

Chapter 15

INTERCONNECT AND MEMORY ORGANIZATION IN SOCS FOR ADVANCED SET-TOP BOXES AND TV

Evolution, Analysis, and Trends

Kees Goossens[†], Om Prakash Gangwal[†], Jens Röver[‡], and A.P. Niranjana[‡]

[†] *Philips Research Laboratories, Eindhoven, The Netherlands*

[‡] *Philips Semiconductors, Sunnyvale, USA*

{Kees.Goossens,O.P.Gangwal,Jens.Roever,Niranjana.AP}@Philips.com

1. Introduction

In this chapter we show that the organization of the communication and memory infrastructures is critical in today's complex systems-on-chip (SOCs). We further show that resource management in the form of scheduling or arbitration is common to them both. The increasing importance of these issues is illustrated by following the evolution of an advanced set-top box and high-definition digital TV application (ASTB) and its SOC implementations over time.

In Section 2, we introduce the application domain (embedded systems for high-volume consumer electronics), and the application (advanced set-top boxes for high-definition digital and analog TV). The computation kernels and the communication (data rates, latencies) needed for real-time audio and video are demanding. Meeting real-time requirements, while minimizing resources for cost-effectiveness is challenging for such large heterogeneous SOCs.

In Sections 4 to 6, we review two existing and one possible future SOC implementation of the ASTB application, along the following axes, introduced in detail in Section 3. We commence with the application itself, including real-time requirements, the computation kernels, the kinds of traffic flowing between them, and the logical memories used by them. The following axes categorize its implementation; the mapping of computation (types and number of IP blocks) and communication. Then we consider the interconnect organization, communication abstraction (how IP blocks interact with the interconnect),

and the memory organization (how the logical memories are implemented). Finally, we review arbitration: how traffic types are supported by the interconnect, and how the system as a whole meets its real-time requirements.

The Viper SOC (Section 4) and its successor Viper2 (Section 5) share some important characteristics (such as a dependence on a single external memory) but have a different philosophy underlying their architecture. The former implements the interconnect grouped by data-rate requirements (leading to low and high-bandwidth interconnects), while the latter groups IP blocks by traffic kind (control traffic, low-latency data traffic, and latency-tolerant data traffic). A possible future implementation (Section 6) builds on Viper2's traffic separation, and additionally integrates multiple on-chip memories and external memories.

In Section 7, we review the evolution of the ASTB application and implementations, and observe some trends.

2. The ASTB Application

We focus on embedded systems for high-volume consumer electronics, in particular, advanced high-quality set-top box and TV systems. The ASTB application comprises audio decoding (e.g. AC3, MP3), video decoding (e.g. MPEG2), video pixel processing (e.g. de-interlacing, noise reduction), and graphics [1, 2]. These functions are combined to provide different products (such as analog, digital, and hybrid TV), as well as different modes of operation within a product. For example, digital TV decodes audio and video before performing further video pixel processing, while analog TV uses noise reduction instead of video decoding, and in hybrid TV both are present.

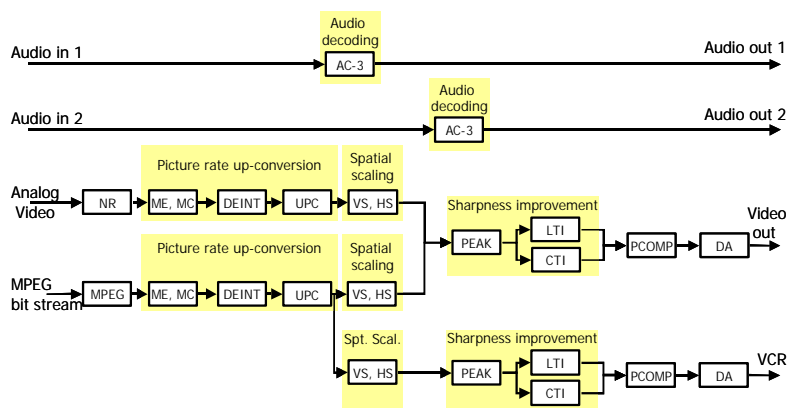


Figure 15.1. Example hybrid (analog and digital) ASTB processing chain.

Figure 15.1 depicts an example hybrid TV application. The audio processing chains for analog and digital remain independent, while the corresponding video-processing chains interact in a convergence to the screen, and a VCR output. We refer to [3, 4] for more information about the particular video functions, such as noise reduction (NR), picture rate up-conversion (motion estimation ME and motion compensation MC, de-interlacing DEINT, up-conversion UPC, etc.), spatial scaling (horizontal scaling HS and vertical scaling VS), sharpness improvement (e.g. peaking, luminance transition improvement LTI), picture composition PCOMP (e.g. mixing and blending of multiple pictures and graphics), and display adaptation DA (color conversion, skin tone correction, blue stretch, green enhancement, etc.). An important characteristic of many of these functions (such as NR, ME, MC) is their *temporal processing* nature, i.e. they use previous, current, and (sometimes) next pictures to generate their output picture. These pictures require significant storage and subsequent retrieval. Table 15.1 lists some typical computation, communication, and memory requirements to implement these functions [5, 6].

Table 15.1. Characterization of ASTB applications. Video pixel processing combines many video-improvement algorithms in a chain. We show typical numbers; however, they depend on the number of (temporal) processing stages.

	computation	in	out	local	memory
audio decoding	100 MOPS	32-640 kbps	5 Mbps	5 Mbps	50 kb
MPEG2 video decoding	4 GOPS	10 Mbps	120 MBps	240 MBps	8 MB
video pixel processing	100 GOPS	360 MBps	360 MBps	360 MBps	4 MB

The audio and video decoding functions conform to standards, such as AC3 and MPEG2. Many implementations complying with a standard are therefore available for these functions [7, 8]. In contrast, video pixel processing functions are often proprietary, and are the differentiating factor for products because they visibly improve the video picture quality [9, 10, 11].

ASTB SOCs have to support various input/output picture sizes, in particular standard (SD) and high definition (HD). Some processing is specific to analog or digital input (e.g. noise reduction is not used for digital input), specific to the display type (e.g. CRT or matrix), and specific to the number of streams processed simultaneously for one screen (e.g. with or without picture-in-picture PIP). Some modes are static, i.e. fixed for a particular product and do not change during its lifetime (e.g. CRT versus matrix). Other modes are dynamic and may change frequently, either triggered by the user (e.g. PIP on/off) or by the environment (e.g. broadcast change from video to film type, or SD to HD for main window).

Audio and video processing have intrinsic real-time requirements: for example, a video field must be displayed at regular intervals (e.g. 1/50th, 1/60th,

or 1/100th of a second). Furthermore, audio and video must be synchronized within strict limits (known as "lip synchronization"). Only a limited amount of processing, such as graphics, has less strict timing restrictions.

Application characteristics From the brief overview of the application domain we extract the following characteristics.

