Architecture and Circuit Techniques for a 2 GHz Advanced High-Speed Bus SoC Interconnect Infrastructure

Alexandre Landry

A thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Applied Science (Electrical Engineering) at
Concordia University
Montréal, Québec, Canada

April 2005

Your file Votre référence
ISBN: 0-494-04379-2
Our file Notre référence
ISBN: 0-494-04379-2

ABSTRACT

Architecture and Circuit Techniques for a 2 GHz

Advanced High-Speed Bus SoC Interconnect Infrastructure

By: Alexandre Landry

A key issue with high performance SoC platforms is how to interconnect their modules to effectively transfer large amounts of data in real-time. Today's most practical communication infrastructures are bus-based due to the small number of processing elements residing on a silicon die. Since the bandwidth of a shared bus goes down with the number of bus masters, hierarchical structures are used to parallelize transfers and to obtain a higher throughput. Hence, a novel shared memory SoC communication infrastructure based on the Advanced High-Speed Bus (AHB) is defined in this thesis.

The objective of this dissertation is to explore various avenues to design a bus operating with a clock in excess of 2 GHz when targeting a 0.18 μm CMOS process. As a first iteration, the fastest circuit techniques are reviewed so as to traverse the learning curve that a designer must experiment with very high-speed designs. To enhance the understanding of high-speed circuit styles, the main cores of an AHB are implemented from a novel, and aggressive, true-single-phase-clocking (TSPC) circuit style. The 2 GHz AHB arbiter has been laid out to prove the performance of the circuit techniques explored with the full-custom SoC infrastructure. In addition, an innovative 2 GHz pipelined memory has been created to respond to the hard IP requirements.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor Yvon Savaria for his excellent guidance during my graduate career. He is surely the giant upon whose shoulders I stand. In the same line of thought, I would like to acknowledge my sincere recognition to Professor Mohamed Nekili for his accurate supervision of this research and his precious help to overcome unusual difficulties. I would like to thank all the members of the Very Large Scale Integration (VLSI) group at Concordia University and give a special thanks to Tadeusz Obuchowicz who took care of difficult CAD tool issues. I gratefully acknowledge the support of the Microelectronics Research Group (GRM) from École Polytechnique of Montréal for their valuable collaboration, particularly Normand Bélanger who reviewed this thesis.

I also express my deepest gratitude to my beloved family and my only sister, Marie-Ève, who have always supported me in all decisions. Finally, I would like to thank my many friends for their understanding and endless support.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ADC:	Analog-to-digital converter.
AHB:	Advanced High-Speed Bus.
AMBA:	Advanced Microcontroller Bus Architecture.
AOI:	AND-OR-INVERTER standard gate.
APB:	Advanced Peripheral Bus.
ASB:	Advanced System Bus.
ASIC:	Application specific integrated circuit.
AXI:	Advanced eXtensible Interface.
C^2MOS:	Complementary CMOS.
CAD:	Computer aided design.
CIW:	Command Interpreter Window.
CMC:	Canadian Microelectronics Corporation.
CMOS:	Complementary metal-oxide semiconductor.
CTMC:	Custom-Touch-Memory-Compiler.
DAC:	Digital-to-analog converter.
DCR:	Device Control Register.
DCVSL:	Differential cascade voltage switch logic.
DMA:	Direct memory access.
DSM:	Deep submicron.
DSP:	Digital signal processing.

DUT:	Device under test.
FIFO:	First-in-first-out register.
FSM:	Finite state machine.
GALS:	Globally-asynchronous-locally-synchronous.
GRM:	Microelectronics Research Group.
HDL:	Hardware description language.
HVL:	Hardware verification language.
IC:	Integrated circuit.
IP:	Intellectual property.
LHS:	Left-hand side.
MCML:	MOS-current-mode-logic.
MESI:	Modified-Exclusive-Shared-Invalid invalidate protocol.
MFSM:	Master finite state machine.
NMOS:	n-type MOS field-effect transistor.
NoC:	Network-on-a-chip.
NORA:	No-race.
OAI:	OR-AND-INVERTER standard gate.
OCP:	Open Core Protocol.
OPB:	On-chip Peripheral Bus.
PCB:	Printed circuit board.
PDN:	Pull-down network.
PE:	Processing element.
PLB:	Processor Local Bus.

PLL:	Phase-lock-loop.
PMOS:	p-type MOS filed-effect transistor.
PUN:	Pull-up network.
QoS:	Quality-of-service.
RC:	Resistor-capacitor.
RHS:	Right-hand side.
RT:	Real-time.
RTL:	Register transfer level.
SFSM:	Slave finite state machine.
SoC:	System-on-a-chip.
SSRAM:	Synchronous static random access memory.
StS:	Standard-to-Specialized.
tr:	Rise time.
tf:	Fall time.
TSPC:	True-single-phase-clocking.
UDSM:	Ultra deep submicron.
VHDL:	Very high-speed integrated circuit hardware description language.
VIH:	Input high voltage
VIL:	Input low voltage.
VLIW:	Very long instruction word.
VLSI:	Very large scale integration.
VOH:	Output high voltage.
VOL:	Output low voltage.

1 INTRODUCTION

1.1 Background

Advances in deep submicron (DSM) technology allow integrating so many transistors on a single silicon die that complete systems can be merged into a single chip. Typically, systems-on-chip (SoC) are conceived from a mixture of custom circuits and preexisting intellectual property (IP) macrocells (sometimes called megacells). Intellectual property macrocells may be designed with two distinct approaches: (1) a soft IP follows a semi-custom flow in which hardware description files are synthesized to obtain the circuits, while (2) a full-custom flow is more appropriate to design a hard IP in which only the masks are available. However, for SoC integration, it is not critical whether we use hard or soft IP megacells. What is really important is to meet the system requirements and reduce time-to-market.

The momentum in the industry hints that SoCs are taking the path that printed circuit boards (PCB) systems took two decades ago. At first, microprocessors were implemented with discrete components over a PCB, and then VLSI improvements allowed packing enough transistors in a chip to merge complete microprocessors on a single chip substrate [1]. In a short time span, electronic systems grew more complex and interconnect standards became most needed to keep up with performance enhancements. Nowadays, PCB interconnect standards span over a large range of applications. SoCs do not escape this movement as their performance is partly set by the

efficiency of the on-chip communication infrastructure used to transfer large amounts of data. To this regard, interconnect standards, tuned for the SoC realm, help achieving high-throughput and reduce design efforts.

Key issues with SoC communication infrastructures are testability, flexibility, and their ability to exchange data at high rates on demand between system modules. For the time being, SoCs are characterized by a small number of processing elements, so a shared memory accessed from a shared bus is a practical SoC interconnect infrastructure to use. To this regard, the Microelectronics Research Group (GRM) of École Polytechnique of Montréal has performed significant work using AMBA standards [2] prior to this research. Hence, different bus standards have been studied in [3], and a novel architecture was invented in [4], [5]. All previous interconnect infrastructures at GRM were implemented with a semi-custom design flow (synthesis followed by placement and routing). This abides by the AMBA specifications since this norm describes synthesizable buses. Yet again, the performance obtained by this design flow is limited by the use of standard libraries, and sparse layouts created by automated placement and routing [6], [7], [8]. Hence, interconnecting high-performance cores with an AMBA bus is a problem that remains unsolved.

1.2 Overview of Thesis

As presented earlier, the design of communication infrastructure's architecture is a critical issue with systems-on-chip. The complete shared memory interconnect fabric proposed in this thesis consists of a hierarchical AMBA Advanced High-Speed Bus (AHB), a group of processing elements and a shared memory acting as a communication buffer. Since the heart of this SoC communication infrastructure operates at a frequency

in excess of 2 GHz, it is imperative to implement the main system cores with full-custom circuits to optimize the speed of the AHB shaping the hard IP.

The purpose of this research is to undergo the learning curve involved with the design and the optimization of high-speed circuit, and the many twists involved with their layout. To enhance the understanding in high-frequency circuit design, a thorough study of dynamic logic is completed so as to isolate the fastest avenues. The work reported in this document spans a wide range of abstraction levels. An implementation as ambitious as the one presented in this dissertation requires a sequence of actions carefully planned to overcome the many obstacles to be encountered in the design process.

To accomplish this research goal, three tasks have been identified:

1. ***Architectural definition***: A chain is never stronger than its weakest link. Defining a robust architecture forms the first link of the design chain. Though the architecture of an AHB is defined by its specifications, the AHB hierarchy, the pipeline structure, and the communication protocol of the entire high-speed hard IP must be described in a first step.
2. ***Circuit Decisions***: Designing sequential circuits operating above the GHz mark is no easy task. Conventional circuit techniques may not be applicable since their structure limits the maximum frequency. A careful study of high-speed methodologies is required to select appropriate sequential logic style.
3. ***Circuits Design***: Once a high-performance architecture is defined and a high-speed sequential logic style is selected, circuits design may start. Finite state machines (FSM), registers, combinational logic blocks, and datapaths need to be designed.

1.3 Summary of Thesis

The design of a very high-performance SoC communication infrastructure is a thorough commitment that forces engineers to work over several levels of abstraction. Based on the tasks identified in this chapter, the complete hard IP is described in the next chapters. State-of-the-art computer aided design tools (CAD) utilized in this project, along with the well-established on-chip communication industry standards are presented in Chapter 2. In addition, Chapter 2 surveys circuit styles known to be the fastest with CMOS IC design. The design flow and circuit techniques used with the novel interconnect infrastructure are presented in Chapter 3. Chapter 4 presents the global architecture of the proposed communication fabric. An ambitious arbitration procedure and the high-level description of a pipelined high-throughput shared memory are described in the same chapter. Circuit implementations with their results are described in Chapter 5, and an AHB arbiter layout is presented to prove the accuracy of the investigated methodology. Finally, Chapter 6 concludes this document and presents the many avenues available for improvements as future work.

2 STATE-OF-THE-ART

Systems-on-chip (SoC) design in the forthcoming ultra deep submicron (UDSM) technologies will be dominated by multi-core (multi-processor) platforms in a number of application areas [9]. A key issue to enable this type of platform is to develop a communication-centric interconnect infrastructure to effectively transfer large amounts of data within the SoC. Obviously, such communication-centric fabrics are characterized by a number of tradeoffs, e.g., with regards to latency, parallel programming capability, design effort and circuit complexity. The requirements of the target application set the boundaries for tradeoffs to be considered. For instance, the type and the structure of the communication fabric is a significant factor that directly influences the performance of a network-on-chip (NoC). Outstanding NoCs allow for record-breaking SoC platforms' performance, as far as the SoC cores can keep up. The computer-aided-design (CAD) tools at hand impact the design flow and the overall complexity envisioned for a design. Inadequate design tools often entail excessive time-to-market delays and possibly project failure.

This chapter lists the CAD tools used for the design of the high-performance communication infrastructure. The main tools and their capabilities are documented so as to shorten the learning curve that other designers could experiment when undertaking a project of similar target frequencies with significant design efforts. Then, various NoC standards are surveyed. Abiding to a standard facilitates design integration since they encourage uniform interfaces with specific protocols. Finally, high-performance circuit

styles are discussed since full-custom design requires selecting the best logic style manually.

2.1 Design Tools List

The Canadian Microelectronics Corporation (CMC) distributes tool sets, for many IC design styles, to its member universities across Canada. Hence, all tools utilized in the design of the high-speed communication fabric were obtained from the CMC.

2.1.1 AMS Environment

The AMS flow uses the AMS Environment^{tm} and a set of tools tuned to facilitate the development of mixed-signal designs [10]. In essence, the AMS Environment is a feature within Cadence's toolbox that enables mixed-signal and mixed-language designs to be developed by analog and digital designers from the same environment. When large full-custom designs are developed using Cadence tools, means to create and tune transistor circuits are required, along with a solution to create a test fixture with functional testbenches to stimulate the digital "analog-like" circuit. A variety of Cadence tools are necessary within the AMS flow to achieve that:

1. Command Interpreter Window (CIW).
2. Cadence Hierarchy Editor [11].
3. AMS netlister [10].
4. AMS compiler [10]
5. AMS Design Prep [10]
6. AMS elaborator [10]
7. AMS simulator [10], using SimVision [12] as waveform viewer.

The AMS Environment presents a familiar interface to both digital designers and analog designers. For instance, analog designers typically work at the transistor level using a graphical editor. On the other hand, digital designers specify circuits using a text editor and they rely on digital synthesis to map the behavioral circuit description into physical circuits. AMS provides means to seamlessly integrate these two types of design entry tools together in the same design flow.

The CIW is the root of the Cadence environment. It is used to invoke different tools, set design options (such as AMS options), and monitor status, warning and error messages issued by Cadence tools. Virtuoso is used to build the target system. Its versatility allows specifying cells at different levels of abstraction: functional, spectre, hspice, schematic, layout, extracted, etc. The Hierarchy Editor is used to complement Virtuoso. It is this editor that recognizes mixed-signal designs within Cadence, and it provides accurate representations of analog/digital interface boundaries used by AMS to automatically insert interface elements. AMS works intimately with the Hierarchy Editor to generate, compile, and simulate Verilog-AMS^{tm} netlists. Finally, the AMS Simulator is used to run the Verilog-AMS netlist and it displays mixed-signal circuit results using SimVision.

2.1.2 Affirma Analog Artist and SpectreSVerilog

Affirma Analog Artist^{tm} is the typical simulator used for analog design simulations within Cadence. It encapsulates a variety of simulators tuned to simulate different types of designs. For instance, Spectre^{tm} [13] and Hspice^{tm} [15] are both used to simulate flat analog designs. On the other hand, SpectreSVerilog^{tm} [13] is a super-set of Spectre as it can simulate mixed-signal circuits. Unfortunately, the waveform viewer

used by Affirma only displays a small number of signals at a time, and it does not support buses. Hence, some workarounds are required to export the waveforms to other waveform viewers.

2.1.3 VerilogXL

VerilogXL[tm] [15] is another tool distributed by Cadence. As its name suggests, it is used to compile, simulate, and synthesize Verilog [16] hardware description language files. Even though, it is tuned for digital designs, it turns out to be a handy tool to elaborate and verify Verilog testbenches to be included later in the AMS test fixtures. Here again, SimVision is used to display simulation results and process them.

2.1.4 Virage memory compiler

Virage offers a specialized memory compiler optimized for various memory structures. Embed-It! Integrator[tm] and Custom-Touch-Memory-Compiler[tm] (CTMC) are used to rapidly create high-performance memories. Synopsys VCS does not recognize memory structures effectively and the results it offers are often disappointing. Unlike Synopsys, CTMC generates high-speed memories on much smaller footprints from a specialized standard cell library. With CTMC, it is possible to generate a memory with its description, and a RTL VHDL and/or Verilog files in less than five minutes.

2.2 Standard On-Chip Bus Overview

Since SoC platforms emerged as feasible VLSI circuits, a number of industrial norms appeared to standardize on-chip communication infrastructures. As of today, large SoCs asking for scalable NoCs are almost inexistent for a number of reasons [17]. It makes sense to use smaller types of interconnect fabrics to link cores on the same

substrate with modern SoCs. As the complexity of those systems will eventually scale up, it is expected that the set of available standards will grow so as to offer a variety of solutions tuned for different requirements, similarly to what happened with printed circuit boards (PCB). Hence, this section surveys the main bus standards in the industry.

2.2.1 Advanced Microcontroller Bus Architecture (AMBA)

ARM^{Ltd} owns one of the most accepted shared buses for SoCs in industry. The reason for it is simple: ARM owns a significant market share of embedded processors that are commonly sold with a built in AMBA interface. The AMBA Specification 2.0 [2] defines a set of three fully synchronous SoC buses: Advanced High-Speed Bus (AHB), Advanced System Bus (ASB), and Advanced Peripheral Bus (APB). Both AHB and ASB buses are used with systems requiring the highest performance. On the other hand, the APB bus is oversimplified. It makes no provisions for multiple bus masters and it is used with modules that are so simple, and yet so slow, that using an AHB or ASB bus would increase their complexity without any gain in return.

The AHB standard supersedes its ASB predecessor. ASB suffers from tri-stated bus signals known to be impossible to test with integrated circuits. It was issued rapidly after SoCs emerged, to fill in the gap between the available standards of that time. Unfortunately, it is inspired from standard PCB buses, which makes ASB maladapted to its environment. The AHB norm was introduced as a remedy to testability problems. In addition, it offers a new set of functions related to split transactions, a feature that is not supported by ASB. Therefore, ASB is not recommended for new designs.

The architecture of an AMBA AHB is illustrated in Figure 2.1. It is characterized by five different actors. Firstly, the bus master is an entity that initiates transfers on the

AHB, i.e., it issues addresses and READ data while it receives WRITE data. Note that the bus master has no authority over the AHB as it must wait for a bus grant issued by the bus arbiter. Secondly, the bus slave is the reciprocal of the bus master. It is the entity that answers the transfer request issued by a master. Hence, it receives addresses and READ data while it issues WRITE data. Thirdly, the bus arbiter rules over the AHB to grant the bus to only one bus master at a time. Many algorithms can be implemented by the arbiter and the specifications leave the arbitration policy open to be chosen by system architects. Fourthly, the AHB decoder interprets addresses to select the one slave corresponding to the target destination. Finally, multiplexed interconnections link all AHB entities together to form the communication infrastructure. Further details are available from the AMBA specification 2.0 [2].

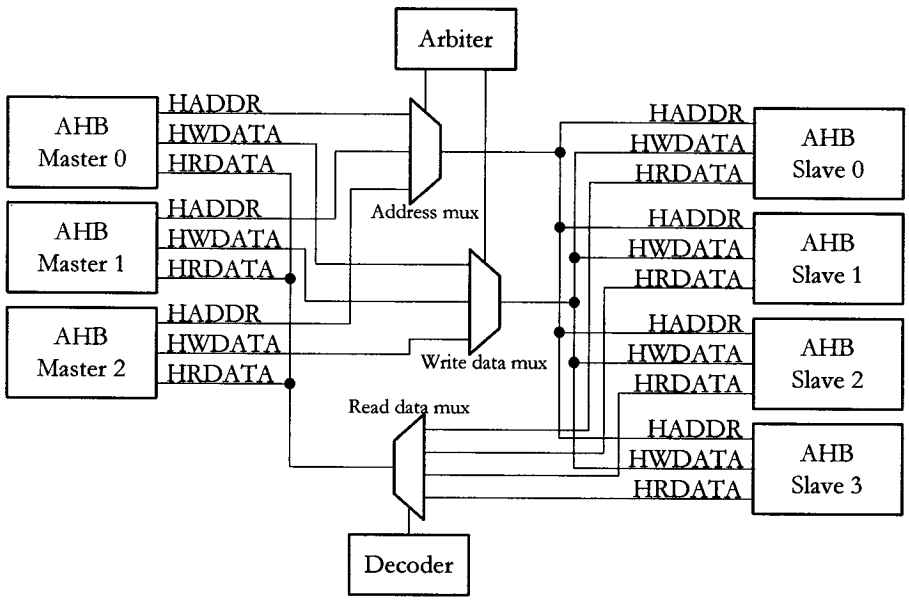


Figure 2.1. Multiplexed AMBA AHB interconnections [2].

An advanced concept of AHB allows defining a hierarchy of buses to effectively augment the bandwidth of a NoC made of AHBs. To achieve that, an interconnect matrix

is used [18], as illustrated by Figure 2.2. The interconnect matrix is shaped as a specialized bridge that accepts multiple AHB "masters" as inputs, and then it multiplexes those AHBs with some arbitration policy toward a number of AHB "slaves". This advanced structure allows the creation of high-throughput communication fabrics where several Advanced High-Speed Buses may evolve in parallel, i.e., concurrently.

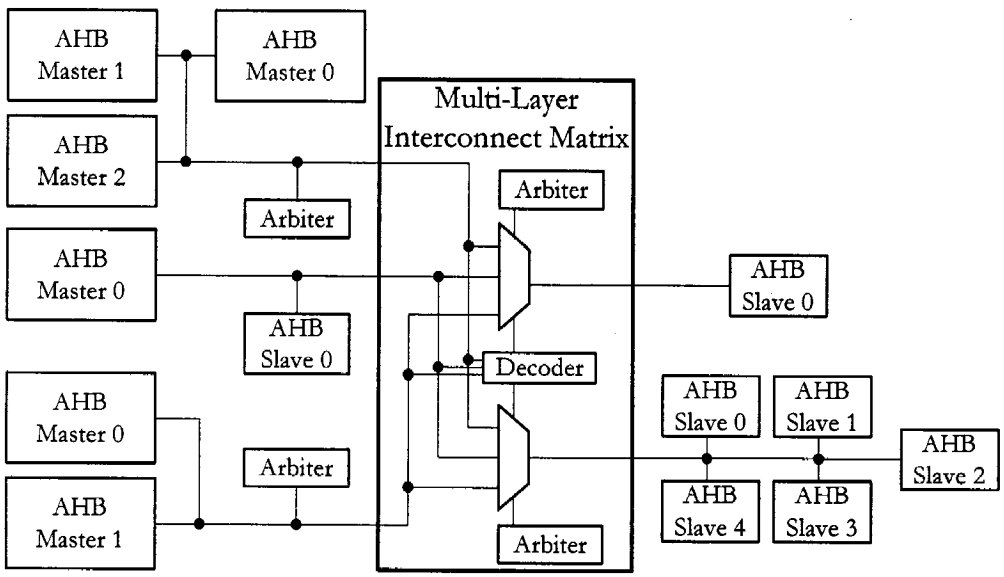


Figure 2.2. Complex Multi-layer AHB system example.

2.2.2 CoreConnect

CoreConnect[tm] is another well accepted on-chip standard bus and it is owned by IBM [19]. It offers similar characteristics as the AMBA AHB standard, yet its typical structure is more complicated than AMBA, as shown in Figure 2.3. The Processor Local Bus (PLB) is similar to an AHB or ASB as it is used where the best performance is needed. The On-chip Peripheral Bus (OPB) is used with low performance cores. Interestingly, the OPB supports multiple bus masters, which is an added value over its equivalent APB. The traffic over the PLB is lightened by the use of a dedicated control bus called Device Control Register (DCR).

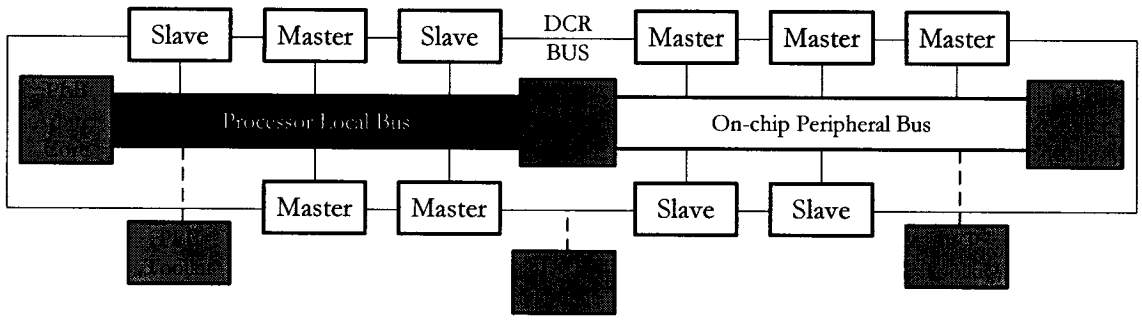


Figure 2.3. CoreConnect organizational diagram (reproduced from [19]).

CoreConnect supports advanced features such as split transactions, decoupled read and write buses to perform two transactions concurrently, address pipelining to reduce overall bus latency, and it is a fully synchronous bus. Even though the many advanced features supported by CoreConnect boost its performance to a higher level than AMBA, its increased complexity makes it less appropriate to implement with full-custom circuits. This is the main reason why CoreConnect was not considered for this research.

2.2.3 STBus

In a multi-processor environment, the system bus is the main shared resource. To this regard, the STBus [20] is a true split-transaction bus. As a matter of fact, CoreConnect PLB and AMBA AHB support split-transaction transfers, but they should occur rarely since they tend to increase the latency. STBus is shaped as a split-transaction bus by definition [21]. The STBus address bus is completely decoupled from the READ data bus and the WRITE data bus, i.e. they are separately arbitrated. This bus is designed to minimize the average latency in a multi-processor system where several simultaneous requests are made for large amounts of data. In addition, the STBus supports shared memory multiprocessing with data coherence using a modified MESI write-invalidate snoopy coherence protocol [21], [22]. The STBus, as any other split-

transaction bus, is extremely complicated to design since each interface must be equipped to keep track of a number of transfers in order to assemble data with the corresponding address. This action is performed with the transaction tags and a tag index.

2.3 Standard Interfaces Overview

An alternative to standard buses that gains acceptance with the SoC industry is to use standard interface protocols to design a custom NoC. Intuitively, standard interfaces define the protocol to exchange data from one instance to the next. Hence, the interconnect architecture is left free to system architects. With this approach, the cores are wrapped into a standard interface and they exchange data with the NoC via the specified unidirectional channel protocol. Simple bus-based NoC to complex interconnect infrastructures, such as butterflies, meshes, or trees [21] can be implemented with such standard interfaces.

