Towards an Efficient Memory Architecture for Video Decoding Systems

Alexsandro C. Bonatto, Marcelo Negreiros, André B. Soares, Altamiro A. Susin
Electrical Engineering Department
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
alexsandro.bonatto@ufrgs.br

Abstract—**Multimedia applications are known to use large amounts of memory. The video modules need also high throughput memory port for coding and decoding high resolution video sequences. The design of a multimedia System-on-Chip (SoC) could implement embedded block RAMs but it is much more cost-effective to use a single external memory at the expense of a multichannel memory controller. This paper presents the design and implementation of an efficient memory hierarchy for a Set-Top Box (STB) SoC with a video decoder. To use efficiently the Double Data Rate (DDR) external memory it must be accessed in burst mode whenever possible. In this paper we develop an analysis and implementation of a four level memory hierarchy targeting data latency reduction and bandwidth optimization of the memory port. The case study is DDR2 SDRAM memory used as the main system video memory in a digital television set-top box implemented on a Virtex-5 FPGA. This paper presents the architecture of the system and shows that the memory hierarchy efficiently uses the DDR characteristics while serving four client processes. The proposed memory architecture can reduce data latency in 78% when compared to a direct demand-access procedure.**

Keywords—Set-top box; digital television; memory hierarchy; memory controlle; digital design

I. INTRODUCTION

A System-on-Chip (SoC) for digital television set-top box is a complex system designed to manage high quality images in high definition videos. A SoC is composed of several modules as processors, video and audio decoding accelerator units, memories and peripherals. Besides multimedia coding and decoding, the system is required to manage the displayed video and compose it with locally generated text and graphics information, referred to as the On-Screen Display (OSD) information. Furthermore, the format of the video output must be scaled to fit the video display format. Video post processing operations are done in real-time over the bitmap image using digital signal processing techniques. Video processing systems require an efficient memory hierarchy to reach real-time performance for the decoding of high definition video sequences. Both the decoded video and the OSD data are stored in the same shared memory to be composed and exhibited by the video output module on a monitor or a television.

The Brazilian Digital Television System (ISDB-T) standard [1] has adopted the H.264/AVC standard [2] and MPEG4-AAC standard [3] for video and audio coding and decoding. In the H.264/AVC video decoding process, a Decoded Picture Buffer (DPB) is needed to store previously decoded frames. For this large size memory, an off-chip DRAM (Dynamic Random Access Memory) is used and the memory accesses are directed to a single memory interface. A memory hierarchy is needed to optimize the interface between the processing modules of the system and the external memory, to improve data locality and to reduce read data latency.

Related works found in literature treating the problem of external memory data accesses do not present a complete analysis of the latency in accessing data in a video processing system containing a memory hierarchy. Chih-Hung Li et al. [4] present an analysis of the size of the synchronization buffer implemented to increase the DRAM access efficiency. They argue that larger block sizes can optimize the external memory access increasing its efficiency. In their work, they showed that transferring the video pictures using smaller block sizes, as 8x8 pixel granularity, can provide a better trade-off considering cost, efficiency, power and real-time requirements. However, this work does not present any analysis of how the synchronization buffer size impacts over data latency in accessing the external memory. In [5] it is presented that the use of data-forwarding caches can reduce off-chip memory reads by 53%, while using a last-frame cache can eliminate 80% of the off-chip reads. However, the use of frame buffer compression techniques can reduce the memory bandwidth but in a multiplexed memory channel the bandwidth reduction is not as important as the efficient memory channel usage.

This work presents an analysis methodology of the memory hierarchy to reach an efficient design of a memory interface for digital video systems. We present an analysis of memory requirements and system behavior when managing high definition video sequences. The design challenge is to fit the huge bandwidth requirements for video decoding and video post processing over a single interface for off-chip DRAM shared by several processing modules. Also, it is necessary to control the data latency in such a way that high definition videos can be decoded and exhibited in real-time. The memory hierarchy design is the key point to reach real-time high definition video decoding and processing, minimizing the occurrence of memory access conflicts.