First, computational requirements vary substantially (three orders of magnitude, cf. the second column of Table 15.1), from low for audio processing to very high for video pixel processing functions. This leads to a large number of heterogeneous computation elements. For example, Viper2 (Section 5) contains 60 dedicated and weakly-programmable IP blocks, two application-domain-specific VLIW media processors, and one general-purpose RISC processor. (A note on terminology: we divide IP *blocks* in two categories: function-specific *cores*, and programmable *processors*.)

Programmable processors offer the flexibility to implement emerging standards for audio and video, and differentiation of products (when integrating SOCs in products). Low cost, power efficiency, and high computational performance are achieved through the use of function-specific cores.

Video decoding and video pixel processing operate on large amounts of data, necessitating large buffers for communication and temporal data. Multi-tasking processors, especially high-performance VLIW processors, have large amounts of program instructions (code), which also requires storage. For high memory requirements, large external (off-chip) memories are more cost effective than on-chip memories.

The interconnection infrastructure plays a central role in the SOC architecture for a number of reasons. The large amounts of (temporal) video data have to be transported between many IP blocks, via memory or directly. This requires a high-performance interconnect. To support the many static and dynamic modes of operation, data transportation mechanisms must be highly configurable. Moreover, to configure the SOC for a mode of operation, a flexible control interconnect must allow IP blocks to be accessed and programmed [12].

Finally, the combination of cores, processors, and interconnect must guarantee the hard-real time requirements of the application. Resource management (arbitration) of computation (e.g. real-time operating systems), communication (e.g. traffic classes with different performance), storage (by the external-memory controller), and their combinations are essential.

3. System Analysis Overview

Before we discuss individual designs, we define the axes along which the designs are presented. Although the axes are not independent (interconnect and memory organizations are strongly related, for example), for every axis an evolution can be identified for successive designs. For each design we discuss

the application, its computational complexity, and the resulting traffic types. These are reflected in the interconnect organization, the communication abstraction (i.e. how IP blocks interact with the interconnect), the memory organization, and finally how all these components are managed or arbitrated. We have some general remarks on a few of these points in this section.

3.1 Traffic Types

Traffic types help us to discuss how communication is mapped to interconnect implementations. The heterogeneity of the processing elements (cf. Section 2), results in a variety of traffic types, based on data rate, latency, and jitter characteristics, see Table 15.2. Based on these traffic types, the three designs have different interconnect partitionings, as we shall see in Sections 4 to 6, and summarized in Table 15.4.

Table 15.2. A classification of traffic types.

label	data rate	latency	jitter	example
LRLL	low	low	low	control traffic
HRLL	high	low	low	cache misses
low-jitter HRLT	high	tolerant	low	“hard-real-time” video
jitter-tolerant HRLT	high	tolerant	tolerant	“soft-real-time” video
jitter-tolerant MRLT	medium	tolerant	tolerant	audio & MPEG2 bitstreams
best effort	tolerant	tolerant	tolerant	graphics

Control traffic originates from control tasks that are usually mapped on one or more processors, which must obtain status information from cores and program them. It has a low data rate, but requires low latency (LRLL) to minimize the system response time, e.g. when the application mode changes.

Multi-tasking processors, such as MIPS and especially high-performance VLIW TriMedia processors, do not have sufficient local memory to contain all instructions (code) and data of the multiple tasks. Instruction and data caches are therefore used to automatically swap in and swap out the appropriate instructions and data. This leads to high (instantaneous) data rates, and requires low latency (HRLL).

Dedicated video-processing cores usually operate on and generate streaming (sequential) traffic with high data rates. They are composed in deep chains without critical feedback loops, and their low-latency requirement can therefore be made less critical by using buffers to avoid underflow. The resulting traffic has a high data rate but is latency tolerant (HRLT). Medium-data-rate latency-tolerant traffic (MRLT) is generated, for example, by audio and MPEG2 processing cores.

Jitter (latency variation) can be handled similarly, and we use the distinction between low-jitter and jitter-tolerant HRLT traffic. IP blocks with the latter

traffic, such as the memory-based video scaler, have an average data-rate requirement but can be stopped when there is no data, and make up by processing at a higher rate later, or by averaging out data bursts. By contrast, low-jitter HRLT IP blocks do not tolerate variations in data rates, because they cannot make up for any lost processing cycles. Examples are video-processing blocks operating at actual video frequencies, where line and field blanking can not be used as slack.

Some processing, like graphics, operate on best effort traffic, in the sense that it gets by on whatever bandwidth and latency it is given.

3.2 Communication Abstraction

As SOCs increase in complexity, the need for IP re-use and associated standardized SOC design methods has resulted in the notion of a platform [13]. A platform structures and standardizes SOC architectures, by regulating the kind of IP blocks that can be used, how they are combined, and how the system is programmed. Separating the computation (processors and cores) from communication (the interconnect core) has many advantages [14, 15]. In particular, we show how the use of the device-transaction-level communication standard (DTL) [16], part of Philips's Nexperia platform, has been key in allowing the interconnect to evolve over time, while the IP blocks remained unchanged. Communication abstraction has been promoted by the VSI alliance, the OCPIP consortium, and ARM, whose respective VCI [17], OCP [18], AXI [19] protocols are similar to DTL.

The use of communication abstractions, such as DTL, has several advantages. The development of cores is simplified because DTL is tailored to the requirements of IP blocks, rather than the interconnect. Moreover, IP blocks become independent from the interconnect, and hence re-usable. The interconnect and IP blocks are glued together by means of (re-usable) adapters. System-dependent customization is restricted to the adapters, instead being implemented by the IP blocks or the interconnect. Examples include little/big endianness conversions, interconnect-dependent sizing of latency-hiding communication buffers. This enables re-use of both the IP blocks and the interconnect.

3.3 Memory Organization

We distinguish several logical memories for *data use*. These are: *algorithmic* memories (e.g. temporal field memories, field to frame conversion memories, line memories of a scaler), *state* memories (e.g. local variables of a hardware or software task), and *decoupling* memories. Decoupling memories even out differences in data production and consumption rates of IP blocks, e.g. due to field and line blanking, horizontal and vertical data access patterns of

video scalers. In the following discussions we omit state memories because, in the designs we discuss, they are always part of the IP block. A second type of logical memories are used to store *instructions* of programmable processors.

Memories that pertain to the communication architecture include pipelining, *packetization*, *latency-hiding*, and *clock-domain-crossing* memories. Pipelining memories are used to increase the operating frequency of interconnects. Packetization memories are required to convert the data of IP blocks to the format used by the interconnect (e.g. 64-bit words). Latency-hiding memories remove or hide latency and jitter introduced by communication networks, memory controllers, RTOS task scheduling, and so on. For example, a 128-byte buffer is used in Viper. A small amount of memory is used to safely cross different clock domains.

All the kinds of logical memories are mapped to (implemented by) physical memories, basically on-chip or off-chip memories. For each of the designs, we show how the different logical memories are mapped to on- or off-chip memories (but we do not further sub-divide in RAM, flash, register files, etc.).

