FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 19, no. 3, December 2006, 405-428

An Overview of On-Chip Buses

Milica Mitić and Mile Stojčev

Abstract: The electronics industry has entered the era of multi-million-gate chips, and thereXs no turning back. This technology promises new levels of integration on a single chip, called the System-on-a-Chip (SoC) design, but also presents significant challenges to the chip designer. Processing cores on a single chip, may number well into the high tens within the next decade, given the current rate of advancements, [1]. Interconnection networks in such an environment are, therefore, becoming more and more important [2]. Currently, on-chip interconnection networks are mostly implemented using buses. For SoC applications, design reuse becomes easier if standard internal connection buses are used for interconnecting components of the design. Design teams developing modules intended for future reuse can design interfaces for the standard bus around their particular modules. This allows future designers to slot the reuse module into their new design simply, which is also based around the same standard bus [3]. In this paper we give an overview of the more popular on-chip bus-based interconnection networks such as AMBA, Avalon, CoreConnect, STBus, Wishbone, etc. The main characteristics of the considered buses in respect to topology, arbitration method, bus-width, and types of data transfers are discussed.

Keywords: On-chip interconnection network, on-chip bus, on-chip communication protocol.

1 Introduction

Shrinking process technologies and increasing design sizes have led to highly complex billion-transistor integrated circuits (ICs). As a consequence, manufacturers are integrating increasing numbers of components on a chip. A heterogeneous SoC might include one or more programmable components such as general purpose processors cores, digital signal processor cores, or application-specific intellectual property (IP) cores, as well as an analog front end, on-chip memory, I/O devices,

Manuscript received

The authors are with Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mails: [milicam,stojcev]@elfak.ni.ac.yu).

and other application specific circuits. In other words, a SoC is an IC that implements most or all the functions of a complete electronic system [4].

On-chip bus organized communication architecture (CA) is among the top challenges in CMOS SoC technology due to rapidly increasing operation frequencies and growing chip size. In general, the performance of the SoC design heavily depends upon the efficiency of its bus structure. The balance of computation and communication in any application or task is, of course, known as a fundamental determinant of delivered performance. Usually, IP cores, as constituents of SoCs, are designed with many different interfaces and communication protocols. Integrating such cores in a SoC often requires insertion of suboptimal glue logic. Standards of on-chip bus structures were developed to avoid this problem. Currently there are a few publicly available bus architectures from leading manufacturers, such as CoreConnect from IBM [5], AMBA from ARM [6], SiliconBackplane from Sonics [7], and others. These bus architectures are usually tied to processor architecture, such as the PowerPC or the ARM processor. Manufacturers provide cores optimized to work with these bus architectures, thus requiring minimal extra interface logic.

This paper gives an overview of the more popular on-chip standardized buses architectures such as AMBA, CoreConnect, Wishbone, STBus, and others, both from an industrial and research viewpoint. The crucial features, including bus topologies, arbitration methods, bus-widths, and types of data transfers are considered.

The rest of this paper is organized as follows: Section II presents background material on CAs, including a survey of typical topologies and communication protocols in use today. Section III, as a central part of this paper, gives an overview of several more popular SoC CAs. In Section IV, for comparison purposes, some common features of the analyzed buses are presented. Concluding remarks are given in Section V.

2 On-Chip Communication Architectures

2.1 Background

The design of on-chip CAs addresses the following three issues [8]:

1. *Definition of CA topology* - defines the physical structure of the CA. Numerous topologies exist, ranging from single shared bus to more complex architectures such as bus hierarchies, token ring, crossbar, or custom networks.
2. *Selection and configuration of the communication protocols* - for each channel/bus in the CA, communication protocols specify the exact manner in which communication transaction occurs. These protocols include arbitration mechanisms (e.g. round robin access, priority-based selection [5, 6],

time division multiplexed access [7], which are implemented in centralized or distributed bus arbiters.

3. Communication mapping - refers to the process of associating abstract system-level communications with physical communication paths in the CA topology [8].

2.2 Topologies

In respect to topology on-chip communication architectures can be classified as:

Shared bus: The system bus is the simplest example of a shared communication architecture topology and is commonly found in many commercial SoCs [9]. Several masters and slaves can be connected to a shared bus. A block, bus arbiter, periodically examines accumulated requests from the multiple master interfaces and grants access to a master using arbitration mechanisms specified by the bus protocol. Increased load on a global bus lines limits the bus bandwidth. The advantages of shared-bus architecture include simple topology, extensibility, low area cost, easy to build, efficient to implement. The disadvantages of shared bus architecture are larger load per data bus line, longer delay for data transfer, larger energy consumption, and lower bandwidth. Fortunately, the above disadvantages with the exception of the lower bandwidth, may be overcome by using a low-voltage swing signaling technique.

Hierarchical bus: this architecture consists of several shared busses interconnected by bridges to form a hierarchy. SoC components are placed at the appropriate level in the hierarchy according to the performance level they require. Low-performance SoC components are placed on lower performance buses, which are bridged to the higher performance buses so as not to burden the higher performance SoC components. Commercial examples of such architectures include the AMBA bus [6], CoreConnect [5], etc. Transactions across the bridge involve additional overhead, and during the transfer both buses remain inaccessible to other SoC components. Hierarchical buses offer large throughput improvements over the shared busses due to: (1) decreased load per bus; (2) the potential for transactions to proceed in parallel on different buses; and multiple ward communications can be preceded across the bridge in a pipelined manner [8].

Ring: in numerous applications, ring based applications are widely used, such as network processors, ATM switches [5, 8]. In a ring, each node component (master/slave) communicates using a ring interface, are usually implemented by a token-pass protocol.

2.3 On-chip communication protocols

Communication protocols deal with different types of resource management algorithms used for determining access right to shared communication channels. From this point of view, in the rest of this section, we will give a brief comment related to the main feature of the existing communication protocols.