2.3.1 Advanced eXtensible Interface

The Advanced eXtensible Interface^{tm} (AXI) [23] protocol is the newest AMBA SoC infrastructure offered by ARM. The AXI protocol is burst-based. Each transaction has the address and control information on the address channel that describes the nature of the data to be transferred. The data is transferred between a master and a slave using a write channel to the slave or a read channel to the master. In write transactions, in which data flows from the master to the slave, an additional write response channel is used to allow the slave to signal completion of the write transfer to the master. The AXI protocol permits address information to be issued ahead of the actual data and enables support for multiple outstanding transfers as well as out-of-order completion of transactions. An

identification tag is associated with every transactions issued between two interfaces. Hence, address and data association is seamlessly performed by the interface. Figure 2.4 shows the AXI channel configuration.

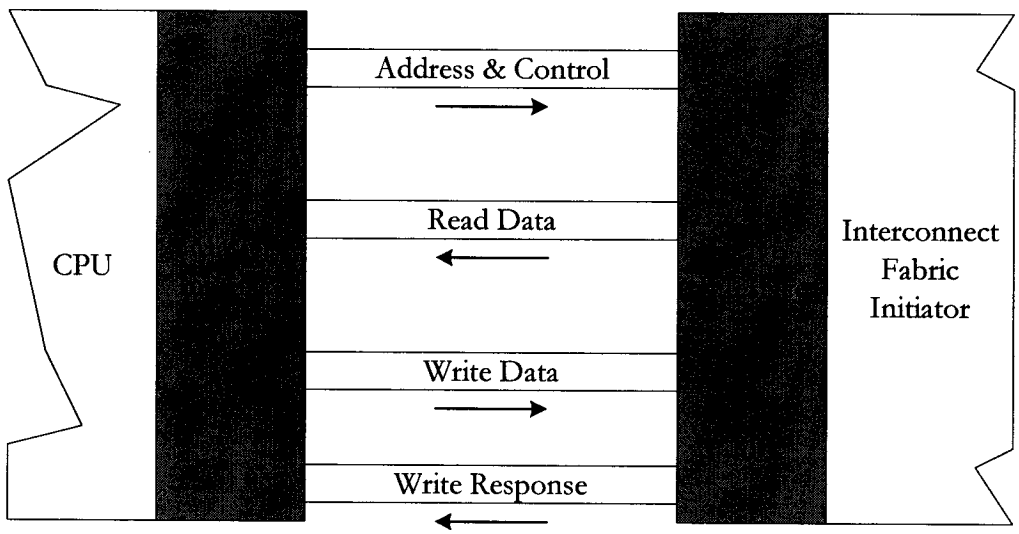


Figure 2.4. AXI channel configuration.

AXI is well suited for high-speed designs since its unidirectional channels allow for register slicing. This feature is used to pipeline interconnects that are known to become increasingly slow with respect to gate delays [24], [25] as the technology scales down. A possible drawback of AXI is that it does not offer dedicated signals to exchange system flags through its normalized interface. Then again, this drawback comes with the advantage of simplicity so as to reduce design efforts and time-to-market.

2.3.2 Open Core Protocol

The Open Core Protocol (OCP) [26] is maintained by an international partnership composed of industrial and academic members. The idea behind OCP is similar to AXI. Here again, different interconnect architectures are possible with OCP as it simply

encapsulates an ASIC core into a standard interface to communicate with the interconnect infrastructure. Figure 2.5 illustrates the resulting system organization (also applicable to AXI). Each SoC core is wrapped into a slave and/or master interface to communicate with the interconnect fabric, which is itself wrapped into standard interfaces, via a unidirectional channel. Unlike AXI, OCP supports several options that are more or less required, depending of the system requirements, which may create diversity within a number of implementations.

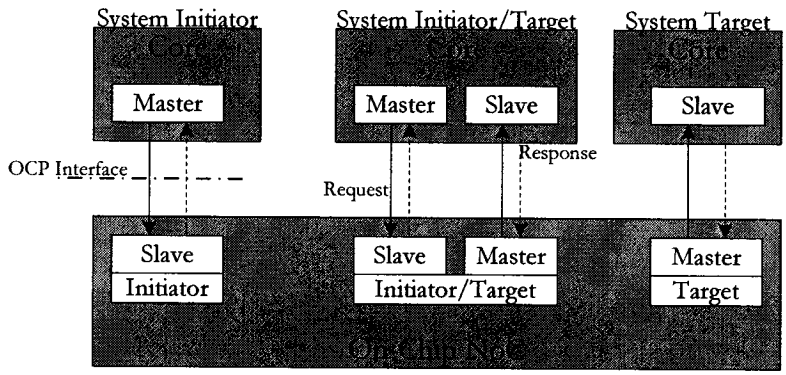


Figure 2.5. System showing an NoC with OCP instances (reproduced from [26]).

2.4 High-Speed Circuit Styles Overview

Designing circuits that operate in the GHz range requires state-of-the-art circuit structures and innovative system architectures. Traditionally, high-speed CMOS logic is achieved with the use of dynamic logic [27]. The circuit style employed to design a circuit makes a significant difference when it comes to challenging the limits of a given technology. Though transistor sizing plays a vital role in circuit speed, the structure of the circuit ultimately sets an upper bound on the maximum achievable performance. Over the past decade, emitter coupled logic, which was widely used with high-performance circuit, has faded away because high-speed CMOS structures become more

and more competitive with more traditional methods (ECL) [27]. This section surveys three sequential logic styles that offer very high performance: no-race (NORA), true-single-phase-clocking (TSPC) dynamic logic, and MOS-current-mode-logic (MCML).

2.4.1 NORA Dynamic Logic Style

In typical CMOS circuits, both static and dynamic CMOS logic are used. For the purpose of system timing, a synchronization strategy [24], [27] is always involved, except for self-timed systems [27]. One of the early popular clocking strategies is clocked CMOS logic (C^2MOS) [27], which uses a non-overlapping pseudo-two-phase clock, as shown in Figure 2.6. Four clocks have to be distributed in such a system in a way that prevents overlaps between two clock pairs. Obviously, clock skew is a serious problem that threatens this requirement and it becomes more apparent with increasing circuit speed [24]. A dead time must be settled between two pairs of clocks to avoid races. Combining clock skew problems with race issues, the dead time must be large enough to provide sufficient stability guaranties. As a result, a significant time is lost with each clock phase to avoid failures and this tends to limit the maximum attainable clock frequency.

NORA stands for NO-RAce CMOS logic [27], [28], and targets the implementation of fast, pipelined datapaths using dynamic logic. NORA uses a true-two-phase clock signal (Φ and Φ'). Forcing some rules on circuit structures avoids races in NORA systems. Two types of stages are used within a NORA pipelined system, Φ-C^2MOS and Φ'-C^2MOS latches (see Figure 2.7). To avoid races, a designer must ensure that the number of inversions between two C^2MOS latches is even. Furthermore, if dynamic nodes are present, the number of static inverters between a latch and the

dynamic node should be even. Finally, the number of static inversion between the last dynamic node and the C^2MOS latch should be even as well. Adhering to the above rules is not always trivial and requires a careful analysis of the logic equations to be implemented. Nonetheless, the elimination of races in case of clock overlap allows speed optimization that qualifies NORA logic as a high-performance design styles.

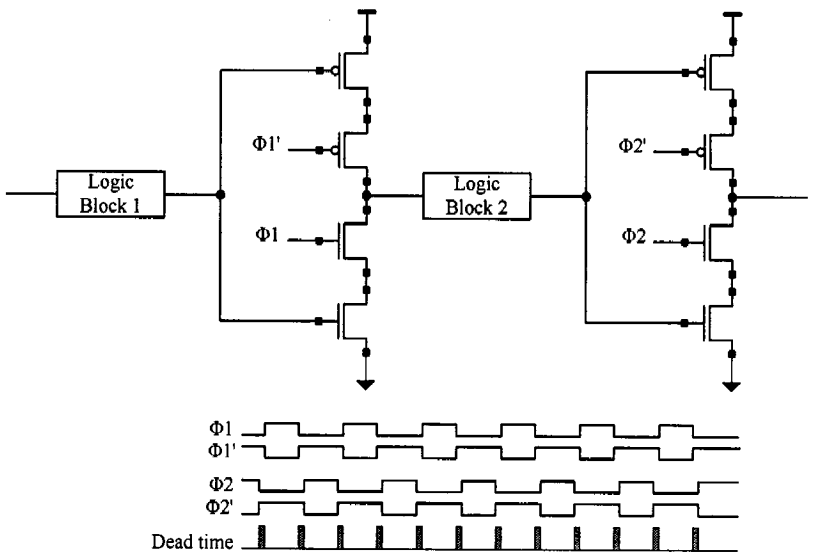


Figure 2.6. C^2MOS dynamic logic.

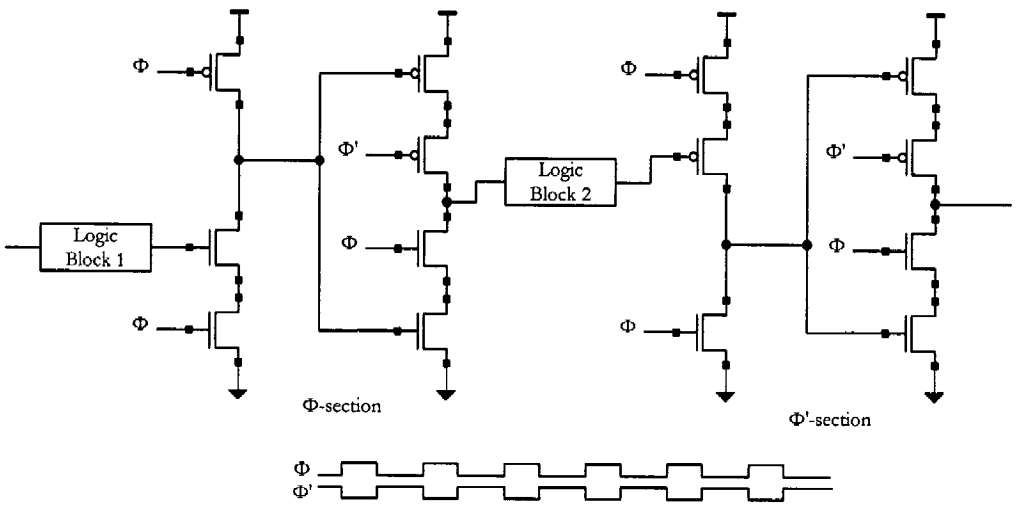


Figure 2.7. NORA dynamic logic.

2.4.2 True-Single-Phase-Clocking Style

NORA logic introduced a dynamic CMOS circuit style that avoids races. It turns out that a further development in clocking strategy is possible, so that a single clock is used to operate dynamic-sequential circuits. The never-ending quest for higher clock rates has led to the proposal of true-single-phase-clocking (TSPC) [27], [29], [30], a powerful design style. Its efficiency partly comes from its use of a single clock phase with no inversion. This leads TSPC to enhanced speed in comparison with NORA logic by eliminating intra-cell clock skew [29]. Yuan and Svensson were the instigators of TSPC logic style. They introduced four level-sensitive latches initially: double-C^2MOS, split-output, TSPC-1, and TSPC-2 (see Figure 2.8). Construction of pipelined-sequential circuits is possible by alternating n-blocks with p-blocks, as with NORA.

Since then, a considerable amount of work has been invested in TSPC and more complex TSPC elements appeared. For instance, edge-triggered TSPC latches were invented so that positive, negative, and double-edge triggered TSPC latches are now commonly used with high-performance systems. For example, a positive-edge triggered TSPC latch may be constructed from the fusion of a p-block with an n-block. In addition, a new variety of TSPC latches makes them appropriate for an increasing number of applications. Non-differential semi-static TSPC flip-flops, dynamic ratio-insensitive differential TSPC latches, static ratio-insensitive differential TSPC latches, and single-transistor-clocked TSPC dynamic and static differential latches are only few examples of the diversity within this powerful design style [30]. Hence, the TSPC circuit technique is not only suitable for high-speed designs, but it also becomes attractive for low-power systems, static and dynamic random access memories for example.

In summary, true-single-phase-clocking has the advantages of simple and compact clock distribution, high-speed, and logic design flexibility. No intra-cell clock skew problem exits with this design style. At a larger scale, clock skew caused by clock delay between different logic blocks in a large system may be minimized with clock distribution networks, and reverse clock distribution [24]. TSPC has been used successfully for the implementation of a number of very high-speed CMOS circuits.

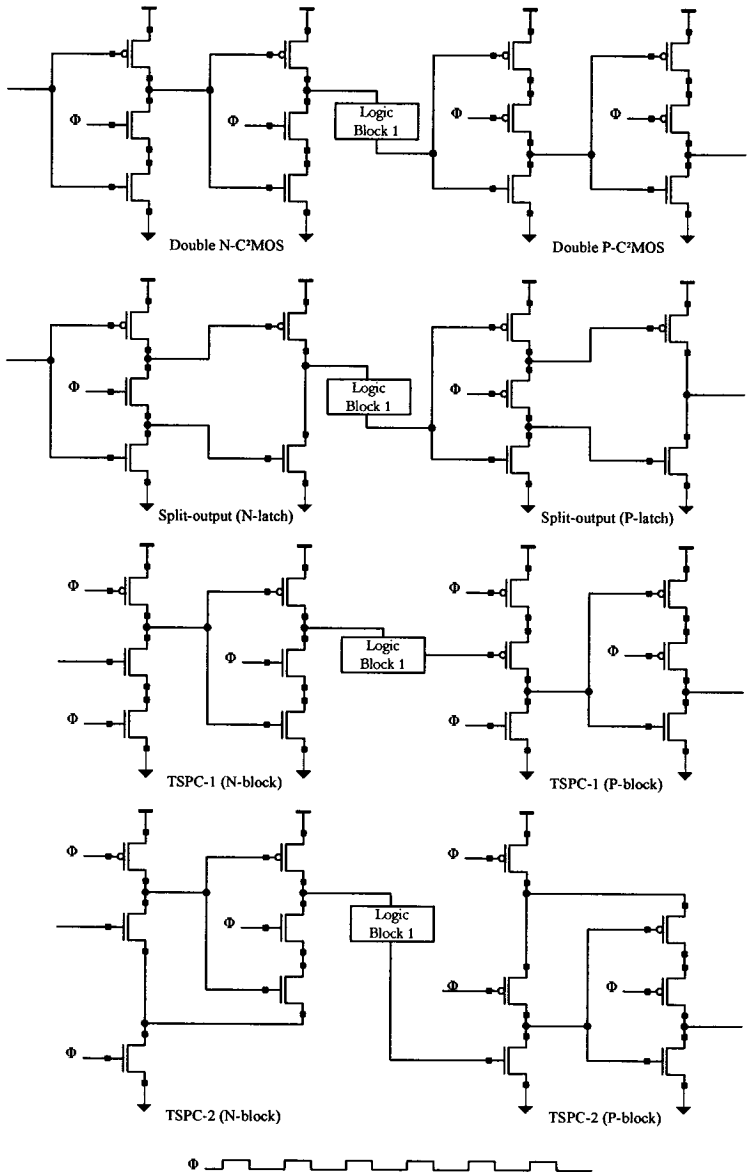


Figure 2.8. True-single-phase-clock latches.

2.4.3 MOS-Current-Mode-Logic

Traditionally, very high-performance digital circuits were implemented using bipolar current-mode-logic (CML) [27]. A MOS-Current-Mode-Logic (MCML) [31], [32] circuit consists of differentially operating MOSFET transistors and a constant current source. Its performance at low voltage is comparable with that of a CMOS circuit and bipolar current-mode-logic circuits. MCML circuits can be used to construct any logic circuits. High-speed compact circuits are feasible, because MCML circuits output complementary signals. An MCML has good characteristics and is widely applicable to logic circuits. It is a useful circuit method for producing GHz processors. It is important to note that the voltage swing is not rail-to-rail with MCML gates but in fact much less, in the order of hundreds of mV [32].

Figure 2.9 shows the theory of operation of an MCML gate with a basic inverter. An MCML gate consists of a differential NMOS pair used to perform logic equations, an NMOS constant-current source used to determine drive current, and two PMOS loads used to determine the voltage swing. Even though MCML is very appealing for high-speed circuits, its design is far more involved than CMOS logic since MCML circuits are much closer to analog design.

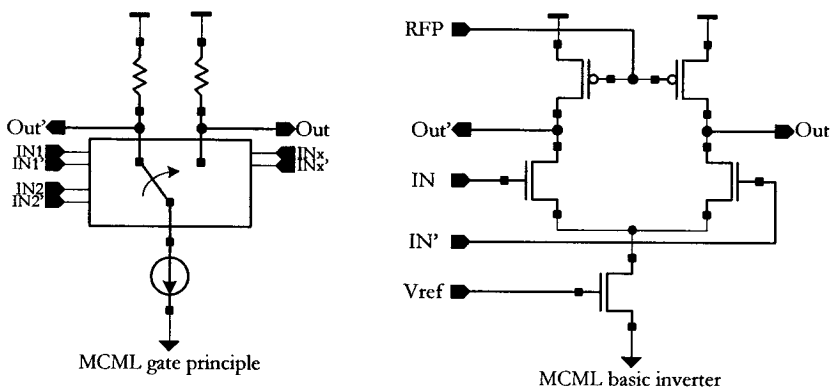


Figure 2.9. MCML method.

2.5 Summary

The high-speed communication infrastructure is implemented using top notch computer aided design tools. Even though the design flow is particularly involved compared with conventional digital logic design methods, a full-custom design flow is used to implement an AMBA AHB SoC infrastructure. The circuits are built with TSPC logic to obtain GHz circuits. AMBA AHB is preferred over other standards for its simplicity and its past history within Microelectronics Research Group (GRM, École Polytechnique of Montréal). Finally, TSPC is selected for its speed, compactness, and ease of use over MCML. MCML can possibly offer better performance, but the complexity of the system to be developed over several levels of abstraction leaves no room to stress design efforts further.

3 FULL-CUSTOM CIRCUIT METHODOLOGY AND TECHNIQUES

As integrated circuits become less expensive and more compact, many new types of products are being introduced, based on digital systems. Consequently, digital logic design is being performed under different motivations. Since each case is different, different design problems are encountered. As a result, different design flows have been established to respond to different needs. For instance, semi-custom design (recently referred to as ASIC design) exists to speedup time-to-market and to reduce design efforts at the expense of reduced speed, higher power consumption, and increased area. On the other hand, deliberate design for high-performance is called full-custom design because each design is manually tuned to high-performance. Digital systems' speed can be significantly improved by deliberate logic design.

This chapter defines the full-custom design flow used to realize a novel 2 GHz AHB SoC infrastructure. The complete design methodology, acquired with experience, is detailed. In addition, a complicated simulation process has been researched in order to fill the gap created by an unusual design approach within Canadian universities. The complexity of the circuit and the relatively complicated set of inputs required for complete stimulus controllability and observability. For instance, a mixed-signal simulation approach has been employed to integrate HDL testbenches into the analog-like circuit simulation process.

Finally, circuit techniques are surveyed in this chapter to further detail the methodology used. Hence, the selection of a TSPC style is explained together with accurate logic design styles and rules enforced in this project.

3.1 Establishing a Design Methodology

Designing a 2 GHz AMBA High-Speed Bus (AHB) [2] is not straightforward. The fastest reported implementation prior to this research adopted a semi-custom design flow. The AHB resulting from the logic synthesis could operate up to 510 MHz according to circuit simulations [33]. This bus is indeed fast considering the numerous bottlenecks raised by limitations of the synthesis process, standard cell libraries, and relatively sparse layouts created by automatic placement and routing [6], [7], [8]. To overcome those bottlenecks caused by the ASIC flow used, a full-custom flow is most appropriate [6], [7], [8]. This led to the implementation of a hard IP operating at 1.4 GHz [34] initially, and that was further improved to sustain a clock rate in excess of 2 GHz [35], the initial target. Nevertheless, achieving a digital VLSI circuit clocked in the GHz range requires careful planning since the best design cannot compensate for an underperforming circuit stage. Hence, this section describes the design methodology followed to develop the 2 GHz AHB fabric.

Even though digital systems' performance can be significantly improved with deliberate logic, every steps of the full-custom design flow is important to achieve performance requirements. Actually, the design of the 2 GHz AHB fabric spans multiple levels of abstraction and follows a long sequence of design stages, as illustrated by Figure 3.1.

3.1.1 Architectural Planning

Defining the bus architecture is the first of many design steps. The structure of a digital system is, without any doubt, a determining factor in obtaining the required performance. This is true regardless of the design flow that is used. Then again, the pressure goes up when a full-custom design is employed. As a matter of fact, the iterative process involved with full-custom design directly threatens time-to-market and profitability. For instance, a product being introduced on the market one year before the competition has a potential of more than twice the profit of the latecomers. The firm that introduces the product has free way to capture the majority of the market share at higher prices, whereas the latecomers can only gather leftovers of the market share at lower prices [8]. Actually, an architectural limitation discovered in a latter design phase may require redoing a significant portion of the design to set the architecture right.

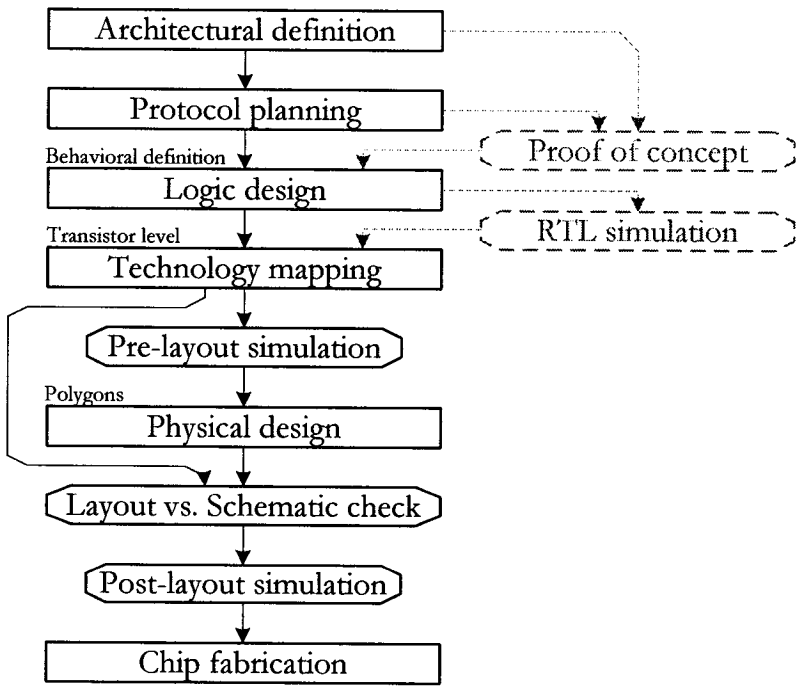


Figure 3.1. Proposed design flow.

To be successful with the architectural design phase, an intimate understanding of the application under development is required. That implies being comfortable with the specifications and requirements of the system, thoughtful planning of modules' interactions, and foreseeing architectural implications on hardware and firmware.

3.1.2 Protocol Design Phase

The communication infrastructure being addressed by this research shapes a network-on-chip (NoC). For this reason, the communication protocol must be meticulously designed since the overall SoC performance depends on the NoC efficacy. Luckily, the AMBA AHB standard helps setting the communication protocol between different systems' actors. As a matter of fact, the AHB standard sets the bus architecture and it defines the signals' behavior. This allows interconnecting modules from different design teams with less effort since a standard interface is strictly compulsory. Nevertheless, the hard IP under development may require going beyond the standard to optimize the SoC infrastructure. In Chapter 4, the hard IP architecture is explained and some amendments to the standard are formulated. Amending the protocol set by the standard brings a risk that must be tackled appropriately. Obviously, a bug in the protocol would force time consuming iterations to fix it.

3.1.3 Logic Design

The logic design phase involves defining the behavioral model of the communication infrastructure. Logic networks with pipelining are described by this design phase. With careful partitioning, it is possible to establish behavioral equations manually. However, when the functions of a partition are too involved for human minds,

it is possible to describe the behavior of the component using a hardware description language to extract the logic functions [8]. The resulting equations will be given under an AOI (AND, OR, and INVERTER) or OAI shape in most cases. Then, it is possible to map those equations into transistor circuits. Yet again, the full-custom circuit resulting from that approach is similar to what CAD tools would create using standard libraries. Hence, the performance gain may be limited.

