DDR SDRAM Memory Controller for Digital TV Decoders

Hadley M. Siqueira[1], Ivan S. Silva[2], Marcio E. Kreutz [1], Edgard F. Corrêa [1]

[1]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte

[2]Departamento de Informática e Estatística
Universidade Federal do Piauí

hadley@lasic.ufrn.br, ivanivan@ufpi.edu.br,
{kreutz,edgard}@dimap.ufrn.br

Abstract. *This paper presents an implementation of a multichannel DDR SDRAM memory controller as a module of a Set-Top Box compliant with Brazilian Digital Television System. A Set-Top Box is comprised by modules that access an external memory sharing the same bus. Thus, it is necessary a multichannel DDR SDRAM memory controller to schedule accesses. This work shows that the implemented system running at 100 MHz can achieve the necessary bandwidth to decode and exhibit HD 1080p resolution videos at 30 frames per second.*

1. Introduction

Brazilian Digital Television System (BDTS) is a technical standard for digital television broadcast developed in Brazil. It is based on the Japanese ISDB-T standard. To receive a transmission, a Set-Top Box (STB) may be used. A STB is a device that connects to a television and an external source of signal, turning the signal into content which is then displayed on the television screen or other display device.

The Brazilian Digital Television System uses H.264/AVC, the latest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG), as video codec. This state-of-the-art video coding standard out performs previous standards by employing bi-predictive motion estimation, spatial prediction and adaptative entropy coding techniques.

The new compression techniques used in H.264/AVC has as penalty an increase in the memory traffic. Memory access optimizations are necessary to handle data in the reference frames while video is decoded and exhibited at real time. Architectures of video processing systems require a single interface to off-chip DRAM memory in order to achieve the necessary storage capacity at low cost [Wolf and Henriksson 2008]. In this context, double data rate synchronous DRAM (DDR SDRAM) [JEDEC 2003] memories have large use in embedded systems because of their low cost and high data storage capacity.

This work is part of a national effort to develop a set-top box compliant with BDTS. The set-top box contains components like audio decoder, video decoder, general processor, graphic processor and memory controller. This paper presents a multichannel DDR SDRAM memory controller design implemented as an IP to be used in the set-top box. It is an extension of the work developed in [Bonatto 2009], that implements the

multichannel memory controller for a H.264/AVC video decoder. Now, the memory controller is extended in order to handle a H.264/AVC video decoder, a general CPU and a graphic processor.

This paper is organized as follow: Section 2 describes the set-top box architecture; Section 3 describes the multichannel DDR SDRAM controller as well as some related works. Results are discussed in section 4 and conclusions in section 5.

2. Set-top box architecture

This work focuses on a set-top box being developed to be compliant with BDTS. It is a prototype being developed as a national effort. Figure 1 illustrates the STB architecture.