Static-priority: employs an arbitration technique. This protocol is used in shared-bus communication architectures. A centralized arbiter examines accumulated requests from each master and grants access to the requesting master that is of the highest priority. Transactions may be of non-preemptive or preemptive type. AMBA, CoreConnect... use this protocol [5, 6].

Time Division Multiple Access (TDMA): the arbitration mechanism is based on a timing wheel with each slot statically reserved for unique master. Special techniques are used to alleviate the problem of wasted slots. Sonics uses this protocol [7].

Lottery: a centralized lottery manager accumulates request for ownership of shared communication resources from one ore more masters, each of which has, statically or dynamically, assigned a number of Xlottery ticketsX [10].

Token passing: this protocol is used in ring based architectures. A special data word, called token, circulates on the ring. An interface that receives a token is allowed to initiate a transaction. When the transaction completes, the interface releases the token and sends it to the neighboring interface.

Code Division Multiple Access (CDMA): this protocol has been proposed for sharing on-chip communication channel. In a sharing medium, it provides better resilience to noise/interference and has an ability to support simultaneously transfer of data streams. But this protocol requires implementation of complex special direct sequence spread spectrum coding schemes, and energy/battery inefficient systems such as pseudorandom code generators, modulation and demodulation circuits at the component bus interfaces, and differential signaling [11].

2.4 Other interconnect issues

We will point now to several interconnect issues that have direct impact on bus organization and its efficiency.

Programming model- consists of a load and store operations. These operations are implemented as a sequence of primitive bus transactions. Modules issuing requests are called masters and those serving requests are called slaves [12].

Split versus non-split buses- If there is a single arbitration for a request response pair, the bus is called non-split. In this case, the bus remains allocated to the master of the transaction until the response is delivered. Alternatively, in a split bus, the

bus is released after the request to allow transactions from different masters to be initiated [13].

Transaction ordering- usually, all transactions on a bus are ordered. However, on a split bus, a total ordering of transactions on a single master may cause performance degradation. This situation is typical when slaves respond to different speed. To solve this problem, recent extensions to bus protocols allow transactions to be performed on connections [14, 15].

Atomic chains of transactions- represent a sequence of transactions initiated by a single master that is executed on a single slave exclusively. During this activity, other masters are denied to access that slave until the end of the first transaction. This mechanism is standardly used to implement synchronization mechanisms between master modules (i.e., semaphores) [13].

Media arbitration- bus master modules access the bus and the arbiter grants access. Arbitration is centralized as there is only one arbiter component. It is also global, since all requests as well as the state of the bus, are visible to the arbiter. When a grant is given, the complete path from the source to the destination is exclusively reserved [12, 13].

Destination name and routing- command address and data are broadcasted on the bus. They reach every destination, only one of each activates, based on the broadcasted address, and executes the requested command [12, 13].

Latency- is caused by the following two factors: a) the access time to the bus, which is the time until the bus is granted; and b) the latency introduced by the bus to transfer the data [12].

Data format- is defined by separate wire groups for the transaction type, address, write data, read data, and return acknowledgments/errors [5, 6, 16, 17].

2.5 Advantages and disadvantages of on-chip-buses and new proposals

In the bus-based design approach IP components communicate through one or more buses usually interconnected by bus bridges. Since the bus specification can be standardized, libraries of components whose interfaces directly match this specification can be developed. Even if components follow the bus standard, very simple bus interface adapters may still be needed. For components that do not directly match the specification, wrappers have to be built. Companies offer very rich component libraries and specialized development and simulation environments for designing systems around their buses. A somewhat different approach is core a based design. In this case, IP components are compliant to a bus- independent and standardized interface and thus are directly connected to each other. Although the standard may support a wide range of functionalities, each component may have an interface containing only the functions that are relevant for it. These components

may also be interconnected through a bus, in which case standard wrappers can adapt the component interface to the bus.

As a conclusion we can say that on-chip-bus-design and on-chip-core-based design methodologies are integration approaches that depend on standardized component or bus interfaces. They allow the integration of homogeneous IP components that follow these standards to be directly connected to each other, without requiring the development of complex wrappers. Let us note that on-chip buses rely on shared communication resources and on arbitration mechanism that is in charge of serializing bus access requests. This widely adopted solution unfortunately suffers from power and performance scalability limitations, and restricted sharing of resources between communicating entities. For bus networks, the bus is occupied by a single communication even if multiple communications could operate simultaneously on different portions on the bus. Therefore a lot of effort has been devoted to the development of advanced bus topologies (e.g. partial or full crossbar, bridged buses) and protocols for better support of route-ability, flexibility, reliability, and reconfigure-ability. Therefore, a systematic way of designing networks with possibly arbitrary topology is gaining the importance [2].

In the long run, a more aggressive approach is needed. For particular needs, the SoC may be built around a sophisticated and dedicated network-on-chip that may deliver very high performance for connecting a large number of components. It seems that this design paradigm shifts towards packetized on-chip communication based on micro-networks of interconnects or networks-on-chip [18].

3 SoC Buses Overview

In the sequel an overview of the more relevant SoC communication architectures will be given. Due to space limitation the discussion will be focused on describing the more distinctive features of each of them.