3.1.4 Technology Mapping

Technology mapping consists of translating logic networks into transistor circuits. Of course, it is possible to use a simple AOI/OAI design style (standard AND, OR, INVERTER / OR, AND, INVERTER gates) to perform this step, but deliberate logic design allows for more aggressive circuit structures. As a result, each equation is studied cautiously so as to research an optimized transistor structure minimizing fanin effects, logic depth, and intrinsic delays. Obviously, technology mapping is the first design phase where a significant speed improvement is achieved from meticulous circuit techniques. The freedom brought by full-custom design often results in unconventional transistor constructs outperforming any equivalent semi-custom designs.

3.1.5 Physical design

The physical design phase is the last phase capable of great speed improvements over automated designs. Performing layout for a full-custom design is a painstaking endeavor to most draftspersons. However, circuits laid out using handcrafted polygons are more compact than those obtained from CAD tools. Needless to say, physical design

is critical to high performance if meticulous done, yet a lousy layout can rapidly destroy all speed gains made previously.

3.2 Simulation Process

Interestingly, the full-custom design flow utilized with the 2 GHz AHB fabric raised a serious problem regarding simulations. The digital design flow [36] maintained by the Canadian Microelectronics Corporation (CMC) is inadequate for full-custom design since it is meant for ASIC design. Intuitively, the 2 GHz AHB fabric resembles more to an analog design, since the work is performed at transistor level in an analog way. For this reason, the analog design flow [37] is more appropriate to the design style employed with the hard IP. However, digital circuits are often characterized by numerous digital signals where a specific bit organization is important. In order to stimulate the circuits with relevant test vectors, or simply to activate a specific state, controllability of the test vectors is required. Achieving this objective with the analog design flow at hand is not easily realized. Furthermore, a great number of signals are to be observed in order to attest the correctness of the design. WaveScan^{tm} [38], the waveform viewer provided by Affirma Analog Artist^{tm} , does not support the notion of bus. For this reason, WaveScan explodes groups of signals to display them, so groups are difficult to read and they consume precious space on the display to analyze the response of the modules. Here again, a solution had to be envisioned to observe the system response to test vectors. Therefore, an alternate simulation flow has been researched in order to apply test vectors from either a hardware description language (HDL) or a hardware verification language (HVL), and to observe them accurately.

The solution for the above stated problem resides in Cadence mixed-signal design tools and Verilog [16] as the hardware description languages used to specify test cases. It is possible to specify functional views into Cadence Virtuoso[tm], so as to create test fixtures including the analog component with a Verilog file to control the stimulus applied to the device under test (DUT). The Hierarchy Editor inserts digital-to-analog (DAC) instance in between the digital stimulator and the DUT to effectively convert the digital signal into an analog signal. Options allow specifying V_{IH}, V_{IL}, t_r, and t_f to shape the signals according to the specifications of the DUT. Similarly, an analog-to-digital (ADC) instance is used to interpret the out coming analog signal by the testbench. Hence, V_{OL}, V_{OH}, and the maximum transition time can be specified for conversion. Figure 3.2 shows a typical test fixture used in the design of the 2 GHz AHB fabric.

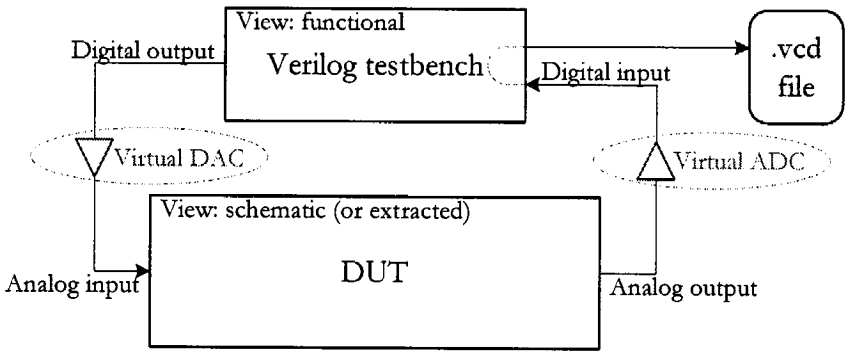


Figure 3.2. Test fixture used for full-custom simulation.

SpectreSVerilog[tm] is a mixed-signal simulator that combines SpectreS[tm] capabilities with Verilog XL[tm] to effectively simulate analog and digital cell views cooperatively. Hence, Verilog XL feds the virtual DAC with test vectors that are applied to the analog cell view. On the other end, the response of the DUT is converted back to digital signals that are dumped in a .VCD file. The .VCD file saves all signals involved in the Verilog file. Hence, observability is possible from Verilog XL since the ADC

provides a mean to read back the DUT response. Afterward, it is possible to enter post-processing in Verilog XL^{tm} and verify the correctness of the design with SimVision^{tm}, the waveform viewer included with Verilog XL. The resulting waveform is similar to a typical RTL simulation. Any signals that remain in between V_{OH} and V_{OL} longer than the maximum transition time specified for the ADC interface are shown with a red rectangle, meaning undefined value. Hence, voltage level problems along with slow transitions are seamlessly traced with this simulation method.

3.3 Circuit Techniques

Designing digital circuits clocked in the GHz range is a thorough commitment that requires state-of-the-art circuit techniques combined with aggressive transistor sizing. The never-ending quest for higher clock rates led to the proposal of true-single-phase-clocking (TSPC) [27], [29], [30], a powerful design style. This dynamic circuit technique allows reaching high frequencies. Its efficiency partly comes from its use of a single clock phase with no inversion. This leads TSPC to improved speed in comparison with the no-race (NORA) [28] as it avoids intra-cell synchronization bottlenecks [29]. Yuan and Svensson [29], [30] proposed four high-speed TSPC latches: TSPC-1, TSPC-2, C^2MOS, and split-output. Even though they claimed that TSPC-1 and TSPC-2 should perform faster, other researchers have shown that C^2MOS and split-output outperforms the former latches [39]. However, beyond this debate, the question of which TSPC style is best suited for a design remains unanswered. This section attempts answering this question based on the experience gained through the design of a 2 GHz communication infrastructure targeting high-end SoCs.

In addition to the TSPC style employed, a discussion on transistor sizing is included in this section, along with the preferred logic styles, and the design rules enforced to improve the probability of success. The aim here is to transmit the knowledge on high-speed circuits, acquired through experience, to possibly shorten the learning curve that other designers targeting this kind of performance could experiment.

3.3.1 Selecting a TSPC Style

To implement 2 GHz AHB cores, three positive edge-triggered latches have been studied: TSPC-2, split-output, and C^2MOS (see Figure 3.3). Even though all candidates meet the target speed, there are timing issues to account for in selecting an optimal style (speed wise). As a matter of fact, besides the transistor count, the optimal TSPC solution should maximize the slack time allotted to useful computation (t_{logic}) and interconnect delays (t_{INT}). Hence, the maximum allowable slack time (t_{SL}) between two sequentially adjacent registers is given using equations (3.1) and (3.2) [24]:

$$t_{SL} \leq t_{CP} - (t_{C\text{-}Q} + t_{SETUP}) \tag{3.1}$$

where,

$$t_{SL} = t_{LOGIC} + t_{INT} \tag{3.2}$$

and $t_{C\text{-}Q}$ is the time from rising clock edge to output valid, t_{CP} is the clock period, and t_{SETUP} is the setup time of a register.

Table 3.1 summarizes various parameters of the TSPC latches obtained by circuit simulations. Even though the hold time of TSPC-2 (Figure 3.3a) outperforms any other style, its high setup time severely hinders the slack time. This is further aggravated by a slower clock-to-output delay, which creates the worst slack time of all three styles. Hence, TSPC-2 is considered the least effective for the design of 2 GHz AHB cores.

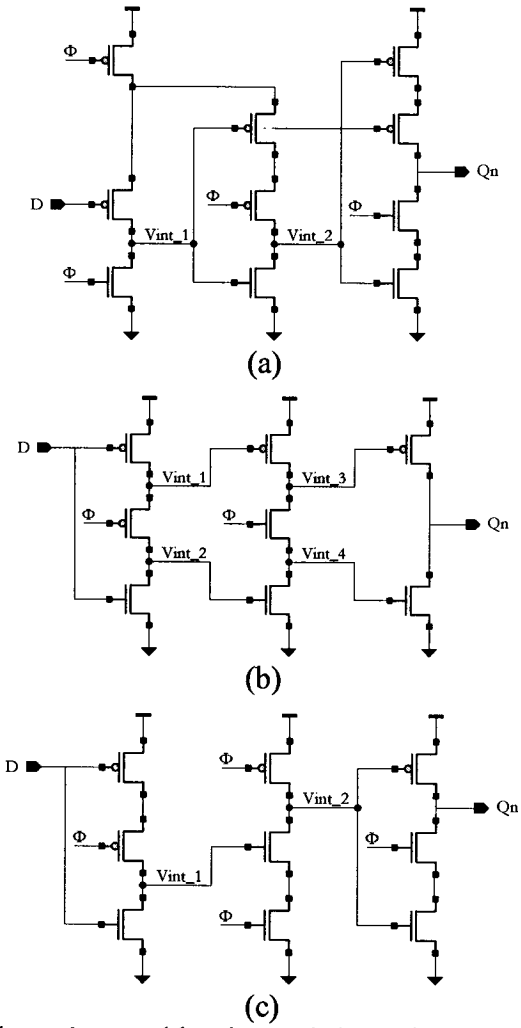


Figure 3.3. Positive edge-triggered latch candidates for the 2 GHz AHB. (a) TSPC-2, (b) Split-output, (c) C²MOS.

TABLE 3.1. COMPARISON PARAMETERS OF THREE TSPC LATCHES.

	TSPC-2	Split-output	C²MOS
Transistor Count	10	9	8
t_{SETUP} (ps)	195	170	50
t_{HOLD} (ps)	0	165	55
$t_{C\text{-}Q\,(MAX)}$ (ps)	125	105	81
t_{SL} (ps)	180	249	345

The reduced clock fan-in of split-output latches (Figure 3.3b) is an advantage that cannot be overlooked, since the power consumption in clock distribution networks is a very significant factor in modern ICs [24]. The reduced transistor count of the split-

output TSPC latch has a tendency to speedup clock-to-output delay, but these latches require a long setup and hold (more than half the clock period). Another possible drawback of the split-output approach is that all internal nodes (Vint_x) do not have a full voltage swing, as some single transistors are used to propagate high and low logic [29]. Therefore, accounting for all factors, split-output latches are used sparingly with the 2 GHz AMBA bus, where there is no logic involved in signals path, as in the case of the datapath. For instance, the datapath is mostly made of 2-to-1 multiplexers, used as an enabling mechanism, and latches. As the bus width and address space increases, the complexity of the datapath explodes. This damps the clock distribution network significantly. For this reason, a reduced clock fanin is appropriate to reduce the burden put forward to the clock drivers.

Finally, C^2MOS (Figure 3.3c) offers the best trade off to obtain the fastest logic as its slack time appears to be the best of all three TSPC techniques. For this reason, it is largely used in the design of the high-speed bus, especially where complex decisions need to be computed between two sequentially adjacent registers. In addition, its internal nodes offer improved stability over split-output latches so that PMOS and NMOS are used more effectively. Also, it is easier to embed logic into C^2MOS latches.

3.3.2 TSPC Device Sizing

The design of a digital circuit operating in the GHz range requires aggressive transistor sizing. Each component must drive a sufficiently large current to overcome the parasitic capacitance and resistance, which are extracted from the process used for the physical design. Since gate delays decrease as the technology scales down, an ever increasing disparity between wire delays and gate delays is seen [24], [25], [40]. This

phenomenon is mainly due to parasitic RC components that, in a sense, create a low pass filter limiting the maximum frequency of a circuit. Hence, it is desirable to size the memory elements in a way that minimizes input load and maximizes output current.

In essence, an edge-triggered TSPC latch is made of three inverting stages. This hints at using buffer theory to gradually boost the current capabilities of each succeeding stage [27]. Furthermore, each device is configured with the minimal gate length specified by the process used, whereas its gate width is optimized for speed. Proper transistor sizing significantly impacts a device performance, as discussed in [6], [8], [27], [29]. To perform TSPC device sizing, the output stage is sized in the first place to match the attributes of an inverter driving a large output load. From that point, the second stage is scaled down by a factor of two and the first stage by a factor of four, with respect to the output stage so as to maximize current driving capabilities within the latch. Figure 3.4 illustrates the buffering theory applied with TSPC stages. This approach generates a small latch input capacitance with respect to its output current capabilities. Hence, the third stage can muddle through with a large fanout, while the fanin of the latch is small to minimize RC delays experienced by the input logic, and to allow for longer interconnects.

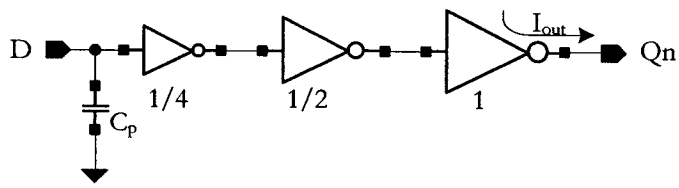


Figure 3.4. Buffering TSPC stages.

3.3.3 Logic style and Design Rules

The maximum clock frequency in synchronous systems is constrained by logic coupled with interconnects delays, from one latching instant to the next [24]. One way to minimize this delay is to decompose the complexity of each logic block so as to create small and manageable sub-blocks that operate within one clock period. Even though this approach suffers from an initial delay, this is acceptable in pipelined structures, yet a practical limit to logic reduction sets the system efficiency [41]. To a certain extent, the logic style employed makes a difference.

The complementary CMOS circuit technique allows the creation of complex logic functions. Unfortunately, this design style is particularly sensitive to RC delays caused by connecting transistors in series. It is quite obvious that the number of transistors interconnected in series should be minimized to obtain high-speed logic.

As a result, a design rule was created to limit the number of transistors used in series. Simple decisions were implemented using pass logic [8], [27] while complex computations were designed with a parallel static-NAND (or a parallel static-NOR) structure close to pseudo-NMOS logic, as described in Chapter 5 and [35]. This is seamlessly implemented, since many logic networks can be reorganized in a parallel static NAND structure with simple algebraic manipulations. Note that with CMOS technology, inverter delays are still small so that extra inverters can be accepted in the blocks.

In addition, to guide logic design, a supplemental rule was enforced with the high-speed AHB circuits to obtain successful "at speed" simulations after layout extraction. With the 0.18 μm CMOS process in use, it was expected that the parasitic capacitances

would double from circuit level simulations to extracted layout simulations, due to the many added wires. This affects the slack time by a factor of two, as suggested by (3.3):

$$t_{SL} = R \cdot 2C_p \tag{3.3}$$

where C_p represents the total parasitic capacitive load attached to a device. Obviously, the capacitive load has a linear relationship with the slack time. Hence, logic blocks were not allowed to consume more than 50% of t_{SL} during circuit level simulations so as to prevent exceeding the time budget after layout extraction.

Finally, keeping the transition times below 100 ps in circuit level simulations ensured further room for degradation. In addition, V_{OH} and V_{OL} were kept within VDD ± 10% to respect industrial practice to this regard.

3.4 Summary

The aim of this chapter is to describe various methodologies used to cope with the design of a novel 2 GHz communication infrastructure. The full-custom design flow employed is explained in details. In addition, a difficult simulation process has been researched and described in details in this chapter. This mixed-signal simulation method is most needed to apply well thought test vectors and to verify the correctness of the high-speed fabric.

Finally, a section is devoted to circuit techniques since deliberate logic has the potential of improving system performance significantly. In addition, designing for high-frequencies is poorly documented in the literature as most books only define the problematic. Hence, metrics for choosing a TSPC style over another are discussed, along with means for sizing the device for high-performance. Preferred logic styles with design rules are also enumerated.

4 AHB INFRASTRUCTURE ARCHITECTURE

The performance of high-end SoCs is characterized by their processors' ability to communicate effectively within the silicon die. Even though a number of communication standards exist to answer this concern, none were designed to interconnect high-speed modules in a simple fashion. As a result, the throughput offered is low, and complex parallel structures need to be constructed to achieve the data crunching rates required by modern applications. A new communication infrastructure, based on the AMBA AHB specification [2], is described in this chapter. Its architecture supports multiple-outstanding data streams through a pipelined shared-memory, specifically designed to meet the data rates required by the new communication fabric. The derived structure is simple, yet powerful, to interconnect a small number of bus masters. In addition, its components support clock rates outperforming every shared bus reported in the literature to our knowledge. The internal targeted frequency is 2 GHz.

This chapter explores the architecture and the operations of the major components of the new communication infrastructure. The master bridges required to link the platform's processor with the high-speed bus are explained. Finally, the structure of the pipelined shared memory is presented with its conflict resolution mechanism, and the arbitration method is detailed.

4.1 AHB Interconnect Megacell Overview

Modern SoCs are typically developed using a semi-custom design flow (commonly referred to as ASIC flow) [6], [7], [8]. This hints that new communication infrastructure interfaces mainly with soft intellectual property (IP) modules, and sporadically with hard IPs when enhanced performance is needed. As a result, the high-performance AHB requires a cautiously designed architecture that combines seamless integration of the modules to provide high-throughput. Obviously, the communication fabric's interfaces must be capable of accepting different clock rates, sometimes much lower than its 2 GHz frequency. This solicits multi-frequency bridges. In addition, several modifications are implemented with respect to the AHB standard so as to obtain the highest possible performance [34]. They range from protocol leveraging techniques to interconnect reconfiguration with an increased pipeline depth. This section introduces the many modules of the new SoC infrastructure together with initial assumptions made to reduce the complexity of this research project.

4.1.1 Clock Domains

As mentioned previously, the 2 GHz communication infrastructure is a hard IP intended to enhance the bandwidth offered by the most demanding SoCs. This suggests that the proposed megacell is surrounded by a number of processing elements that sometimes operate from a much lower clock frequency. This situation is the result of performance bottlenecks caused by the synthesis, placement, and routing processes [6], [7], [8]. Hence, a module developed using an ASIC flow and a 0.18 μm CMOS technology typically operates up to 200 to 300 MHz. This phenomenon must be taken into consideration when designing a high-performance communication infrastructure.

Three distinct clock domains have been considered in designing the 2 GHz AHB fabric. The first domain, referred to as the standard AHB domain, is formed by the processing elements and their local AHB. All standard AHBs share an equal fraction of one specialized AHB's bandwidth. Hence, the standard AHB domain works from a clock frequency of 125 MHz. The second clock domain encapsulates specialized READ and WRITE AHBs. Both specialized AHBs operate with a 2 GHz clock frequency. The third clock domain of interest is again encapsulated by the hard IP. It provides a 500 MHz clock frequency to synchronize the operation of the high-throughput shared-memory. Figure 4.1 illustrates the three clock domains and their components. It is important to mention that synchronization of the different domains is beyond the scope of this research. For the time being, it is assumed that the clocks are generated and synchronized by a phase-lock-loop (PLL) [42].

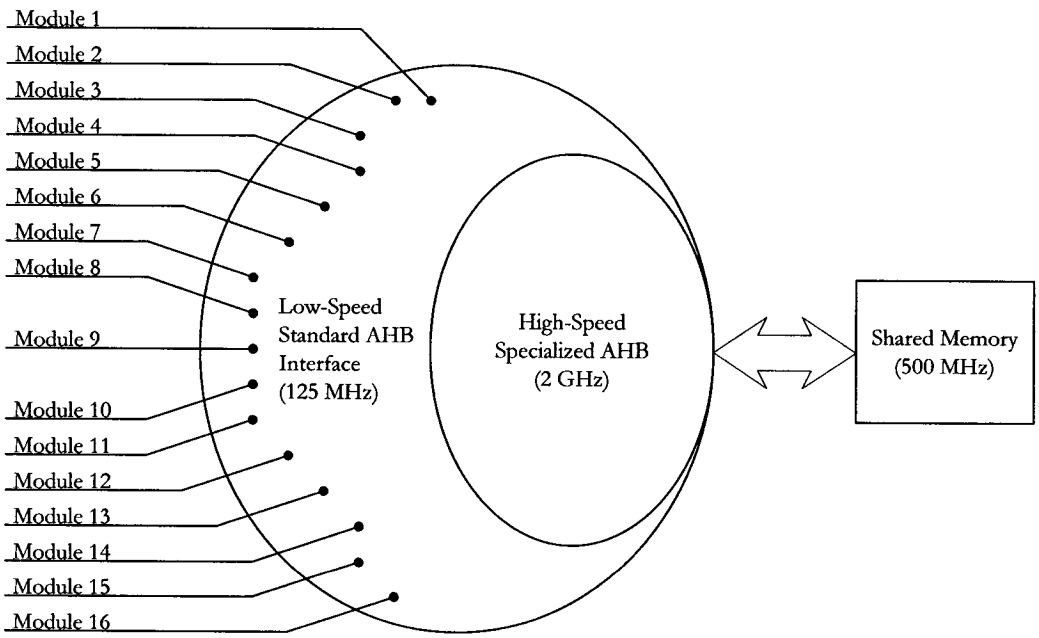


Figure 4.1. Clock domains division.

The new high-speed bus infrastructure does not include advanced synchronization mechanisms such as a globally-asynchronous-locally-synchronous (GALS) wrappers, asynchronous FIFOs, nor synchronizers to decouple the clock domains. For this reason, clock edge alignment is required from the PLL to avoid synchronization failures. This imposes a limitation on the clock rates that a designer may select. As a matter of fact, the clocks of the standard AHB and the clock of the memory must have a period that is a multiple of the period of the 2 GHz clock used by the specialized AHB. This safeguard is required to ensure that edges from interacting clocks coincide with each other on a regular and predictable basis.

Note that even though a unique clock frequency is assumed for the standard AHB domain, it may use multiple clock frequencies in practice, since typical SoCs are composed of several modules of different nature often provided by different vendors. Then again, to simplify the complexity of the first implementation of the high-speed AHB infrastructure, the standard AHB domain has been constrained to use only one clock frequency.

4.1.2 Architectural Overview of the Communication Fabric

The architecture of an Advanced High-Speed Bus is mostly determined by its specifications [2]. However, there exist some grey spots that give room for customization. The new hard IP makes effective use of those grey spots to elaborate an architecture slightly different from what has been specified to effectively provide an aggregate bandwidth that corresponds to a 4 GHz clock speed. Thus, the heart of the high-speed bus is derived from the AHB standard. Still, to obtain the fastest possible speed out of this interconnect infrastructure, READ and WRITE operations have been

separated on two physically independent AHB, referred to as specialized AHB. In addition, all superfluous signals to the shared memory were screened out from the implementation to simplify the circuits. Finally, the pipeline depth has been adjusted to optimize the throughput of the specialized READ and WRITE AHBs.

The specialization made to the high-speed AHB buses is not forbidden by the standard. In addition, it allows doubling the effective bandwidth since the READ and the WRITE buses operate concurrently. Furthermore, several processors offered by ARM (the owners of AMBA standards) are available with specialized AHBs, here also to increase the processors' bandwidth. As an example, the ARM1136J(F)-S processor [43] is offered with five concurrent AHB master interfaces: one interface is dedicated to instruction fetch, the second to data write, the third to data read, the fourth to peripheral access and the fifth handles direct memory access (DMA) only.

The proposed communication infrastructure guarantees high-performance access to all the modules it services. An AHB interface supporting the basic features of the standard is provided to facilitate modules integration with the high-performance communication infrastructure, as shown in Figure 4.2. Standard AHB interfaces are used by the processing elements as AHB slaves to access the high-performance communication fabric. Each specialized bus resembles strangely to what has been specified by the standard, except for the AHB decoder which is absent from this design. Banning the decoder is perfectly acceptable since the memory bank forms a single slave. The memories are accessed upon a specific clock phase and not upon the address criteria as in conventional shared-buses. Hence, it is the bus arbiter that determines which SSRAM from the memory bank is accessed at what time. This is not against the

specifications since the preferred arbitration method depends on the specifics of a particular application. It is thus left open to the designer.