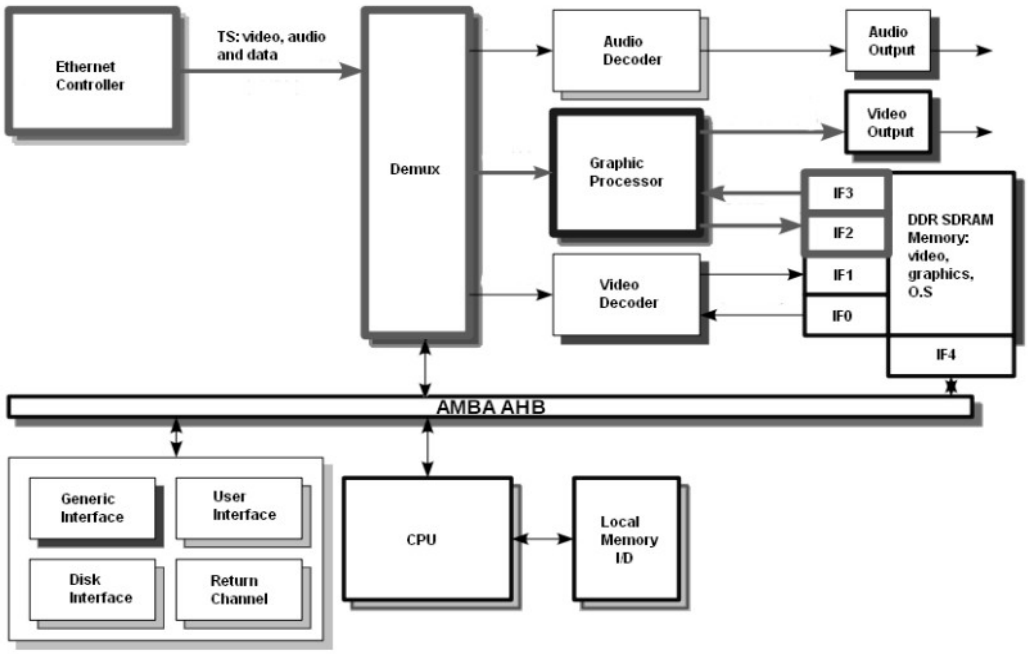

Figure 1. Set-top box architecture

The ethernet controller sends a Transport Stream (TS) to the demultiplexer. The demultiplexer divides the TS in audio, video and data information and sends them to the modules: audio decoder, video decoder and CPU, respectively, through bus. Some data, e.g legends, are send directly to the graphic processor.

Three modules access the DDR SDRAM memory: CPU, H.264/AVC video decoder and graphic processor. These three modules are connected to the multichannel DDR SDRAM memory controller. The memory controller is represented by the IF0, IF1, ..., IF4 blocks in Figure 1.

The CPU is a general processor and processes user inputs and software that comes in transport stream. It can also run a middleware. The video decoder decodes the H.264 frames and stores the decoded frames in the Reference Picture Buffer (RPB) memory, which is a region on DDR SDRAM memory reserved for frames storage. The Motion Compensator (MC) module and filter are responsible for the video decoder

accesses to memory.

The MC architecture is presented in [Zatt 2007]. It contains a local tri-dimensional cache structure that is used to store the requested reference pixels while the reconstruction process is executed. As the video decoding process has an unpredictable behavior, the MC module can access the main memory at different data rates. Also, the region of pixels used to reconstruct the image can be different in each decoded macroblock. The MC cache uses an addressing scheme based on: the x and y pixel block coordinates and the reference picture number. The memory controller uses this information to send a region of pixels to the MC with size 32x16 luminance pixels and two 16x8 chrominance pixels.

The filter output generates decoded pixels in a sequence of macroblocks. The filter does not generate any addressing information and the decoded pixels are stored in the reference picture buffer indexed by the memory controller. Graphic processor takes decoded frames, combine them with an overlay and sends the new generated frame to a display device. It works by reading two lines: one of the actual frame being exhibited and the other from an overlay. It can also store a new overlay for future use.

Images are stored in the form of macroblocks of pixels (MBs) in the Reference Picture Buffer (RPB) and the granularity of data transfers to the reference memory is a macroblock organized in Luminance (Y) and Chrominance (Cb and Cr) information. The CPU has no defined granularity, since it works with other data than macroblocks.

3. Multichannel SDRAM Controller

The main purpose of the multichannel memory controller is to guarantee Quality of Service (QoS) between the Processing Units (PUs) accessing external memory. This means to split available data channel bandwidth and fast access permissions to read or write shared data during the decoding processes. The aim of this scheme is to optimize resources in order to achieve real time decoding.

The implemented multichannel DDR SDRAM memory controller is divided in an arbiter module, a data-path module and a DDR SDRAM memory controller. It is not a new design, but an extension of the multichannel memory controller implemented in [Bonnato 2010]. The original design supports three clients. In this work, it is extended to support five clients and some slightly modifications are also made in order to provide a more modularized system. Figure 2 shows a block diagram of the proposed memory controller.