The paper presents in Section II the digital television set-top box SoC architecture and multi-channel memory hierarchy.

This work was supported by CAPES, CNPq and Finep, Public Brazilian Institutions for Education, Research and Technology Development.

Section III presents the memory hierarchy modeling for digital video systems and a memory hierarchy design analysis based on the memory model presented in this work. System behavior analysis and synthesis results are presented in Section IV. Finally, conclusions are presented in Section V.

II. DIGITAL TELEVISION SET-TOP BOX ARCHITECTURE

In this section it is presented the digital TV set-top box architecture compliant to the Brazilian digital TV standard and the description of its main processing modules. Also, it is presented the implemented memory hierarchy and an analysis of this implementation.

A set-top box is composed by several processing units, treating data at many points of the data flow and in different levels. Data processing can be grouped in three main categories: input and output data processing, which includes input data demultiplexing and video composition; signal decoding, which includes video and audio decoders and system control, user interface and application processing. Fig. 1 presents a block diagram of the main modules of the set-top box from the memory hierarchy view. In this system implementation, there are five memory channels accessing the external DDR2 memory: H.264 video decoder pixel output and MC cache input; display video controller and OSD data and CPU bus program and data interface.

There are four memory hierarchy levels in this system, from the local memories (level 0) to the external memory (level 3). The command bus is shared between all the memory interface channels (IF#) to provide one-way information about read or write data accesses from the processing modules to the external memory. Each IF channel is composed by a memory buffer and an address generator. The memory buffer is used to control the data access granularity to the main memory.

A. *H.264 Advanced Video Coding Standard*

ITU-T H.264/AVC [2] is the latest video coding standard defined by VCEG and JVT incorporated into the MPEG-4 specifications as Part 10. This video standard supports 8, 10 and 12-bit in 4:2:2 and 4:4:4 YCbCr color schemes. Fig. 2 shows the video reconstruction path in an H.264/AVC video decoder standard, starting from the compressed video bitstream and finishing in the YCbCr video that is stored in the DPB.

In Fig. 2, the syntax elements to inter-frame and intra-frame video reconstruction and the compressed residual data are extracted from the compressed video bitstream by the entropy decoding block. The residual data is decoded using fixed or variable length binary codes in the entropy parsing module to be processed by the inverse transform and quantization steps (IT and IQ). Using information decoded from the bitstream, the decoder creates a prediction block using intra or inter prediction modes. Inter-prediction (or Motion Compensation – MC) is the hardware module which reconstructs an actual frame from the reference frames stored in the DPB. Intra-prediction reconstructs each image block from its neighborhood, composed of already reconstructed macroblocks of pixels. Prediction and residual information are combined together to produce output image samples.

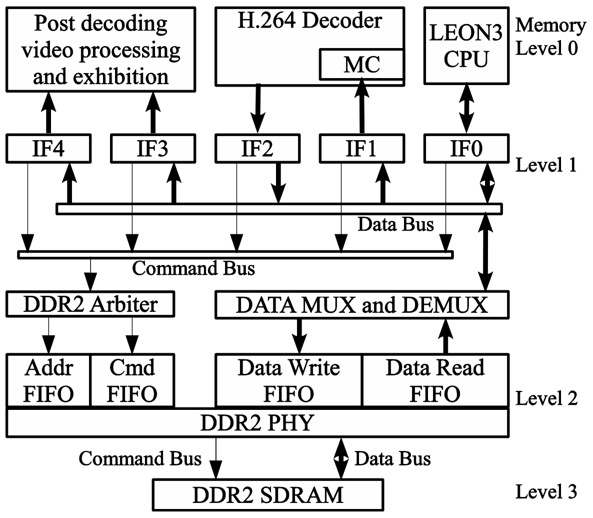


Figure 1. Block diagram of the memory system architecture for the SBTVD set-top box with a detail in the proposed memory hierarchy.

