Heterogeneous Multi-Core Platform for Consumer Multimedia Applications

Peter Kollig and Colin Osborne
NXP Semiconductors

Southampton, U.K.
Peter.Kollig@nxp.com

Tomas Henriksson
NXP Semiconductors Research

Eindhoven, The Netherlands
Tomas.Henriksson@nxp.com

***Abstract*—This paper presents a multi-core SoC architecture for consumer multimedia applications. The comprehensive functionality of such multimedia systems is described using the example of a hybrid TV application. The successful usage of a heterogeneous multi-core SoC platform is presented and it is shown how specific challenges such as inter-processor communication and real-time performance guarantees in physically centralized memory systems are addressed.**

***Keywords*—component; multiprocessor; TV; physically centralized memory system**

I. INTRODUCTION

Multi-core architectures have recently attracted substantial attention because of the increasing difficulty to push processor core speeds beyond the few GHz mark already reached some years back. Therefore the computer devices industry has recently focused on instantiating the same processor core multiple times (dual-core, quad-core) and improving communication mechanisms between the multiple cores. In contrast to this, the consumer devices industry has always looked at heterogeneous compute platforms that utilise a mix of industry-standard CPU, fixed-point DSP, VLIW, and function-specific HW cores, an example being the Nexperia platform [7], [14]. An important advantage of the heterogeneous platform approach is that algorithms can be executed on the processor core that is best suited for them. Functional subsystems, consisting of several co-operating algorithms, are implemented on a single processor core, possibly supported by function-specific HW cores. The functional subsystems have well defined communication interfaces, which make debug and system integration effort low. Recent advances in CMOS technology allow integration of an ever growing number of processor cores on a single die [12]. This high level of integration offers a cost reduction, whilst at the same time increasing competition for usage of scarce shared HW resources.

The aim of this paper is to describe the functionality and architecture of state-of the-art SoCs for consumer multimedia applications. We focus on multi-core aspects and discuss the challenges and solutions caused by the provision of shared HW resources.

The rest of this paper reviews selected prior work on multi-processor architectures in section II. Section III reviews the kind of multi-media applications found in a typical Consumer Electronics SoC. Section IV introduces the underlying SoC architecture and Section V investigates specific performance challenges in the design of large multi-core SoCs. Trends and a recent SoC implementation are discussed in Section VI and Section VII draws conclusions.

II. RELATED WORK

Culler et al. classify multiprocessors according to their communication architecture, i.e. how they cooperate to solve a large problem [1]. Although based on general purpose computing, the classification is also useful for multimedia applications. The three categories are shared address, message passing, and data parallel processing. It is a logical classification, where each of the three categories can be realized on any physical architecture, e.g. most operating systems implement a message passing abstraction on top of a physically centralized memory architecture. The latter two categories are better suited for multimedia because there is a high level of fine-grained data parallelism in many of the algorithms and a system typically is constructed as a pipeline of functional subsystems, which pass video frames and audio frames from one functional subsystem to the next.

For multimedia, general purpose processors have been extended with multimedia specific instructions, e.g. [2] and [4]. Many of those extensions aim to utilize fine-grained data parallel processing. Although successful for applications such as video decoding on PCs, the power usage of general purpose processors is too high and computational capacity is too low for high-end consumer electronic SoCs.

A different architecture evolved from the digital video decoders used in set-top boxes, DVD players, and digital TVs, e.g. [3]. This architecture is based on several function-specific HW cores that are orchestrated by a microcontroller. That heterogeneous architecture allows for SoCs with low power consumption and high computational performance at the expense of flexibility. With the advance of manufacturing technology more and more function-specific HW cores have been integrated together with programmable processors on a SoC. Nomadik is a specific implementation of this architecture style that makes use of a distributed system object component

978-3-9810801-5-5/DATE09 © 2009 EDAA

(DSOC) programming model [5]. This general architecture, consisting of a microcontroller supported by several domain- or function-specific HW cores is the style used in most NXP products, e.g. [7]. This is because it offers high computational performance at acceptable power consumption. Furthermore, it has been proven that the development effort is challenging but manageable.

