DEVELOPING A MULTICHANNEL HIGH SPEED DDR SDRAM MEMORY CONTROLLER: A CASE STUDY FOR H.264/AVC DECODER

Alexsandro C. Bonatto, Andre B. Soares, Altamiro A. Susin
{bonatto,borin,susin}@eletro.ufrgs.br

Signal and Image Processing Laboratory (LaPSI)
Federal University of Rio Grande do Sul (UFRGS)

ABSTRACT

Embedded systems used to process high definition video sequences require storing large amounts of data while processing. These systems frequently employ dedicated hardware architectures which are more efficient to process video signals because of the data parallelism. Hardware modules perform data processing sharing one main memory used to store several reference frames. This shared memory has as main characteristics high capacity to store data and high bandwidth. In this paper we propose memory hierarchy architecture to integrate a DDR SDRAM memory controller in a H.264/AVC video decoder hardware implementation. The memory hierarchy contains an arbiter used to control data access priority between the hardware modules.

1. INTRODUCTION

In high performance video processing systems, an efficient memory hierarchy design is the key point to reach real time capacity while the decoding process of full high definition video sequences is executed. The memory hierarchy can be understood as the organization of the data storage elements and the way that they are accessed. In embedded systems, memory capacity is a very limited hardware resource. Generally, it is composed by local and external memories. External memories have more capacity to store data at low cost because they are manufactured in large scale. In this context, double data rate synchronous DRAM memories (DDR SDRAM) have large use in embedded systems because their low cost and high data storage capacity.

H.264/AVC [1] is the latest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). The H.264 video standard is defined in three different profiles: *baseline*, *main* and *high*. This work is focused on the hardware implementation of the *main* profile decoder.

Video processing systems computational complexity is formed by two major components: time complexity and space (or storage) complexity [2]. Space complexity is measured by the amount of memory required to hold all the reference frames and other information while video is being processed. Time complexity is measured by the approximate number of operations required to execute an algorithm. When considering an architecture design, the time complexity of an algorithm's execution is directly dependent of the total memory bandwidth available. Thus, efficient processing architectures are that have best balance between those two complexities, performing less memory accesses while processing data. The design of an efficient video decoder must consider the number of accesses to the reference memory and the way as these accesses are performed.

In the H.264/AVC video decoder, a predefined number of decoded video frames are stored into a reference memory and are used in the decoder process. The major amount of data storage complexity in the video decoder is required to store the reference frames and is limited on about 12.5 mega-bytes of data. The reference memory is accessed by different processing units (PU), each one of them interfacing data in different ways and in different levels of dependency. In digital video processing systems, the design of a memory hierarchy is necessary to store the reference frames and control the read and write requests from different PUs.

This paper presents an analysis of data access behaviors from different PUs of a hardware implementation of the H.264/AVC video decoder. A multichannel DDR SDRAM memory controller with access requests arbiter is designed and simulated. The multichannel controller controls data access requests optimizing the reference memory use by the H.264/AVC decoder. In this first implementation it is used a round-robin arbiter scheme with fixed time-slot (time-division multiplexing) to allow PUs to access the memory channel.

This paper is organized as follows: section 2 presents the hardware architecture of the H.264 decoder; section 3 presents the multichannel DDR SDRAM controller architecture proposed; the simulation results are presented in section 4 and the conclusions are discussed in section 5.

2. H.264 DECODER ARCHITECTURE

The video decoder hardware architecture used as reference in this work is presented by Agostini in [3]. The proposed architecture is organized in five main PUs: *motion compensation* (MC)*; intra-frame compensation* (intra)*; filter*; *inverse transform* (IT) and; *inverse quantization* (IQ). The PUs are implemented in VHDL language to be prototyped and validated over a FPGA platform. The entropy decoder and control processes are performed by an embedded processor. Also, this processor sends the video coded input bitstream to the decoder and controls the video output. The MC process and the reference frames memory are not yet fully integrated with the decoder.

The temporal differences between frames are processed by the MC to generate one actual frame. This process is the most computationally demanding in the video decoder. Also, MC is the decoding process that generates more stored data requests. The reference frames used by the MC process are previously decoded and

stored in an shared memory (or main memory). These frames are produced after full frame decoding, which are the output of a filter block in H.264. Finally, video output module needs to fetch decoded frames in the original order, even when frames are decoded in an arbitrary order. This produces the need to store these frames at the external memory. The video decoding process at level 4.0 uses four high definition (HD) 4:2:0 (1920x1080) reference pictures, representing an amount of 12.5 mega-bytes of stored data.