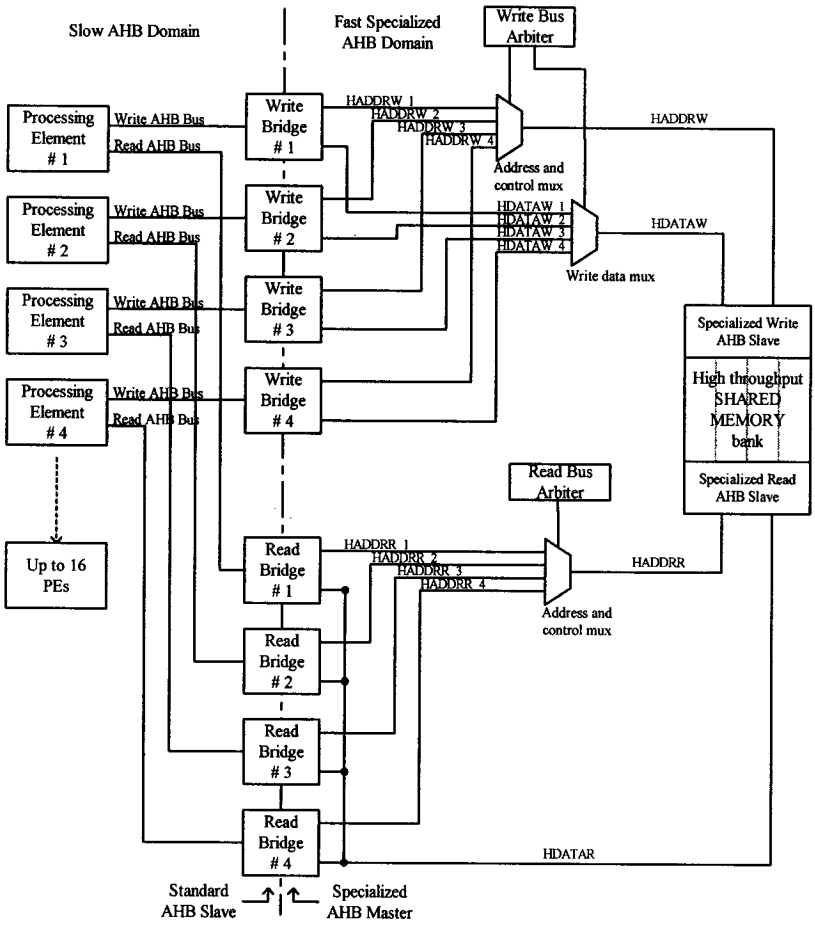


Figure 4.2. Overall structure of the proposed communication infrastructure.

The example shown in Figure 4.2 represents the hypothetical SoC used to create the new communication infrastructure. Sixteen AHB compliant processing elements (PEs), communicate through the novel 2 GHz AHB fabric. Each PE can perform a READ and a WRITE operation concurrently via their local AHBs that are directly connected to the hard IP. Hence, with a bus operating at 2 GHz, it is possible to guarantee a bandwidth of 125 MHz with a one cycle latency delivery time up to sixteen processors [34].

The high-speed AHB fabric is best suited for a hierarchical bus model. This concept is known as a Multi-Layer AHB, and it was introduced by ARM in 2001 [18]. A Multi-Layer AHB uses an interconnect matrix that multiplexes several AHB masters interacting with a varying number of AHB slaves. In a sense, this interconnect matrix can be seen as a special bridge routing transfers inside a hierarchy of AHBs. The interconnect matrix that ARM suggests is indeed flexible and offers a high throughput since it allows many transfers to occur concurrently. However, when many PEs compete to access the same resource (e.g. a shared memory, a FIFO, etc.), the bandwidth of the Multi-Layer AHB goes down to that of a single semi-custom AMBA AHB bus. The novel AHB fabric queues the transfers and processes them one-by-one through a collection of high-speed bridges creating, for the soft-cores connected to its ports, the illusion that their requests was served with no apparent latency by the 2 GHz communication infrastructure.

4.1.3 Specialized AHB signals

The specialized READ and WRITE AHBs are designed as light versions of the standard. Except for the increased pipeline depth on the READ bus, all signals interact as specified by the standard. Then again, all unused signals were screened out from the 2 GHz interconnect infrastructure to ease its design. For this reason, a list of remaining signals is provided in Table 4.1. Note that suffixes W and R are used to differentiate the WRITE bus from the READ bus. For further details on AHB signals interactions, the reader is referred to AMBA specification 2.0 [2].

TABLE 4.1. SPECIALIZED AMBA AHB SIGNALS.

Name	Source	Description
HCLKF	Clock source	This clock times all bus transfers. One clock cycle is equivalent to one bus cycle. All signal timings are related to the rising edge of HCLKF.
HRESETn	Reset controller	This signal is used to reset the system and the bus. HRESETn the only is active LOW signal.
HADDRW[4:0] HADDRR[4:0]	WRITE/READ bridges (master)	The 5-bit address bus is used to specify the memory location to be accessed.
HTRANSW[1] HTRANSR[1]	WRITE/READ bridges (master)	Indicates the type of the current transfer. 1 = NONSEQ and 0 = IDLE.
HDATAW[31:0]	WRITE/READ bridges (master)	The 32-bit write data bus is used to transfer the data from the WRITE bridge to the interleaved memory.
HDATAR[31:0]	Interleaved memory (slave)	The 32-bit read data bus is used to transfer the data from the interleaved memory to the READ bridge.
HBUSREQWx HBUSREQRx	WRITE/READ bridges (master)	A signal used to notify the arbiter that bridge x requires access to the bus.
HGRANTWx HGRANTRx	Micro-programmed arbiter	This signal is used to indicate to bridge x that it is currently the highest priority master. In contrast with the standard, bridge x gets ownership of the bus only on HGRANTW/Rx.
HMASTERWx[3:0] HMASTERRx[3:0]	Micro-programmed arbiter	These signals from the arbiter indicate which bus master is currently accessing the bus. They are used by the micro-programmed arbiter to follow up on the access sequence.

4.2 Interleaved Memory

The availability of a high-throughput memory is a key to the success of the new AHB communication fabric. Indeed, the development of a memory supporting a 2 GHz clock rate is extremely difficult [27]. For this reason, the shared memory rapidly turned into a serious treat to the feasibility of the 2 GHz AHB infrastructure. The architecture of the high-speed communication infrastructure relies on a high-performance memory to act as a communication buffer. However, if the shared memory offers a bandwidth inferior

to the 2 GHz AHB, the throughput of the communication fabric is degraded to that of its buffer. Therefore, the shared memory has been subject to intensive research.

It was decided early that designing both an AHB and a memory supporting a clock rate of 2 GHz was unacceptable considering the resources at hand. The fastest embedded memory available through the Canadian Microelectronics Corporation (CMC) is a synchronous static random access memory (SSRAM) from Virage [44]. This memory compiler provides a high-density memory with a clock frequency up to 500 MHz when targeting a 0.18 µm CMOS process. The gap to reach the required performance is still important, but pipelining is possible to create a memory bank emulating a 2 GHz memory.

The memory bank is made of four dual-port random access memories from Virage to create a memory sustaining two billion data accesses per second on both ports (Figure 4.3). Each memory port interfaces to one of the specialized AHB so as to offer this aggregate bandwidth of four billion data accesses per seconds.

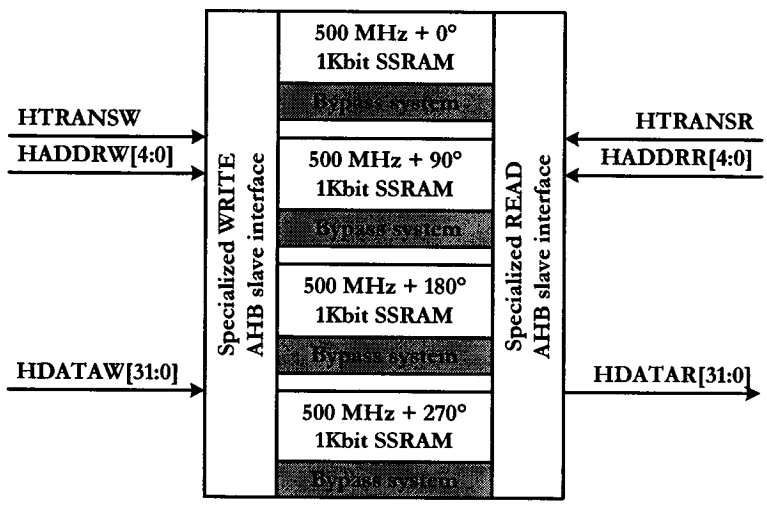


Figure 4.3. Block diagram of the high-throughput interleaved memory.

The four SSRAMs are pipelined with an interleaved clock, i.e. the memories' clocks are shifted by 90° with respect to one another so as to obtain a memory bank clocked at 2 GHz. A timing diagram reveals how the four SSRAMs' clocks are organized in Figure 4.4. A 90° phase shift on a 500 MHz clock results in a delay of 500 ps. This delay is equivalent to one 2 GHz clock cycle. Thus, the 90° phase shift provides a mean to align one SSRAM clock edge with every positive clock edge on the specialized AHB. This method generates a memory that accepts a new address and completes a new operation every 500 ps.

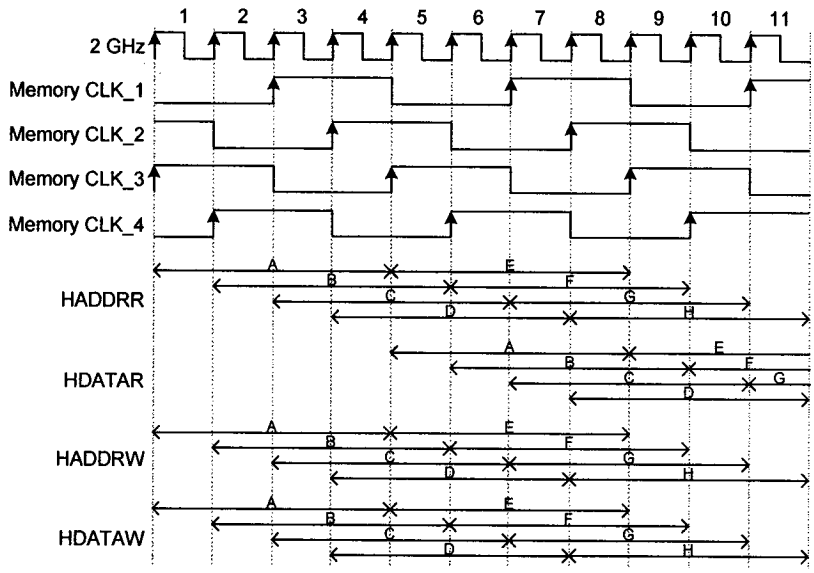


Figure 4.4. Interleaved memory timing diagram.

As mentioned previously, the pipeline depth of the specialized AHB had to be increased from two to eight levels as a means of optimization. The motivation to increase the depth of the AHB pipeline becomes evident from the memory operations. On the high-speed AHB, a new address phase is initiated every clock cycle. However, when a read operation is attempted, the 500 MHz memories are slow to provide data with respect to the high-speed AHB. This is where an increased pipeline depth becomes necessary.

When a memory accepts a new address, six 2 GHz clock cycles are needed by the SSRAM to fetch the data. To avoid wasting four bus cycles, a new access is initiated on each memory until the arbiter rolls back to the first memory with the first data item ready to be delivered. This principle is illustrated in Figure 4.4.

It should be noted in Figure 4.4 that the specialized READ bus (denoted by HADDRR and HDATAR) and the specialized WRITE bus (denoted by HADDRW and HDATAW) are processed differently by the buffer memory. As a matter of fact, a READ operation is performed over two 500 MHz clock cycles whereas a WRITE is committed over only one cycle. For this reason, WRITE and READ operations have been split over two independent AHB systems. This is required to simplify the circuits, and it allows developing a specialized AHB optimized to take advantage of the shared memory operations. Hence, for a READ operation, it is appropriate to issue the address phase on the high-speed AHB as soon as possible to obtain the data within the allowed time frame. For a WRITE operation, it is preferable to hold on the address until the standard AHB delivers the data to be written to the memory. As a result, the specialized WRITE AHB issues the transfer late whereas its converse bus issues the transfer early.

A READ cycle is more critical than a WRITE cycle. This is explained from real-time deadlines envisioned to acquire the READ data. This is why READ transfers are issued as soon as addresses are available: to provide time to return the READ data to its PE. Then again, WRITE operations also have real-time deadlines to respect, but the PEs do not have to wait on the WRITE cycle to be completed to proceed with their work as long as a guarantee that the write will be committed properly is provided. This is always the case with the SSRAMs under use: a write operation is guaranteed to be completed.

For this reason, the processing elements may consider a WRITE cycle completed as soon as the 2 GHz bridge takes delivery of the data.

Another important component required by the interleaved memory bank is the memory wrapper. It is used to decouple the 2 GHz clock domain from the 500 MHz. The duration of each bus cycle on the fast AHB is of short duration compared with the memory cycles. It is for this reason that wrappers need to be used. Their task consists in sampling and holding address, control, and data (data only for WRITE operations) bits for the duration of a memory cycle.

Finally, there exists a possibility that the READ AHB and the WRITE AHB attempt an access to the same memory location simultaneously. If this situation occurs, the data being read may be corrupted. An easy way to handle address contention is to raise an error flag to re-attempt the memory access. However, this strategy may severely hinder the system's determinism since the probability of failure over a small address space is substantial. As an example, consider the head and tail pointers associated with each virtual buffer in the memory. The probability of address contention over one of these pointers cannot be overlooked. Unfortunately, every failure tends to build cycles of latency which ultimately leads to clogging SoC communications. A better way to resolve this type of conflict is to equip the shared memory with a bypass system, so as to directly route the data being written by the WRITE bus onto the READ bus via the write-through port provided with the SSRAM. The bypass mechanism reacts only upon conflict detection.

In case of address contention, the internal operation of Virage's SSRAM prioritizes the WRITE over the READ operation. Therefore, when a conflict arises, the

WRITE operation is committed successfully and the data being read must be considered as corrupted. It is the responsibility of the designer to detect address contention. Figure 4.5 shows the bypass system. This circuit exploits the write-through feature offered by Virage's SSRAM. In effect, when activated, the SSRAM's write-through mechanism copies the data from DA to QA in the successive memory cycle. Therefore, the data to be read may be corrupted if it is taken from the memory cell itself, but the SSRAM has a mechanism that holds the write data long enough to be used by the specialized READ bus. The role of the bypass mechanism is only to detect address contention, and select the alternate data source in case of positive conflict detection. Figure 4.6 shows a timing diagram illustrating the operation of the bypass circuit. As it is shown, the address contention has no incidence on the successful completion of the read operation since the alternate source (QA) provides valid data with no extra latency. Therefore, the deterministic behavior of the high-speed communication infrastructure is preserved, which is an important characteristic for real-time applications.

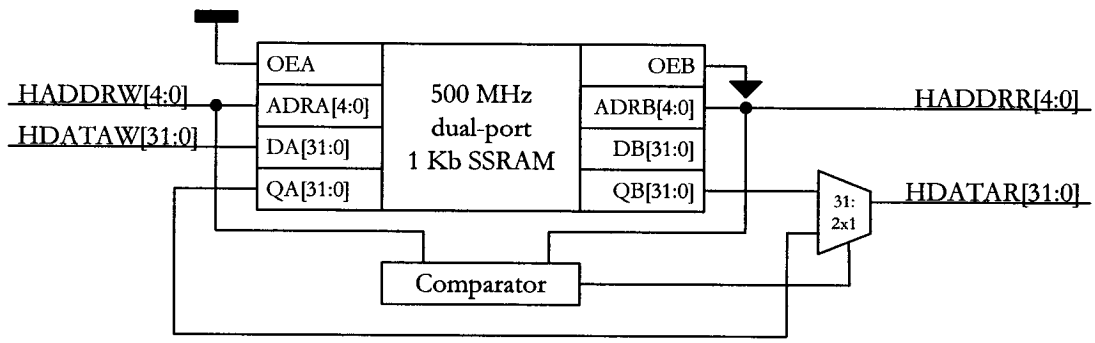


Figure 4.5. Block diagram of the bypass system.

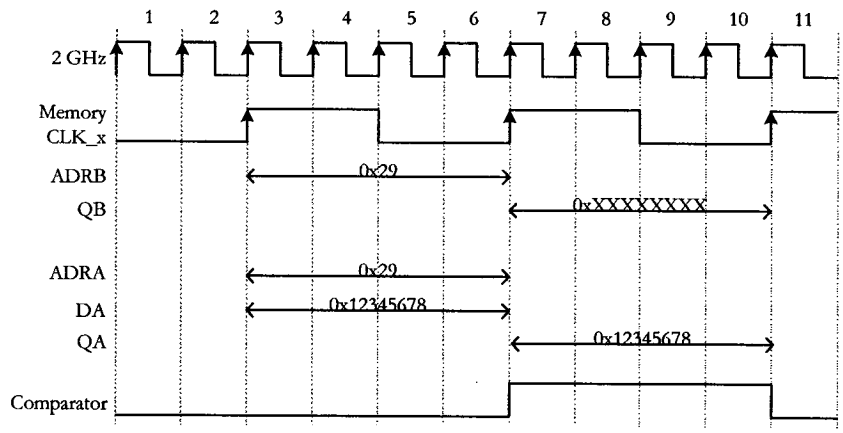


Figure 4.6. Timing diagram of a conflict resolution.

The specialized AHBs operate in a similar way of a Multi-Layer AHB interconnect matrix [18], [35]. As a matter of fact, a Multi-Layer Interconnect Matrix physically multiplexes several AHBs toward a number of slaves at low-speed. The new AHB based interconnect infrastructure time-multiplexes several AHBs towards a shared resource at a high-speed. Thus, the new solution avoids resource conflicts by satisfying many simultaneous transfers at a very high-speed, but this can only work with a proper arbitration mechanism.

4.3 Very Long Instruction Word Arbiter

As mentioned in Section 4.2, the high-throughput dual-port memory bank can support a bandwidth of 4 GHz. This allows emulating a multi-port memory provided that a "phase-aware" arbitration mechanism is developed to manage the traffic on the shared bus. Indeed, the arbiter must be aware of which memory bank a processing element writes shared data into so that it can be read by the destination processor. This adds, on the arbiter, the extra burden of granting the shared bus on a specific memory phase determined by the physical location to be accessed. Figure 4.7 shows an example

defining the difficulty produced by independent memory phases. The arbitration policy used is a naïve implementation of a round-robin static arbiter (Figure 4.7a). The target data to be implemented by this arbitration policy is as shown in Figure 4.7b. However, when the memory phases are taken into account, a discontinuity in the data flow is revealed. This is due to a side effect caused by the operation of the memory bank. The communicating PEs are not mapped to the same physical memory within the memory bank (Figure 4.7c). Hence, they cannot communicate since their requests are never phased together. This naïve arbitration policy results in the dataflow discontinuity illustrated in Figure 4.7d.

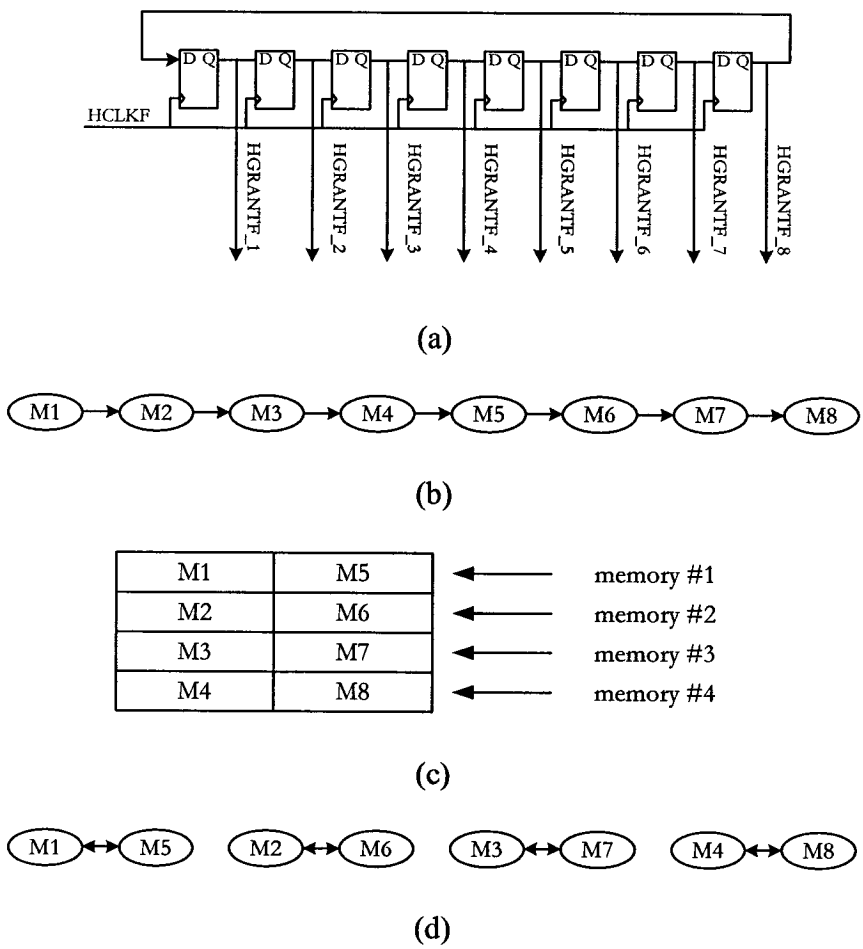


Figure 4.7. Arbitration with phases: problem definition. (a) Round-Robin arbiter, (b) Target dataflow, (c) Memory map, and (d) Resulting dataflow.

To design a "phase-aware" arbitration schedule for the above example, distinct schedules must be executed on the READ and on the WRITE bus. Different schedules makes possible mapping communicating entities on the same physical memory. Hence, a solution to the above problem is shown in Table 4.2.

TABLE 4.2. PROPER SCHEDULING FOR A CONTINUOUS DATAFLOW.

PEs schedule	Memory phases															
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
WRITE AHB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	--
READ AHB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	--

Indeed, the arbitration schedule shown in Table 4.2 is simple to establish since the hypothetical application is simple. Nonetheless, a real application with dynamically changing bandwidth needs and deadlines may be more difficult to compile with the unconventional memory bank designed for the very high-speed AHB. There exist two fundamental approaches to design arbiters: those that execute a static schedule and those that adapts dynamically to incoming requests. Yet, both arbitration methods must account for memory phases. To obtain better determinism and ease the design phase, the specialized READ and WRITE AHBs use a pre-defined arbitration schedule. In addition, the arbiter provided to both high-speed AHBs is structured in a way that resembles typical very long instruction word (VLIW) micro-programmed processors [1], [45]. VLIW machines typically result in simpler architectures and simpler circuits. However, an effort must be made to determine an arbitration schedule at compile time [1], [45]. This task is better performed with a predictable application [46]. It is realistic to assume a predictable application behavior with the new communication infrastructure since most SoCs are used with digital signal processing (DSP) platforms. Predictability is a

characteristic of most DSP applications. Therefore, a VLIW arbiter appears particularly attractive with a DSP platform, such as a video converter platform, where multiple outstanding data streams are present simultaneously. The VLIW arbiter creates a channel between a source and a destination processor by forcing the communicating entities to access the specialized AHB upon the same memory phase following a pre-determined bus access schedule.

Figure 4.8 details the block diagram of the micro-programmed arbiter. Its architecture is not so different from that of a modern microprocessor. The arbiter is composed of three main components that interact closely. First, the memory of the micro-program holds 64-bit words. Each 64-bit word represents a sequence of up to 16 bus masters. Second, the arbitration controller programs the shift register with those long words acquired from the memory. Finally, the shift register circulates the long word thoroughly. Its output stage is composed of a pipelined logarithmic decoder to generate HGRANTF. HMASTERF[3:0] signals are four bits representing a unique bus master. A feedback path from the shift register to the controller is provided in order to trace the status of the access sequence on the bus. Monitoring the access sequence lets the controller execute complex schedules where multiple long words are required to run an application. Moreover, this mechanism allows loading a new application by reprogramming the arbitration sequence online. Such a feature is interesting for SoCs that are flexible enough to execute a set of different applications. To go further, the content of the micro-program memory may be altered online with a new schedule [46].

Finally, the micro-program approach has been proven extremely powerful in implementing a microprocessor's controller [1], [45]. A wide variety of schemes can be

accomplished from this type of architecture. Even though the VLIW arbiter executes a statically determined schedule, it offers great flexibility and a performance level that cannot easily be challenged by other arbitration policies. Yet again, a significant amount of work must be accomplished in order to acquire an absolute schedule. The difficulty to establish the arbitration schedule resides in the complexity, the predictability, and the efficiency of the target application.

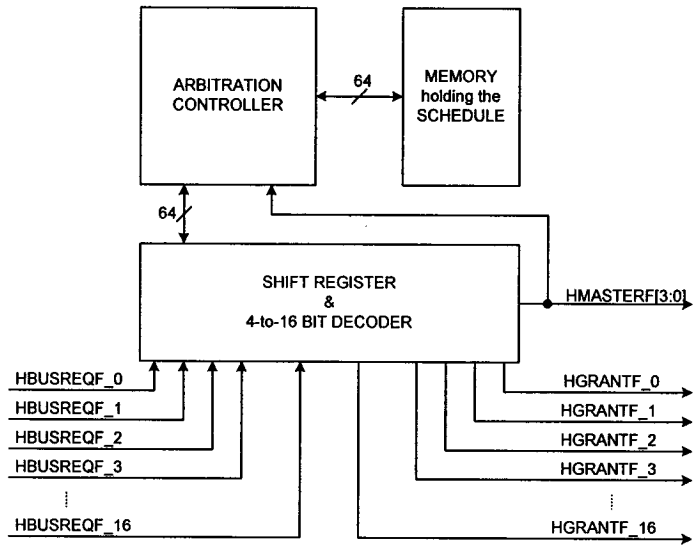


Figure 4.8. Block diagram of the VLIW micro-programmed arbiter.