The development challenges include mapping of functional subsystems (e.g. video decoding) to SoC HW resources, choosing the right level of flexibility, and managing the design complexity. Some mapping challenges have been tackled by several automation approaches, e.g. [6]. The algorithms try to choose the best suited SoC HW resource option, but most of them fail because of the multitude of operational modes of a multimedia SoC, or because they put too stringent rules on the hardware they support. The most fundamental reason why those automated mapping attempts are not needed for advanced multimedia SoCs is however because only one implementation exists of most functional subsystems, so there are no options to choose from. There is no time or budget to try out several different implementations. Therefore the mapping is and will continue to be done by hand to a large extent.

III. MULTIMEDIA APPLICATIONS

Multimedia applications implemented on Consumer Electronics (CE) devices span a vast range of functionality, from audio decode such as MP3 [8] via video decode such as H.264 [9] up to advanced picture quality (PQ) processing such as frame rate up-conversion and Motion Accurate Picture Processing (MAPP) [10]. Hybrid TV solutions are a very good example because they are virtually capable of executing any of these multimedia applications.

A. Audio Processing

Audio processing in a hybrid TV consists of analog TV standards decode, digital audio decode and audio post processing.

- The analog standards decoder is used for demodulation and decoding of analog sound standards, e.g. FM A2, BTSC, EIAJ, NICAM, etc.
- The digital audio decoder is used for compressed digital audio formats including MPEG1 layer 1, 2, 3, MPEG4-AAC, MPEG4-HE AAC, etc.
- Audio post processing includes support for Dolby Virtual Surround, SRS TruSurround, bass enhancements and others. Of particular importance are algorithms that improve the sound quality of small speakers such as those fitted into today's thin LCD panels

Besides the high computational complexity of the above functions, there is also a large number and diversity of audio related interfaces, including analog IF processing, SPDIF and I2S for PCM and compressed digital audio, as well as on-chip DACs for line outputs.

B. Video Decoding

The situation for video decoding is similar to that for audio decoding. Generally there are a number of video sources in a hybrid TV, including:

- Analog video standards such as PAL or NTSC. These are provided by an analog tuner and traditionally decoded in function-specific HW cores.
- Digital uncompressed video data such as HDMI or DVI.
- Digital compressed video data provided by digital tuners/demodulators, solid state storage or, increasingly, from Internet enabled systems.

Early systems with compressed digital video support supported only MPEG2, but today, the adoption of HD formats has seen a transition to the higher compression H.264 and VC1 coding standards. Besides these well established formats, there has been a proliferation of an increasing number of new codecs such as Sorenson, VP6/VP7, On2, etc, caused by the recent introduction of Internet based video distribution.

C. Picture Quality Enhancement

PQ enhancement can be divided into two categories. The first category addresses the repair of noise and improvements of sharpness in analog systems as well as the repair of compression artifacts seen in digital systems. The second category enhances video quality by adding new information such as frame insertion based on motion adaptive algorithms. Some enhancement algorithms are generic and applicable for all displays whereas other enhancement algorithms are specifically developed for one display type.

D. Web 2.0 Functionality

Increasingly, digital TVs also process content provided by internet related services. This requires the addition of graphics acceleration, flexible video decoding and web browsing functionality.

IV. GENERAL SOC AND SYSTEM ARCHITECTURE

The previous section introduced multimedia applications implemented in hybrid TV systems. The mapping of the functional subsystems onto SoC HW resources is based on a number of considerations:

- Support – of industry standards. This is especially important for processors that are programmed by the set makers. Industry standard CPU cores such as the ones from ARM and MIPS have extensive tool chain and library support that eases the application design and debug.
- Performance – computationally intensive algorithms such as HD H.264 decode cannot be implemented cost effectively on a general purpose processor because of the computational complexity. Instead a function-specific HW core is used.

TABLE I. MAPPING OF APPLICATION TO SOC HW RESOURCES

Algorithm class	Resource mapping	Reasoning
Analog standards audio decode	Fixed point DSP	Legacy
Digital compressed audio decode	VLIW processor	Flexibility, availability of codecs, cost of HW
Audio postprocessing	Fixed point DSP	Performance, flexibility
Analog standards video decode	Function-specific HW	Performance, legacy
Digital uncompressed video decode	Function-specific HW	Performance, legacy
Digital compressed video decode – established codecs	Function-specific HW with control processor	Performance, cost of HW
Digital compressed video decode – new codecs	VLIW processor	Flexibility, standards evolution
Picture quality: artefact repair	Function-specific HW, partly on VLIW processor	Performance, flexibility, cost of HW
Picture quality: motion adaptive picture processing	VLIW processor with HW co-processor	Performance, flexibility, cost of HW
Content browsing and control	Industry standard CPU	Support