3.1 AMBA bus

AMBA (Advanced Microcontroller Bus Architecture) [6, 19], is a bus standard devised by ARM with aim to support efficient on-chip communications among ARM processor cores. Nowadays, AMBA is one of the leading on-chip busing systems used in high performance SoC design. AMBA (see Fig. 1) is hierarchically organized into two bus segments, system- and peripheral-bus, mutually connected via bridge that buffers data and operations between them. Standard bus protocols for connecting on-chip components generalized for different SoC structures, independent of the processor type, are defined by AMBA specifications. AMBA does not define method of arbitration. Instead it allows the arbiter to be designed to suit the

applications needs, the best. The three distinct buses specified within the AMBA bus are:

- ASB (Advanced System Bus) - first generation of AMBA system bus used for simple cost-effective designs that support burst transfer, pipelined transfer operation, and multiple bus masters.
- AHB (Advanced High-performance Bus) X as a later generation of AMBA bus is intended for high performance high-clock synthesizable designs. It provides high-bandwidth communication channel between embedded processor (ARM, MIPS, AVR, DSP 320xx, 8051, etc.) and high performance peripherals/ hardware accelerators (ASICs MPEG, color LCD, etc), on-chip SRAM, on-chip external memory interface, and APB bridge. AHB supports a multiple bus masters operation, peripheral and a burst transfer, split transactions, wide data bus configurations, and non tristate implementations. Constituents of AHB are: AHB-master, slave-, arbiter-, and Xdecoder.
- APB (Advanced Peripheral Bus X is used to connect general purpose low-speed low-power peripheral devices. The bridge is peripheral bus master, while all buses devices (Timer, UART, PIA, etc) are slaves. APB is static bus that provides a simple addressing with latched addresses and control signals for easy interfacing.

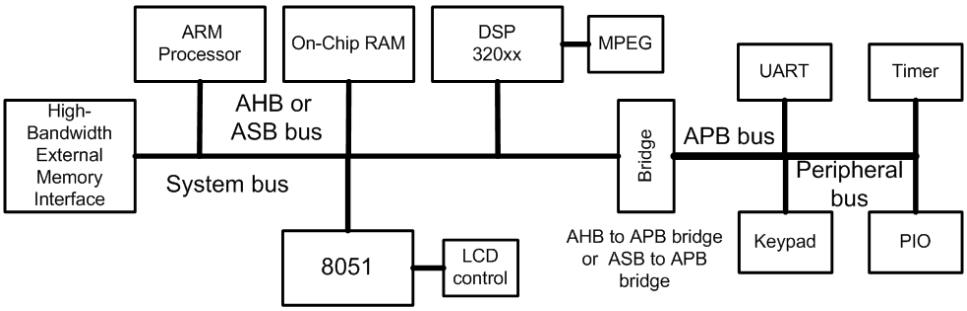

Fig. 1. AMBA based system architecture.

Recently, two new specifications for AMBA bus, Multi-Layer AHB and AMBA AXI, are defined. [6, 20]. Multi-layer AHB provides more flexible interconnect architecture (matrix which enables parallel access paths between multiple masters and slaves) with respect to AMBA AHB, and keeps the AHB protocol unchanged. AMBA AXI is based on the concept point-to-point connection.

Good overview papers related to AMBA specifications are references [6,20,21].

3.2 Avalon

Avalon bus (see Fig. 2) is a bus architecture designed for connecting on-chip processors and peripherals together into a system-on-a-programmable chip (SOPC). As an AlteraXs parameterized bus Avalon is mainly used for FPGA SoC design based on Nios processor [22, 23].

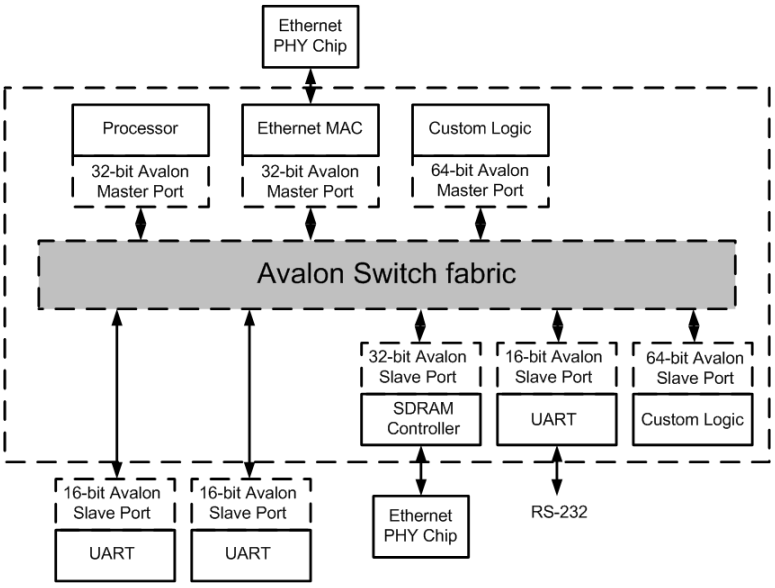

Fig. 2. Avalon bus based system

Avalon has a set of predefined signal types with which a user can connect IP blocks. Avalon is a synchronous interface and specifies the port connections between master and slave components and specifies the timing by which these components communicate. Basic Avalon bus transactions transfer one data item 8-, 16-, 32-, 64-, or 128-bits wide. Avalon uses separate address, data and control lines.

This bus supports multiple bus masters. Masters and slaves interact with each other based on a technique called slave-side (distributed) arbitration.

The Avalon bus model (switch fabric) provides the following services to Avalon peripherals connected to the bus: data-path multiplexing, address decoding, wait-state generation, dynamic bus sizing, interrupt priority assignment, latent transfer capabilities, and a streaming Read and Write capabilities [22, 23].

AlteraXs SOPC Builder, as a system development tool, automatically generates the switch fabric logic that supports each type of transfer supported by the Avalon interface.

3.3 CoreConnect

CoreConnect [5] is an IBM-developed on-chip bus. By reusing processor, subsystem and peripheral cores, supplied from different sources, enables their integration into a single VLSI design. CoreConnect is a hierarchically organized architecture. It is comprised of three buses that provide an efficient interconnection of cores, library macros, and custom logic within a SoC (see Figure 3).