4.3.1 Traffic Analysis

As previously mentioned, there are several parameters involved in achieving quality-of-service (QoS) under real-time constraints. The 2 GHz AHB fabric outperforms the processing elements (PE) by completing multiple memory accesses while the PEs are attempting a single access. To generate a proper access schedule, it is important to be aware of the time window available to complete the transfers, and the permissible time to provide the response to a PE. Since timing is particular to every PE, their scheduling priority may be much different. In addition, the application to be run

plays a significant role in bus utilization. Hence, a cautious traffic analysis is the key to avoid clogging the communication infrastructure.

Before timing parameters are further examined, some simplifying assumptions must be formulated to facilitate the arguments:

1. The PEs place their requests upon the 2 GHz communication fabric simultaneously;
2. All PEs run on the same system clock, i.e. 200 MHz;
3. Each PE is guaranteed a communication bandwidth of 125 MHz;
4. For more realistic results, interfacing with an ARM946 processor [47] is assumed.

At a later time, the above simplifying assumptions will be relaxed to give a more appropriate picture of the possibilities offered by the new high-speed communication infrastructure.

Since the delivery time is an important parameter, a study has been conducted to discover different ways of accelerating the response time of the hard IP. It turns out that it is possible to exploit aggressive, yet realistic, timing parameters to steal precious nanoseconds to the PEs. Thus, it is viable to initiate a transfer on the 2 GHz AHB before it is officially made available by the PEs.

Figure 4.9 shows how it is possible to capture the address before its time. The shaded area represents the address exclusion zone where the address appears unstable to the hard IP. The time from the rising edge of HCLK to HADDR valid appears as T_{ova}. To get an accurate estimate of what this parameter could be, ARM processors have been studied. Regardless of the family of ARM processors, the upper bound on T_{ova}

corresponds to 30% of the system clock (not to be confused with the bus clock). T_{pahb} models the propagation delay for the address to travel along the standard AHB. This parameter is hard to estimate because it varies greatly with the length of the AHB wires (dependant on placement and routing), the parasitic capacitance on the wires (dependant on the number of cores), and other electrical parameters [24], [27], [40]. However, AHB-Lite links, with a single slave, are assumed in this research, and the associated wires are expected to be short. Therefore, a T_{padr} value of approximately 0.5 ns is assumed at this time. As a result, the address can be sampled at the rising edge at the end of the sixth cycle. This allows the hard IP to begin transfers early on the READ bus to deliver a response as early as possible. Hence, in the example of Figure 4.9, the address could be issued as early as the sixth HCLKF cycle on the hard IP.

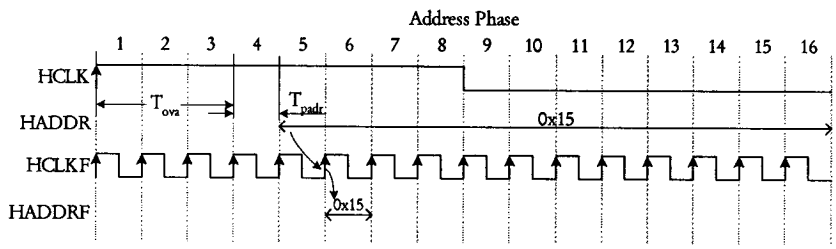


Figure 4.9. Exclusion zone to steal the address to a processing element.

Now that the timing encroachment feature is unveiled, there are other parameters associated with the data exclusion zone to consider. As a matter of fact, once the hard IP has accessed the aimed data, some parameters must be considered to deliver it safely according to the particulars of each PE. Figure 4.10 defines the exclusion zone to respect when returning data to its PE. The data is fetched from the memory on the ninth cycle. The tenth cycle is needed to transmit data to the READ Bridge while the eleventh cycle is used to latch the data. Meanwhile, READ data appear on the standard AHB. Then, T_{prd}

represents the propagation time required by a word to travel across the standard AHB. Yet again, its value is assumed to be around 0.5 ns. The most significant parameter is represented by T_{isrd} (input setup time on HDATAR), and it must last at least 30 % of an ARM processor's system clock. Hence, 1.5 ns of setup time are required to satisfy this parameter. The last compulsory timing constraint is T_{ihrd} which models the hold time on the data bus. T_{ihrd} must be greater than zero, meaning that READ data must remain valid beyond the rising HCLK edge.

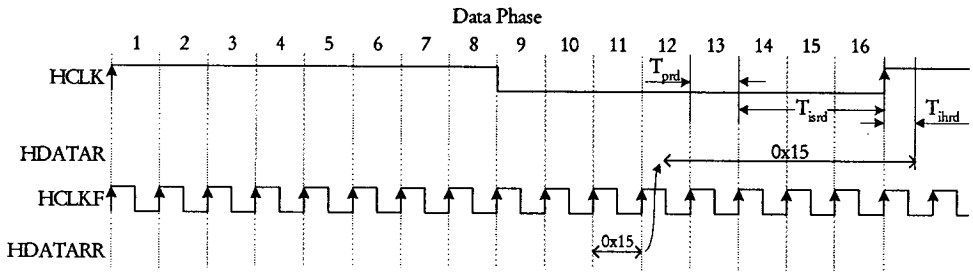


Figure 4.10. Exclusion zone to deliver READ data.

Figure 4.11 illustrates an example with a burst of 16 simultaneous READ requests. Again, the two exclusion zones for address and data words are shown at the extremities of the timing diagram. The time slot left in between the exclusion zones represents the time window available to commit all READ transfers. It can be observed that addresses are issued early on the 2 GHz AHB fabric, henceforth encroaching the time credited to each PE to transfer addresses. As explained earlier, this is acceptable if the address timing is guaranteed. If there is insufficient knowledge about the system timing, we advise against the time-stealing feature. As a result, request A is sampled early by the bridge and is immediately issued on the specialized READ AHB at the sixth bus cycle. Address A is delivered 500 ps prior to the rising clock edge of memory number 4. Though the address is delivered at the last minute, it easily meets the setup

time requirements of the SSRAM memory. The time needed by the SSRAM memories to fetch data is too long to steal time from those elements. Hence, data word A is latched at the end of the eleventh bus cycle by the memory wrappers, and READ data is issued on HRDATAR during the next bus cycle. As a result, data A could appear on HRDATA before the data phase begins. The READ Bridge has the responsibility to hold those data and deliver them with the proper data phase.

It should be observed that the last few transfers are more critical. The 2 GHz AHB infrastructure serializes all accesses. The first request to proceed has plenty of time to be executed, as shown in Figure 4.11, but the last transfer is more critical. In this example, if the exclusion zones grow larger, then it would not be possible to satisfy a complete burst of 16 READ operations with zero latency. There are two ways to cope with this case: (1) reduce the bus frequency of the PEs to extend the time window, or (2) resign to accept one (or more) cycle of latency for transfers exceeding the time window.

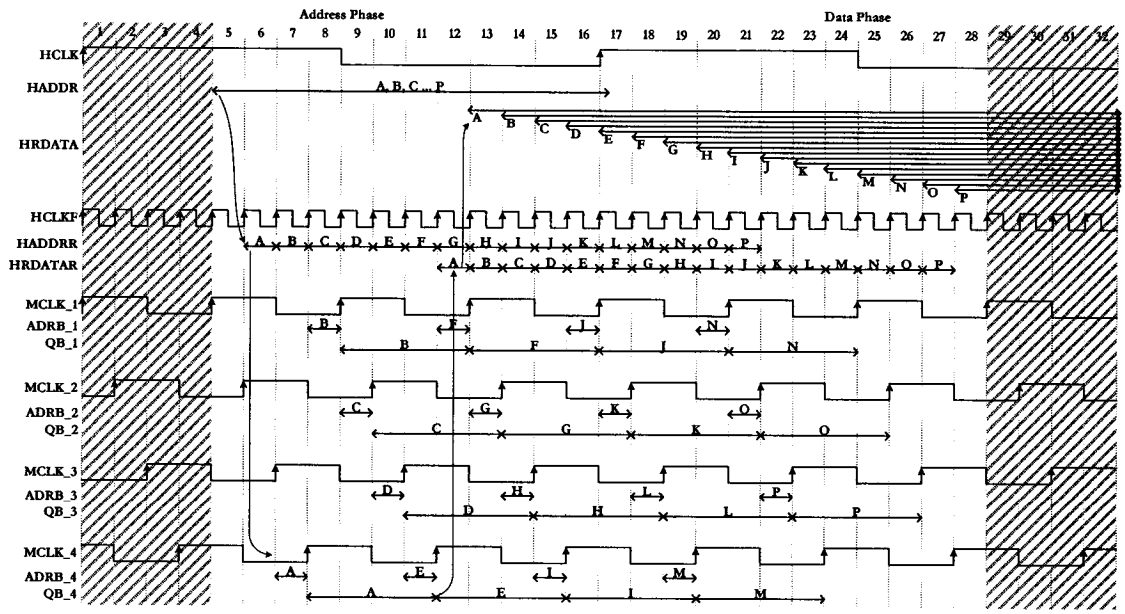


Figure 4.11. Performing 16 READ cycles with no apparent latency.

At this point, it should be mentioned that a WRITE cycle does not have to compel to the same data exclusion zone. For a WRITE cycle, the guarantee that the operation will be committed successfully is a sufficient condition for a PE to complete the transfer. Therefore, when a WRITE operation is processed, the 2 GHz communication infrastructure only needs to return an OKAY response with HREADY asserted to satisfy the processors. This feature contrasts with a READ cycle since the data must be returned with an OKAY response for a processor to complete the transfer. For this reason, READ operations are considered more critical than WRITE operations.

Figure 4.12 illustrates the difference with data exclusion zones when a burst of 16 WRITE operations is attempted. In this example, time stealing is performed on the address bus and on the data bus. However, it is not required to steal time for a WRITE operation to complete with no apparent latency. The hard IP acquires the address in the same manner as for a READ operation. The difference stems from data origin: it is issued by the processors. This contrasts with a READ operation where the data is issued by the shared memory. For this reason, the communication infrastructure must acquire the data from the processors in a safe manner, i.e., in a manner respecting the exclusion zones at the beginning of the standard AHB cycle. From Figure 4.12, the address is sampled before the data phase, but the WRITE Bridge does not issue it until the data phase begins. At the beginning of the data phase, the bridge waits for the exclusion zone to expire and it receives the data. At this point, address A is issued to the hard IP, and its associated data follows immediately on the 22^{nd} bus cycle. The memory cannot fail to write the data, unless it is damaged. Therefore, the specialized WRITE Bridge does not have to wait for the memories to return an OKAY response. The PEs are notified of a

successful transfer even if the actual WRITE operation has not yet been completed. This feature allows keeping the normal AHB pipeline depth at two levels, as required by the AMBA specifications [2].

Note that in both cases, READ and WRITE transfers, the pipelined activities of the standard AHB is not compromised. The data phase can overlap with the next address phase without clogging the 2 GHz AHBs. As a matter of fact, the next wave of transfers would simply line up with the present sequence, as with the transfers shown with dashed lines in Figure 4.12. The same principle applies for the specialized READ AHB.

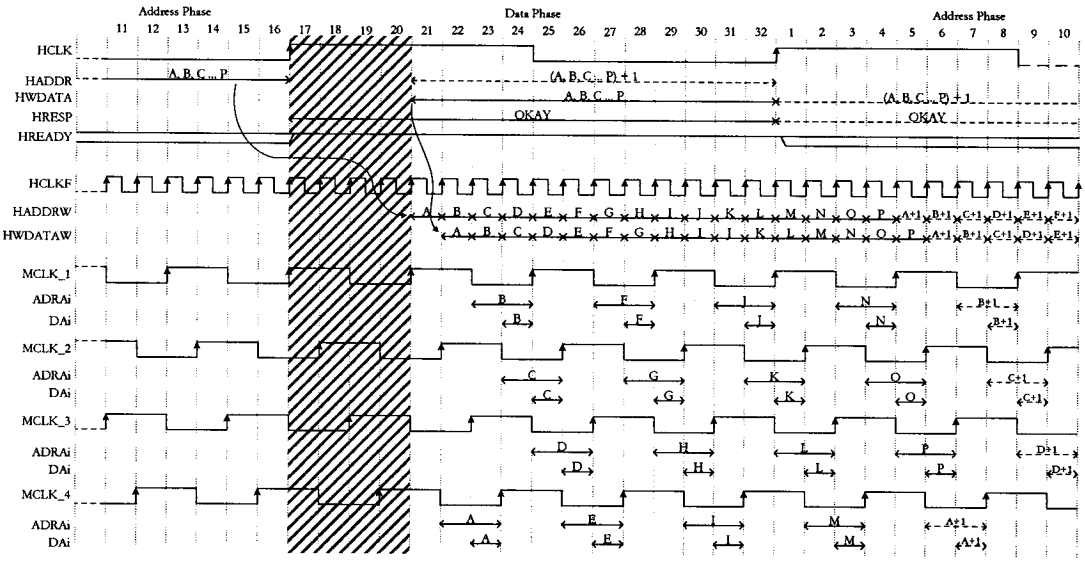


Figure 4.12. Performing 16 WRITE cycles with no apparent latency.

4.3.2 Timing Model

All parameters described above can be equated to model the upper bound on the time window available for a burst of READ transfers. In addition, the behavior of the high-speed READ bus can be modeled in a way representing the minimum amount of time required to perform a burst of transfers. The resulting relationship is expressed by the following equation and its timing parameters are defined in Table 4.3:

$$(adrR+W+rdR+N-1)\cdot t_{HCLKF} \leq (2+L)\cdot t_{HCLK} - (t_{ova}+t_{padr}+t_{isadr}+t_{ovrd}+t_{prd}+t_{isrd}) \quad (4.1).$$

The left hand side (LHS) of equation 4.1 defines an accurate model of the minimum time required to perform a burst of N transfers with 100 % bus utilization. For instance, it is possible that some masters do not require the total bandwidth dedicated to them for the entire application execution. In this case, the N parameter is adjusted to reflect a reduced total utilization. Hence, the above relationship must be revisited each time an application enters a new task requesting different resource scheduling needs.

TABLE 4.3. TIMING PARAMETERS DEFINITION.

Symbol	Parameter
adrR	Cycles required for address phase on the fast AHB.
W	Number of wait states
rdR	Cycles required for data phase on the fast AHB.
N	Maximum number of masters.
t_{HCLKF}	HCLKF cycle time.
L	Tolerable latency expressed in HCLK cycles.
t_{HCLK}	HCLK cycle time.
t_{ova}	Rising HCLK to HADDR valid.
t_{padr}	HADDR propagation time.
t_{isadr}	HADDR input setup time.
t_{ovrd}	Rising HCLKF to HRDATA valid.
t_{prd}	HRDATA propagation time.
t_{isrd}	HRDATA input setup time.

In contrast with the LHS, the right hand side (RHS) may not be unique: it is most likely dependant on individual processors attached to the hard IP and on the application. As a matter of fact, the L parameter modeling the latency is particular to each PE's architecture, role, and task evolution. Other parameters, such as t_{ova} and t_{isrd}, vary according to the implementation of the processing core. In addition t_{padr} and t_{prd} vary in

accordance to the distance of the PE (wire length) and the complexity of the standard AHB. For these reasons, the time window varies greatly from each PE's point of view. There are two ways to cope with this variation: (1) consider only the worst case scenario, and (2) adjust the arbitration schedule according to the particulars of every PE. The first solution is the easiest avenue; however, it may hinder the performance of the communication infrastructure. The second solution is more complex since it requires intimate knowledge of the system at hand, but it allows for improved performance when an first come first served schedule is compiled.

4.4 Master Bridges

To interface the new interconnect infrastructure with the SoC cores, two types of bridge are needed: one for the READ bus and the other for the WRITE bus. Even though their implementation varies in many points, the tasks accomplished by these entities are numerous. In effect, the bridges provide a standard AHB slave interface for SoC cores, while a specialized AHB bus master initiates the transactions on the high-speed bus fabric. Finally, protocol filtering is performed by the bridge since a variety of unused signals and features of AHB were screened out. Hence, the only transfer modes kept for the very high-speed AHB are IDLE and SIMPLE transactions, without any possibility of inserting wait states or signaling errors.

Leaving out burst transfer mode may look like a severe drawback, however it is not. The PEs interconnected with the high-speed bus can still perform transfers in burst mode. The internal operation of the hard IP is hidden from the processors sitting on the SoC. Therefore, PEs transfer bursts of data while the bus arbiter multiplexes simple

transfers from all PEs on the high-speed bus. This mechanism allows processing multiple concurrent bursts by the interleaved memory.

The illustration below (Figure 4.13) shows the internal organization of the READ master bridge. Its architecture is divided in a way similar to that of a typical microprocessor [1], [45]: the first logical division is the controller (shown in light grey), then the datapath, and finally the input and output interfaces. This section explores algorithms and functionalities of both bridges.

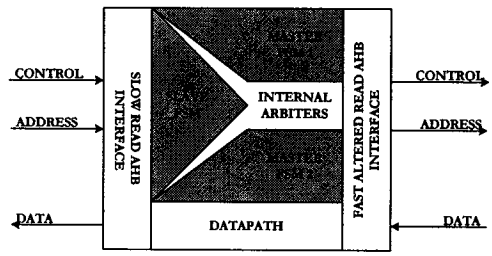


Figure 4.13. Block diagram of the READ master bridge.

4.4.1 READ Bridge

The READ Master Bridge is controlled by three finite-state machines (FSM): The first FSM operates as a slave at the speed of the slow standard AHB. The slave FSM receives a request from a PE and it launches one of the fast FSMs. The slave FSM has two states and it monitors the state of the standard AMBA AHB. Table 4.4 details the state-phase relationship of the slave FSM with the standard AHB. Note from Table 4.4 that, even though the slave FSM never inserts wait states, it remains sensitive to extended data phases. The master FSMs personify the masters of the 2 GHz specialized AHB. They are alternately triggered by the slave FSM. This is required to reflect the pipelined architecture set by the AHB standard. Thus, at the moment where a master FSM launches an address phase, the other may complete a data phase. This process keeps on

repeating endlessly. This strategy works correctly provided that some arbiters are planned to allocate shared resources.

Table 4.5 describes the improved pipeline structure. A first discrepancy from the AHB standard should be noted on the READ master FSMs in Table 4.5. As a matter of fact, states T3 to T7 inclusive form illegitimate wait states. The reason forcing this delay between address and data phases was explained in details in Section 4.2. Though this is a deviation from the AHB standard, it is not difficult to implement. With a standard AHB, data input buffers are enabled by signal HGRANT delayed by one bus cycle. For the specialized READ AHB, the same mechanism is used but data input buffers are enabled by HGRANTR delayed by six bus cycles. Adding wait states to the AHB protocol is the key to increase the AHB pipeline depth. Then again, the gain in throughput obtained greatly justifies this deviation from the AHB standard.

Figure 4.14 shows a detailed block diagram of the READ master bridge. The bridge controller is kept apart from the datapath's implementation. This divide-and-conquer methodology is often used in complex architectures such as microprocessors. Though the controller of the READ Bridge has been greatly simplified to ease its construction, the full-custom design methodology in use further complicates implementation of this core.

TABLE 4.4. TRACING THE STATE OF THE STANDARD AHB.

Standard AHB cycle	1	2	3	4	5	6	7
Slave FSM state	T0	T1	T0	T0	T1	T0	T1
Standard AHB phase	Adr 0 Rd 1	Adr 1 Rd 0	Adr 0 Rd 1	Adr 0 Rd 1	Adr 1 Rd 0	Adr 0 Rd 1	Adr 1 Rd 0
HREADY	1	1	0	1	1	1	1

TABLE 4.5. SPECIALIZED READ AHB PIPELINE STRUCTURE.

	Transfers													
Arbitration phase	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Address phase		A	B	C	D	E	F	G	H	I	J	K	L	M
Wait States (Data fetch)			A	B	C	D	E	F	G	H	I	J	K	L
				A	B	C	D	E	F	G	H	I	J	K
					A	B	C	D	E	F	G	H	I	J
						A	B	C	D	E	F	G	H	I
							A	B	C	D	E	F	G	H
Data phase								A	B	C	D	E	F	G

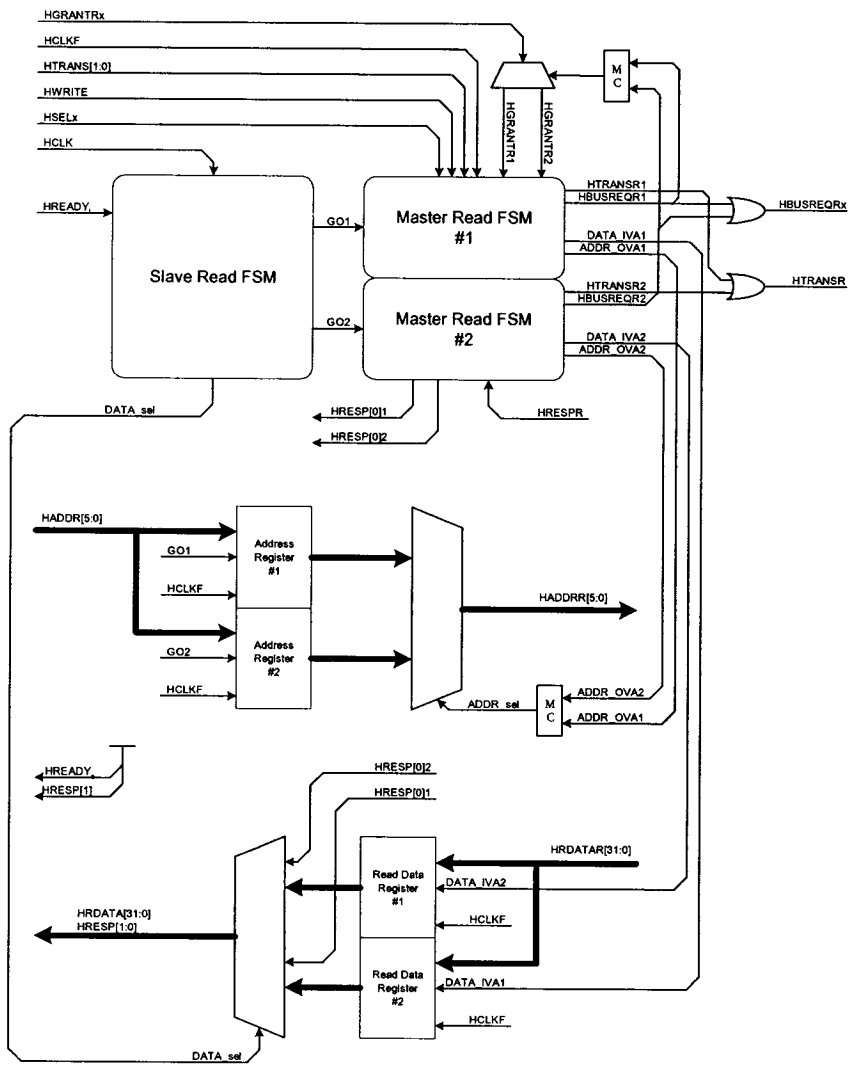


Figure 4.14. Detailed READ Master Bridge block diagram.

The datapath has one particularity resulting from the controller. In fact, it has a double input single output buffer to accommodate the two master FSMs (see Figure 4.14). This architecture is necessary to in order to enable the bridge to collect as many samples has required by the two master FSMs. The single output is cooperatively controlled by slave and master FSMs. In addition, some arbiters are provided to resolve resource sharing conflicts between both master FSMs.

4.4.2 WRITE Bridge

The WRITE Bridge is simpler in its control logic than the READ Bridge. This is mainly due to its conformity to the AHB standard. The controller's operation of the WRITE Bridge is close to the READ Bridge's controller, but no extra wait states are added. The same pulse mechanism is used to trigger one of the sister FSMs, but the pulse activates upon reception of the write data. In contrast with READ master FSMs, the WRITE master FSMs trigger only when address and data are available for the transfer. WRITE transfers are performed as specified by the AHB standard. Latches within the interleaved memory wrappers stabilize address and data for safe SSRAM sampling. The last variation with the WRITE Bridge stems from the datapath, whose data direction has been adjusted to point toward the interleaved memory. For this reason, the slave FSM does not control the output data. Then again, the WRITE Bridge always provides an OKAY response with HREADY stuck-at-one. Furthermore, this bridge remains sensitive to wait states insertion on the standard AHB.

Some glue arbiters are required to ensure stability of the controller since, under special circumstances, both master FSMs compete for the same resources. In fact, there is a possibility that one master FSM catches up on the other. As an example, such a

possibility arises when one master FSM stalls waiting for a positive HGRANTF and its sister FSM is launched. At this point, both master FSMs reach the same state and they will both proceed with a transaction when HGRANTF is asserted. For this reason, an internal arbiter is required to let only the first master FSM that asserted HBUSREQF proceed with HGRANTF. The other FSM has to wait for the next bus grant. Similarly, another possibility of conflict exists with the selection lines of the address multiplexer.