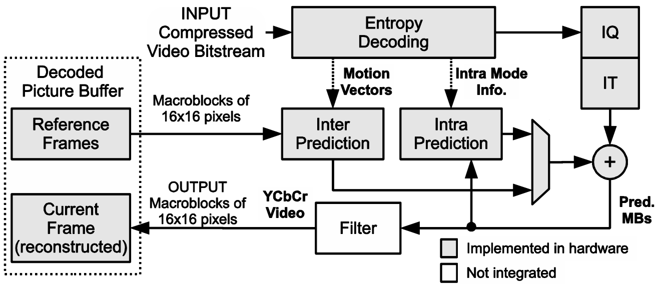


Figure 2. Block diagram consisting of the main hardware modules of the H.264 Advanced Video Decoder.

The H.264/AVC standard treats video pictures in small regions of 16x16 pixels, called macroblock of pixels. Moreover, the macroblock can be subdivided into an 8x8 block of pixels, a 4x4 block of pixels or a 4x1 line of pixels. The video decoder hardware architecture used in this work [6] produces pixels samples grouped in a line of pixels (LoP), containing four pixels. A sequence of LoPs is generated in the decoder output to form the macroblock of pixels. The video decoder has two memory interface channels: the decoded pixel output and the motion compensation cache channels. The inter-prediction module (Fig. 2) in this video decoder architecture contains a cache of 32 x 2.5 macroblocks used to reconstruct the current decoded macroblock.

B. *Multichannel Memory Controller*

The Multichannel Memory Controller (MMC) designed in this work implements a multiplexed interface from several modules to a single external memory, as is shown in Fig.1. An off-chip 64-bit wide DDR2 SDRAM memory running at 200 MHz is used as main system memory. The H.264 video decoder decodes digital images that are stored in the off-chip memory. Also, the MMC implements a data/instruction interface for the CPU through a bus standard interface.

The CPU generates the OSD image and stores it in the main system memory. Two memory channels are implemented to allow the composition of the decoded video and the OSD by the video processing blocks. The MMC supports the OSD feature, including a graphics processor output and a CPU interface, and channels to manage the decoded video sequences: the MC channel, video decoder channel and video output channel. Each channel interface consists of an address generator and a data buffer. The address generators are used to organize the macroblocks of pixels in reference pictures. In the case of video display and the OSD, the buffer has a size of 240 macroblocks. This is necessary to store two entire lines of macroblocks, considering full HD video resolution. One line is loaded while the other line is being exhibited.

An arbiter is implemented in order to control how each process access data in the main memory. Each process has a priority in the implemented arbitration scheme, considering a separation between high or low latency and high or low bandwidth. Each process accessing the off-chip memory is classified as latency sensitive (LS) or bandwidth sensitive (BS) processes. An LS process is characterized to require immediate access to the main memory channel and to use the channel for few memory transactions. On the other side, a BS process is characterized by the generation of a continuous access sequence to the main memory in long transactions, without the need of immediate access. In this implementation, the best approach to multiplex the memory channel between processes is using a priority based arbitration scheme with preemption. The arbitration algorithm with preemption is shown in Fig. 3.

In this arbitration scheme, a higher priority process (LS process) can preempt a current transfer (BS process), allowing one single transfer. Afterward, the lower priority channel resumes its transfer. The LS processes also are characterized to perform intermittent accesses to the memory channel. This characteristic is important to avoid starvation of BS channels or lower priority LS channels. Data transfer size is adjusted to each IF channel according to the module requirements.

C. Graphics Generation and Exhibition

The On-Screen Display channel is a dedicated channel of the multichannel memory controller used to provide CPU generated graphics data to the video processing blocks. It is the source of OSD data. The OSD channel is responsible for reading the CPU generated image from memory and transmitting it to the video processing blocks. The OSD channel provides pixels in YCbCr 4:4:4 format to the video processing blocks.