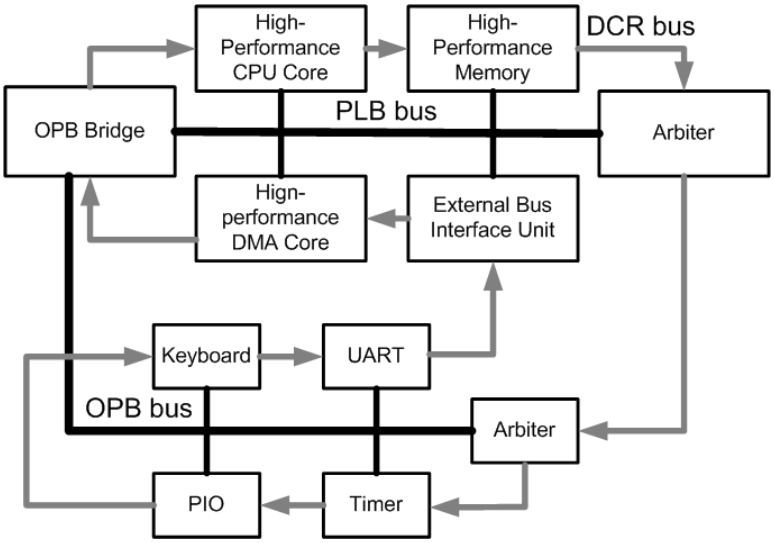

Fig. 3. CoreConnect bus based system

PLB (Processor Local Bus) X is the main system bus. It is synchronous, multi-master, central arbitrated bus that allows achieving high-performance and low-latency on-chip communication. Separate address, and data buses support concurrent read and write transfers. PLB macro, as glue logic, is used to interconnect various master and slave macros. Each PLB master is attached to the PLB through separate addresses, read-data and write-data buses, and other control signals. PLB slaves are attached to PLB through shared, but decoupled, address, read data, and write data buses. Up to 16 masters can be supported by the arbitration unit, while there are no restrictions in the number of slave devices [19].

OPB (On-chip Peripheral Bus) - is optimized to connect lower speed, low throughput peripherals, such as serial and parallel port, UART, etc. Crucial features of OPB are: fully synchronous operation, dynamic bus sizing, separate address and data buses, multiple OPB bus masters, single cycle transfer of data between bus masters, single cycle transfer of data between OPB bus master and OPB slaves, etc. OPB is implemented as multi-master, arbitrated buses. Instead of tristate drivers

OPB uses distributed multiplexer. PLB masters gain access to the peripherals on the OPB bus through the OPB bridge macro. The OPB bridge acts as a slave device on the PLB and a master on the OPB.

DCR bus (Device Control Register bus) X is a single master bus mainly used as an alternative relatively low speed datapath to the system for: (a) passing status and setting configuration information into the individual device-control-registers between the Processor Core and others SoC constituents such as Auxiliary Processors, On-Chip Memory, System Cores, Peripheral Cores, etc; and (b) design for testability purposes. DCR is synchronous bus based on a ring topology implemented as distributed multiplexer across the chip. It consists of a 10-bit address bus and a 32-bit data bus. CoreConnect implements arbitration based on a static priority, with programmable priority fairness.

3.4 STBus

STBus is an on-chip bus protocol developed by STMicroelectronics [16]. It represents a set of protocol, interfaces and architectural specifications intended to implement the communication network of digital systems. The STBus interfaces and protocols are closely related to the Virtual Component Interface (VCI) industry standard.

STBus implements both the protocols definition and the bus components. The following protocols are used [16, 24]:

- Type I (Peripheral protocol) - is a simple synchronous handshake protocol with limited set of available command types, suitable for register access and slow peripherals. No pipelining is applied. Type I acts as a Request-Grant protocol. Only limited operation code and length are supported.
- Type II (Basic Protocol) - is more efficient than Type I because it supports split transactions and adds pipelining features. The transaction set includes read/write operation with different sizes (up to 64 bytes) and also specific operations like Read-Modify-Write and Swap. Type II is equivalent to the Request-Grant-Valid protocol. Transactions may also be grouped into chunks to ensure allocation of the slave and to ensure no interruption of the data stream. This protocol is typically suited for External Memory controllers. A limitation of this protocol is that the traffic must be ordered and transactions must be symmetric (i.e. the number of the requesting cells equals to the number of the response ones).
- Type III (Advanced protocol) - is the most efficient, as it adds support for split transactions, out-of-order executions and asymmetric communications (i.e. the number of cells might differ between request and response). Type III is mainly used by CPUs, multi-channel DMAs and DDR controllers.

The STBus is modular and allows master and slaves of any protocol type and data size to communicate, through the use of appropriate type/size converters. A wide variety of arbitration policies is also available, such as bandwidth limitation, latency arbitration, LRU, priority-based arbitration, etc.

The components interconnected by the STBus can be either initiators (initiates transactions on the bus by sending requests, such as CPUs or ASICs) or targets (responds to requests, such as memories, registers or dedicated peripherals). Initiators can load or store data through the STBus backbone (see Fig. 4). Some resources might be both initiators and peripheral/targets.

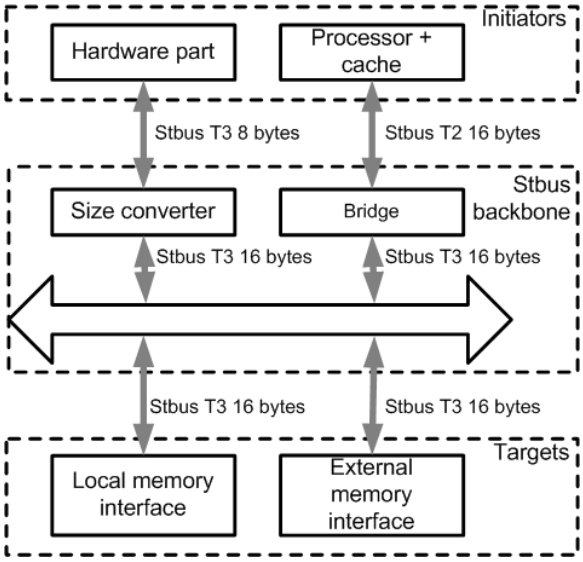

Fig. 4. STBus interconnect

STBus based system includes three kinds of components [24]:

- Switch or node- this block arbitrates and routes the requests and responses. Different kinds of arbitration are possible, including: fixed priorities, variable priorities, dynamic priorities, latency based, bandwidth based and least recently used.
- Converter or bridge domain- converts the request from the protocol to another, for example from basic protocol to advanced protocol
- Size converter: is used between two buses of same type of different widths. It includes buffering capacity.
- STBus can instantiate different bus topologies such as [19]:
- Single shared bus-suitable for simple low-performance implementations. This bus characterizes minimal wiring area but limited scalability.