Data-path is a multiplexer plus additional logic that handles clients' access requests. When it detectes that a client wants to access memory, it sends to arbiter a signal composed of the client ID and the kind of access (read or write). Arbiter module has all the necessary logic to schedule accesses and decides which client will gain access to memory. When it decides, it sends a signal to the selector input of the multiplexer inside data-path. The DDR SDRAM memory controller is responsible for the comunnication between the choosen client and DDR SDRAM memory. It is also responsible for pre-charges and refresh commands that are necessary to keep memory's data consistent. DDR SDRAM memory controller is presented in [Bonatto 2008].

Three different arbiters were designed, each implementing a different arbitration scheme. One of them implements a round-robin and the other two implement a preemptive scheme, with each giving different priorities for differente modules.

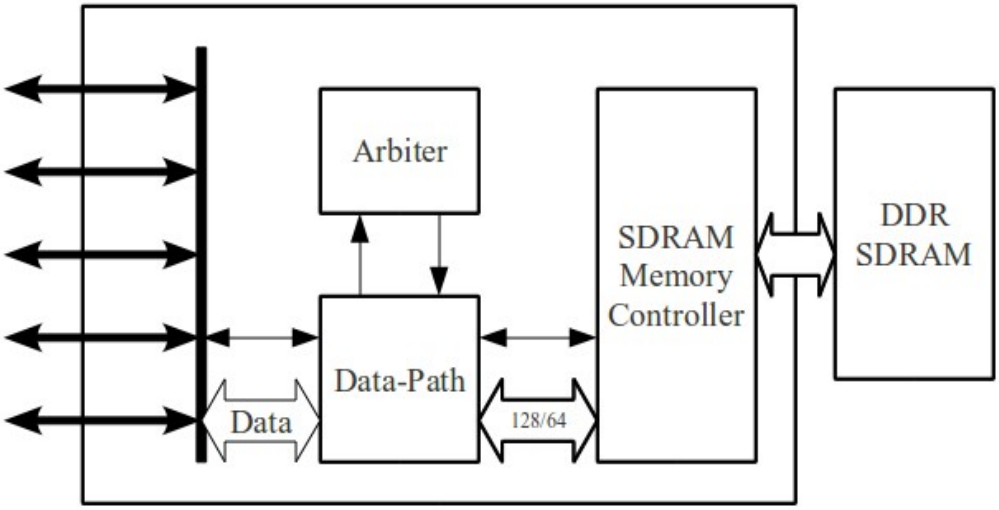

Figure 2. Multichannel memory controller architecture

3.1. Buffers

The most common operations on a DDR SDRAM memory are write and read. Before one can read or write to memory, it is necessary to setup it: bank and line need to be activated, address needs to be on the address bus among other things. These steps takes some cicles and while setup takes place, no data can be transferred between a client and memory. Once setup is done, it is possible to transfer large quantity of data, as long as data is stored on same line, due to large data bus and data transfer occurring on both rising and falling edge of clock. If data needs to be stored on a different line, the actual activated line should be deactivated and the new line activated. This process also needs some clock cicles and no data can be transferred as well.

As it is impossible to transfer data while memory is configured, it is necessary to minimize the time wasted configuring memory in order to maximize the ammount of data that can be transferred. Buffers can help increase maximum data transfer if application has some spatial or temporal locality, reducing unecessary repetitive accesses to memory, leaving memory free to other clients' accesses.

For each client a buffer was implemented. Buffers are not coupled with memory controller. Thus, it is an optional module that can be used when the clients present local or temporal spatiality patterns accesses. This is the case of all the modules of the set-top box that access memory. Figure 3 shows the multichannel memory controller integrated with the clients and buffers.