The video channel is a dedicated channel of the multichannel memory controller used to provide decoded video data to the video processing blocks. It is the source of decoded video. The video channel gets the correct image from the buffer of decoded video frames to be transmitted to the following video processing blocks. The DPB manages the buffer addressing for video decoded pictures storage and retrieval. It also is responsible for frame rate conversion by providing the same decoded video picture multiple times. The video channel is responsible for converting the data output format of the decoder in a raster format used by the display. Since the decoded image is stored in macroblocks, the video channel must read a full line of macroblocks from the current decoded video frame.

```
if (LS_request){   //latency sensitive request
  while (LS_request) {
    allow_data_transfer(LS_channel);
  } // end of LS request
else if (BS_request){   //bandwidth sensitive request
  while (BS_request) {
    allow_data_transfer(BS_channel);
    if (LS_request){   // interrupt BS for LS transfer
      allow_data_transfer(LS_channel);
    } // end of LS transfer
  } // end of BS transfer
} // end of LS request
```

Figure 3. Arbitration algorithm with preemption.

III. DIGITAL VIDEO SYSTEM MEMORY MODEL

In this section we describe a general memory model for multiple processing modules sharing the same memory port. The memory model for a digital video decoding system requires an organization in memory levels in order to optimize data accesses to the memory port. In a complex video system, the main memory is shared between several modules like processors, video decoders, video output controllers and other peripherals. We define each memory module interface as a channel. Also, different channels access data in the main memory with different requirements: access granularity, bandwidth and latency. First of all, it is defined that:

1. *Main memory* is the system memory where several processing modules share information;

2. *Granularity* is the number of data units transferred continuously between a processing module and the main memory, either for writing or for reading;

3. *Latency* is the time that a processing module has to wait to perform a data transfer with the main memory, either for writing or for reading data;

4. *Effective Memory Bandwidth* is the memory bandwidth used to transfer data, without considering cycles used for memory operation as bank and row activation and memory auto-refresh.

Latency becomes an issue in a system with several modules sharing the same main memory. In video decoding systems, latency is critical to reach real-time decoding and depends on several factors: how data is transferred between the channels and memory, the granularity of data transfers and memory bandwidth. Data transfers granularity depends on how the system is built and how it processes data. Data transfer bandwidth is dependent on picture size and frame rate. Picture size is determined by the number of sampling points and pixel depth (or bits per pixel).

A. General View of the Memory Hierarchy

We will quantify the system latency using levels of memory with different sizes and relate this with bandwidth and granularity, in order to reach the optimum memory hierarchy design. Fig. 4 presents a general view of a memory hierarchy containing four memory levels. An arbitration scheme must be used to manage the switching activity between channels considering each channel's priority. Latency in the memory levels 0 and 1 are not critical, because these levels have a very predictable behavior. The latency problem is in the levels 2 and 3 of the memory hierarchy, because these levels are shared between several processes in the system.

The design of the memory hierarchy of the SoC must consider the characteristics of different access types to each channel connected to the external memory. We propose a model in this work to find the best fit of the memory system at different levels in video processing systems. The latency becomes a major problem in a system containing several levels of memory and channels sharing a single memory bus [7]. Therefore, the memory design is intended to reduce the effect of latency over the data transfers between the SoC and external memory. The memory hierarchy model design considers different buffer sizes in each memory level. Also, it is considered different data transfer granularities between the memory channels and the external memory. The smaller the data transfer granularity, the longer the time to transfer the information. This occurs because the need to send row and bank active and precharge commands to the SDRAM [8].

B. Memory Model for Data Latency

In this model, we calculate the memory hierarchy latency from the response time (**t_{RESP}**), formed by three parts as shown in (1) that are related to the system memory levels: **$t_{REQ}(c)$** is the time between sending a command request (RD or WR) to the arbiter and receiving the acknowledge; **$t_{WAIT}(c)$** is the time waiting in the internal queue at the memory controller and; **t_{PROC}** is the command processing time in the memory controller. Fig. 4 illustrates the memory model and each time interval in the different memory levels for a channel **c**.

$$t_{RESP}(c) = t_{REQ}(c) + t_{WAIT}(c) + t_{PROC} \qquad (1)$$

The time required by the arbiter to process the request (**t_{REQ}**) for a channel **c** is presented in (2). This equation considers the channel priority in the arbiter processing order and the data transfer granularity for each channel. Equation (2) represents the worst case scenario, when all channels are requesting access to the main memory at the same time. The granularity is represented by **n_{ACC}** that is the number of consecutive accesses to write the commands in the queue **M** (Fig. 4). When a channel **c** writes data to the memory, it takes two clock cycles (***k*** =2), however for reading data it takes one clock cycle (***k***=1).