3.4 Arbitration

The ASTB application can be analyzed in terms of its computation and communication, but, additionally, it has real-time constraints for audio and video. To meet real-time requirements computation, communication, and memory resources must be arbitrated, or managed. For example, a simple first-come first-serve bus arbiter cannot distinguish low-latency from latency-tolerant traffic, and cannot offer differential data-rate services. Both are important in meeting real-time requirements. Two key issues in each of the designs are the arbitration of critical resources (such as external-memory bandwidth), and managing the interaction of various arbiters (e.g. those of external memory and on-chip interconnect).

4. Viper

Viper [1] is a highly integrated multimedia SOC targeted at advanced set-top box and digital TV (ASTB) applications. It provides progressive SD, and interlaced SD and HD outputs. Viper's main functions are video decoding (e.g. MPEG2), audio decoding (e.g. AC3), and video pixel processing to improve picture quality and to convert between image formats (e.g. de-interlacing). It simultaneously supports two video streams (e.g. one high definition of 1920x1080 pixels interlaced at 60Hz, and one standard definition of 720x480 pixels interlaced at 60Hz) and three audio streams.

4.1 Computation Mapping

Viper contains two processors (a MIPS-PR3940 and a TriMedia TM3218) and 50 function-specific cores. The cores include video-processing modules such as MPEG2 decoders (with high computation and communication requirements), audio/video input/output modules (e.g. MPEG2 transport-stream parser) and general-purpose peripherals (e.g. UART interface, USB controller). As noted in Section 2 their communication requirements and memory usage vary significantly.

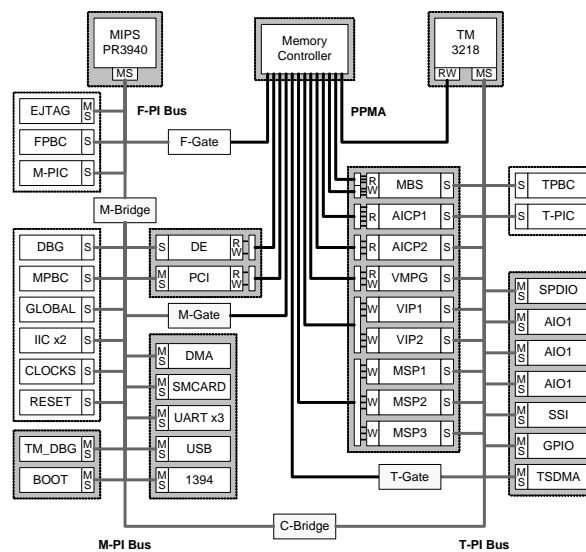


Figure 15.2. Simplified block diagram of Viper.

4.2 Communication Mapping

Memories are required to a) store instructions and data for both processors, b) function as an algorithmic memory (in particular temporal data, and for vertical scaling), and c) decouple IP blocks if their processing rates do not match. On-chip memories cannot be used in many of these cases because they would be too large. Hence it is advantageous to merge all data and instruction memories in one external memory. This results in high data rates for the off-chip memory from streaming cores (HRLT), and from data and instruction accesses of the programmable processors (HRLI). To hide the latency to the background memory the MIPS and TriMedia processors all have data and instruction caches. In Viper, all IP blocks are masters, i.e. they autonomously write to and read from the memory. In Figure 15.2 master and slave ports are

labeled M and S, and all read (R) and write (W) ports are masters. This avoids the need for a central DMA agent that has to be programmed by a processor, which could be a bottleneck. Given the particular combination of many masters and one slave, it makes sense to optimize the combination of external-memory controller and interconnect, as we shall see in the next section.

The status and program registers of all cores can be accessed at locations in a global memory map. This control traffic is generated by a few masters (the programmable processors) requiring access to many slaves (the cores), in contrast to the data traffic. Control traffic has low data rates, but requires low latency, to reduce the system response time (e.g. when the application mode changes).

4.3 Interconnect Organization

A single-hop broadcast interconnect, such as a shared bus (multi-master and multi-slave) that connects all IP blocks (including external memory), is not a feasible solution for a SOC like Viper. It cannot fulfill the bandwidth requirements of high-data-rate traffic, and certainly cannot offer the required low latency for control (LRLL) and cache traffic (HRLL). In Viper the traffic is therefore partitioned over several different interconnects, with *traffic separated on the basis of data-rate requirements*. There are two interconnects: one for low-data-rate (LRLL, control traffic) to medium-data-rate traffic (MRLT, audio and encoded MPEG2), and one for high-data-rate traffic, both low latency (HRLL, cache misses) and latency tolerant (HRLT, video). Below, we first discuss each interconnect in turn, and then how they interact.

The low to medium-bandwidth interconnect The idea underpinning this interconnect is that low latency can be ensured for control traffic by isolating it from high-data-rate traffic. Two 32-bit PI busses [20] are used: M-PI (see Figure 15.2) for control traffic from the MIPS and DMA traffic from streaming cores, and T-PI for control traffic of the TriMedia. MRLT DMA traffic is added to the M-PI bus to increase its utilization. The two busses are connected by a bridge (C-BRIDGE, with a master and slave port on both sides), to allow both processors access to all cores for programming.

The high-bandwidth interconnect The high data-rate requirements to access external memory cannot be fulfilled with a tri-state bus, such as the PI bus. Even with large tri-state drivers, it would be too slow because of a high capacitive load due to the large number of IP blocks. Observe, however, that all high-data-rate traffic is destined to the off-chip memory, that is, many masters communicate with a single slave. This leads to the design of a specialized 64-bit point-to-point memory-access interconnect (PPMA) that connects the IP blocks directly to the memory controller, see Figure 15.2. The PPMA allows

multiple active high-speed transactions because, essentially, it consists of a set of independent direct (non-pipelined) wires from the IP blocks to the memory controller. The MIPS and memory controller are re-used from earlier designs, and have different interfaces. As a result, the MIPS is connected to the memory controller using a PI bus (F-PI) running at processor speed (the M-PI bus runs more slowly), via a gate (F-GATE, which has only a slave port at the MIPS side). As discussed next section, this arrangement is not ideal because it increases the MIPS's latency when accessing the external memory.

The memory controller can efficiently schedule multiple outstanding transactions, as all client requests are visible. However, in this interconnect design, the very large number of long global (top-level) wires running from the many IP blocks to the single memory controller significantly complicated clock-tree balancing and global timing closure.

IP blocks are connected to the PPMA by means of adapters that packetize (format) the data of the IP blocks to 128-byte data bursts. The packetization memory that is required for this is merged with the latency-hiding memory. The latter hides the access latency an IP block would see before it is served by the memory controller. The use of adapters is essential to implement the communication abstraction, further discussed in Section 4.4.

Combining the two interconnects To allow cores on the PI busses access to the external memory, two additional gates (M-GATE and T-GATE) are used. The F-PI and M-PI MIPS PI busses are connected by the MIPS bridge (M-BRIDGE). The interconnects are summarized in Table 15.4.