- Flexibility – evolving standards require flexibility in implementations so that new codecs can be added without the need for a new SoC.
- Legacy – functions such as analog TV standards have been stable for many years and efficient implementations are available, mapping to different SoC HW resources is normally not justifiable
- Re-usability – implementation, integration and verification are time consuming tasks and sometimes it is appropriate not to implement a function on the most optimum SoC HW resource in order to make it reusable in future SoCs

The remainder of this section looks at the mapping of functional subsystems to SoC HW resources and introduces a SoC architecture that allows an efficient SoC implementation and subsequent system integration.

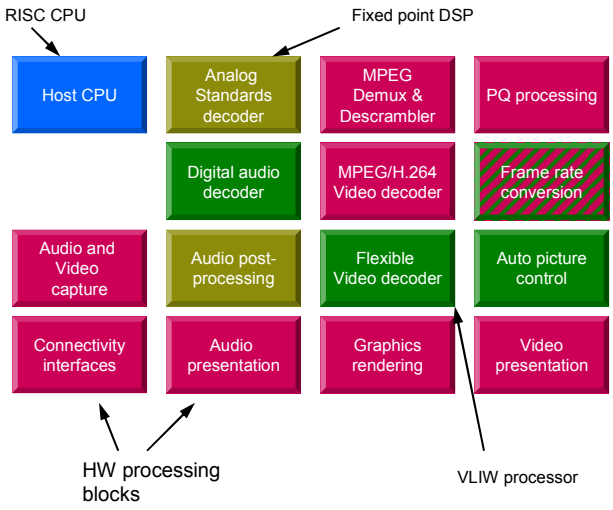

Figure 1. Application mapping

A. *Application Mapping*

The mapping of functional subsystems to SoC HW resources within a typical hybrid TV SoC is sketched in Figure 1. More detailed explanations are given in TABLE I. These describe the functional subsystems, the mapping to SoC HW resources, as well as the principle reasons for the mapping to a specific SoC HW resource.

The application mapping obtained contrasts to homogenous multi-core processor architectures where all processor cores are of the same type, where the challenge is in finding sufficient application level parallelism that allows scheduling of multiple threads onto the identical processor cores.

B. *SoC Architecture Drivers*

This section investigates the choice of processor cores and considers the SoC infrastructure that allows an efficient mix of different types of processor cores and function-specific HW cores to access shared resources in the SoC such as the interface to external SDRAM.

1) *Processor cores*

Investigation into the mapping of functional subsystems of multi-media applications to SoC HW resources has shown that a number of different processing core types are needed to implement all features and functions in cost and power efficient implementations. This includes:

- Host CPU – Industry standard cores such as MIPS 24k are used. Typically, these have a large application code base (tens of Mbytes) and thus need to access code and data in an external SDRAM memory.
- VLIW processor cores – NXP TriMedia based cores are used. This provides scalability and processing power and it allows us to exploit the ecosystem of software implementations of video and audio codecs. The VLIW processor cores exploit fine-grained data parallelism. Code and data segments are typically large, so the VLIW processor cores need access to SDRAM. TriMedia cores are often supported by local function-specific HW cores.
- Embedded control CPUs – small to medium sized code base, for architectural and commercial reasons often use processor cores from the same processor provider

as the host CPU (for example MIPS 4k). For architectural consistency, such processors are hooked up to the same bus structures as the host CPU or VLIW processors.

- Fixed point DSP – generally deeply embedded into the SoC architecture; executing code out of small on-chip memories. Often, DSP cores are connected into the SoC infrastructure via HW semaphore mechanisms.
- Function-specific HW cores – massively parallel computation cores that make use of fine-grained parallelism as well as coarser parallelism. Some are connected to external real-time interfaces, e.g. HDMI, and therefore need real-time performance. They typically process large data sets (e.g. video frames) and thus need access to SDRAM.

2) SoC infrastructure

Additional drivers for the architecture of the SoC are:

- Physically centralized memory architecture to optimize the total cost of electronics in the total application.
- Data paths that decouple functional subsystems by using communication through system main memory, thus implementing message passing between functional subsystems.
- Shared, hierarchical control buses to provide communication paths whilst at the same time avoiding congestion on local bus segments.
- On-chip semaphores and interrupt connectivity to provide HW support for inter-processor communication (IPC).