These two conflicts are removed with arbiters made of Muller-C elements [27] (their implementation will be discussed further in Chapter 5). The last problem to foresee is related to synchronization. The slave FSM is clocked at the rate of the standard AHB whereas the fast FSMs are clocked at 2 GHz. A synchronization mechanism must be provided to bridge the two clock domains. Triggering a fast FSM from a naïve association to the slave FSM's state is not possible. Consider the timing diagram in Figure 4.11. It is clearly shown that some transfers (e.g. transfer A) can complete within the address phase. A simple association with the slave state could cause a fast FSM to launch twice to perform the same transfer since the slave FSM does not toggle its state fast enough. The same situation is possible with any transfer when wait state are inserted by another slave on the standard bus. For this reason, the slave FSM launches a fast FSM through a synchronization register that generates a short pulse whose waveform is shown in Figure 4.15.

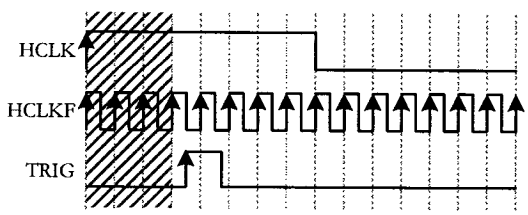


Figure 4.15. Triggering pulse waveform.

4.5 Summary

In this Chapter, a high-performance AHB-compliant communication infrastructure for high-end SoC platforms is presented. It uses a time multiplexed high-speed AHB to interconnect modules operating at variable frequencies. Means to create a high-throughput interleaved memory serving the high-speed bus are discussed. In addition, a simple, yet powerful, arbitration mechanism is detailed to provide good quality-of-service under real-time constraints. The SoC platform is reconfigurable by altering the content of the arbiter's memory. Support for multiple outstanding data streams is possible.

5 CIRCUITS IMPLEMENTATION

Designing high-speed digital systems clocked in the gigahertz range is a painstaking endeavor that requires state-of-the-art circuit techniques combined with aggressive transistor sizing. Even though architectural innovations may lead to improved clock rates, low level circuit design is the only way to achieve maximum performance. This chapter explores the collection of circuits used to implement three decisive modules that compose the 2 GHz AHB fabric: (1) the Specialized-to-Standard (StS) AHB Bridge, (2) the specialized VLIW AHB arbiter, and (3) the pipelined memory bank and its wrappers. Unlike the fastest AHB reported in the literature prior to this research [36], this implementation is achieved from a full-custom design methodology [6], [7], [8]. Utilizing the interconnect architecture presented in Chapter 4, the circuits necessary to built this communication infrastructure are defined. Feasibility of the proposed hard IP is supported by circuit simulations of its main modules.

To further support the circuit methodology characterized in this chapter, circuit simulation results of the VLIW AHB arbiter, after layout extraction, are detailed. Achieving a digital circuit clocked in the GHz range requires careful planning to keep speed attributes with parasitic capacitances that appear after layout extraction, a factor largely dominated by interconnects as the technology is scaled down [27], [28], [44]. Nevertheless, laying out a decisive component represents an accurate metric to verify the feasibility of the design style employed to achieve a 2 GHz AHB communication infrastructure.

5.1 Specialized-to-Standard AHB Bridge

Due to its complexity, the Specialized-to-Standard (StS) AHB Bridge consumes more resources then any other module used to build the interconnect infrastructure. Its circuit complexity comes, in majority, from the multiplexed datapath. Then again, the controller required to manage StS Bridge's activities is mostly irregular since it involves an innovative group of circuits, all different in their functions and sizes, molding its structure. For this reason, emphasis is given to the controller circuits since the datapath can be implemented with multiplexers and TSPC latches using more intuitive methods. In addition, due to their great similarities, only the WRITE Bridge is discussed in this section. The circuits involved with the READ Bridge can seamlessly be deduced from the following discussion.

5.1.1 Master Finite State Machine

Perhaps the most challenging component to design in this system, the Master FSM (MFSM) combines complex high-speed decisions with large fanins and fanouts. The interaction between the WRITE Slave FSM (SFSM) and both WRITE Master FSMs is described by Figure 5.1. The SFSM triggers the MFSMs in an alternate manner via the signal TRIG (represented by a dotted arrow). Hence, the WRITE SFSM acts as a master to the WRITE MFSM, which is composed of four states: IDLE (T0), ARBITRATE (T1), ADDRESS (T2), and DATA (T3). Note that the arbitration phase may be skipped if the bus master is already in possession of the specialized AHB bus. Obviously, to get the fastest finite state machine possible, a one-hot encoding method [52] must be used. Table 5.1 lists the state equations used to implement the MFSM.

Splitting the traditional D-input of TSPC latches means that we can use pairs of nodes Set' and Reset to control the TSPC latches. This technique hints at minimizing the input function complexity, since the logic input equation is decomposed into smaller physically independent circuits. With high-speed digital circuits, it is better to create a component from many small Boolean equations than from a single large one. Similarly, the input load splits in proportion to transistors geometries given that the D-equation of TSPC latches is broken into pairs of equations Set' and Reset (denoted by S'Tx and RTx in Table 5.1, respectively).

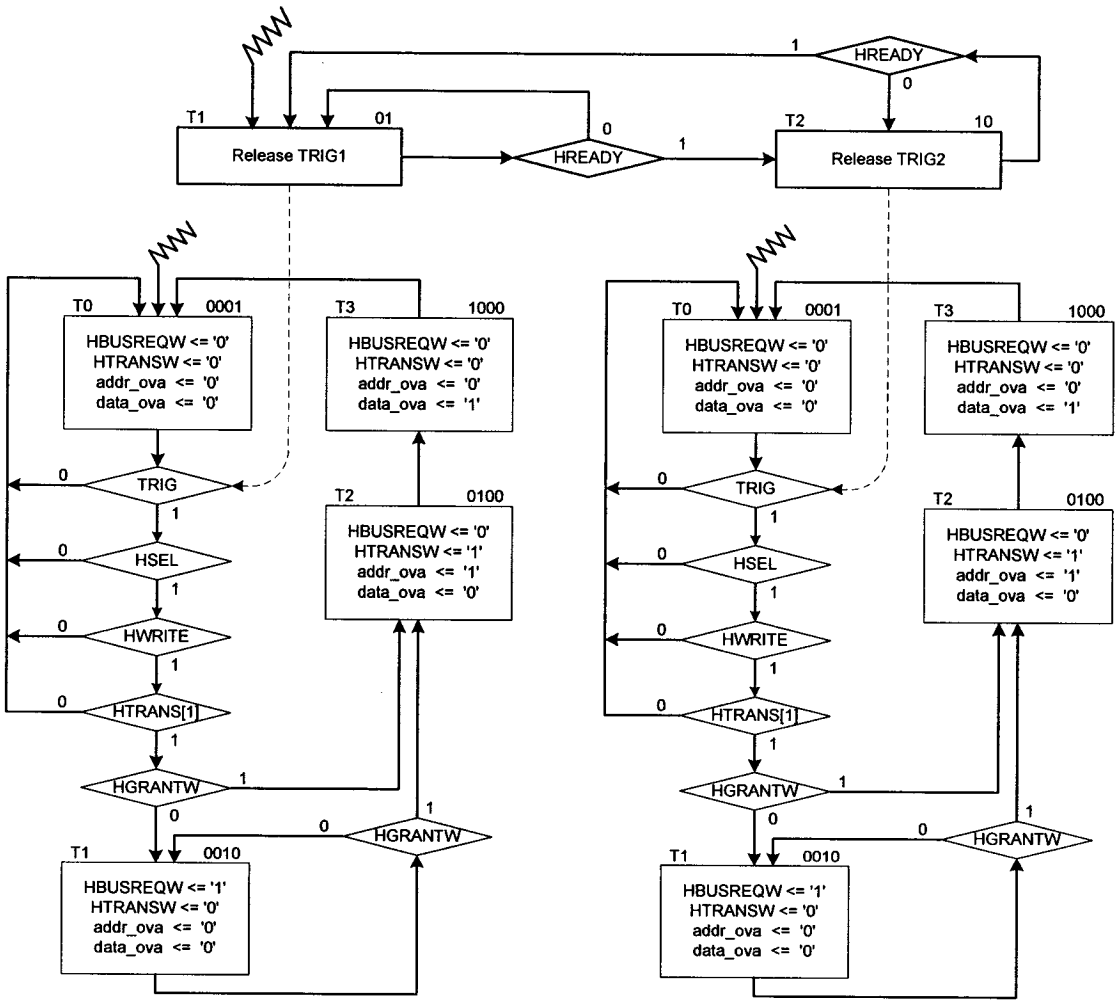


Figure 5.1. Controller's state diagram.

TABLE 5.1. STATE EQUATIONS FOR THE WRITE MASTER FSM.

State Name	State Equations	State Encoding
IDLE	RT0 = T0 · (TRIG + HSEL + HWRITE + HTRANS)	0001
	S'T0 = T3 + HRESETn'	
ARBITRATION phase	S'T1 = HGRANTF' (T0 · (TRIG + HSEL + HWRITE + HTRANS) + T1)	0010
	RT1 = T1·HGRANTF	
ADDRESS phase	S'T2 = HGRANTF (T0 · (TRIG + HSEL + HWRITE + HTRANS) + T1)	0100
	RT2 = T2	
DATA phase	S'T3 = T2	1000
	RT3 = T3	

Implementing the WRITE MFSM requires well thought-out circuit structures. This is necessary due to the large fanin experienced by the majority of the states. As a matter of fact, several circuit styles were explored to implement the logic functions: C^2MOS, pseudo-NMOS, and pass transistor logic [8], [30]. It turned out that these circuits are not well suited to large fanins since the RC delay created by long transistor chains limits the maximum clock frequency. In addition, the states of the WRITE MFSM cannot be further partitioned since the AMBA standard sets the protocol to use. To speed up critical circuits, series of transistors are systematically reorganized into faster parallel networks.

Figure 5.2 shows how the WRITE MFSM is implemented with a group of non-conventional circuits. All memory elements are designed with a positive edge-triggered C^2MOS TSPC latch [30], [33]. The traditional D-input is split into Set' and Reset nodes (shown by arrows 1 and 2 in Figure 5.2, respectively) to minimize the complexity of the input logic blocks. This is acceptable provided that Set' and Reset circuits assert in a mutually exclusive manner. In addition, a keeper [8], [30] (indicated by arrow 3 in

Figure 5.2) is placed after the input stage to guarantee internal stability of the dynamic latch in the advent that both Set' and Reset inputs stay negated for a long period of time. The feedback inverter from the keeper is small enough to ensure the speed of the latch, yet it drives enough current to overcome the leakage current [8], [30], that slowly discharges the internal node between the first and the second stage. This circuit method prevents this dynamic latch from loosing its state provided that the clock keeps running. Therefore, the MFSM can be left in any state for a long period of time without risking consistency of the system's state. This is attractive with embedded systems since most systems spend the majority of their life in a dormant state [53].

In summary, a cautious design is required to guarantee that both Set' and Reset input are not activated simultaneously. Failure to provide this guarantee may result in system malfunction since both PUN and PDN could turn on simultaneously. With such a situation, the value in which the output settles into is as random as tossing a coin. However, it may be acceptable in some circumstances to use embedded ratioed-logic that could violate the above requirement. In Figure 5.2, this is the case with HRESETn that may be asserted together with Set' of some states. Proper transistor sizing is used to ensure that the PDN wins against the PUN when this rare, but possible, system reset occurs. On the other hand, both Set' and Reset inputs may stay inactive for an infinite period of time provided that the clock pulse is uninterrupted. The weak inversion keeper maintains the charge on the internal nodes to secure the stability of the dynamic latch.

The circuits used to generate Set' and Reset inputs are similar to a C^2MOS NAND structure. They are called parallel static NAND circuits [38], and an example is shown with arrow 4 in Figure 5.2. Obviously, these methods differ in their pull-down networks

(PDN). The series of transistors created by a large fanin with a complementary CMOS approach is avoided with the new parallel structure. To achieve maximum speed, the depth of the pull-up network (PUN) is limited to one decision level plus one inversion. Note that with CMOS technology, inverter delays are still small so that extra inverters can be accepted to generate input complements. Therefore, the speed of parallel static NAND circuits is practically independent of their fanin.

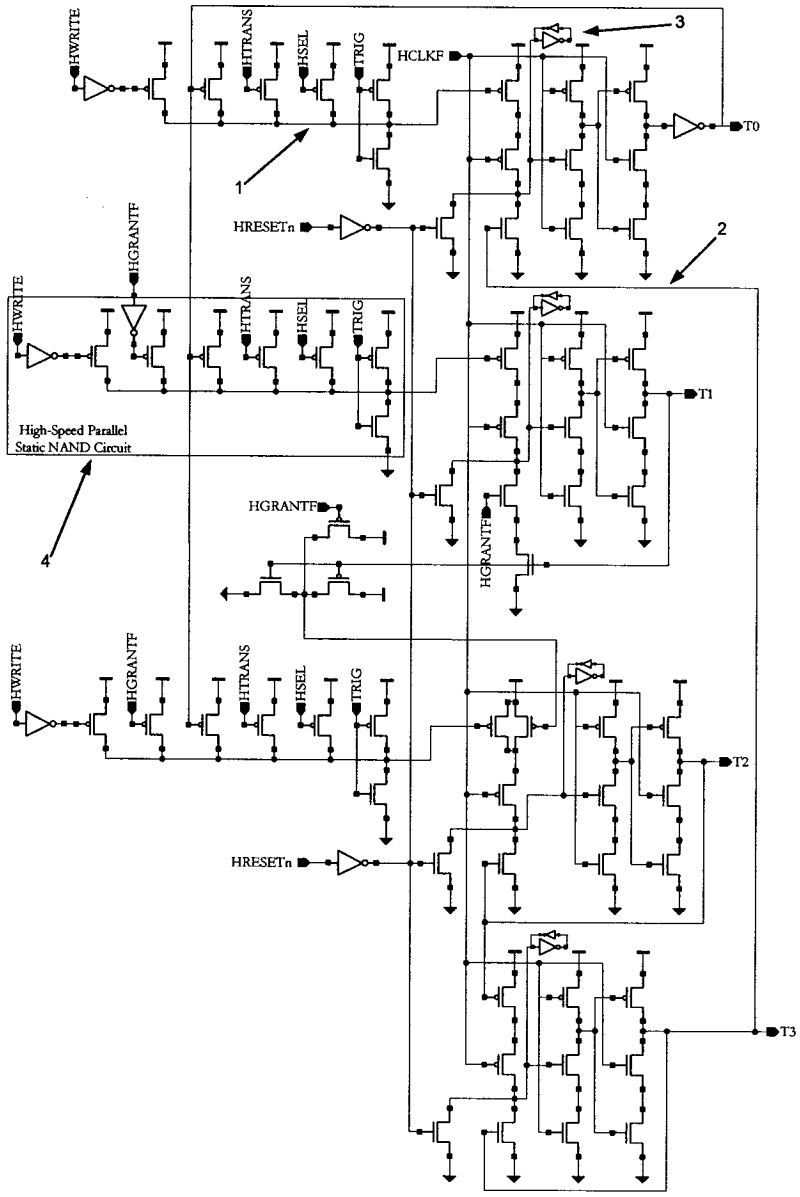


Figure 5.2. Circuits of the Master Finite State Machine.

To reduce power consumption, the last signal to be asserted is connected to the NMOS pull-down transistor. In the case of the MFSM, this pull-down transistor is connected to TRIG. As mentioned in Chapter 4, a signal TRIGx is connected to each MFSM in order to control their activation. This signal stays asserted with logic '1' for a duration of one clock cycle and then goes negated until a new request needs to be served. Hence, the parallel static NAND structure is said to be in evaluation phase when TRIG is asserted. A '0' is produced to activate Set' input if all PMOS are in cut off region. If a minimum of one condition fails, any PMOS that turns on drives a current sufficiently large to override the pull-down network and to force a '1' on the latch input. Note that the PMOS attached to TRIG is required since TRIG may drive Set' to '1' like any other signals participating in the PUN's logic. This circuit method borrows the idea of ratioed-logic in pseudo-NMOS design style [30]. Then again, the power consumption is reduced and its speed properties are guaranteed since it is forbidden to stack transistors in series in the pull-up network. Furthermore, it is applicable to multiple Boolean equations since any sum of products (or products of sum) can be reorganized in a NAND only structure from basic algebraic manipulations. Interestingly, the parallel static NAND circuit reminds of dynamic logic design, but it is different. In effect, the output is always connected to one of the power supply rails making this configuration static.

Set' and Reset inputs of state T0 are logically inversed, i.e., T0 goes to '0' when Set' is asserted and it goes to '1' when Reset is asserted, since its output goes through an additional inverter. This is done to simplify the reset logic in the advent that HRESETn goes asserted. With a one-hot encoding method, only one state gets asserted and all other states are negated. This is required to ensure stability of the MFSM. Since it is easier to

embed logic in the PDN of the C^2MOS TSPC latch than in its PUN, system reset logic was embedded in the PDN for all state. The PDN is faster since it is made of only one NMOS initially, and NMOS are faster than PMOS [6], [8], [30]. Therefore, T0's output had to be inverted in order to embed system reset logic in its PDN.

5.1.2 Slave Finite State Machine

Even though the Slave Finite State Machine (SFSM) is not as complex as the MFSMs, its importance cannot be overlooked. The role of the SFSM is to witness the state of the standard AHB attached to it, and to trigger one MFSM in the advent a new request comes in. Obviously, its clock runs at the speed of the standard AHB, i.e. 125 MHz. Here again, non-conventional circuits are used to speed up the operations of the SFSM. As explained previously, the SFSM snoops on a bus operating at 125 MHz, but its intrinsic delays must challenge those of the MFSM since some signals (e.g. TRIG) are intercommunicating. Furthermore, compiling a complex schedule with non-uniform standard bus rates may require a SFSM to operate much faster than 125 MHz. For this reason, it is designed with the same philosophy as the MFSM to get the fastest possible operation.

To follow up with the state of the standard AHB, two states are needed, they are named: (1) ADR1 (associated with state T0), and (2) ADR2 (associated with state T1). A simple system analysis allows proving that two states are sufficient. From the AMBA specifications [2], an address phase taking place on the i^{th} bus cycle is accompanied by Data phase *i-1*. Similarly, Address phase *i+1* is takes place with Data phase *i*. Associating a state with these two possibilities allows covering adequately all states of the standard AHB since the SFSM loops back infinitely. In addition, the SFSM is

designed to be sensitive to signal HREADY. When HREADY is negated (i.e. 0), the SFMS freezes in its current state until HREADY is asserted again. This avoids false positives when another slave inserts wait states on the standard AHB, or responds with a non-OKAY HRESP[1:0]. Table 5.2 shows the relationship of a sequence of transfers on the standard AHB with the state of the SFSM.

TABLE 5.2. SFSM STATE RELATIONSHIP WITH THE STANDARD AHB.

Bus cycle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
SFSM State	T0	T1	T0	T0	T1	T0	T1	T1	T1
Addr phase	ADR 0	ADR 1	ADR 2	ADR 3	ADR 4	ADR 5	ADR 6	ADR 7	ADR 8
Data phase	DAT 1	DAT 0	DAT 1	DAT 2	DAT 3	DAT 4	DAT 5	DAT 6	DAT 7
HREADY	1	1	0	1	1	1	0	0	0

The circuits of the SFSM are effectively built using techniques invented to design MFSMs. Here again, a C^2MOS positive edge-triggered TSPC latch is used with disconnected PUN and PDN, as shown in Figure 5.3. The PDN activates upon assertion of HREADY and the local latch output. For instance, when HREADY and T1 are both set to '1', the PDN pulls Vint_1 to logic '0' as to reset T1. State T1 controls the PUN of the other latch. Therefore, when T1 goes low, it enables the next state's PUN to pull Vint_1 high and to assert T0 upon the next rising clock edge. If HREADY is pulled low by another slave on the standard AHB, the PDN is instantly disabled and the SFSM stalls until HREADY resumes logic '1'. A keeper is used again for means of state refreshing in the advent that a C^2MOS latch PUN and PDN would remain off for a long period of time. A toggle latch style is sufficient to implement this small state machine, and a one-hot encoding method is used to unify the design styles and to associate one latch per MFSM.

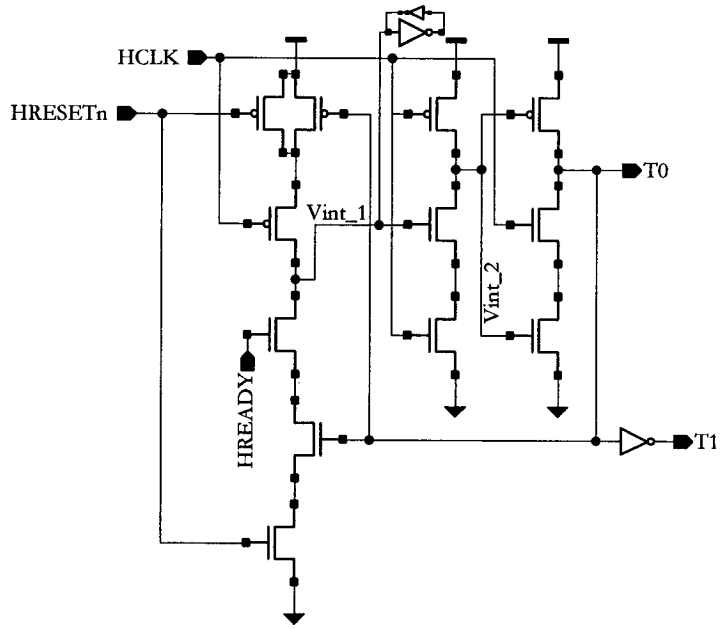


Figure 5.3. Circuits of the Slave Finite State Machine.

5.1.3 Synchronizing Register

Since AHB contains two pipeline levels, the bridge must process two transfers that are in different states. To do so, two MFSM are used in parallel for simplicity. Then again, this strategy brings a problem related to clock synchronization. As mentioned in Section 4.1.1, the different clock domains are synchronized via a PLL, whose design is beyond the scope of this thesis. Thus, the question on how to communicate between different clock domains remains unanswered.

The SFSM regulates the triggering of the MFSMs. It monitors the incoming standard AHB to detect state changes and it takes proper action toward the next MFSM to be activated. A synchronizing register is needed to signal the MFSM when to start. The synchronizing register generates a pulse, named TRIGx, for a duration of one HCLKF cycle. The time at which TRIGx is pulled to '1' depends on t_{ova} (time to output valid). The shape of signal TRIGx has previously been presented in Figure 4.15.

Coming back to the MFSM state equations (Table 5.1), signal TRIG must be asserted to enable the state machine to leave state T0. The state diagram, shown in Figure 5.1, suggests that TRIGx, represented by the dashed arrows, is generated directly by the SFSM. It turns out that this is not accurate as a synchronizing register is needed between the SFSM and the MFSM. This is necessary to model an accurate t_{ova} and to adapt to two different clock domains.

Figure 5.4 illustrates the synchronizing register used to perform this action. It follows the same philosophy as TSPC circuits previously discussed. A 2-input C^2MOS NAND gate detects the enabling conditions. A logical '1' propagates down the register upon reception of a positive transition of HCLK. The delay is modeled by the fast clock (HCLKF) according to valid address input delays exhibited by the standard AHB. The chain resets synchronously upon assertion of the signal TRIG or HRESETn (system reset). A pulse of one HCLKF cycle duration is created by the register. This is ensured by the circuit shown at the bottom right of Figure 5.4 that synchronously resets the chain of latches upon assertion of signal TRIGx. When HCLKF is '0', all Reset inputs are negated. On the other hand, when HCLKF is '1', TRIG is connected to all Reset inputs through the pass gate. At this point, if TRIG is '1', the chain of latches is reset together with TRIG. The state of the slave FSM determines which synchronizing registers will be launched since one synchronization register is associated to each SFSM states.

5.1.4 High-Speed Resource Sharing Arbiters

Operating two independent MFSM raises problems associated with resource sharing. This situation emerges since the two MFSMs snoop on the same interface signals. In addition, their independence makes them unaware of the state of their sister

FSM. For these reasons, arbiters are required to allocate shared resources within the StS Bridge on a first-come-first-served basis. Three arbiters are needed in the design of the MFSM (READ or WRITE): one to allocate HGRANTF to only one MFSM, and the others to control address and data datapaths' multiplexers.