In Section 4.6 we discuss the performance and arbitration issues of the chosen interconnect scheme, in particular the role of the gates.

4.4 Communication Abstraction

In this design, there are two different interconnects: a point-to-point PPMA of 64 bits wide, and tri-state PI busses of 32 bits wide. PI busses are used for the control traffic and the MIPS PPMA interface because IP blocks were only available with tri-state PI-bus interfaces. Alternative bus implementations (multiplexed or wired-OR) were therefore not considered although tri-state busses can cause testability and lay-out problems. The F-GATE is another penalty (in terms of additional latency for the MIPS cache misses) because the memory controller does not use the PI bus protocol. These observations motivate the communication-abstraction concept introduced in Section 3.2, i.e. IP blocks use an abstract point-to-point interface and protocol suitable for them, which is converted to different interconnect protocols by means of adapters. In fact, communication abstraction was already used to connect IP blocks to the PPMA (except for those on the F-PI bus), and proved to be very successful in re-using these IP blocks in Viper's successors, even as interconnects evolved.

4.5 Memory Organization

Most logical memories are mapped on the external memory, because they are too large to be kept on chip. This includes algorithmic memories (such as SD/HD fields for temporal processing), decoupling memories (e.g. SD/HD fields), and the instruction memories for the processors. The only exception is the line memories of the memory-based scaler, which are mapped to on-chip local memory. Both processors use instruction and data caches to hide the latency of accessing data and instructions in the external memory.

Latency-hiding memories, which hide the variations in data-access latency (jitter) to the external memory, are kept on chip. This jitter is introduced by the memory controller when it arbitrates between the urgent cache misses of the processors (HRLI), the heavy data rates of streaming traffic (HRLT), and the remaining traffic (e.g. best-effort graphics). These latency-hiding memories are implemented in the adapters (the boxes marked R and/or W in Figure 15.2). They are merged with the packetization memories that are required to convert the IP block's data to the format used. The interconnect, memory controller, and overall system requirements determine this format. None of the interconnects (PI busses, PPMA) are pipelined. The bridges and gates contain only memory to synchronize clock domains. They are therefore *circuit-switched*, that is, they make an end-to-end connection between the master and the slave, occupying all intervening interconnects. While this simplifies transaction handling, it also causes a performance bottleneck, as we shall see below.

4.6 Arbitration

No arbitration takes place in the PPMA. Instead the memory controller considers all outstanding requests from all IP blocks connected to it. The memory controller optimizes the bandwidth to the external memory. The run-time-programmable arbitration scheme uses time-division multiplexing, with two priorities per slot. The higher priority guarantees a maximum latency to low-jitter clients, and the lower priority allows other clients to use the bandwidth left over.

Traffic with similar characteristics is coalesced before entering the PPMA, conceptually adding a first level of round-robin arbitration. This includes both the (multiple) read and write ports of a single core (e.g. VMPI in Figure 15.2), and multiple cores (e.g. VIP1 and VIP2). This reduces the number of top-level wires to the memory controller.

The arbitration of the PI busses proceeds independently, except when addressing a slave behind a bridge or gate. In this case, both busses are locked until the slave has responded (non-pipelined circuit switching). This works well for the inter-PI bus bridges (M- and C-BRIDGE) and F-GATE. However, when accessing the PPMA through the M-GATE and T-GATE a transaction can

be stalled for some time, depending on the traffic class of the master (e.g. best effort). During this time the F-PI bus is locked. For example, a DMA access of the USB module on the M-PI bus can in this way stall the MIPS for some duration, if the MIPS tries to program a core at the same time. For this reason, the latency of control traffic can vary considerably: from 10-20 cycles without DMA interference, to 100+ cycles with interference.

Hence, the way in which traffic types are separated (by data rates rather than latency) and mapped (on separate, yet interacting interconnects) causes interacting arbitration schemes (of the PI busses and memory controller). This complicates guaranteeing the real-time behavior required by the application, and forces overdimensioning of parts of the system (in particular the PI bus frequency). These observations suggest improvements for Viper's successors.

5. Viper2

Viper2 is a successor of Viper, and targets mid to high-end analog, digital, and hybrid (both analog and digital) TV, including wide-XGA plasma and LCD displays. Viper2 extends Viper by handling 100Hz interlaced and 60Hz progressive displays. It upgrades the conversion of interlaced to progressive output video from SD to HD. It also includes advanced video improvement algorithms, such as motion-compensated Digital Natural Motion for SD pictures.

5.1 Computation Mapping

Viper2 contains three processors (one MIPS PR4450 and two TriMedia TM3260) and 60 function-specific cores. The cores are similar to those of Viper, but have higher computational and communication requirements due to HD output.

5.2 Communication Mapping

Viper2 has one external memory, like Viper. The second TriMedia adds more HRLC cache-miss traffic, and the larger number of cores, with higher data rates, load the external memory close to its maximum.

5.3 Interconnect Organization

Recall that in Viper, the interconnects are defined by data-rate requirements: PPMA for high-data-rate traffic, and the PI busses for low to medium-data-rate traffic. Their interaction via the gates leads to performance issues on the PI busses (cf. Section 4.6). For Viper2, this problem would have been more acute with the increased number of IP blocks. This is addressed by *partitioning the interconnect on the basis of traffic types* instead. Cache misses (HRLC), streaming data (MRLT and HRLT), and control traffic (LRLC) are separated in three interconnects, respectively. No bridges are required between these inter-

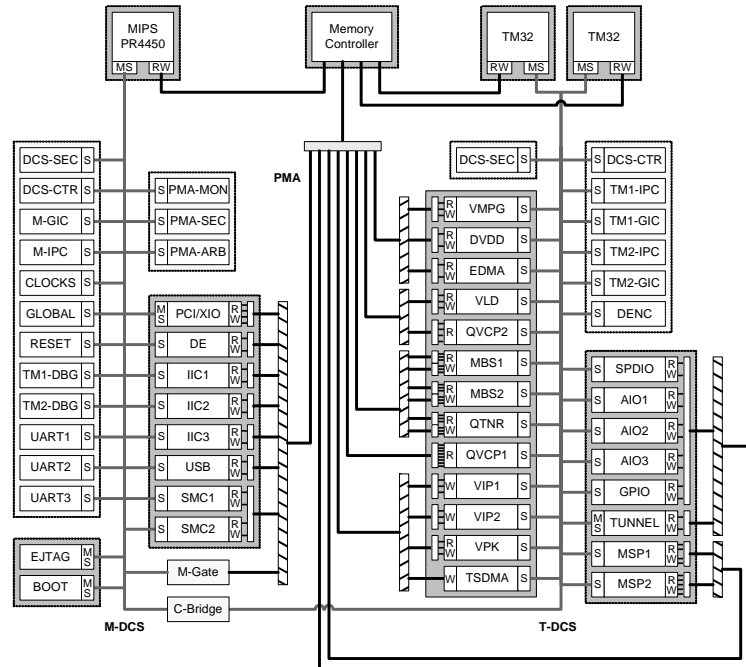


Figure 15.3. Simplified block diagram of Viper2.

connects, and hence there is no interaction between their arbiters. This avoids the traffic interactions of Viper. (Observant readers will notice the M-GATE in Figure 15.3, which, however, is only used during booting and debug.) Below, we describe each of the interconnects, and return to arbitration issues in Section 5.6. The interconnects are summarized in Table 15.4.