$$t_{REQ}(c) = \sum_{i=0}^{c} k(i) \cdot n_{ACC}(i) \qquad (2)$$

The time waiting (**t_{WAIT}**) for the command queue to process all buffered commands depends on the command buffer size (**n_{CMD}**) and the number of consecutive accesses (**n_{ACC}**) multiplied by the time to process the queued commands and memory row changes, as is presented in (3). The time needed to perform a memory row change is represented by **t_{ACTPRE}**.

$$t_{WAIT}(c) = k(c) \cdot n_{CMD} + t_{ACTPRE} \cdot \frac{n_{CMD}}{n_{ACC}(c)} \qquad (3)$$

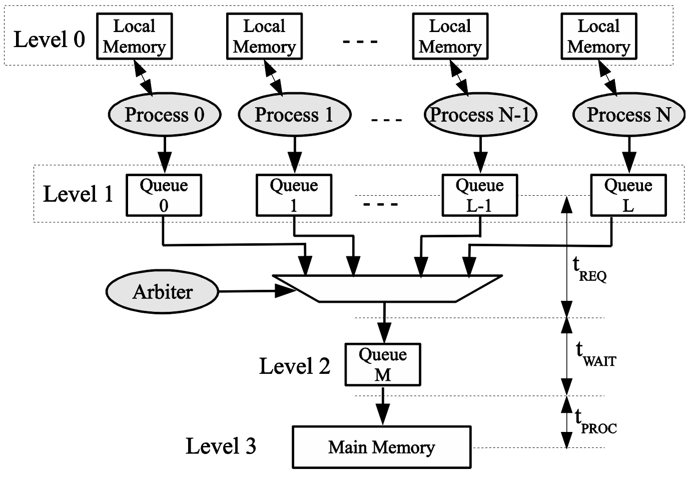


Figure 4. Memory hierarchy levels with *L+1* channels for *N+1* system processes: *L0* is the local SRAM memory; *L1* is the process to main memory queue; *L2* is the memory controller to main memory queue and; *L3* is the main memory.

As is presented in (3), **$t_{WAIT}(c)$** can be reduced by either reducing the command buffer size (**n_{CMD}**) or by augmenting the number of consecutive accesses (**n_{ACC}**) to the memory. The effective memory bandwidth is increased by performing multiple consecutive DRAM accesses, reducing the row activation and bank precharge cycles (**t_{ACTPRE}**).

The memory controller needs a process time (**t_{PROC}**) to execute one read or write command. The process time varies with the need to execute memory bank or row changes. Also, it depends on the internal architecture of the memory controller. In our computation, it will be used a fixed number of cycles for **t_{PROC}**, because the row change and granularity effects has been already computed in (3).

The reduction of **t_{RESP}** is the way to maximize the effective memory bandwidth through an efficient memory hierarchy design. This analysis aim adjusts the queue sizes and access granularity for transfer data with the system main memory.

C. SoC Memory Hierarchy Design and Analysis

An analysis of system main memory bandwidth necessary for this SoC is presented in Table I considering different video and OSD resolutions. The frame rate for interlaced video is 30 frames per second while for progressive video is 60 frames per second. In high-definition videos, nearly half the bandwidth of the memory is used in the data transfer of video and graphics.

Memory bandwidth increases significantly with an efficient management of data transfer, as presented in the last two rows of Table I. In the presented case, data transfers between memory and system use a single access for read or write 32 bytes. In the case of using eight bursts, the effective memory bandwidth increases about five times when comparing to single burst access. This occurs due to the reduction of memory bank and row activation and precharge.