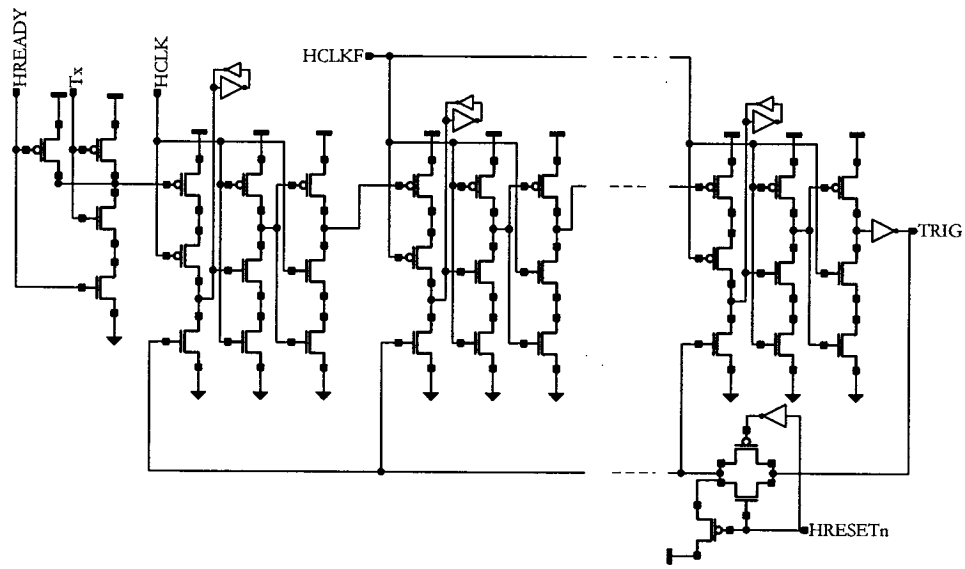


Figure 5.4. Circuits of the synchronizing register.

Asserting HBUSREQFx is a trouble-free task to perform from either MFSM. A simple OR gate is sufficient to fulfill this task. Then again, complications arise when HGRANTFx is returned by the bus arbiter. It may happen that both MFSM are awaiting this signal under the condition that they are both triggered, and the first MFSM to activate is delayed by the bus arbitration process. When these conditions are satisfied, the second MFSM activated shall catch up on the first one. As a result, both MFSM awaits HGRANTFx and a danger exist that the wrong MFSM proceed with the transfer, or even worst, they could both process their transfer, henceforth contending the 2 GHz WRITE AHB. Actually, the contention problem could be solved in a straightforward manner by fixing MFSM priority in advance. Yet again, the AHB standard requires in-order

completion, i.e., two transfers cannot be swapped on the bus since they would not complete in the order they are issued.

To keep the speed attributes of the 2 GHz AHB fabric, a simple yet fast arbitration mechanism is created to demultiplex HGRANTFx toward the proper MFSM. The arbiter is made of two Muller-C elements [30] controlling a 1-to-2 demultiplexer, as illustrated by Figure 5.5. Since the demultiplexer stands on the path of HGRANTFx, it is preferable to set the channel as soon as possible. For this reason, two Muller-C elements are used in parallel to speedup the creation of the select signal and its complement. This strategy allows saving one inverter delay. In addition, the demultiplexer is constructed from a structure reminiscent of a differential cascade voltage switch logic (DCVSL) gate [8], [30]. Yet the PDN do not always act in a complementary manner as since they can both be turned off concurrently. For this reason, both PMOS must remain open as with ratioed logic gate. This ensures the negation of HGRANTF_A and HGRANTF_B when both PDNs are turned off. Table 5.3 defines the truth table of this arbiter. Finally, two pull down transistors are used to initialize the arbiter. At power up, the state in which the arbiter will settle into is uncertain. For this reason, it is imperative to force a stable value at the output of the Muller-C elements.

The last type of arbiter is required to solve conflicts in the datapath. The address bus in Figure 4.14 illustrates a good prelude for the conflict to be solved. When both MFSM stall at state T1, they compete to control the address datapath. To solve this conflict, the pair of Muller-C elements discussed previously is used to control the group of 2-to-1 multiplexers on a first-come-first-served basis. The same approach is used to control the data output multiplexers.

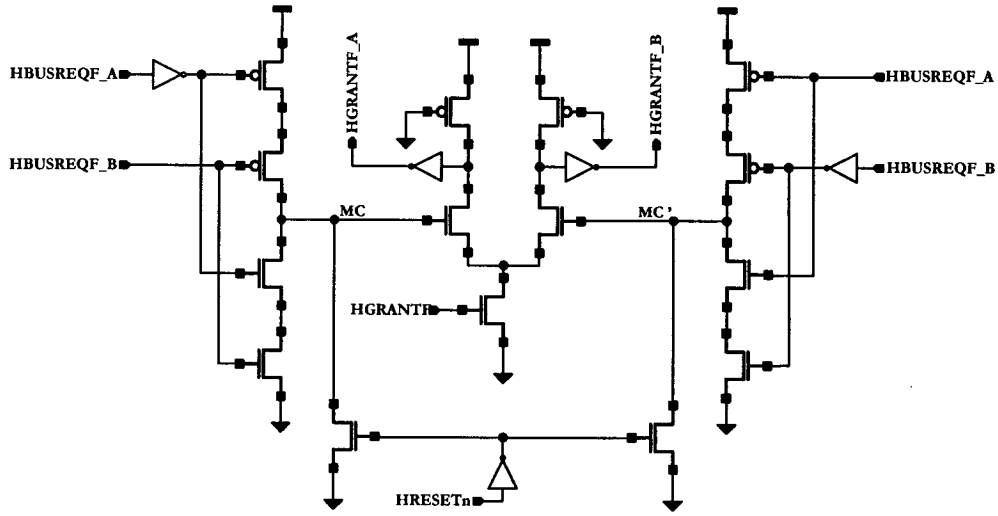


Figure 5.5. Circuits of HGRANTF arbiter.

TABLE 5.3. TRUTH TABLE OF HGRANTFX ARBITER.

A	B	MC	HGRANTF	HGRANTF_A	HGRANTF_B
0	0	No change[a]	0	0	0
0	1	0	0	0	0
1	0	1	0	0	0
1	1	No change[a]	0	0	0
0	0	No change[a]	1	No change[a]	No change[a]
0	1	0	1	0	1
1	0	1	1	1	0
1	1	No change[a]	1	No change[a]	No change[a]

a. No change means that the output kept its previous value, it could be '0' or '1'.

5.1.5 Address and Data Datapath

The datapath is implemented with a group of 2-to-1 multiplexers and latches. The positive edge-triggered latches used are of type split-output TSPC. Even though spilt-output latches appear slower than C^2MOS latches, they were deemed advantageous for the datapath design. In particular, their speed satisfies the requirements of the 2 GHz AHB infrastructure and they lower the clock load by a factor of two as opposed to

C^2MOS TSPC latches. Considering the quasi absence of logic along addresses and data paths, the slight speed reduction is not significant since they do not constitute critical paths.

Figure 5.6 illustrates a circuit slice of the address datapath. Its input multiplexer is used to emulate a pseudo-regenerative mechanism for the split-output latch. When TRIG1 is asserted, it selects the input address for the next sample. On the other hand, when TRIG1 is negated, the output of the split-output latch is fed back to its input. This enabling mechanism allows refreshing the dynamic latch when it is disabled. Subsequently, output multiplexers enable the set of latches associated with the controlling MFSM to propagate on the 2 GHz AHB infrastructure. This output stage is controlled by the Muller-C arbiter discussed in Section 5.1.4. Note that the slice shown in Figure 5.6 contains an odd number of inversions. This is required since another multiplexer placed on the specialized bus restore the number of inversion to an even number. Hence, the true data value is sampled by the shared-memory module.

5.1.6 Implementation Results

The high-performance StS Bridge was implemented using Cadence Virtuoso^{tm} at the transistor level. Its functionality was verified through detailed circuit-level simulation using the Verilog test benches and the SpectreSVerilog^{tm} simulator, as explained in Chapter 3. In addition, circuit techniques and design rules stretched in Chapter 3 were followed to improve immunity of the StS Bridge to speed reduction caused by parasitic elements unveiled by layout extraction. Unfortunately, a bug with the simulation tools prevented a combined simulation of the bridge's controller and datapath. For this reason, simulation results are presented separately for these two entities.

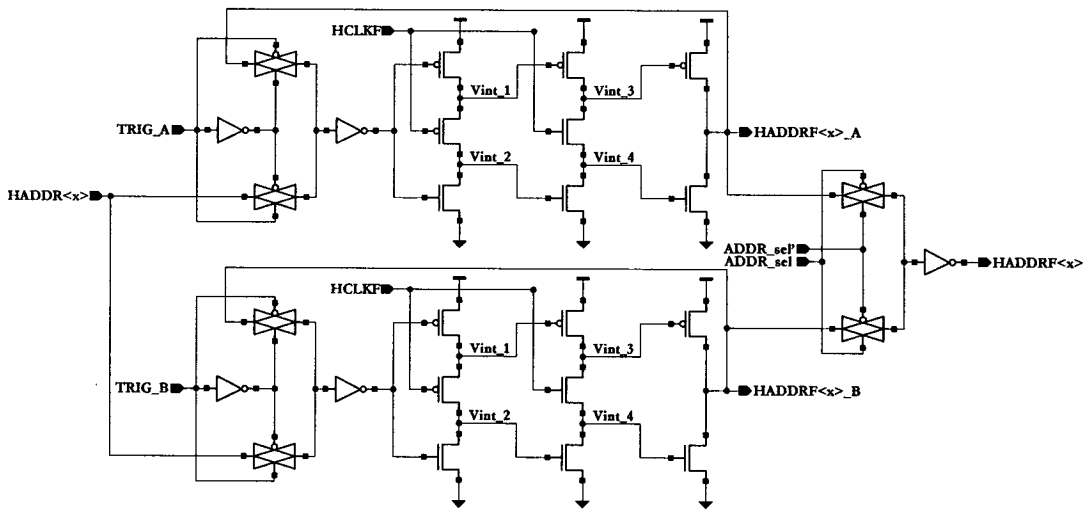
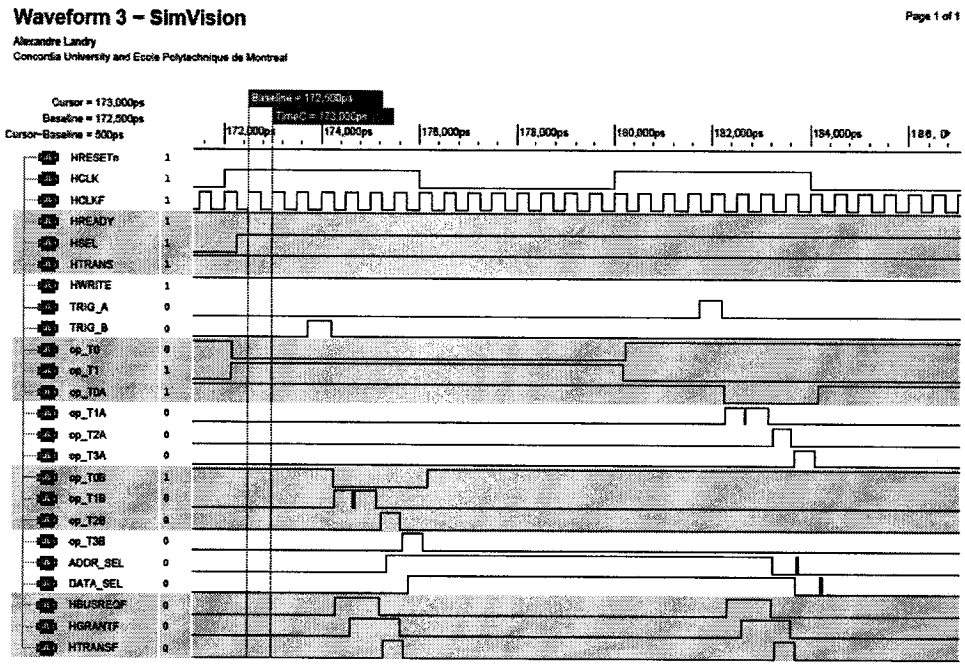
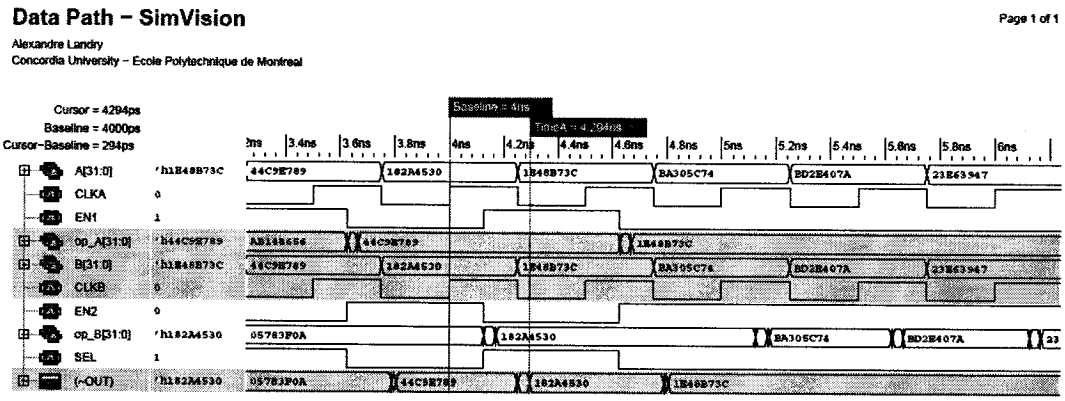


Figure 5.6. Circuit slice of the address datapath.

Figure 5.7a illustrates simulation results for the StS WRITE Bridge clocked at 2 GHz. In this simulation, all standard AHBs operate at 125 MHz. In a first operation, a slave FSM triggers the master FSM B (via TRIG_B) which initiates a bus arbitration process for a WRITE operation to the high-performance share-memory. Upon reception of the grant signal, an address phase follows then the MFSM enters the data phase to complete the transfer. Note that MFSM B is triggered only when address and data are available to it. On the next standard AHB cycle, MFSM A is triggered and it performs a WRITE cycle. Some glitches are visible on some signals. This is acceptable with synchronous sequential circuits if glitches occur in a dead time between two positive clock edges. Figure 5.7b shows simulation results of the datapath. For example, when EN1 goes asserted, A[31:0] is sampled on the rising edge of CLKA. The data appears on OUT~ (OUT~ is the complemented value of OUT to improve clarity) when SEL goes to '0'. And finally, Table 5.4 summarizes some circuit and simulation statistics of the WRITE StS Bridge.



(a)



(b)

Figure 5.7. StS WRITE Bridge (a) controller and (b) datapath output waveform.

TABLE 5.4. STS WRITE BRIDGE CIRCUITS AND SIMULATION SUMMARY.

Speed (GHz)	Power supply (V)	Max. transistion time[a] (ps)	No. of active devices	Area (μm^2)	Temperature (°C)
2.2 GHz	1.8	< 100	1933	975.8	27

a. Time measured between the 10% and 90% marks of full voltage swing..

5.2 Very-Long-Instruction-Word AHB Arbiter

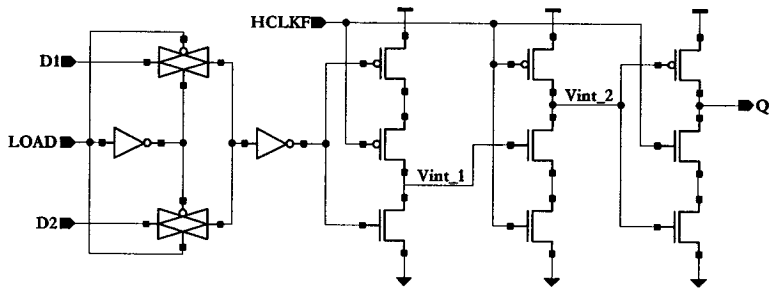
The very-long-instruction-word (VLIW) AHB arbiter used by the 2 GHz communication infrastructure is structured as a micro-programmed processor [1], [37], [49]. Its design is split into three logical entities: (1) the controller which is responsible for the access schedule, (2) the shift register that cycles a sequence of sixteen HMASTERF<3:0> to access the AHB, and (3) a modified logarithmic Barrel shifter organized as a 4-to-16 bit decoder. For the time being, the controller's function is performed by an external dedicated processor, so only the shift register and the decoder were designed to arbitrate the access sequence on the 2 GHz bus.

5.2.1 Shift Register

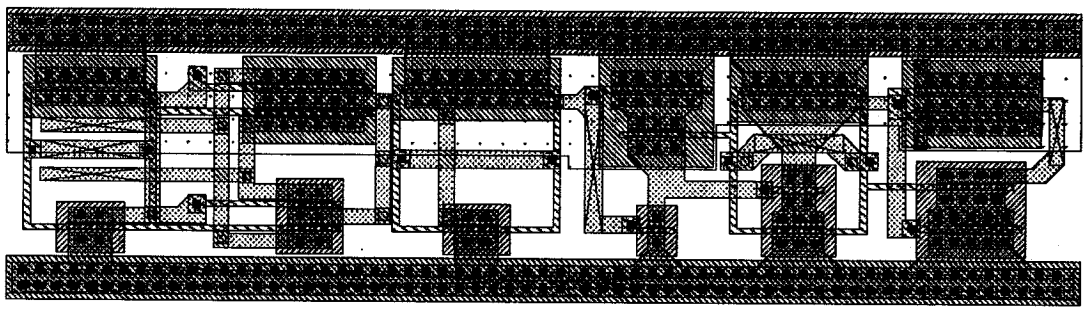
The shift register is designed to hold a complete sequence. This implies providing enough stages to hold a sequence of sixteen bus grants (represented by HMASTERF<3:0>), which corresponds to the maximum number of bus masters operating onto an AHB. Even though the arbitration sequence is determined at compile time with a VLIW approach, it is not possible to use a simple round-robin shift register that circulates one bit at a time around the sixteen bus master, since the target application may require reconfiguring the sequence of accesses as it progresses [37]. As a result, a 4-bit wide shift register is designed to hold encoded binary numbers representing a unique bus master and a load capability is provided to ensure programmability of the register.

Figure 5.8a illustrates the transistor organization of one register cell. It is shaped by one C^2MOS positive edge-triggered TSPC latch, one 2-to-1 multiplexer designed from pass logic, and one inverter to obtain an even number of inversion at the output of the C^2MOS latch. All transistors in series have been sized in a progressive manner to

compensate for added internal nodes' capacitances, as shown in Figure 5.8b. This precaution is needed the most with a large fanin, yet again this design targets clock frequencies in the gigahertz range, so progressive transistor sizing helps reaching that target even with small fanins [6], [30]. The cell illustrated hereinafter forms the basic building block of most circuitry of the AHB arbiter, henceforth offering a regular structure to ease manual placement and routing in the physical design phase.



(a)



(b)

Figure 5.8. C²MOS latch with 2-to-1 multiplexer cell. (a) Schematic, (b) Layout.

5.2.2 Decoder

Decoding four bits into sixteen to generate HGRANTFx signals is a task that could not be accomplished in a straightforward manner within the allowed time budget. To avoid instability on the AHB bus, it is important that only one HGRANTF goes asserted at a time. Therefore, pipelining is considered to break down the logic along the

critical path. Since the decoder is made of a shift register, it is possible to start the decoding process early with respect to the output to take advantage of pipelining.

As a result, a pipelined logarithmic Barrel shifter is envisioned to perform decoding [54]. This shifter is constructed from four 2-to-1 multiplexer levels. The total shift value is decomposed into powers of two to provide up to sixteen positions with four levels. Hence, the i^{th} stage either shifts over 2^i or passes the data unchanged, depending on the value of HMASTERF<i>. For instance, decoding binary number "1010" is equivalent to shift over five bits, the first stage is set to shift mode, the second to pass mode, the third to shift mode, and the final stage to pass mode.

The structure of the decoder is shown in Figure 5.9. It should be noted that the structure of the logarithmic shifter is rather unconventional. In effect, a simplification is made to minimize the circuit complexity by truncating the stages where only zeros are to be inserted, so as to create a triangular structure. As a matter of fact, only one '1' propagates through each decoder stage at a time. This implies that a limited number of '1' is inserted at the first stage. Examining the behavior of the logarithmic decoder reveals that the i^{th} stage only needs to be 2^i bits wide. The balance, i.e. $M-2^i$, is condemned to pass or to shift only '0' values, where M represents the width of the shifter in bits. Hence, with proper wiring, those stuck-at-zero cells can be removed. Again, pass logic is used to construct the fastest possible multiplexer. A C^2MOS register decouples adjacent levels to minimize the number of transistor interconnected in series. With this approach, the basic cell developed for the shift register could be reused to implement the logarithmic shifter, here again with a regular shape.

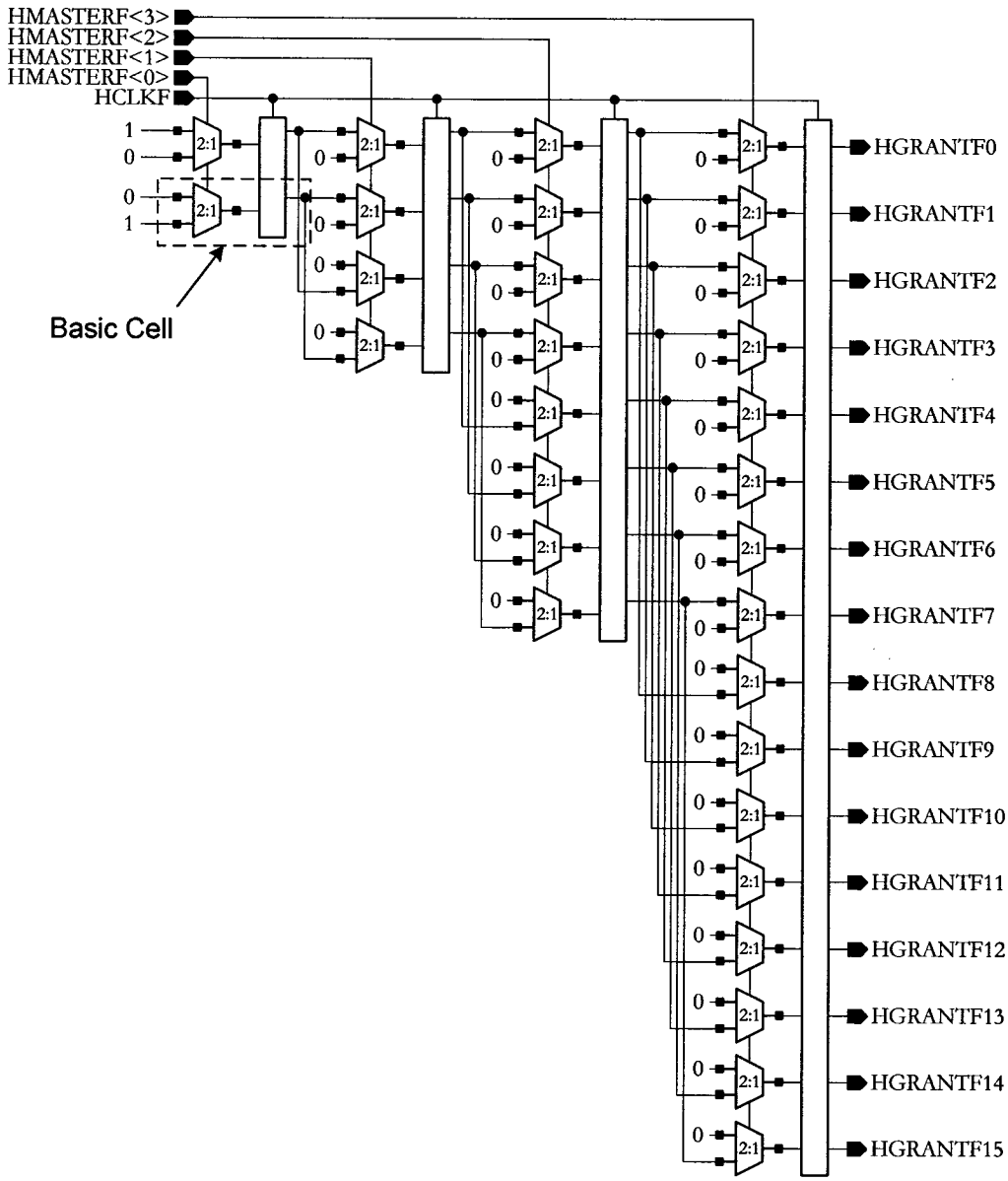


Figure 5.9. Structure of the 4-to-16 bit logarithmic decoder.