The device control and status interconnect (DCS) The DCS interconnect differs from Viper's PI busses in several ways. First, the traffic to be transported over the DCS interconnect is homogeneous (only low-data-rate low-latency traffic), in contrast to the PI busses in Viper. Only single-word transactions are allowed, to ensure low latency. Second, it is a synchronous or asynchronous wired-OR bus, not a tri-state bus, to lower the length and number of wires, and so increase its frequency and improve its testability. Timing closure is alleviated by structured post-lay-out netlist modification. Further, to ensure low latency, the load on the DCS busses is kept to less than 0.5% (one data word per cycle corresponds to 100%). Finally, to hide the significant difference in speed between the processors and the core being accessed, posted writes and reads can be used. A posted read accesses a (possibly out-of-date) copy of a core's registers in the adapter of the core, which is in the fast DCS

clock domain instead of the slower core's clock domain. By the combined effects of these improvements the latency of the control interconnect of Viper2 is the same as in Viper, despite the increase in the number of IP blocks.

The bridge (C-BRIDGE in Figure 15.3) between the MIPS (M-DCS) and Tri-Media (T-DCS) interconnects functions as before (i.e. both interconnects are locked when addressing a core through the C-BRIDGE).

The high-bandwidth low-latency interconnect The three processors all require low-latency access to the external memory for their cache misses. They are therefore connected directly to the memory controller, using non-pipelined wires. Viper's F-PI bus and F-GATE to connect the MIPS to the memory controller have been eliminated.

The pipelined memory-access interconnect (PMA) The pipelined memory-access interconnect (PMA) is Viper2's medium to high-bandwidth latency-tolerant interconnect. In Viper, the 64-bit point-to-point PPMA has direct wires between the memory controller and all IP blocks with high data rates (except for the MIPS). In Viper2, this is no longer feasible because the Viper2's increased chip area results in longer wires, and the medium-data-rate cores on the PI bus are moved to the PMA. Moreover, the number of medium to high-data-rate IP blocks has increased, as well as their data-rate requirements (to deal with HD instead of SD pictures).

The PMA therefore contains two innovations. Both hinge on the fact that many masters communicate with a single slave, like in Viper. First, the outstanding transactions from multiple cores are presented one at a time to the memory controller, to reduce its complexity, and to decouple the PMA communication arbitration from the memory arbitration (breaking global arbitration into local subarbitrations). The memory controller is now independent of the number of cores, making it more re-usable. We return to the PMA arbitration in Section 5.6.

Second, the point-to-point wires of PPMA of Viper have been replaced by a pipelined multiplexed interconnect to reduce the length and amount of wires, to ease lay-out and timing verification. A tree topology clearly fits well with exposing a single transaction to the memory controller, as transactions of different cores converge towards the top. In Figure 15.3 the cores on the dark shaded background connect to a PMA of 8 nodes (shown as hatched boxes), using 28 ports, but it is clearly scalable to a larger number of masters. Note that the tree topology is motivated by lay-out and timing closure, and that a node in the tree is not necessarily a point of arbitration (further discussed in Section 5.6).

The adapters glueing the cores and PMA together contain combined memories for clock-domain crossings, packetization (to convert data from the cores

to 128-byte PMA bursts), and pipelining and latency-hiding memory (to overcome the latency and jitter introduced by the memory controller and PMA).

5.4 Communication Abstraction

In Viper, re-use of legacy IP blocks required the F-PI bus with its associated side effects (cf. Section 4.4). This motivates the creation of IP blocks that are independent of the interconnect, which was already applied to Viper's PPMA interconnect. Many of Viper's IP blocks were re-used in Viper2 without change, proving the value of the methodology. Even the adapters that connected the IP blocks to the PPMA required only minor modifications for the PMA (e.g. resizing of the packetization and latency-hiding buffers).

The DTL (device-transaction-level) protocol [16] that is used has several profiles, which are related to the traffic types. The MMIO (memory-mapped IO) profile is chiefly used for control traffic (LRLI). The MMBD (memory-mapped block data) and MMSD (memory-mapped streaming data) profiles are used by IP blocks to communicate via shared on- or off-chip memory. MMBD and MMSD are used predominantly for reading and writing, respectively. Cache misses (HRLI) use MMBD, while streaming (audio MRLT and video HRLT) cores use MMSD or MMBD.

The use of DTL ensures that IP blocks can transparently connect to any of the interconnects (direct IP to IP communication, PMA, DCS interconnect), making it easier to move IP blocks within a design, and to re-use them across designs, with possibly different interconnects.

5.5 Memory Organization

All algorithmic memories (such as SD/HD fields for temporal processing), the decoupling memories, and the instruction code for the processors are mapped to external memory. There are two exceptions. The first, like in Viper, is the line memories of the memory-based video scaler (MBS), which are mapped to on-chip local memory of the MBS. Second, a small local memory is introduced in one place to (significantly) reduce the bandwidth pressure on external memory. Although this introduces another slave, it statically connects only two IP blocks, and hence is not part of the PMA.

All processors use instruction and data caches to hide latency.

The adapters contain latency-hiding memories for cores, to even out variations in data access latency to external memory. This variation is due to the combined effects of the memory controller (like in Viper), and the PMA arbitration (new in Viper2). The adapters contain packetization memories to convert IP-block data to a format suitable for efficient transport over the PMA. The packetization memories and latency-hiding memories are merged for area efficiency.

There are no pipeline memories in the DCS. The PMA is a pipelined circuit-switched interconnect. This means that every node in the tree contains some pipeline stages, to decrease wire lengths, allow higher operating frequencies, and ease timing closure. Circuit switching entails that a transaction that is accepted occupies all nodes on the path from the IP block to the memory controller. To eliminate run-in and run-out of the pipeline, and hence loss of bandwidth, transactions are prefetched as close as possible to the top of the tree, based on their scheduled order.

5.6 Arbitration

The two DCS interconnects are arbitrated independently, using a round-robin scheme (Table 15.3). M-DCS and T-DCS have 6 and 4 masters, and 32 and 38 slaves, respectively. The bridge is circuit switched; it locks both interconnects when addressing a slave at the other side of the bridge.

The memory controller has four inputs, one for each processor and one for PMA. The PMA offers a sequentialized view on the cores, i.e. the multiple active transactions of IP blocks are offered one at a time to the memory controller. The memory controller arbitrates its inputs using a round robin to aim for low latency. To guarantee a maximum latency, burst lengths are limited.