TABLE I. MEMORY BANDWIDTH FOR DECODING AND PROCESSING

	Decoded Video and OSD Sizes [c]				
	1080i 1080p	***1080i 720p***	***1080i 480p***	***480p 480p***	***240p 240p***
OSD resolution (width x height)	1920 x 1088	1280 x 720	720 x 480	720 x 480	320 x 240
Video resolution (width x height)	1920 x 1088	1920 x 1088	1920 x 1088	720 x 480	320 x 240
Total BW used [a]	30.6 %	23.2 %	19.7 %	7.0 %	1.5 %
Total BW using single burst [b]	182.1 %	139.4 %	118.4 %	41.8 %	9.3 %
Total BW using eight bursts [b]	34.2 %	26.2 %	22.2 %	7.8 %	1.7 %

a. For 3.2e9 peak bandwidth for a DDR2 SDRAM off-chip with 64-bit wide and 200 MHz.

b. For 200e6 clock cycles per second to transfer data in a 64-bit bus wide.

c. 1080i is 1920x1088 interlaced and 1080p is 1920x1088 progressive.

For the memory interface, it is defined that a minimum block transfer size is 1 burst and the maximum is 72 bursts. A set of 1x4 and 4x4 pixel occupies less than 32 bytes that are transferred in a single data transfer in burst length equals to 4. A block of 16x16 pixels (macroblock) in YCbCr 4:2:0 format is transferred in 12 consecutive bursts and in YCbCr 4:4:4 in 18 bursts.

Table II presents the analysis of the time spent by the arbiter in processing command requests, based in (2). Table III presents the impact in time processing the queue for different L2 buffer sizes and granularity based in (3). For this calculus, $\boldsymbol{k}$ equals to 2 cycles and $\mathbf{t_{ACTPRE}}$ equals to 6 clock cycles. As Table III presents, the increase in command buffer size augments the time that a buffered command waits until be processed by the memory controller. The increase in the granularity of data accesses causes the reduction in time to process the command queue. However, the increase in the $\mathbf{n_{ACC}}$ parameter causes an augment in the waiting time to the other channels, as is shown in Table III. Thus, there is a compromise between the granularity of access and size of command buffer.

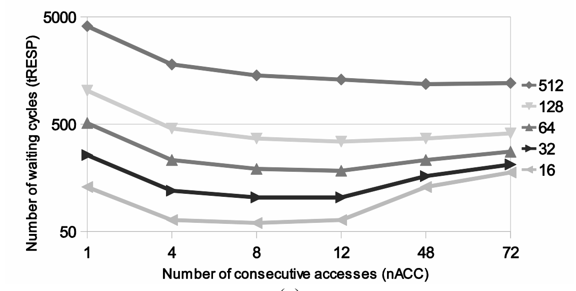
Combining the data of Tables II and III comes to the curves shown in Fig. 5a and Fig. 5b. The minimum points showed in both curves are between granularities of 8 and 12 bursts. Also, because the video decoder uses a granularity of macroblocks, in this implementation we choose to design a memory hierarchy using $\mathbf{n_{ACC}}$=12. The granularity to minimize the response time of access is set to 12 consecutive bursts. This means that every 12 consecutive accesses to the same channel **c**, the arbiter can preempt the channel **c** and pass control to another channel of higher priority (**c-1**). Likewise, when the channel **c** ends a sequence of 12 hits, the arbiter returns control to the channel that was preempted.

TABLE II. NUMBER OF CLOCK CYCLES TO GIVE ACCESS TO A CHANNEL ($\mathbf{t_{REQ}}$) USING DIFFERENT DATA TRANSFER GRANULARITIES AT THE WORST-CASE SCENARIO.