- Full crossbar-intended for high-performance systems. Wiring area is large.
- Partial crossbar-used in medium performance systems, represents a good compromise with respect to the previous two proposals.

3.5 Wishbone

Wishbone [25] bus architecture was developed by Silicore Corporation. In August 2002, OpenCores (organization that promotes open IP cores development) put it into the public domain. This means that Wishbone is not copyrighted and can be freely copied and distributed.

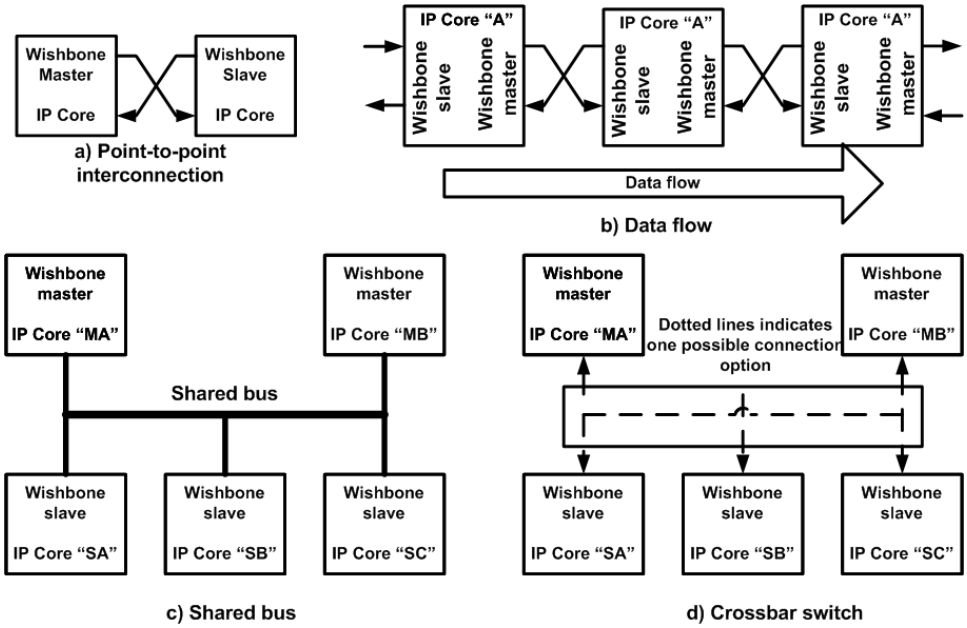

Fig. 5. Possible Wishbone interconnections

The Wishbone defines two types of interfaces, called master and slave. Master interfaces are IPs which are capable of initiating bus cycles, while slave interfaces are capable of accepting bus cycles [19]. The hardware implementations support various types of interconnection topologies (see Figure 5) such as:

a) point-to-point connection- used for direct connection of two participants that transfer data according to some handshake protocol
b) dataflow interconnection- used in linear systolic array architectures for implementation of DSP algorithms
c) shared bus- typical for MPSoCs organized around single system bus

d) crossbar switch interconnection- usually used in MPSoCs when more than one masters can simultaneously access several different slaves. The master requests a channel on the switch, once this is established, data is transferred in a point-to-point manner.

The Wishbone supports different types of bus transactions, such as read/write, implementing blocking/unblocking access. A Read-Modify-Write transfer is also supported.

Wishbone doesnXt define hierarchical buses. In applications where two buses should exist, one slow and one fast, two separated Wishbone interfaces could be created.

Designer can also choose arbitration mechanism and implements it to fit the application needs, best.

3.6 CoreFrame

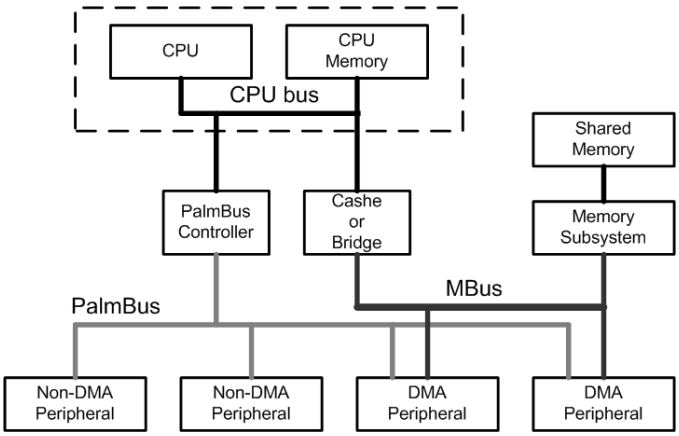

Fig. 6. CoreFrame architecture based system structure

The CoreFrame [26] architecture is low power high-performance on-chip interconnect architecture for integration of SoC blocks. From a high-level point of view, the CoreFrame architecture (see Figure 6) is viewed as a system of three buses (CPU bus, PalmBus and MBus). The CPU bus is connected to PalmBus via PalmBus controller and to the MBus through a cache or bridge. The PalmBus and MBus are independent parallel buses, rather than a hierarchy of buses. Concurrent activities may be achieved on both buses maximizing available bandwidth resources. To avoid three-state buffering, CoreFrame doesnXt use shared signal lines. Instead it uses point-to-point signals and multiplexing. Communication between subsystems is carried out through shared memory variables.