Table 15.3. Arbitration overview.

location	traffic	aim	method
DCS	LRL	low latency	round robin
memory controller	HRL	low latency	round robin with cut off
PMA top	low-jitter HRL	maximum latency	time-division multiplexing
PMA top	jitter-tolerant HRL	minimum bandwidth	priorities
PMA top	best effort	best effort	round robin
adapters	(all)	coalescing	round robin

Recall from Section 3.1 that jitter-tolerant HRL cores have an average data-rate requirement but can be delayed when there is no data, whereas low-jitter HRL cores cannot be delayed. Best-effort cores can operate on whatever bandwidth and jitter they are given.

The PMA uses the fact that many masters contend for a single slave, and arbitrates low-jitter cores using time-division multiplexing (to guarantee a maximum latency), jitter-tolerant cores with priorities (to guarantee a minimum bandwidth), and best-effort cores with a round robin. These arbitration mechanisms are applied (prioritized) in the order listed in Table 15.3. (The TDMA slots are skipped when unused. Note that priorities alone do not guarantee a minimum bandwidth, but their combination with system invariants does.) The arbitration scheme is fully programmable at run time.

The multi-stage arbitration scheme of PMA is motivated by performance requirements, and maps only partially to the physical tree of PMA nodes, which is driven by back-end issues. A node in the physical tree is not necessarily a point of arbitration. In fact, only at the top of the PMA does arbitration take place.

Like in Viper, traffic with similar characteristics is coalesced in the adapters before they enter the PMA, conceptually adding a first level of round-robin arbitration. This includes both the (multiple) read and write ports of a single core (e.g. QVCP1 in Figure 15.3), and multiple cores (e.g. AIO1 to AIO3, SPDIO, and GPIO).

6. An Example Future SOC

In this section we take a leap into the future, and describe a speculative SOC, based on an extrapolation of Viper and Viper2. We present a design that illustrates the trends that we foresee, but intermediate and hybrid solutions are very likely between Viper2 and the future SOC sketched here.

Future applications will contain more advanced video-processing functions, at higher picture resolutions. Examples are motion-compensated high-definition noise reduction and temporal up-conversion, as well as a move to MPEG4 and advanced graphics. We further expect higher and more dynamic data rates.

6.1 Computation Mapping

The number of processors increases to support emerging media-processing applications, and (system-integrator-defined) differentiation of products. The number of function-specific cores also increases, to efficiently implement standard or proprietary application kernels (e.g. PixelPlus and subpixel luminance-transient improvement).

6.2 Communication Mapping

As the number of processors increases, the amount of data and instruction cache misses (HRL) grows. This is an undesirable trend, because low latency is hard to guarantee for more than a few users of any shared resource, whether it is an external memory or an interconnect. There are several (partial) solutions. First, minimize the use of caches, e.g. reduce multi-tasking to decrease code memory size and increase locality, or improve memory management such as software prefetching in combination with scratch-pad memories. Second, lower the dependency on low latency, e.g. by using hardware multi-threading, or by increasing the emphasis on streaming instead of random-access traffic. Third, use fewer shared resources, e.g. use multiple memories, and interconnects that allow concurrent accesses such as switches and networks.

Control traffic (LRL) essentially suffers from the same low-latency problems. However, current solutions can be used in the near future because the available head room can accommodate the increase in (low-data-rate) control traffic.

MRLT and HRLT traffic increases because of the growth in the number of function-specific cores, which tend to implement streaming computation. Increasing picture dimensions from standard to high definition, also boosts HRLT traffic (e.g. sixfold for temporal up-conversion).

Thus, future interconnects must address the rise in communication needs, but architectural opportunities to limit the increase in traffic types that are hardest to implement, low latency in particular, should also be exploited.

6.3 Interconnect Organization

We see several trends that affect communication.

First, the number of processors grows, resulting in more masters and LRL traffic for the control interconnect. Busses like DCS are single-hop broadcast media, for which latency increases for two reasons. The arbiter frequency slows down because global arbitration must take more masters into account, and because the wires from IP blocks to the arbiter become longer. Moreover, without concurrent transactions and with limited operating frequency, only a subset of masters can have low latency.

Second, the increasing number of processors gives rise to more cache-miss (HRL) traffic, which cannot be supported by only one slave (the external memory). Again, with a single shared resource, not everyone can have low-latency access. Hence multiple memories (slaves) are required. They are probably external because processor instructions are too large to fit on chip. Some kind of switch must connect multiple masters (processors) to multiple slaves (memories).

Third, we have seen that the amount of HRLT traffic grows due to larger pictures. Communicating only via a single external memory is no longer feasible, for bandwidth reasons, and we foresee a combination of multiple off-chip and on-chip memories. The former, while not ideal for energy dissipation and pinning, is indispensable because HD (temporal) video data is too large to be kept on chip. The latter reduce the bandwidth pressure on external memories, and lower the latency and power to access data. In Viper2 a similar technique is used once (cf. Section 5.5), but shared on-chip memories will gain in number and importance. In both cases, we see a growing number of slaves, which the PMA interconnect alone cannot address.

Finally, both Viper and Viper2 contain IP blocks (tunnels) to communicate with off-chip components, either in a system-on-package or multi-chip setting.

Similar examples are USB, PCI, and PCI EXPRESS. A further rise in the use of these interfaces (and hence masters and slaves) is likely.

Networks on chip (NOC)

From the preceding discussion we conclude that any future interconnect must deal with many masters and many slaves, with high data rates. Busses cannot fulfill the bandwidth requirements. Switches [12] fare better because they offer concurrent master-slave communications, but are not scalable to the extent we require (50+ masters, and 50+ slaves). The PMA interconnect is optimized for a single slave, and multiple instantiations would be necessary. Multiple switches, or networks on chip (NOC) [21, 22, 23, 24] are scalable, and can solve many of the issues listed here. A NOC consists of a collection of routers (or switches) that transport data in packets. Adapters, now called network interfaces, connect routers to IP blocks and packetize the transactions of the IP blocks. The remainder of this section compares NOCs with the other interconnects.

First, we observe that wires connecting IP blocks are underutilized (as little as 10% [25]). Both PMA and NOCs reduce the number of wires that interconnect IP blocks by sharing them. However, instantiating PMA multiple times to address multiple slaves would increase the number of wires. Therefore, NOCs are better scalable in the number of attached slaves, as illustrated in Figure 15.4.

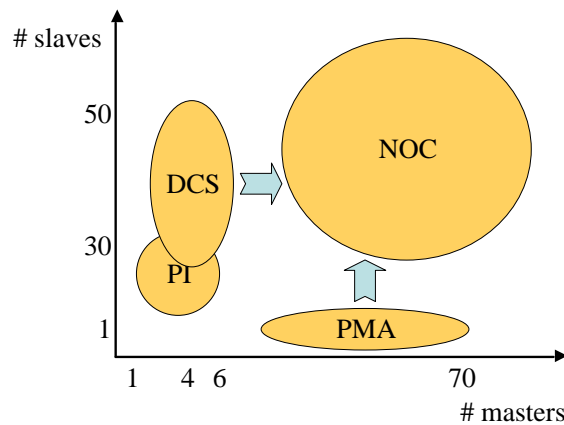


Figure 15.4. Interconnect evolution.