An example for the resulting data and control paths is given in Figure 2. Functional subsystems such as transport stream (TS) de-multiplexing or video decode are decoupled by paths through the main memory. This design style allows developing, debugging and validating of functional blocks in complete isolation. Softening of real time constraints through memory decoupling reduces overall system integration effort and reduces risk of dynamic system behavior issues.

Control paths are described by green arrows in Figure 2. This shows that the application running on the Host CPU sends control commands to the various subsystems, some in function-specific HW cores (demux), some in mixed function-specific HW and SW (PQ processing) or on a control CPU (video

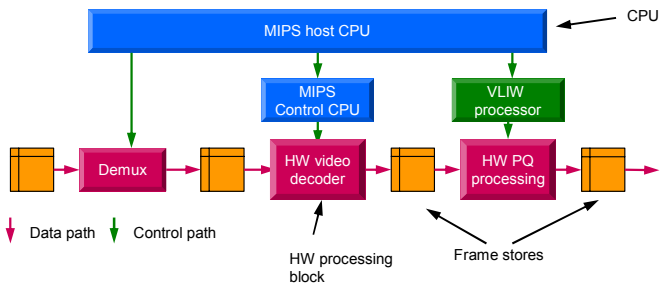

Figure 2. Data and control paths

decode). The actual control communication is mostly implemented via global semaphores and interrupts. This communication style decouples functional subsystems and thus eases overall system integration.

Data flow is shown as red arrows in Figure 2. This shows how the frames move from one processing core to the other via the frames stores in SDRAM. A frame of video or audio data is only accessed by one processor or function-specific core at a time. When a frame has been completed, the data is flushed from caches and local scratchpad memories into SDRAM before the processing core signals that the frame is ready. Data coherency is explicitly managed in software. The use of temporal algorithms for picture quality enhancement implies that each video processing core owns several video frames. This makes the processing latency from video input to display a tough challenge, which is partly addressed by processing cores that execute several algorithms on the same frames in one temporal loop.

A conceptual view of the multi-core SoC implementation is shown in Figure 3. This shows a number of different processors types (host CPU, VLIW, DSP) that connect via a shared control bus. The control bus is segmented and isolated using bus bridges so that control traffic is localized, yet providing the possibility to access control registers on other bus segments. A global semaphore and interrupt module is provided. All processor cores and function-specific HW cores have access to the physically centralized external SDRAM memory via the memory subsystem, shown in the top of Figure 3.

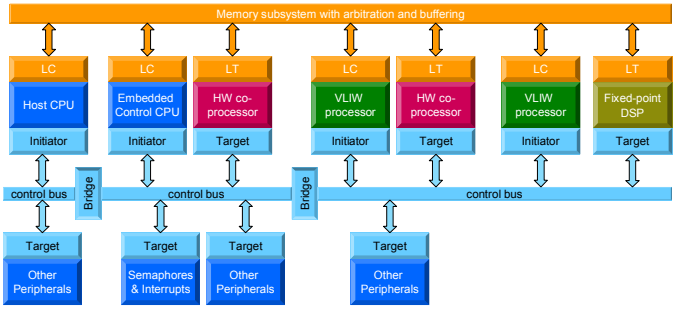

Figure 3. Conceptual view multi-core SoC architecture

V. Challenges in Memory Subsystem Design

One of the main challenges in multi-core SoC design is the design of the memory subsystem [11]. All the processing cores are masters (in AXI terminology) and the Main Memory Interface (MMI) is slave. The MMI is thus shared between all the processing cores. The example in Figure 4 shows how two processor cores and many function-specific HW cores are connected to the memory subsystem.

The MMI needs to have high utilization in order to keep the total electronics cost low. The wider the MMI, the more expensive the system gets. A wide MMI is expensive because several SDRAM devices are needed, the SoC needs to have many pads, and a package with many pins must be used. The processor cores and the function-specific HW cores that all need access to the SDRAM via the MMI initiate traffic with

different characteristics and have different requirements on the service they need from the MMI. Some require high bandwidth, others require low bandwidth. Some have large transactions, others have small transactions. Some have well-aligned transactions, others do not. By well-aligned we mean that the address of the transaction is a multiple of the transaction size. Some can tolerate long access latency whereas others require short access latency.