A PalmBus represents a master-slave interface with a single-master intended for communications between the CPU and peripheral blocks. It is not used to access memories. The PalmBus is designed for low-speed access from the CPU core and it provides the I/O backplane and allows the processor to configure and control peripheral blocks. Timings of the bus are synchronous with the CPU core. PalmBus is also designed with low-power consumption in mind [27].

The MBus is designed for high-speed accesses to shared memory from the CPU core and peripheral blocks. The MBus protocol is optimized for both ASIC-type implementations and data transfers to a data memory devices [28].

3.7 Marble

Marble (Manchester AsynchRonous Bus for Low Energy) developed at the Manchester University is on-chip two channel micropipeline bus with centralized arbitration and address decoding which operates without global clock pulse. It is intended to provide interconnections of asynchronous macrocells within the VLSI ICs [29].

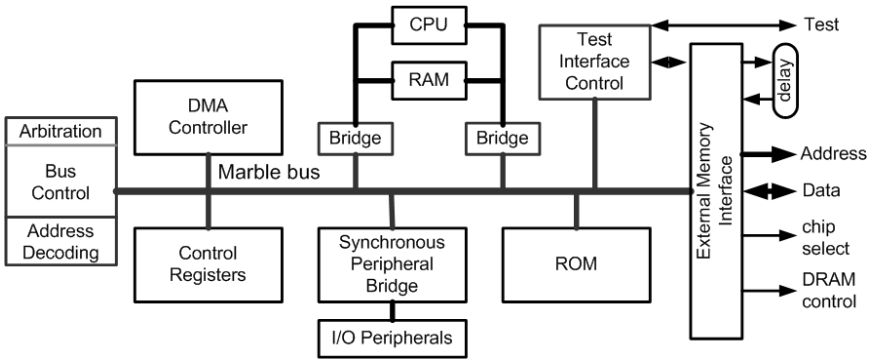

Fig. 7. The AMULETH3H system

MARBLE is based on a split-transfer architecture allowing transfers between different initiators and targets to be interleaved without the needs for retries, thus giving low energy operation and low latency.

A MARBLE bus consists of two asynchronous multipoint channels. One of these channels carries the command from the initiator to the target, returning either an accept or defer status. The other multipoint channel carries a response from the target to the initiator (and the read or write data in the appropriate direction). The two channels are used in a decoupled transfer scheme with loose coupling between channels in order to implement split transactions [30].

The interconnection provided by MARBLE is used in AMULET3H microprocessor (see Figure 7). It is intended to connect CPU core and DMA controller to

RAM, ROM, and other peripherals [30].

In general, MARBLE demonstrates that all the features of a high-speed on-chip macrocell bus can be implemented efficiently in a fully asynchronous design style.

3.8 PI bus

The PI (Peripheral Interconnect) bus was developed by several European semiconductor companies (Advanced RISC Machines, Philips Semiconductors, SGS-THOMSON Microelectronics, Siemens, TEMIC/MATRA MHS) within a framework of European project (OMI, Open Microprocessor Initiative framework) [Error! Reference source not found.]. PI bus is an open standard published by OMI. For SoC design purpose PI bus System Toolkit is developed. VHDL codes for master, slave and control units are freely distributed. In addition, synthesis scripts for different ASIC and FPGA technologies, and examples of system solutions are available [9].

PI bus is a synchronous bus with un-multiplexed address and data signals that supports operation of multiple masters and bridges. It is an on-chip bus used in modular, highly integrated SoC designs. PI bus is designed for memory mapped data transfers between its bus agents. Bus agents are on-chip modules equipped with PI-bus interface and connected via PI-bus signals. A PI-bus agent acts as a PI-bus master when it initiates data read/write operations, since bus ownership has been granted to the agent. A PI-bus agent who is addressed at PI-bus operation acts as a PI-bus slave when it performs the requested data read/write operation. Typical masters are processor modules, coprocessors, or DMA modules, while typical slaves are on-chip memory and input- output interfaces to the external world (see Figure 8) [9].

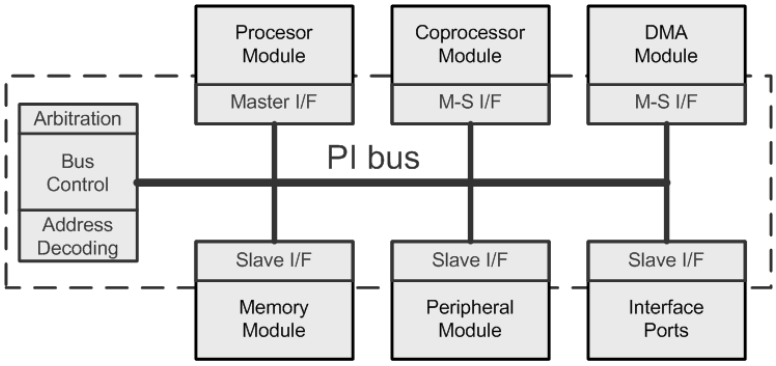

Fig. 8. TModules of a PI bus connected system

The main features of PI-bus are: 1) processor independent implementation and

design; 2) demultiplexed operation; 3) clock synchronous; 4) peak transfer rate of 200 MHz (50 MHz bus clock); 5) address and data bus scalable (up to 32 bits); 6) 8-, 16-, 32-bit data access; 7) broad range of transfer types from single to multiple data transfers; and 8) multi-master capability. The PI-bus does not provide: a) cache coherency support; b) broadcast; c) dynamic bus sizing; and d) unaligned data access [9].