Second, a NOC is scalable in the sense that adding more routers results in more bandwidth. A NOC copes well with many masters and slaves because many transactions can take place concurrently. There are several reasons for

this. First, a NOC has distributed arbitration, unlike a bus or switch. This removes a major bottleneck, cf. Section 6.6. Second, in the appropriate topology (such as a mesh or (partial) fat tree, e.g. Figure 15.5) there are many independent paths that can be used simultaneously. Finally, packet switching is commonly used in NOCs, instead of circuit switching, employed in Viper2's PMA. As interconnects increase in size (number of routers), their diameter (distance between master and slave) grows, and reserving wires end to end (from master to slave) for the duration of the transaction becomes inefficient. The set-up and tear-down phases of the circuit take longer, causing congestion (blocking other transactions) [26]. (Fundamentally, multiple interconnects with circuit-switching bridges, such as those in Viper and Viper2, suffer from the same problem, discussed in Section 4.6.) Packet switching reduces these problems, by allowing pipelined transactions (on a single path), possibly at the cost of higher latency. NOCs can therefore offer tremendous bandwidth between many masters and slaves [27].

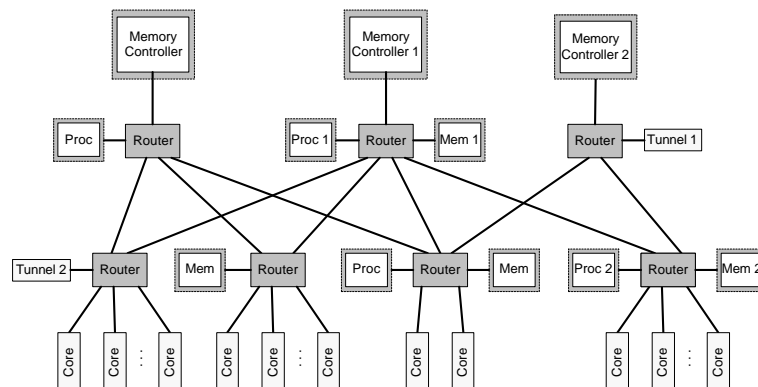


Figure 15.5. Simplified block diagram of future SOC.

Third, while NOCs scale in terms of bandwidth, this is not so clear cut for latency. Packet switching has both positive and negative effects on the latency. High operating frequency of (point-to-point) links and routers, and concurrent transactions reduce the latency, while arbitration per router, and possible congestion may increase it. In any case, placing IP blocks with latency-critical communication close to one another in the NOC topology reduces the number of router hops, and hence the latency (e.g. proc 1 and memory controller 1 in Figure 15.5).

Finally, a major advantage of NOCs is their ability to offer differentiated services [28]. This means that different traffic types can be implemented on a single NOC, and that different traffic types can be multiplexed on a single net-

work port. In Section 6.6 we discuss the strong relation between NOC services and NOC arbitration.

6.4 Communication Abstraction

The re-use of Viper2 IP blocks in a NOC is straightforward due to the use of DTL. The adapters from DTL required little modification to change from Viper's PPMA to Viper2's PMA internal communication protocol. However, moving from Viper2's circuit-switching PMA to a NOC's packet switching is more elaborate (e.g. end-to-end flow control, transaction reordering, distributed memory), but existing IP blocks are unaffected.

A NOC has the ability to offer different services to different connections, on a single network port. For example, a multi-tasking processor can request a connection to shared memory per task, with different properties per connection, such as bandwidth, latency, transaction ordering, and flow control. In fact, this is essential when multiplexing several logical communications over USB or PCI EXPRESS to off-chip components. It also eases the design of real-time multimedia applications [24], like those discussed here. If we want to take advantage of this capability, DTL must be extended to deal with connections (optional for backward compatibility). This tendency can already be observed in the appearance of thread and connection identifiers in OCP and AXI.

6.5 Memory Organization

In Section 6.3 we argued that multiple external memories will be likely in future SOCs to cope with increasing HRLC cache traffic and HRLT traffic (for algorithmic and decoupling memories).

We also foresee multiple on-chip memories for low power, to lower data access latency, and to relieve pin and bandwidth pressures. Viper2's processor and cache memory model can be extended to a memory hierarchy, as is illustrated in Figure 15.5. Proc 2 has its own cache (not shown), but can overflow to memory mem 2, which is relatively close (one hop), or memory mem 1, further away (two hops, but still on chip), or one of the external memories memory controller 1 or memory controller 2 (two hops, but passing through a memory controller). In fact, *all* memories are accessible to any of the IP blocks, but at non-uniform cost (although possibly within a uniform address space, i.e. NUMA). Similarly, all IP blocks can be programmed by any of the processors. A multi-master multi-slave NOC interconnect is therefore useful for both data and control traffic, as suggested by Figure 15.4.

On-chip memories can function as caches or scratch pads (for data and instructions). By keeping inter-IP-block communication on chip the latency and jitter introduced by memory controllers is eliminated, reducing the size of latency-hiding memories (cf. Section 5.5).

Like for Viper2, the network interfaces contain latency-hiding memories for streaming cores, to even out variations in data access latency. If the NOC offers low-latency and/or low-jitter communication, these and on-chip latency-hiding memories can be reduced. The network interfaces also use some memory to packetize the data to the format used by the NOC routers, and to cross clock domains. NOCs have pipelined routers (even in circuit-switched variants), and sometimes pipelined links too, to increase the operating frequency of the network. The *ÆTHEREAL* NOC from Philips, for example, provides a combined guaranteed-bandwidth-and-latency service with a router pipeline of three words deep [27].

6.6 Network Services and Arbitration

A major advantage of NOCs is their ability to offer differentiated services. This means that different traffic types can be implemented on a single NOC by means of a protocol stack [14], and different traffic types can be multiplexed on a single network port. Different DTL profiles and traffic coalescing, like that of Viper and Viper2, is then taken care of by the network (interface). In particular, the *ÆTHEREAL* NOC [28] offers guaranteed bandwidth, and best-effort connections, that are useful for the traffic types listed in Table 15.2.

However, sophisticated global arbitration such as PMA's, is more expensive when using the distributed arbitration of NOCs. Time-division multiplexing is relatively cheap, but distributed priority- or rate-based arbitration is not acceptable, in terms of buffering cost of routers [27]. The PMA interconnect can take advantage of its tree topology for its arbitration, but this is harder even in regular NOC topologies such as fat trees and meshes. NOC services are therefore less expressive and flexible than PMA arbitration. Moreover, it is harder for distributed arbitration to be of the same quality as global arbitration (cf. contention and congestion). However, given the abundance of bandwidth [27], this can be addressed by bandwidth overallocation, with best-effort traffic consuming unused capacity.

