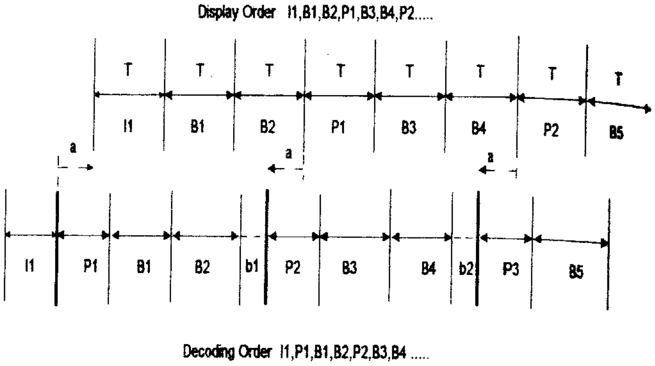

Display Order I1,B1,B2,P1,B3,B4,P2.....
T
T
T
T
T
T
T
T
I1
B1
B2
P1
B3
B4
P2
B5
a
a
a
I1
P1
B1
B2
b1
P2
B3
B4
b2
P3
B5
Decoding Order I1,P1,B1,B2,P2,B3,B4 .....

Figure 4 Relationship between decoding and display order

Mobile.m2v

	I picture		P picture		B picture	
	Ave.	Max.	Ave.	Max.	Ave.	Max.
Bits per one block	64	274	47	414	34	323
VLD : 20 bytes						
decoding cycles	499	622	535	695	545	851
bus utilization	51.95 %		72.62 %		95.22 %	
Ave. FIFO cycles : 11 Max. FIFO cycles : 15 (initial) The number of the MB in I frame exceeding 667 decoding cycles is 0 (0.0 %) The number of the MB in P frame exceeding 667 decoding cycles is 25 (0.05 %) The number of the MB in B frame exceeding 667 decoding cycles is 4421 (3.38 %)						
VLD : 16 bytes						
decoding cycles	500	622	536	697	545	850
bus utilization	52.11 %		72.68 %		95.22 %	
Ave. FIFO cycles : 11 Max. FIFO cycles : 15 (initial) The number of the MB in I frame exceeding 667 decoding cycles is 0 (0.0 %) The number of the MB in P frame exceeding 667 decoding cycles is 31 (0.06 %) The number of the MB in B frame exceeding 667 decoding cycles is 4474 (3.42 %)						
VLD : 8 bytes						
decoding cycles	502	622	538	697	546	852
bus utilization	52.64 %		72.71 %		95.20 %	
Ave. FIFO cycles : 11 Max. FIFO cycles : 15 (initial) The number of the MB in I frame exceeding 667 decoding cycles is 0 (0.0 %) The number of the MB in P frame exceeding 667 decoding cycles is 36 (0.07 %) The number of the MB in B frame exceeding 667 decoding cycles is 4571 (3.50 %)						

Table 1 Decoding cycles per MB and bus utilization under different VLD buffer size