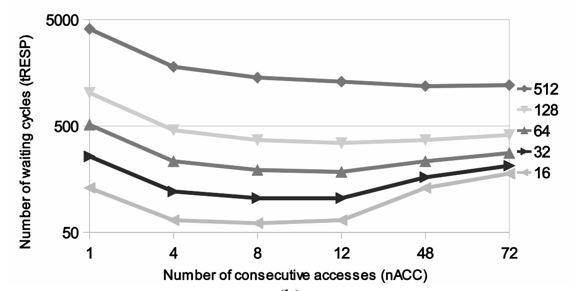
Access Channel	L	Number of Consecutive Accesses ($\mathbf{n_{ACC}}$)				
		1	*4*	*8*	*12*	*48*
CPU channel	0	2	8	16	24	96
MC Cache	1	3	9	17	25	97
Decoder channel	2	5	11	19	27	99
OSD channel	3	6	12	20	28	100
Video channel	4	7	13	21	29	101

TABLE III. NUMBER OF CYCLES EXPENDED TO PROCESS A COMMAND QUEUE IN DIFFERENT SIZES AND ACCESS GRANULARITIES AT THE WORST-CASE SCENARIO.

Commands Buffer Size ($\mathbf{n_{CMD}}$)	Number of Consecutive Accesses ($\mathbf{n_{ACC}}$)				
	1	*4*	*8*	*12*	*48*
512	4096	1792	1408	1280	1088
128	1024	448	352	320	272
64	512	224	176	160	136
32	256	112	88	80	68
16	128	56	44	40	34



(a)



(b)

Figure 5. Response time analysis for read data in the external memory in different scenarios. The charts present a comparison between access granularity and buffer size for the CPU channel (a) and for the MC channel (b). Data latency increases with the increase in the command buffer size.

IV. MEMORY HIERARCHY HARDWARE IMPLEMENTATION RESULTS AND ANALISYS

In this section an analysis of the simulation and synthesis results are presented together with an evaluation of the external memory bandwidth and data latency in the video processing architecture. The STB SoC was implemented in VHDL language and verified using a Xilinx-Digilent ML509 board [9,10]. The implementation results for a Xilinx XC5VLX110T FPGA are presented in Table IV.

TABLE IV. SYNTHESIS RESULTS FOR XILINX XC5VLX110T FPGA OF SOME SET-TOP BOX SOC HARDWARE MODULES.

SoC Hardware Module	Slice Regs	Slice LUTS	Block RAMs	Mem. Level
Available in FPGA	***69,120 (100%)***	***69,120 (100%)***	***148 (100%)***	**-**
Multichannel Memory Controller	2,714 (4%)	2,739 (4%)	40 (27%)	L1
PHY DDR2 controller	2,277 (4%)	1,749 (3%)	5 (3%)	L2
H.264/AVC Video Decoder	9,244 (13%)	12,805 (18%)	43 (29%)	L0
Video Processing	1,408 (2%)	5,456 (8%)	13 (9%)	L0
Leon-3 CPU	4,868 (7%)	6,618 (9,5%)	29 (20%)	L0

The set-top box prototype with the proposed memory hierarchy has been validated on FPGA board using real video sequences (one-seg level 3.1) from broadcast television signals in the ISDB-T Brazilian standard [1]. The memory hierarchy was implemented to perform 12 consecutive accesses ($\mathbf{n_{ACC}}$) to the external memory with a variable commands buffer size ($\mathbf{n_{CMD}}$).

The system currently does not have an RF frontend (antenna input). Therefore, the captured video bitstreams have been stored in a prototype board memory to be read and processed by the H.264 video decoder. While decoding of H.264 video in real-time by the video decoder, the CPU generates graphics and closed captions. Both CPU and video decoder are sharing the external DDR2 SDRAM memory simultaneously and video processing and exhibition occurs at real-time (60 FPS).

An analysis of external memory bandwidth utilization is presented in Table V, considering the memory accesses needed to generate and display the decoded video and the OSD video. These results were obtained from behavioral simulation. Data showed in Table V compare the maximum waiting times for data and for a permission to use a channel for the CPU and MC Cache. Both modules are critical for system performance, because a large delay to read data from memory can stall the CPU and system bus or the video decoder.

The display frame rate is set to 60 FPS. A low frame rate (5 fps) is considered in the generation of the OSD graphics by the CPU. The channel format of the video channel is YCbCr 4:2:0, while the OSD video is in the YCbCr 4:4:4 format. Each channel has an associated priority in the arbitration scheme, CPU and MC have the highest priorities and are classified as LS channels, and decoder has intermediate priority and also is an LS channel. OSD and video output have lower priorities and are classified as BS channels.