The MMI is typically a DDR2 or DDR3 interface between the multimedia SoC and one or more SDRAM devices. The DDR2 and DDR3 protocols require a certain order of DDR commands to read and write data. It is not possible to transfer data every clock cycle over the interface. Some clock cycles are wasted because of bank conflicts, read/write turn-around time, and refresh operations. The challenge is to maintain high utilization of the MMI and at the same time satisfy the service requirements of all the processor cores and the function-specific HW cores.

Typically, programmable processors require low average access latency, because their cache miss penalty is strongly related to the access latency, indicated by LC (Latency Critical) in Figure 3. There is on the other hand no restriction on the latency of one single transaction because the deadlines for the tasks on the programmable processors are in the range of milliseconds and each task execution includes thousands of transactions.

Many function-specific HW cores can make use of prefetching read accesses and posted write accesses. Such function-specific HW cores are thus latency tolerant as long as the memory subsystem can buffer the requests and/or data during the transactions. This is indicated by LT (Latency Tolerant) in Figure 3. Some function-specific HW cores require data to be available at exact points in time and cannot be stalled. The display controller is an example of such a function-specific HW core because it has to send pixel data to the display based on an external clock signal. If the display controller prefetches data long in advance and the memory subsystem buffers the data, then it can be mathematically guaranteed that there will always be data on time for the display controller if the MMI guarantees to service transactions from the display controller with a limited latency [12].

To be able to service the programmable processor cores with sufficiently low latency, most of the function-specific HW cores must be latency tolerant and the MMI arbiter must make use of that fact. It is a challenge to make the function-specific HW cores latency tolerant, especially for data-dependent algorithms such as video decoding and state-of-the-art picture quality enhancement algorithms. It is also a challenge to design an arbitration algorithm that takes all requirements of all processing cores into account. A complicating fact is that these requirements change over time, for example when the user changes TV channel or chooses to watch a DVD.

In NXP SoCs, arbitration algorithms are applied that make full use of the latency tolerance of the traffic from the function-specific HW cores. Thereby the memory subsystem manages to deliver low latency service to the processor cores while maintaining high utilization of the MMI.

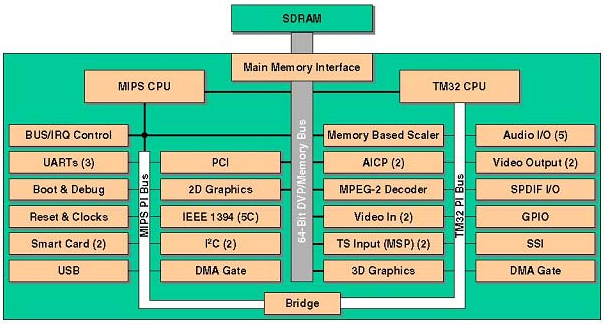

Figure 4. Block diagram of one NXP DTV SoC

VI. TRENDS AND DISCUSSION

The Nexperia heterogeneous multi-core platform described in this paper has successfully been used for DTV SoCs for more than 10 years [7] [13] [15] [16] [17] [18] and recently been implemented as part of NXP's global DTV platform TV550 [14]. TV functionality that was once implemented on several separate SoCs has now been almost fully integrated on one SoC. The underlying SoC architecture utilises standardised architecture elements such as a physically centralized memory architecture, split control buses and efficient inter-processor communication. The platform is likely to remain unchanged for several years to come as long as the physically centralized memory architecture remains the cheapest option to realize multi-media functionality.

Some key characteristics of selected SoCs from the Nexperia family are shown in TABLE II. The two right-most columns show the number of latency critical (LC) processor cores and the number of latency tolerant (LT) function-specific HW cores respectively. Some key trends can be extracted easily. The number of latency critical processor cores stays low. The number of latency tolerant function-specific HW cores on the other hand grows over time. The reason is that stable functional subsystems are moved to optimized latency tolerant implementations over the years. This is a necessary trend to fit ever more functionality on the SoC and to be able to design a memory subsystem that serves the latency critical processor cores with sufficiently low latency. The SDRAM bandwidth has increased over the years, but not faster than the core clock frequencies and thereby the bandwidth need per core. During the generations of TV SoCs built on the Nexperia platform there has thus been only a slight growth in processor cores on the SoCs.