3.9 OCP

OCP (Open Core Protocol) [14] is an interface standard that interconnects IP cores to on-chip bus. The OCP defines a comprehensive, bus-independent, high-performance and configurable interface between IP cores and on-chip communication subsystems. A designer selects only those signals and features from the palette of OCP configurations needed to fulfill all of an IP coreXs unique data, control and test signaling requirements. Existing IP cores may be inexpensively adapted. Defining a core interface using the OCP provides a complete description for system integration. The main features of OCP interface are: 1) Master - slave interface with unidirectional signals: 2) Driven and sampled by the rising edge of the OCP clock; 3) Fully synchronous, no multi-cycle timing paths; 4) All signals are strictly point-to-point (except clock & reset); 5) Simple request / acknowledge protocol; 6) Supports data transfer on every clock cycle; 7) Allows master or slave to control transfer rate; 8) Configurable data word width; 9) Configurable address width; 10) Pipelined or blocking reads; 11) Specific description formats for core characteristics, interfaces (signals, timing & configuration), performance [15].

Some of the standard on-chip buses, such as AMBA and SiliconBackplane XNetwork, use OCP. Communication requirements concerning IP core can be described using this protocol format. OCP interface is a user-settable, so the designer can define interface attribute, such as address and data bus width. Beside basic OCP version, there are four extensions: Simple Extension, Complex Extension, Sideband Extension and Debug and Test Interface Extension. Basic OCP includes only data flow signals and is based on simple request and acknowledge protocol. However, the optional extensions support more functionality in control, verification and testing. Simple Extension and Complex Extension support burst transaction and pipelined write operations. In addition, Sideband Extension supports user-defined signals and asynchronous reset. Also, Debug and Test Interface Extension supports JTAG (Join Test Action Group) and clock control. This is the reason why, when integrated in SoC, the OCP protocol allows debugging and IP block test generating.

Figure 9 presents SoC design based on the OCP protocol.

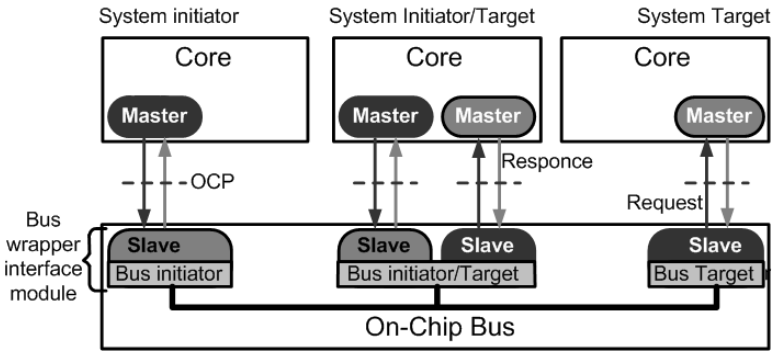

Fig. 9. Wrapped bus and OCP instances

3.10 VCI (Virtual Component Interface)

The Virtual Component Interface [15] (VCI) is an interface rather than a bus. Thus the VCI specifies: a) a request-response protocol; b) a protocol for the transfer of requests and responses; and c) the contents and coding of these requests and responses. The VCI does not touch areas as bus allocation schemes, competing for a bus, and so forth.

There are three complexity levels for the VCI: Peripheral (PVCI), Basic VCI (BVCI), and Advanced VCI (AVCI). The PVCI provides a simple, easily implementable interface for applications that do not need all the features of the BVCI. The BVCI defines an interface that is suitable for most applications. It has a powerful, but not overly complex protocol. The AVCI adds more sophisticated features, such as threads, to support high-performance applications. The PVCI is a subset of the BVCI, and the AVCI is a superset of the BVCI.

BVCI and AVCI make use of a Xsplit protocol.X That is, the timing of the request and the response are fully separate. The initiator can issue as many requests as needed, without waiting for the response. The protocol does not prescribe any connection between issuing requests and arrival of the corresponding responses. The only thing specified is that the order of responses corresponds to the order of requests. In the AVCI, requests may be tagged with identifiers, which allow such requests and request threads to be interleaved and they response to arrive in a different order. Responses bear the same tags issued with the corresponding requests, such that the relation can be restored upon the reception of a response.

As an interface, the VCI can be used as a point-to-point connection between two units called the initiator and the target, where the initiator issues a request and the target responds (see Figure 10).

The VCI can be used as the interface to a wrapper, which means a connection to

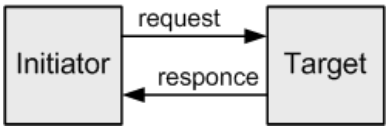

Fig. 10. VCI is a Point-to-Point Connection

a bus. This is how the VCI allows the VC to be connected to any bus. An initiator is connected to that bus by using a bus initiator wrapper. A target is connected to that bus by using a bus target wrapper. Once the wrappers for the bus have been designed, any IPs can be connected to that bus, as depicted in Figure 11.

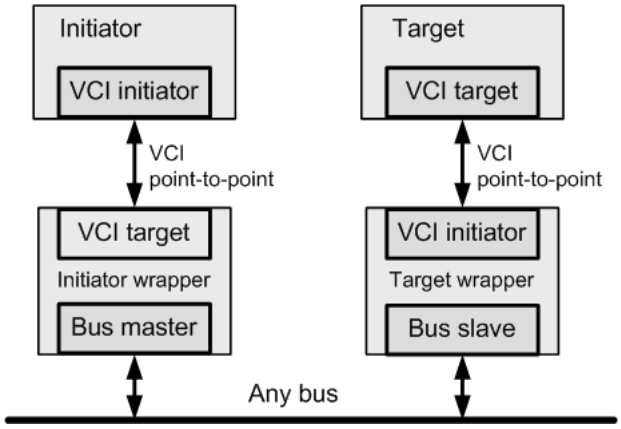

Fig. 11. Two VCI Connections Used to Realize a Bus Connection

3.11 SiliconBackplane Network

Sonics Network [7] consists of a set of architectures and SoC design tools. Defined architectures are SiliconBackplane for on-chip interconnection and MultiChip for of-chip interconnection. SiliconBackplane implements two-level arbitration, based on TDMA and round-robin. SiliconBackplane Network is network on a chip that connects IP blocks in a SoC. Network isolates the system of IP blocks from network by requiring all blocks to use single bus interface protocol OCP (Open Core Protocol). Every IP block communicates via wrapper, which Network calls an agent, using OCP. Agents communicate with each other through Network. As systems requirements change, OCP protocol and Network network support modification of many systems parameter in real time. System requirements relate to, for example, selection of arbitration scheme, definition of address space, etc. An agent is generated using tool Fast Forward Development Environment, developed by Sonics.