MC and filter have buffers with capacity to store nine macroblocks. Graphic processor has 4 buffers. Two of them are used to store the lines of a frame and overlay to be exhibited. The other two are used to store the generated lines of a new overlay and next line of frame and overlay to be exhibited. Each of these buffers can hold an entire line of a 1920x1080 frame. CPU has a simple buffer with capacity for nine macroblocks.

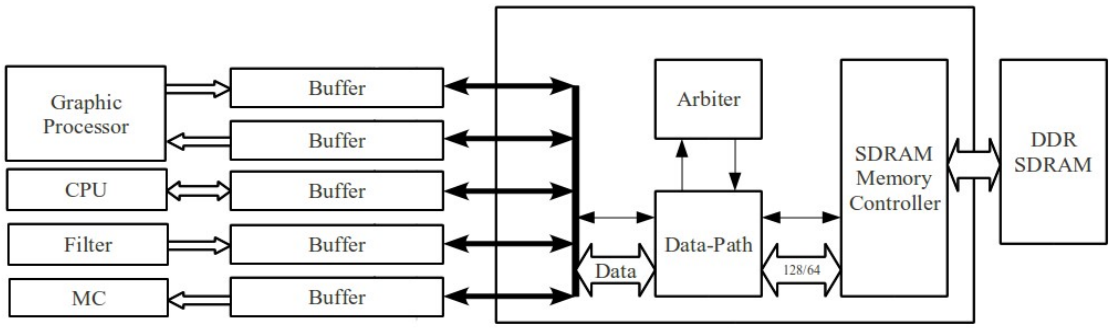

Figure 3. Multichannel memory controller, buffers and clients

3.2. Memory Layout

To maximize data transfer, some memory layouts were defined to store decoded frames and the actual frame being exhibited. Motion compensation process takes a macroblock on a frame already decoded. The coordinates of this macroblock are similar to the one being decoded. The next macroblock to be decoded may have in common some search area search with the previous decoded macroblock. To take advantage of this, decoded macroblocks are stored in a tiled manner. When a new search area needs to be fetched only the necessary macroblocks are updated.

Graphic processor, on the other hand, needs data to be stored in a raster scan order, this leads to duplication of frames, with the same frames being stored in both tiled manner and in a raster scan manner. While MC reads data in a tiled manner, graphic processor reads the same frames in a raster manner.

Although more data is transferred, the high quantity of data that is stored at one time reduces configuration wasted cycles. Also, the use of spatial and temporal localities together with buffers leads to less data retransmited. The motion compensator may need the same data several times, fetching data tiled and get stored on buffers reduces the quantity of access to RAM.