5.2.3 Layout and Implementation Results

Full-custom design is most often used to attain highest performance or smallest size. In contrast to semi-custom design, full-custom design requires handcrafted transistor layouts with manual placement and routing [6], [7], [8], [30]. Obviously, an irregular full-custom structure of VLSI complexity rapidly becomes a painstaking

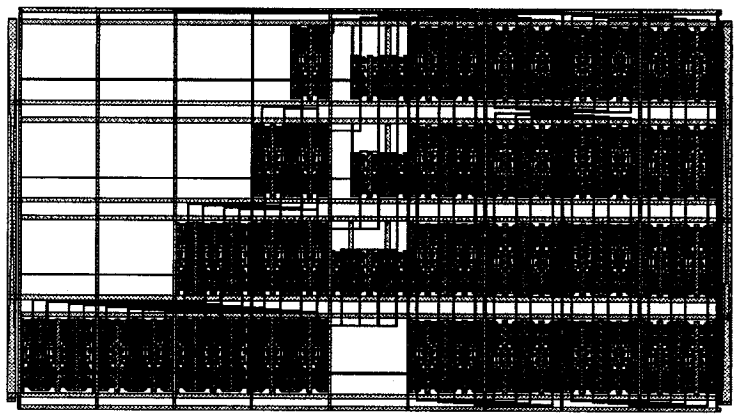
endeavor to which many draftspersons are not disposed to commit. To validate the circuit techniques used in the implementation of the 2 GHz AHB infrastructure, a layout implementation is most needed. Amongst all modules shaping the communication infrastructure, the AHB arbiter subsystem is the most regular, yet its complexity is high enough to give significant results of much credibility. Therefore, the high-speed VLIW AHB arbiter was implemented and laid out using Virtuoso^{tm} layout editor from the Cadence toolbox [55]. Its functionalities were verified using exhaustive test benches and the SpectreS^{tm} simulator. All circuits are laid out using 0.18 μm TSMC CMOS process. The main techniques, i.e. the true-single-phase-clock technique and the effective arbiter structure, have been proven experimentally. All parasitic resistors and capacitors were extracted from the layout to get the most accurate results prior to a chip fabrication.

The resulting overall structure of the high-speed arbiter is shown in Figure 5.10a. It comprises 1708 transistors and the rectangle area is 5095.68 μm^2. The architecture is bit sliced to ease physical design. Scanning the layout from left to right, the decoder is the triangular structure on the left, signal buffers are inserted in the center, and finally the shift register is the rightmost array. A subset of the simulation results is illustrated in Figure 5.10b. Decoded signals HGRANTF6, 7, 14, and 15 are shown to exhibit the performance of the ultra-fast arbiter. In addition, HMASTER<3:0> is exploded in this figure to show the matching between decoder results and the value of HMASTER<3:0>. The maximum frequency was explored and physical level simulations established that the arbiter operates at up to 3 GHz at 27 °C. This undoubtedly meets the initial speed target of 2 GHz. The maximum frequency is reduced to 2.77 GHz when the worst case temperature, i.e. 125 °C, is simulated. This speed reduction is normal since material's

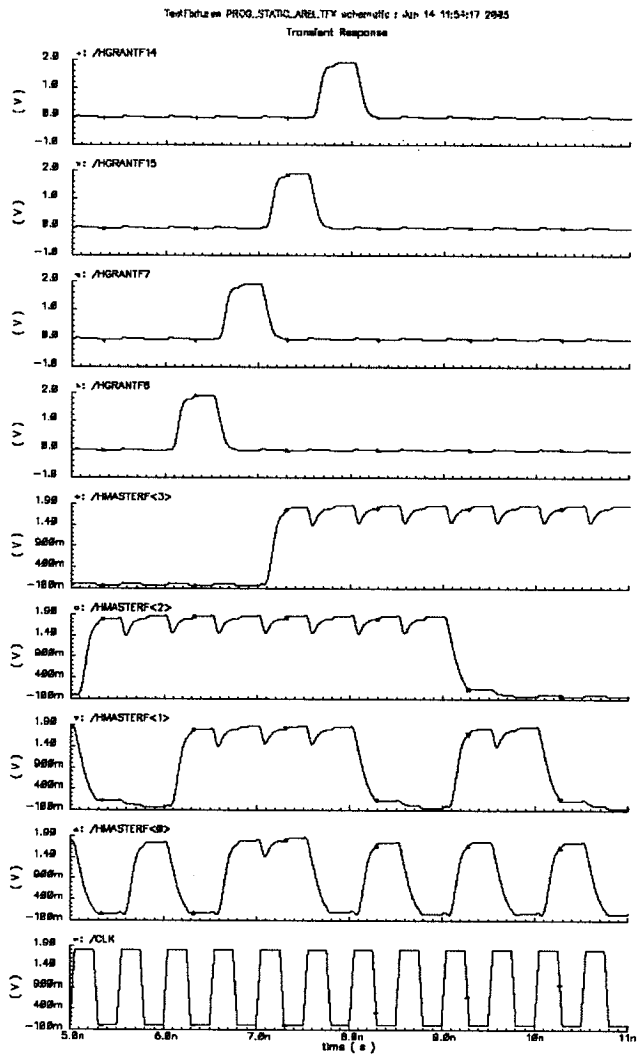
conductivity degrades as the temperatures scales up [6], [27]. Yet again, the initial speed objective is met even at worst temperature.

5.3 High-throughput Memory and its Wrappers

Designing an embedded memory is a sustained and thorough process, especially when the target clock frequency is very high. The learning curve involved with full-custom memory design is unacceptably high. Therefore, a semi-custom design is considered for this component. The Canadian Microelectronics Corporation (CMC) distributes a memory compiler named Custom Touch Memory Compiler^{tm} (CTMC is manufactured by Virage) [48] to its member Canadian universities. This memory compiler is a powerful tool that can substantially reduce design efforts. From a simple graphical user interface (GUI), it is possible to synthesize a double port SSRAM memory that can sustain operating at up to 500 MHz. In contrast to Virage, the Synopsys tools cannot compile a memory as effectively. In effect, the Synopsys tools fail to recognize memory structures. This implies that it builds it from discrete logic gates with all the drawbacks it involves: numerous gates to form a memory cell, a sparse layout created by automatic placement and routing tools, and the memory resulting from the synthesis process is slow due to those cumulative drawbacks. Obviously, Virage's memory compiler tackles memory structures with a standard library optimized to build high-speed or compact memories.



(a)



(b)

Figure 5.10. (a) Layout of the 4-to-16 bit decoder, (b) Simulation results.

Hence, CTMC was used to synthesize a 1 Kb dual port SSRAM memory that works up to 500 MHz with minimal efforts. Yet again, four SSRAMs are needed to build the interleaved memory, hereby calling for a wrapper interface to decouple the clock domains and to unite the four memories into a single entity. In addition, a comparator is needed to compare the WRITE address with the READ address applied to a SSRAM so as to avoid conflicts, as explained in Section 4.2. When a conflict is detected, the comparator raises a flag and the data is taken from an alternate source: the write-through data port of the memory.

5.3.1 Memory Wrappers

The memory wrappers form the unique slave on each specialized AHB. Even though the shared memory is constructed from four SSRAMs operated in an interleaved manner, it is considered as one logical AHB slave. For this reason, the memory is not accessed with the address as it is common with typical shared buses. It is rather accessed on a time basis, i.e. on a specific 500 MHz clock phase. Each wrapper on a specialized AHB is connected to a different clock net. This allows triggering one wrapper at a time to sample the address and data on the specialized AHB.

On the WRITE AHB, the address and the data word are both sampled by the wrappers. Their role is to stabilize address and data long enough for the memory to safely sample those signals. This is required to bridge the 2 GHz clock domain with the 500 MHz one. In effect, the specialized WRITE AHB is so fast compared to an isolated SSRAM that, without wrappers, the address phase terminates before the SSRAM samples it. Hence, address and data wrappers on the WRITE AHB extend address and data

phases according to specific requirements set by the SSRAM. Similarly, READ wrappers sample the address in the same manner, but obviously not the data bus.

The wrappers are implemented with fast logic in a straightforward manner. The basic cell presented in Figure 5.8a is reused to implement the latching wrappers. The input multiplexer is used as an enabling circuit that feeds back the latch's output Q to its D1 input when it is not enabled. Signal HTRANSW conveys the presence of a valid transfer when it is set to '1'. Hence, connecting HTRANSW to LOAD input in Figure 5.8a creates an enable signal that activates the basic cell when valid address and/or data words need to be sampled on input D2. Obviously, this mechanism works properly only if the latch is clocked at the speed of the specialized bus, i.e. at the rate set by HCLKF.

5.3.2 Address Comparator

The address comparator is an important component complementing the wrappers. The SSRAM has no internal mechanism to detect address contention. The address comparator fulfils that task as an external component that works jointly with READ and WRITE wrappers. Since both specialized AHBs operate on the same clock domain, their addresses are sampled at the same time by the memory wrappers. Yet again, a danger exists that both addresses are identical. The address comparator is used in association with the SSRAM's write-through mechanism to effectively resolve address contention issues (see Section 4.2 for further details).

Figure 5.11 shows the implementation of the 5-bit wide comparator. It is constructed using typical pseudo-NMOS logic style. The PDN consists of five XNOR PDNs embedded in one gate. This is acceptable since the number of transistors in series in the PDN is kept small, i.e. it does not exceed two. In addition, all transistors' size is

scaled up by a factor to compensate for a higher parasitic capacitance at the output. Hence, the speed of this 5-bit XNOR comparator is virtually independent from its fanin. If at least one bit is different between both addresses, the PDN is enabled and MATCH goes to '0'. Similarly, if the bit pattern of both addresses is identical, the PDN is turned off and MATCH is drive to '1' by the PMOS. The outcome of the comparison is saved by a C^2MOS TSPC latch to be used when data are being issued by the SSRAM. Actually, a 2-to-1 multiplexer selects which data source from the SSRAM is to be returned to the StS READ Bridge from the address comparison upshot. This explains why the comparator's result needs to be secured until the data phase on the specialized READ AHB goes on.

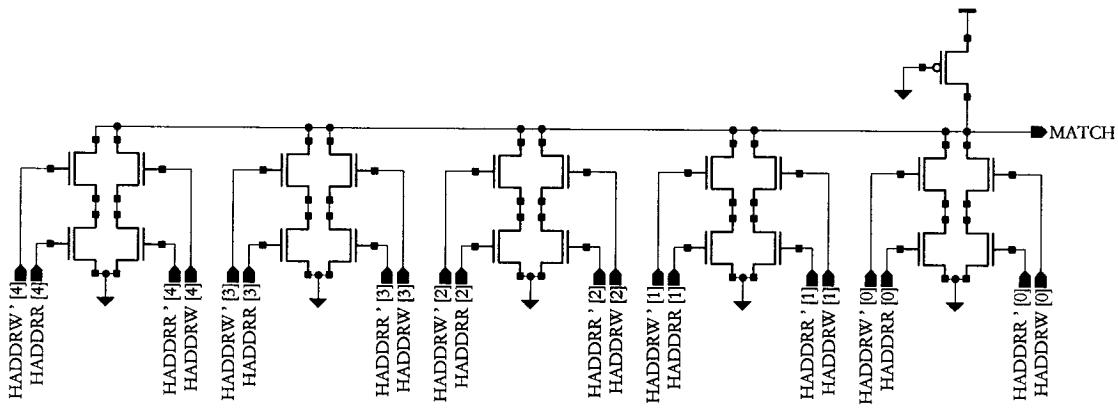


Figure 5.11. 5-bit XNOR comparator transistor organization.

5.4 Summary

This chapter focuses on circuit details used to implement the 2 GHz AHB infrastructure. At the beginning, unconventional circuits used to create the StS Bridge are described. Fast, TSPC-based logic can be derived from the circuit methodology utilized to shape the master finite state machines. As a matter of fact, a novel circuit

design style, based on TSPC, is effectively used to implement StS Bridge. A parallel static NAND gate has been invented to speedup decisions within the 2 GHz AHB fabric. In addition, unconventional C^2MOS positive-edge triggered latches are created to be used as S'R flip-flops. Their design ensures the stability of their output as long as the clock is active.

In addition, a 2 GHz VLIW AHB arbiter was laid out using a 0.18 μm CMOS process from TSMC to validate and prove the value of the techniques used to implement the high-speed communication infrastructure. Most notable, post layout simulations, performed with all parasitic capacitances extracted, demonstrated that the AHB arbiter operates at up to 3 GHz at room temperature (i.e. at 27 °C).

Finally, memory wrappers are presented with the implementation of an address comparator capable of solving address contention issues. As a result, address contention is dealt with by the wrappers with no extra latency injected in the system.

It is important to mention at this point that complete system validation, i.e. simulations after system integration, could not be performed due to signal inaccuracies created within the simulation process. This clearly results from a simulator limitation or a simulator bug.

6 CONCLUSION AND FUTURE WORK

This chapter briefly outlines possible extensions to this dissertation that can lead to future research avenues. These extensions include: (1) using MCML to implement the circuits of the high-speed AHB fabric, (2) establishing a bypass mechanism to speedup high-throughput memory, (3) implementing improved split-transaction features, (4) inserting register slices, and (5) making the hard IP asynchronous with respect to its environment. In addition, this chapter contains the conclusion and summary of this dissertation.

6.1 Future Work

1. ***System Validation:*** Despite significant efforts to validate the communication infrastructure through extensive simulations, this task remains to be done. A limitation with the simulator created a discrepancy between applied test vectors and what was sensed by the device under test. It is expected that AMS Environment provides all means to fulfill this task seamlessly, but the current design is incompatible with this flow. The importance of the system validation cannot be overstretched and it is subject to ongoing efforts.
2. ***MCML Circuits:*** As stated in Chapter 2, MCML circuits have the potential to perform faster than TSPC circuits since the voltage swing is greatly reduced with MCML style. It is expected that further speed improvements are possible with this aggressive circuit style. Furthermore, a lower logic swing

may reduce the power dissipated by the hard IP. MCML gates produce complemented outputs. This opens a door to generate differential signals to transmit data across a longer distance with improved noise immunity.

3. ***Bypass Mechanism:*** The bypass mechanism suggested is different from the one described in this thesis, which is required to resolve address contention on the dual-port memory. A high-speed register bank, capable to challenge the frequency of the high-speed AHB communication infrastructure could be used to store one word with rapid access time. If the READ operation is performed before a second WRITE operation, the wait states could be bypassed to provide data with reduced latency. Obviously, this is a complex improvement since a tag field is needed for a controller to determine where to fetch the data from. This improvement is based on the principle of cache memory in modern microprocessors [21].
4. ***Improved Split-Transaction Feature***: This extension is tightly coupled with extension 2. As a matter of fact, extension 2 implies that two response times are possible. To achieve that, a mechanism must be provided so as to notify the master how long it will take before the transfer completes, and the VLIW AHB arbiter needs to account delayed responses since it directly impacts bus traffic.
5. ***Slicing Wires with Registers:*** Insertion of register slices is a method borrowed to AMBA AXI. The impact of this strategy is to speed up the maximum clock frequency over long distances since a long wire is broken in

multiple segments with shorter delays. Hence, insertion of register slices correspond to an extension of the pipeline introduced in this research.

6. ***Asynchronous Interfaces:*** One of the main challenges with today's CMOS chips is to distribute the clock over a large circuit area while maintaining a clock skew within tolerable bounds. Keeping in mind that the hard IP should facilitate system design by lowering time-to-market, a flexible interface would be most welcome. Hence, a standard-to-specialized AHB bridge that totally decouples the clock by being insensitive to the standard AHB clock would ease the integration of SoC modules. This principle is called globally-asynchronous-locally-synchronous (GALS) [56] and it constitutes one of the main obstacles to the conception of large SoCs.

6.2 Conclusion and Summary

The design of communication infrastructure's architecture is a critical issue with systems-on-silicon. The complete shared-memory interconnect fabric invented in this research consists of a hierarchy AMBA Advanced High-Speed Bus (AHB), a stack of processing elements and a shared memory acting as a communication buffer. Since the heart of this SoC communication infrastructure operates with a frequency in excess of 2 GHz, it is imperative to implement the main system cores with full-custom circuits to optimize the speed of the AHB utilized to build the hard IP.

The purpose of this research is to undergo the learning curve involved with the design and the optimization of high-speed circuits, and the many subtleties involved with their layout. To enhance the understanding in high-frequency circuit design, a thorough

study on dynamic logic is completed so as to isolate speediest avenues. The work reported in this document spans over a wide range of abstraction levels. An implementation as ambitious as the one presented in this thesis requires a sequence of actions carefully planned to survive the many obstacles to be encountered in the design process. To accomplish this research goal, three tasks have been identified, including (1) architectural definition, (2) circuit decisions, and (3) circuit design.

There are several new and significant contributions in this dissertation. These contributions are as follows: (1) a new architecture supporting multiple data stream in real-time is defined; (2) means to implement a pipelined memory to obtain a 2 GHz dual-port memory from four 500 MHz SSRAMs are provided; (3) a novel VLIW AHB arbiter, understanding the concept of phases necessary to operate the pipelined memory, is presented; (4) a new parallel static NAND structure, nearly fanin independent, is proposed; (5) an aggressive, yet unconventional, TSPC style is developed; and (6) circuit techniques to achieve beyond-1GHz circuits are synthesized. In addition, from this research, several extended research fields are possible.

7 LIST OF REFERENCES

[1] Hayes, J. P., *Computer Architecture and Organization*, McGraw-Hill, 3rd ed., 1998.

[2] ARM, Technical Specification: *AMBA Specification,* Doc No.: ARM IHI-0011A, Issued: May 2001.

[3] Marc Bertola, *Conception, réalisation et étude d'une plate-forme générique basée sur le protocole AMBA AHB*, M.A.Sc. thesis, Universite de Montréal, April 2003.

[4] Jean Pepga Bissou, *Conception de haut niveau d'une plate-forme SoC et de son système d'interconnexion*, M.A.Sc. thesis, University of Montréal, March 2003.

[5] Martin Dubois, *Modélisation et conception d'une plate-forme de traitement et transmission de signaux vidéo numérique*, M.A.Sc. thesis, University of Montréal, July 2004.

[6] Weste, N. H. E., K. Eshraghian, *Principles of CMOS VLSI Design*, Addison Wesley, 2nd ed., 1992.

[7] Smith, M. J. S., *Application-Specific Integreted Circuits*, Addison Wesley, 1997.

[8] Chen, W.K., *The VLSI Handbook*, CRC Press, 1999.

[9] P. Magarshack, P. G. Paulin, "System-on-Chip Beyond the Nanometer Wall", Proceedings of Design Automation Conference (DAC) 2003, June 2003.

[10] Cadence, User Guide: *Cadence AMS Environment User Guide,* Product Version 5.0, September 2003.

[11] Cadence, User Guide: *Cadence Hierarchy Editor User Guide,* Product Version 5.0, March 2003.

[12] Cadence, User Guide: *SimVision Analysis Environment User Guide,* Product Version 4.1, March 2003.

[13] Cadence, User Guide: *Spectre Circuit Simulator User Guide,* Product Version 5.0, June 2003.

[14] Cadence, Reference Manual: *Hspice/Spice Interface and Spice 2G.6 Reference Manual,* Product Version 5.0, October 2003.

[15] Cadence, User Guide: *Verilog-XL User Guide*, Product Version 4.1, November 2002.

[16] Palnitkar, S., *Verilog HDL*, Prentice Hall, 2nd ed., 2003.

[17] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, Jan. 2002, pp. 70-78.

[18] ARM, White Paper: *Multi-Layer AHB: Overview,* Doc No.: ARM DVI 0045A, Issued: 2001.

[19] IBM, Technical Specifications: *64-Bit Processor Local Bus,* Doc No.: SA-14-2534-01, Issued: May 2001.

[20] B. Ackland, A. Anesko, D. Brinthaupt, S. J. Daubert, A. Kalavade, J. Knobloch, E. Micca, M. Moturi, C. J. Nicol, J.H. O'Neil, J. Othmer, E. Sackinger, K.J. Singh, J. Sweet, C. J. Terman, and J. Williams, "A Single-chip, 1.6-Billion, 16-b MAC/S Multiprocessor DSP," *IEEE Journal of Solid-State Circuits,* March 2000, pp. 412-424.

[21] Culler, D. E., J. P. Singh, Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann, 1999.

[22] Handy, J., *The Cache Memory Book*, New York: Academic, 1993.

[23] ARM, Technical Specifications: *AMBA AXI Protocol*, Doc No.: ARM IHI 0022A.

[24] E.G. Friedman, "Clock Distribution Networks in Synchronous Digital Integrated Circuits," *Proceedings of the IEEE,* Vol. 89, No. 5, May 2001, pp. 665-692.

[25] International Technology Roadmap for Semiconductors, Interconnect: Semiconductor Industry Association, 2004.

[26] OCP-IP Association, Technical Specifications: *Open Core Protocol Specification,* Doc No.: 161-000125-0001.

[27] Rabaey, J.M., *Digital Integrated Circuits,* Prentice Hall, 1996.

[28] N. Goncalves and H. J. De Man, "NORA: A Racefree dynamic CMOS technique for pipelined logic structures," *IEEE Journal of Solid State Circuits,* vol. SC-18, pp. 261-266, 1983.

[29] J. Yuan, C. Svensson, "High-Speed CMOS Circuit Technique," *IEEE Journal of Solid-State Circuits*, February 1989, pp. 62-70.

[30] J. Yuan, C. Svensson, "New Single-Clock CMOS Latches and Flipflops with Improved Speed and Power Savings," *IEEE Journal of Solid-State Circuits,* January 1997, pp. 62-69.

[31] M. Yamashina, H. Yamada, "An MOS Current Mode Logic (MCML) Circuit for Low-Power Sub-GHz Processors," IEICE Transaction on Electron, vol. E75-C, NO. 10, October 1992, pp. 1181-1187

[32] J. M. Musicer, J. Rabaey, "MOS Current Mode Logic for Low Power, Low Noise CORDIC Computation in Mixed-Signal Environments," *ISLPED 2000*, pp. 102-107.

[33] J.P. Bissou, M. Dubois, Y. Savaria, G. Bois, "High-Speed System Bus for a SoC Network Processing Platform," *IEEE International Conference on Microelectronics,* December 2003, pp.194-197.

[34] A. Landry, Y. Savaria, M. Nekili, "A Beyond-1 GHz High-Speed Bus for SoC DSP Platforms," *IEEE International Conference on Microelectronics,* December 2004.

[35] A. Landry, Y. Savaria, M. Nekili, "A Novel 2GHz Multi-Layer AMBA High-Speed Bus Interconnect Matrix For SoC Platforms," *IEEE International Symposium on Circuits and Systems*, To be published in May 2005.

[36] Canadian Microelectronics Corporation, Tutorial: *Digital IC Design Flow,* Doc No.: ICI-134, July 2004.

[37] Canadian Microelectronics Corporation, Tutorial: *Tutorial on CMC's Analog IC Design Flow,* Doc No.: ICI-098-1, September 2000.

[38] Cadence, User Guide: WaveScan User Guide, Product Version 5.0, September 2003.

[39] S. Ghannoum, D. Chtchvyrkov and Y. Savaria, "A comparative study of single-phase clocked latches using estimation criteria," *IEEE International Symposium on Circuits and Systems*, 1994, pp.347-350.

[40] R. Ho, K. W. Mai, M. A. Horowitz, "The Future of Wires," *Proceedings of the IEEE,* vol. 89, NO. 4, Apr. 2001, pp. 490-504.

[41] M.S. Hrishikesh, N.P. Jouppi, K.I. Farkas, "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," *IEEE International Symposium on Computer Architecture,* 2002, pp. 14-24.

[42] Baker, R.J., H. W. Li, D. E. Boyce, *CMOS: Circuit Design, Layout, and Simulation*, IEEE Press, 1997.

[43] ARM, Technical Reference Manual: *ARM1136JF-S and ARM1136J-S,* Doc No.: ARM DDI 0211D.

[44] Virage Logic, Software User Guide: *Embed-it! Integrator / Custom-Touch Memory Compiler,* Release 3.4.4, August 2003.

[45] Hennessy, J. L., D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 3rd ed., 2003.

[46] M. T. J. Strik, A. H. Timmer, J. L. van Meerbergen, G.J. van Rootselaar, "Heterogeneous Multiprocessor for the Management of Real-Time Video and Graphics Streams," *IEEE Journal of Solid State Circuits,* Nov. 2000, pp. 1722-1731.

[47] ARM, Technical Reference Manual: *ARM946E-S,* Doc No.: ARM DDI 0155D, Issued: August 2001.

[48] Mano, M., *Digital Design,* Prentice Hall, 1991.

[49] K. J. Nowka, G. D. Carpenter, E. W. MacDonald, H. C. Ngo, B. C. Brock, K. I. Ishii, T. Y. Nguyen, J. L. Burns, "A 32-bit PowerPC System-on-a-chip With Support for Dynamic Voltage Scaling and Dynamic Frequency Scalling," *IEEE Journal of Solid State Circuits,* vol 37, pp. 1441-1447, November 2002.

[50] A. Landry, Y. Savaria, M. Nekili, "Cicruit Techniques for a 2 GHz AMBA AHB Bus," *IEEE Northeast Workshop on Circuits and Systems,* to be published in June 2005.

[51] D. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems,* PhD Thesis, Stanford University, October 1984.