The first decoding step of the coded video bitstream input is the entropy decoder. This process generates the control elements for MC and Intra prediction processes. Also, the entropy decoding generates the image residual information that is added to the predicted frames, either motion compensated or intra-frame predicted. The generated images are filtered before the video output and before they are stored in the main memory.

Fig.1 illustrates the video decoding process organized as the data flow through the PUs implemented. Each hardware module in the decoder contains local memory resources used to process local data information. Due to the different data access behaviors from the PUs that interfacing with the main memory, The video output generator exhibits a line of pixels of the entire image, which in the case of full HD video, is composed by 120 macroblocks (MBs) in wide. Therefore, a line of MBs is read from the main memory and stored into the video output buffer. The decoder output (i.e. filter) generate a sequence of MBs and each one is stored in a buffer of 2 MBs size to be send to the main memory. The MC contains a local cache structure that is responsible to request reference pixels stored into the main memory to perform the reconstruction process. Those buffers are necessary to optimize the DDR SDRAM available bandwidth, storing temporary data on buffers to allow sharing the memory channel between PUs.

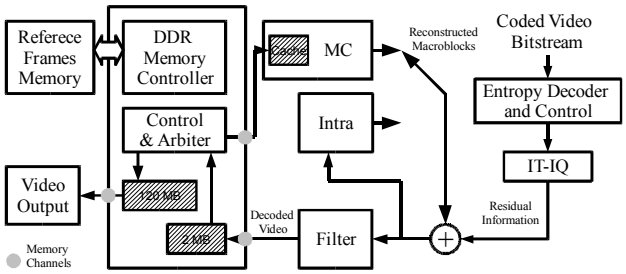

Fig 1. H.264 video decoding process flow and the processing units.

One shared memory is necessary to store several reference frames used by the MC process. In H.264/AVC, slices are formed by motion compensated blocks from past and future (in temporal order) frames. The past and future frames are not fixed just to the immediate frames, as in early standards. Each macroblock in a bi-predictive slice (B slice) can be predicted from one or two reference frames, using past and future frames. The reference frames are organized in two lists: 0 and 1.

This architecture requires a frame buffer to hold output images generated by the filter. If a frame buffer is not available or, if it is necessary to share the storage resources, the main memory has to be accessed also by the video controller. The main memory controller has to balance and optimize the data accesses sequences requested by different modules. This is necessary to reduce the waiting time of hardware modules to access main memory while processing, stalling the video decoder.

Therefore, a multichannel memory controller module is required to control the data accesses sequences.

As the video decoding process has an unpredictable behavior, the MC module can access the main memory in different rates. Also, the three hardware modules are connected to the memory controller sharing the same command and address bus, being necessary to have an arbiter controlling the data requests.

In the next section will be explained the external DDR SDRAM memory controller and also the multichannel memory architecture, as a part of the memory hierarchy.

3. MULTICHANNEL MEMORY CONTROLLER

This section explains the main characteristics of DDR SDRAM memory [4] and a multichannel controller architecture is proposed.

3.1. Double Data Rate SDRAM

Double data rate memories contain three buses: a data bus, an address and a command bus. The command bus is formed by the signals column-address strobe (CAS), row-address strobe (RAS), write enable (WE), clock enable (CKE) and chip-select (CS). The data bus contains the data signals (DQ), data mask (DM) and data strobe signals (DQS). Address bus is formed by address (ADDR) and bank address (BA) signals. These memories operate with differential clocks CK and CKn, which provides source-synchronous data capture at twice the system clock frequency. Data is registered either in the rising edge of CK and CKn. The memory is command activated starting a new operation after receive a command from the controller.

Data words are stored in the DDR memory organized in banks and pages and each memory page contains 2^{10} data words. Fig. 2 illustrates the timing diagram for a RD operation in DDR memory. Data is transferred in bursts, sending or receiving 2, 4 or 8 data words in each memory access.

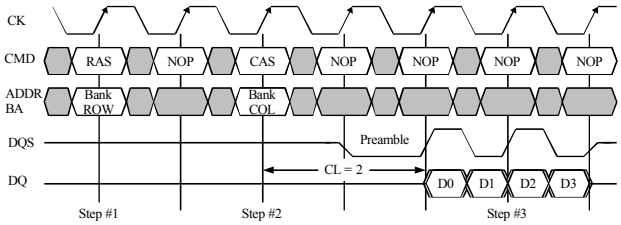

Fig. 2: Timing diagram of reading data.