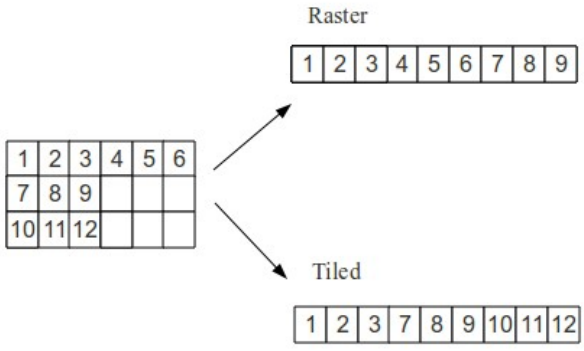

Figure 4. Raster and tile patterns

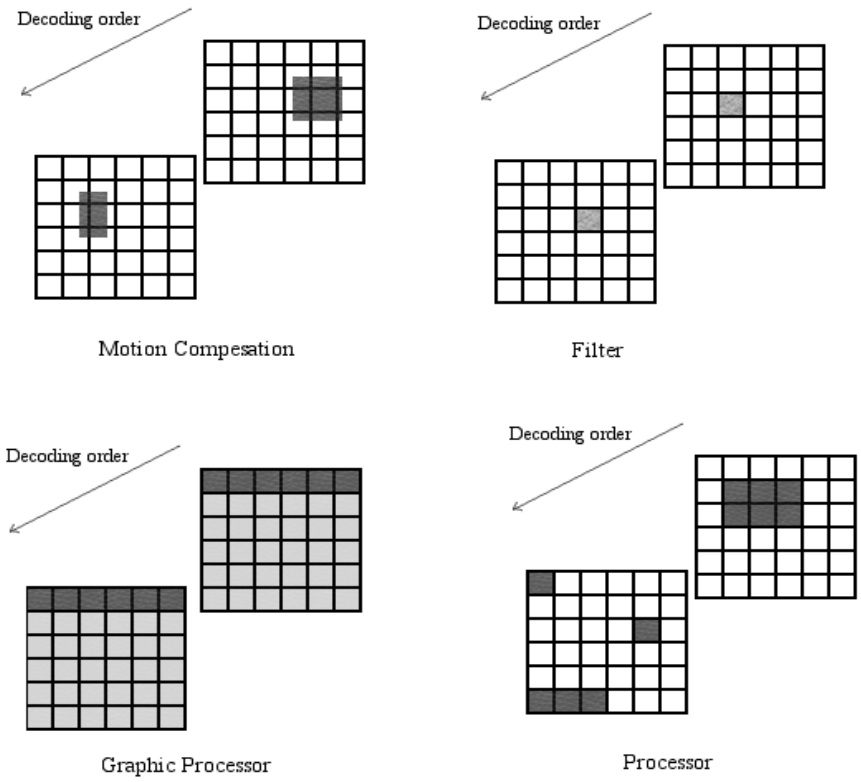

Figure 4. Accesses patterns of clients

3.3. Accesses patterns

Figure 4 shows how data is stored in raster and tiled manner. These two patterns make possible to fill buffers in a faster way because data is grouped together and can be transferred in burst. Figure 5 shows the accessed regions from in the actual and previous frames. Regarding MC and graphic processor, the shaded regions on MC shows that searched macroblocks have a tiled pattern and not a raster scan pattern. The light shades in graphic processor means that all lines of macroblocks are going to be read. The black lines means that a line is read in a raster scan order.

Table 1 summarizes the necessary bandiwidth for each client.

Table 1 – Required bandwidth for each client

MC	Filter	Graphic processor		CPU
		overlay + frame	frame	
294MB/s	89MB/s	180MB/s	89MB/s	not know

4. Experiments and Analysis

Three arbiters were implemented, each implementing a different arbitration scheme described bellow. To simulate the clients, state machines that execute a program stored in a ROM are used. The program mimes the access patterns of motion compesator module, filter, graphic processor and CPU.

A program can have four kinds of instructions: a write instruction, a read instruction, a wait instruction and a goto instruction. Write and read instructions has four parameters: the initial address, the total quantity of data to be transferred and the minimum quantity of data that is going to be transferred if arbiter asks the client to leave the channel. The wait command is used to wait between accesses and has as parameter the time, in cicles, to wait. The goto instruction is used to jump to a different instruction. With these four commands, it is possible to simulate periodic and aperiodic accesses to memory.

To simulate the motion compensator module MoCHA [Zatt 2007], log files of the execution of MC were generated with time of access and quantity of data transferred logged. A program based on these informations was coded. Filter and graphic processor have a known periodic pattern access. CPU program tries constantly to access memory.

4.1. Arbitration Schemes

Three arbitration schemes were tested. The first scheme is a round-robin. Table 2 shows the results of bandwidth utilization achieved by each module.

Round-robin proves to be inefficient, since the decoding process stalls because MC can not achieve 294MB/s. On the other hand, the CPU can achieve a maximum of 907MB/s.