Latency reduction is achieved by dimensioning in a properly way the buffer size in level 2 of the memory hierarchy, adjusting the $\mathbf{n_{CMD}}$ parameter in the system. Table V shows the wait time increase with the increase of the buffer size. Also, with the increase of the buffer size there is a reduction in the time waiting for a permission to access the memory, because there is a reduction of channel switching.

When the buffer size approaches the size of a consecutive data transfer, there is an increase in the occurrence of channel switching. Also, the increase of time waiting for data is more significant than the time for permission. With this analysis we confirm that the use of a 32 positions buffer size for commands to the main memory is the best implementation in this memory hierarchy. This approach can reduce data latency in almost 78%.

TABLE V. AVERAGE TIME OF WAITING FOR DATA ($\mathbf{t_{RESP}}$) AND WAITING FOR THE ARBITER ($\mathbf{t_{REQ}}$) FOR DIFFERENT L2 BUFFER SIZES.

Memory Channel	**Parameter**	**Commands Buffer Size (n_{CMD})**				
		16	***32***	***64***	***128***	***512***
CPU	t_{REQ}	68	69	42	42	40
	t_{RESP}	91	123	213	309	902
MC	t_{REQ}	38	23	21	19	11
	t_{RESP}	53	56	94	96	94

V. CONCLUSIONS

We present in this paper an implementation analysis methodology of an efficient memory hierarchy in terms of latency and buffer sizes. Increasing the memory transfer granularity is better for managing multiple channels because it reduces the switching between channels. Increasing the number of consecutive bursts in the DRAM causes an increase in latency because a large number of commands is queued in the memory controller. Therefore, the command queue should be processed as soon as possible to reduce latency in the memory port.

The presented memory hierarchy architecture is designed to optimize data transfers from several channels to one main memory. Consecutive memory accesses are grouped and scheduled in memory buffers and channel arbitration is performed to increase the effective memory bandwidth. Also, the paper shows that the memory hierarchy can approach the effective bandwidth used of the peak bandwidth required when processing video.

Results obtained in simulation shown that the implemented architecture with multiple channels sharing the same memory causes an effective memory bandwidth reduction in just about 12.5%, if compared to a single memory channel system. The effective memory bandwidth reduction in the memory port is balanced with the data latency reduction that can be reduced by 78% with proper buffer sizing.

REFERENCES

[1] ABNT. NBR15604:2007 – Terrestrial Digital Television – Receivers. 1st ed. 30/11/2007.

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

[3] ISO/IEC 14496-3:2005 – Information technology – Coding of audio-visual objects – Part 3: Audio, Dec. 2005.

[4] C-H. Li, W-H. Peng, and T. Chiang, "Design of memory sub-system with constant-rate bumping process for H.264/AVC decoder," Consumer Electronics, IEEE Transactions on , vol.53, no.1, pp.209-217, Feb 2007.

[5] D. F. Finchelstein, V. Sze, and A. P. Chandrakasan, "Multicore processing and efficient on-chip caching for H.264 and future video decoders," IEEE Transactions on Circuits and Systems for Video Technology. Vol. 11, Issue 11, Piscataway, USA. 2009.

[6] A. B. Soares, A. C. Bonatto, and A. A. Susin, "Integration issues on the development of an h.264/AVC video decoder SoC for SBTVD set top box". In Proceedings of the 24th symposium on Integrated circuits and systems design (SBCCI '11). ACM, New York, NY, USA, pp.-125-130, Sep 2011.

[7] K.-B. Lee and T.-S. Chang, Essential issues in SOC Design – SoC Memory System Design, Springer Netherlands, 2006.

[8] JEDEC, JESD79-2F: DDR2 SDRAM specification, JEDEC Solid State Technology Association, Virginia, USA, 2009

[9] Xilinx, "Xilinx University Program XUPV5-LX110T Development System". http://www.xilinx.com/univ/xupv5-lx110t.htm.

[10] Xilinx, "ML505/506/507 Evaluation Platform User Guide", UG347, October 7, 2009.