Data memory contents are accessed by page activation, using the *row-address strobe* (RAS) command (step #1). After this, the memory controller sets the column address, called as a *column-address strobe* (CAS) command (step #2). In the case of a RD operation, data is available after

the *CAS Latency* (CL) which can be 2, 2.5 or 3 clock cycles (step #3). The data words D0:D3 are transmitted edge aligned with the strobe signal DQS after the CAS latency. DQS is a bidirectional strobe signal used to capture data DQ. To change to another memory page, is necessary to deactivate the current page. This is done by using the *pre-charge* (PRE) command and takes about 10 cycles after the last data access operation.

When interfacing with DDR SDRAM memories, latency becomes a problem if each consecutive data access is performed in different memory rows. In this case, changing memory row requires the execution of *Active* and *Pre-Charge* commands. Frequent row changes reduce the effective data bandwidth, degrading the memory interface speed.

3.2. Multichannel Memory Controller

The main purpose of the multichannel memory controller is to guarantee Q*uality of Service* (QoS) between the PUs accessing the external memory. This means to generate equilibrated data bandwidth and fast access permissions to read or write shared data in the decoding processes. Its internal architecture is illustrated in the Fig.3. It is designed with three access interfaces with respective command, address, data and control signals. The multichannel interface uses a simple protocol where an acknowledge (ACK) signal gives permission to the module after the received access request. There are three main hardware modules: an external memory controller, an arbiter and a data-path. The external memory controller was implemented as an intellectual property (IP) module, presented before in [5].

The arbiter controls the access of each hardware module to the time-division multiplexed memory channel. An internal table stores the maximum occupation in number of clock cycles of the memory channel by each hardware module.

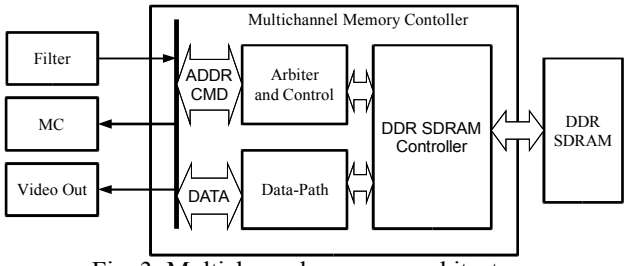

Fig. 3: Multichannel memory architecture.

Video output and filter modules access the main memory at a constant rate, in the exhibition rate. For 1080p 4:2:0 video sequences, filter send data to be stored in the main memory at an effective rate of 245,760 macro-blocks per second, i.e. 94.3 mega-byte per second. Data is send in bursts while the macro-blocks are decoded, and is stored in the memory. This macro-block rate is the same for the video output. Either the video output and the filter requires local memory resources to store MBs.

The MC architecture to be implemented with the controller proposed in this paper is presented by Azevedo in [6]. As the referred author mentions, the memory throughput is mainly needed when the main profile is considered (because the bi-prediction support) and as well high resolution videos (as HDTV). The total memory throughput of 956.448 mega-bytes per second is necessary for decoding HDTV video sequences. In order to optimize memory accesses, the author implements a three dimensional data cache architecture and techniques of read only the necessary samples and interleaved samples stored at the main memory. The use of a cache can reduce in more than 62% of external memory accesses. Using this approach, the total bandwidth for MC accessing external is about 382.57 mega-bytes per second.

Tab.1 summarizes the main memory access rates interfaces estimated for decoding 1080p 4:2:0 H.264/AVC video sequences.

Tab 1. Data access behaviors for 1080p 4:2:0 video.

	Data Direction	Data Behavior	Access Rate (Mbyte/s)
MC	Read	Line of pixels	382.57
Video Out	Read	macro-block	94.3
Filter	Write	macro-block	94.3

The total bandwidth required to interface with the main memory is about 571.17 mega-bytes per second. For a DDR SDRAM memory, running at a clock rate of 100 MHz and data width of 64 bits, the peak bandwidth is 1,600 mega-bytes per second. This memory interface is sufficient to implement the hardware video decoder architecture.

Data access latency is not a problem when more than one read or write commands are requested to the memory. The read and write commands can be concatenated and data can be accessed continuously.

The memory controller can schedule the received commands from the hardware modules to better organize memory data accesses. This simple operation can increase the effective number of data access rates. Local memory caches, as the one proposed in [6] complete the memory communication optimization task.