TABLE II. KEY CHARACTERISTICS OF SOME SELECTED SOCS BUILT ON THE NEXPERIA PLATFORM

Name	Year	Tech. [nm]	core freq. [MHz]	SDRAM bandwidth	#LC	#LT
8525	2001	180	100	1.1 GB/s	2	36
8550	2003	120	240	1.6 GB/s	3	68
8535	2004	90	290	2.1 GB/s	2	56
8543	2007	90	333	2.7 GB/s	3	74
85500	2009	45	500	4.3 GB/s	4	95

The latest Nexperia SoC included in TV550 is manufactured in leading edge 45nm process technology with extremely high levels of integration. Two RISC, two VLIW and three embedded DSP cores power the various functions in the system. A large number of function-specific HW cores, high speed communication interfaces and application specific interfaces such as analog video and display panel interfaces are present. The driver platform API uses Linux interfaces for simple addition of middleware and application SW.

VII. Conclusions and Future Directions

A modern hybrid TV SoC features several processor cores, typically of different types and each selected because it implements some functionality efficiently. For example, the SoC used in the NXP TV550 platform has seven processor cores, all of which are of a different type. In SoCs for consumer multimedia applications, it is thus possible to choose adequate processor cores for each function and thereby finding optimum implementations in terms of die area and power consumption. The chosen heterogeneous processor architecture enables also a SW partitioning where the various functional subsystems implemented on a specific processor have clearly defined SW APIs with low real time constraints. This eases SW debug and subsequent system integration.

The Nexperia platform has been used for several SoC generations and is scalable for the future. New SoCs can be derived either by selectively optimizing implementation of stable functionality in function-specific HW cores for lower cost SoCs, or by adding new or improved algorithms on processor cores to make high-end SoCs that enhance the TV viewing experience.

References

[1] D. E. Culler et al. Parallel Computer Architecture. ISBN 1-55860-343-3. Morgan Kaufmann Publishers, Inc. 1999.

[2] R.B. Lee et al. Refining Instruction Set Architecture for High-Performance Multimedia Processing in Constrained Environments. Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors. 2002.

[3] C.-H. Lin et al. LOW POWER DESIGN FOR MPEG-2 VIDEO DECODER. IEEE Transactions on Consumer Electronics, Vol. 42, No. 3, August 1996

[4] I. Kuroda and T. Nishitani. Multimedia Processors. Proceedings of the IEEE, Vol. 86, No. 6, June 1998, pages 1203-1221.

[5] P. G. Paulin et al. Distributed Object Models for Multi-Processor SoC's, with Application to Low-power Multimedia Wireless Systems. DATE proceedings 2006.

[6] A. Tumeo et al. Ant Colony Optimization for Mapping and Scheduling in Heterogeneous Multiprocessor Systems. SAMOS proceedings 2008, pages 142-149.

[7] S. Dutta et al. Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems. IEEE Design & Test of Computers September–October 2001, pages 21-31.

[8] Watkinson, John, 1994: The Art of Digital Audio (Oxford: Focal Press)

[9] ISO publication page: ISO/IEC 14496-10:2005 – Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding

[10] H. van der Heijden and J. van Gurp: NXP's Motion Accurate Picture Processing (MAPP) gets the edge in fast-action HDTV, NXP Semiconductors, http://www.nxp.com/acrobat_download/other/news/publications/picture_processing_mapp.pdf

[11] P. van der Wolf and T. Henriksson. Video Processing Requirements on SoC Infrastructures. DATE proceedings 2008.

[12] T. Henriksson et al. Network Calculus Applied to Verification of Memory Access Performance in SoCs. Estimedia 2007, pages 21-26.

[13] NXP global DTV platform TV550, http://www.nxp.com/applications/tv/digital_tv/tv550/

[14] Claasen, T.A.C.M.: System on a chip: changing IC design today and in the future, Micro, IEEE, Volume 23, Issue 3, May-June 2003 Page(s): 20 – 26.

[15] S. Mutz and P. Durieux. Heterogeneous Multiprocessing for Efficient Multi-Standard High Definition Video Decoding. Hotchips 2006.

[16] J. Geerlings et al. A Single-Chip MPEG-2 CODEC for DVD+RW. ISSCC 2003, pages 40-41.

[17] S. Dutta. Architecture, design, verification, and validation of multi-processor SoCs for DTV, ASTB, and media processing applications. ISIE 2002, pages 17-21.

[18] S. Dutta. Architecture and implementation of multi-processor SoCs for advanced set-top box and digital TV systems. SBCCI 2003, pages 145-146.