7. Conclusions

The advanced set-top box and hybrid TV (ASTB) application is demanding in terms of computation (high-definition video pixel processing), memory (temporal video data), and communication (high data rates) requirements. The first results in heterogeneous computation elements (function-specific cores, various processors). Instruction, algorithmic, and decoupling memories are all large and mapped in off-chip memory. The communication infrastructure is critical because it must connect many IP blocks with high data rates, in a flexible manner for product differentiation and run-time mode changes. Finally, the

application centers on real-time audio and video, which means that the system resources (memories, interconnect) must be carefully managed (arbitrated).

Viper's interconnect is separated by data-rate requirements, resulting in a low- to medium-bandwidth interconnect (for LRLI and MRLT) consisting of two bridged PI busses, and a high-bandwidth interconnect (PPMA for HRLI and HRLT) of dedicated wiring (Table 15.4). Utilization of both interconnects is high, but the circuit-switched M-GATE between the interconnects causes interference of arbiters for different traffic types, making real-time guarantees more intricate.

Table 15.4. Mapping traffic types to interconnect structures.

	LRLI	MRLT	HRLT	HRLI
Viper	PI	PI	PPMA	PPMA
Viper2	DCS	PMA	PMA	point to point
future	DCS / NOC	NOC	NOC	point to point / NOC

To avoid this, Viper2's interconnect is separated by traffic kind, resulting in three independent interconnects: two bridged DCS interconnects for LRLI control traffic, dedicated wiring for HRLI cache-misses, and the PMA interconnect for MRLT and HRLT audio and video. The PMA uses a sophisticated global arbitration scheme (Table 15.3) that distinguishes low-jitter HRLT, high-jitter HRLT traffic, and best-effort classes, for a high utilization.

Future systems will use multiple on- and off-chip memories, increasing the number of masters and slaves, see Figure 15.4. This motivates a move to multi-hop interconnects, such as networks on chip (NOC). NOCs are scalable in the number of masters and slaves, in bandwidth, and to a lesser extent in latency. NOCs can offer differentiated services and very high bandwidth, but their distributed arbitration favors scheduling simpler than that used in Viper2's PMA.

The challenge for future SOCs for real-time applications is to define advanced memory organizations (e.g. a hierarchy of on- and off-chip memories), and to offer different communication services (different traffic classes) with an interconnect structure that is both scalable and cost efficient, e.g. a NOC.

References

- [1] Santanu Dutta, Rune Jensen, and Alf Rieckmann. Viper: A multiprocessor SOC for advanced set-top box and digital TV systems. *IEEE Design and Test of Computers*, pages 21–31, Sept-Oct 2001.
- [2] M. Brett, B. Gerstenberg, C. Herberg, G. Shavit, and H. Liondas. Video processing for single chip DVB decode. In *IEEE Transactions on Consumer Electronics*, volume 47, pages 385–393, August 2001.

- [3] O. P. Gangwal, J. G. Janssen, S. Rathnam, E. B. Bellers, and M. Durranton. Understanding video pixel processing applications for flexible implementations. In *Proceedings of Euromicro, Digital System Design*, pages 392–401, September 2003.
- [4] G. de Haan. *Video Processing for Multimedia Systems*. ISBN: 90-9014015-8, September 2000. Eindhoven.
- [5] E. G. T. Jaspers, P. H. N. de With, and J.G.W.M. Janssen. A flexible heterogeneous video processor system for television applications. In *IEEE Transactions on Consumer Electronics*, volume 45, pages 1–12, February 1999.
- [6] R. Kramer. Consumer electronics as silicon engine. *International Electron Devices Meeting (IEDM)*, pages 3–7, 1999.
- [7] H. Yamauchi, S. Okada, K. Taketa, Y. Mihara, and Y. Harada. Single chip video processor for digital HDTV. *IEEE Communications Magazine*, pages 394–404, August 2001.
- [8] Markus Rudack, Michael Redeker, Jörg Hilgenstock, Sören Moch, and Jens Castagne. A large-area integrated multiprocessor system for video applications. In *IEEE Design and Test of Computers*, pages 6–17, January 2002.
- [9] G. de Haan. IC for motion-compensated de-interlacing, noise reduction, and picture-rate upconversion. In *IEEE Transactions on Consumer Electronics*, volume 45, pages 617–624, August 1999.
- [10] G. de Haan and J. Kettenis. System-on-silicon for high quality display format conversion and video enhancement. In *Proc of ISCE'02*, pages E1–E6, September 2002.
- [11] M. Schu, D. Wendel, C. Tuschen, M. Hahn, and U. Langenkamp. System-on-silicon solution for high quality consumer video processing—the next generation. In *IEEE Transactions on Consumer Electronics*, volume 47, pages 412–419, August 2001.
- [12] Jeroen A.J. Leijten, Jef L. van Meerbergen, Adwin H. Timmer, and Jochen A.G. Jess. Stream communication between real-time tasks in a high-performance multiprocessor. In *Proceedings of Design, Automation and Test in Europe Conference*, pages 125–131, 1998.
- [13] K. Keutzer, S. Malik, A. Richard Newton, Jan M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.
- [14] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip intercon-

- nect woes through communication-based design. In *Design Automation Conference*, pages 667–672, June 2001.
- [15] Drew Wingard. Socket-based design using decoupled interconnects. Chapter 15, this volume.
- [16] Peter Klapproth. Architectural concept for IP-re-use. In *VLSI ASP DAC*, December 2002.
- [17] VSI Alliance. Virtual component interface standard, 2000.
- [18] OCP International Partnership. Open core protocol specification, 2001.
- [19] ARM. *AMBA AXI Protocol Specification*, June 2003.
- [20] Open Microprocessor Initiative. *OMI/PI-Bus specification*, OMI 324: PI-Bus Rev. 0.3d edition, 1994.
- [21] Pierre Guerrier and Alain Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of Design, Automation and Test in Europe Conference*, pages 250–256, 2000.
- [22] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
- [23] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage. Networks on silicon: Combining best-effort and guaranteed services. In *Proceedings of Design, Automation and Test in Europe Conference*, pages 423–425, March 2002.
- [24] Kees Goossens, John Dielissen, Jef van Meerbergen, Peter Poplavko, Andrei Rădulescu, Edwin Rijpkema, Erwin Waterlander, and Paul Wielage. Guaranteeing the quality of services in networks on chip. In Axel Jantsch and Hannu Tenhunen, editors, *Networks on Chip*, chapter 4, pages 61–82. Kluwer, 2003.
- [25] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, June 2001.
- [26] André DeHon. Robust, high-speed network design for large-scale multiprocessing. A.I. Technical report 1445, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, September 1993.
- [27] E. Rijpkema, K. G. W. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of Design, Automation and Test in Europe Conference*, pages 350–355, March 2003.
- [28] Andrei Rădulescu and Kees Goossens. Communication services for networks on silicon. In Shuvra Bhattacharyya, Ed Deprettere, and Juer-gen Teich, editors, *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*, pages 275–299. Marcel Dekker, 2003.