4. FUNCTIONAL SIMULATION AND ANALYSIS

Data pattern generators were used to simulate the access behavior for the video decoder hardware modules. The multichannel controller arbiter was implemented performing a Round-Robin scheduling scheme, where each data channel have a fixed time-slot to access the shared memory. Different time-slot sizes were used to made a performance comparison between the two main factors of QoS: the available bandwidth and; the fast access permission to use the memory channel. In this implementation of the video decoder, each PU that interfaces with the main memory contains a local memory used to store data while processing.

Tab. 2 summarizes the main information of each PU sharing the main memory regarding the required PUs bandwidth. Also, each PU are classified as bandwidth or latency dependent. In the case of a latency dependent process, the PU require to use the memory channel immediately.

Tab 2. processing units and the memory dependence.

	MC	Video Out	Filter
Sensitivity	latency	bandwidth	latency
Mean channel request interval (clock cycles)	385	48 828	407
Maximum consecutive Access (clock cycles)	61	2 958	40

The maximum consecutive access in clock cycles represents the total clock cycles used from the memory bandwidth for each data access, including the extra cycles for activate and deactivate the memory pages.

The simulation setup was done using a previously recorded data access pattern obtained with MC simulation. The data pattern generators was used with the designed multichannel memory controller with a simple round-robin arbitration scheme. Tab. 3 shows the simulation results for different time-slot (TS) controlled by the arbiter.

Tab. 3. Simulation results and accesses behaviors for PUs (in clock cycles).

	MC		Video Out		Filter	
	Access	Wait	Access	Wait	Access	Wait
TS32	41	32	46	102	23	38
TS64	42	50	69	154	23	54
TS128	42	99	116	242	23	25
TS256	42	118	170	283	23	135
TS512	42	129	274	229	23	147

As the simulation results shown, the increase in the time interval of using the memory channel, bigger is the wait time for other PUs. Also, it can be seen that processes with bandwidth sensitivity increase the relation between access and wait by increasing the time interval.

The memory controller is able to share the DDR SDRAM memory between PUs, but some penalties are detected when using this kind of arbitration. The processes that are latency sensitive may not access the memory channel as faster they need because the bandwidth sensitive process. This can cause delays in the decoding process and degradation of the real-time capabilities of the video decoder.

5. CONCLUSIONS

In video decoder architectures, an efficient memory hierarchy is necessary to allow real-time decoding. The high volume of information requires a large external memory to store several reference frames. Local memory reduces needed external memory bandwidth allowing dedicated hardware modules to execute local processing tasks. Nevertheless, memory channel multiplexing is needed between different hardware modules.

This work presented an external multichannel memory controller which permits to define different memory bandwidth for each processing module. The controller has reuse facilities as it maintains the characteristics of the single channel controller presented in [5].

Data pattern generators can be used to simulate the data access behavior of the real application, if the modules are modeled correctly. As the video output have constant bit-rates, the estimable throughput can be concerned to the motion compensation process. The overall system integrating the multichannel controller and the data pattern generators can be implemented in a hardware development platform to validate the memory hierarchy architecture using a real external memory. The controller was successfully tested considering the communication needs of the main modules of an H.264 decoder.

Future works include on board testing of the controller and the study of different memory channel arbitration schemes.

6. REFERENCES

[1] ITU-T Recommendation H.264 – Advanced video coding for generic audiovisual services, Video Coding Experts Group, Mar. 2005.

[2] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 13, no. 7, pp. 704–716, July 2003.

[3] L. V. Agostini, A. P. A. Filho, W. T. Staehler, V. S. Rosa, B. Zatt, A. C. M. Pinto, R. E. Porto, S. Bampi1, and A. A. Susin, "Design and FPGA Prototyping of a H.264/AVC Main Profile Decoder for HDTV," *Journal of the Brazilian Computer Society*, vol. 12, pp. 25–36, 2007.

[4] JEDEC, JESD79: Double Data Rate (DDR) SDRAM Specification, JEDEC Solid State Technology Association, Virginia, USA, 2003.

[5] A. C. Bonatto, A. B. Soares, A. A. Susin. "DDR SDRAM Controller IP Designed for Reuse," In: *IP based electronic system conference & exhibition - IP 08*, France. Design and Reuse, pp. 175-180, 2008.

[6] A. Azevedo, B. Zatt, L. Agostini, and S. Bampi, "MoCHA: a Bi-Predictive Motion Compensation Hardware for H.264/AVC Decoder Targeting HDTV," in Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on, May 2007, pp. 1617–1620, 2007.