Table 2 – Results for scheme I

Timeslot (cicles)	MC	Filter	Graphic processor		CPU
			overlay + frame	frame	
64	245MB/s	89MB/s	182MB/s	93MB/s	336MB/s
128	190MB/s	89MB/s	184MB/s	95MB/s	508MB/s
256	135MB/s	67MB/s	186MB/s	95MB/s	720MB/s
512	87MB/s	43MB/s	188MB/s	95MB/s	907MB/s

The second scheme is a preemptive one: it gives the MC and filter priority over the graphic processor and CPU. Thus, MC and filter can preempt graphic processor and CPU and these two are scheduled by a round-robin scheme.

Table 3 – Results for scenario II

Timeslot (cicles)	MC	Filter	Graphic processor		CPU
			overlay + frame	frame	
64	305MB/s	90MB/s	179MB/s	95MB/s	181MB/s
128	304MB/s	89MB/s	180MB/s	95MB/s	270MB/s
256	305MB/s	89MB/s	177MB/s	95MB/s	273MB/s
512	305MB/s	89MB/s	179MB/s	95MB/s	270MB/s

In scenario II, it is possible to achieve real time decoding, since all modules got their necessity of data supplied. The CPU can only achieve 273MB/s now. This is a significant reduction when compared to scenario I.

The last scheme is also a preemptive one. It gives priority to MC and filter, that can preempt the graphic processor which in turn can preempts CPU. In this scheme, CPU has a little better performance when compared to scenario II. This little increase of perfomance is because graphic processor finishes it accesses faster, as CPU can not preempt it anymore. This leave a bigger timeslice to CPU access memory before the next access of graphic processor to memory. In scheme III, all modules can achieve the necessary bandwidth.

These results were possible due to the usage of buffers, that make possible to do less context switch. Since data are stored in buffer, clients can make a bigger burst instead of smaller ones. When MC and filter can preempt other modules, it is observed an access pattern where only filter, MC and processor access memory, while graphic processor stays idle waiting for next access.

Table 4 – Results for scheme III

MC	Filter	Graphic processor		CPU
		overlay + frame	frame	
304MB/s	89MB/s	182MB/s	93MB/s	285MB/s

4.2. Simulation and Synthesis

The multichannel controller was developed in VHDL and synthesized to a Xilinx Virtex-2 Pro FPGA using the ISE (Integrated Software Environment) tool from Xilinx and occupies 1480 slices and can run at 114 MHz. Buffers occupies 37% of avaiable FPGA memory.

Mentor Modelsim Xilinx Edition and GHDL tools were used to simulate the entire design with a VHDL DDR SDRAM model from Micron. Results in Tables I, II and III were generated from functional simulation.

5. Conclusions

Results show that the proposed system running at 100 Mhz can achieve the necessary bandwidth necessary to decode and exhibit Full HD resolution videos.

Three arbitration schemes were tested. The one that provided the best results uses a scheme of hierarchical preemptions. Local memory reduces the external memory bandwidth requirements, allowing dedicated hardware modules to execute local processing tasks. Future works include the study of different memory addressing organizations using different banks to store luminance and chrominance information.

References

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

P. V. Wolf, T. Henriksson. “Video processing requirements on SoC infrastructures,”In Proceedings of the Conference on Design, Automation and Test in Europe. DATE

'08. ACM, New York, NY, pp.1124-1125, 2008.

JEDEC, JESD79: Double data rate (DDR) SDRAM specification, JEDEC Solid State Technology Association, Virginia, USA, 2003.

BONATTO, A. C. ; SOARES, A. B. ; SUSIN, A. A. . High Efficiency Reference Frames Storage for H.264/AVC Decoder Hardware Implementation. In: IEEE Latin American Symposium on Circuits and Systems, 2010, Iguaçu Falls, Brazil. IEEE Latin American Symposium on Circuits and Systems, 2010. p. 304-307.

B. Zatt, A. Azevedo, L. Agostini, A. Susin, S. “Memory hierarchy targeting bi-predictive compensation for H.264/AVC decoder,” VLSI, 2007. '07. IEEE Computer Society Annual Symposium on, 446, 2007. Bampi. motion ISVLSI pp.445

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.