Basic building blocks of SiliconBackplane Network are given in Figure 12.

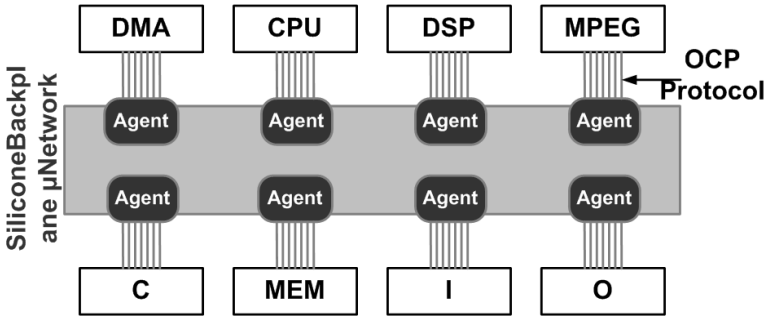

Fig. 12. SiliconBackplane Network constituents

3.12 Presented SoC bus features overview

As it is shown in Table 1, almost all of the analyzed SoC buses with exception of SiliconBackplane and CoreFrame are open standards. Their owners and the latest versions are also presented in Table 1.

Table 1. Status of the represented SoC buses

bus name	bus owner	status [Open Standard/ Licensed]	version	year
AMBA	ARM	OS	Rev. 2.0	1999
Avalon	Altera Corporation	OS	1.3	2005
CoreConnect	IBM	OS	2.9 (32 bit PLB)	2001
			3.5 (64 bit PLB)	2001
			4.6 (128 bit PLB)	2004
			2.1 (OPB)	2001
			2.9 (DCR)	2000
Wishbone	OpenCores	OS	Rev. B.3	2002
SiliconBackplane	Sonics	L	III	2002
CoreFrame	Palmchip Corporation	L	Rev. 1,01	2002
Marble	The Uni. of Manchester	n/a	n/a	1999
PI bus	OMI	OS	Rev. 0.3d	1996
hline OCP	OCP-IP	OS	Rev. 2.1	2005
hline CVI	VSIA	OS	Rev. 2.0	n/a

We will point now to some crucial properties of the prevalent existing on-chip interconnects, i.e., buses, where the communicating modules are directly connected.

Some bus protocols allow out-of-order responses per connections in their advanced modes [17], but both requests and responses arrive at the destination in the same order.

The atomic operations are implemented in one of the following two ways: a) the central arbiter locks the bus for exclusive use by the master requesting the atomic chain; and b) the central arbiter does not grant access to a locked slave. AMBA and CoreConnect use a mechanism of locking the slave and the bus for a short time. On the other hand VCI and OCP do not lock a bus, i.e., it can still be used by other modules, however, at the price of a longer locking duration of the slave.

In non-split bus (like VCI or OCP), arbitration takes places when a transaction is initiated. As a result, the bus is granted for both request and response. In a split bus, requests and responses are arbitrated separately.

For buses with centralized arbitration, the access time is proportional to the number of masters connected to the bus. The transfer latency itself is constant and relatively low, because modules are linked directly. However, the speed of transfer is limited by the bus speed, which is relatively slow.

Some modern bus interfaces like VCI, OCP, AMBA and CoreConnect allow pipeline transactions. This means that concurrently with sending the address of a read transaction, the data of previous write transaction can be sent, and the data from even earlier read transaction can be received [5, 6, 12, 13].

The main features of the analyzed SoC buses, including network topology, bus arbitration method, types of data transfer, and bus width, are given in Table 2. A brief comment concerning their properties follows.

The main limitations of buses are scalability and restricted sharing of resources between communicating entities. In general, traditional interconnects like buses, point-to-point wires, and regular topologies suffer from poor resource sharing in the time and space domains, leading to high contention or resources utilization.

With exception of Marble all analyzed buses are synchronous.

Avalon, CoreFrame and OCP define point-to-point connections. PI bus is a unilevel shared bus. AMBA, CoreConnect, Marble, and Lotterybus are hierarchical buses. SiliconBackplane is an Interconnection network, while Wishbone supports almost all topology types including point-to-point wiring, crossbar, hierarchically organized bus topology, common bus, etc.

Most of the analyzed SoC buses donXt define an arbitration mechanism, and support design of an arbiter, within a SoC, regarding to specific applicationXs requirements. Exceptions from this approach are Marble and CoreConnect that define static priority. SiliconBackplane supports the two-level arbitration, while Lotterybus a lottery tickets.

Table 2. SoC buses features overview

	Topology						Arbitration						Bus width		Transvers						
Name	Point-to-point	Ring	Unilevel shared bus	Hierarchical bus	Interconnection network	Synchronous/Asinhronous	Static priority	TDMA	Lottery	Round-robin	Token Passing	CDMA	Data bus width [bit]	Address bus width [bit]	Handshaking	Split transfer	Pipelined transfer	Burst transfer	Broadcast	Multicast	Operating frequancy
AMBA	-	-	-	×	-	S	7*	7*	7*	7*	7*	7*	8*	32	×	×	×	×	n/a	n/a	11*
Avalon	×	-	-	-	-	S	13*	13*	13*	13*	13*	13*	1-128	1-32	-	-	×	×	-	-	n/a
Core Connect	-	1*	-	1*	-	S	4*	-	-	-	-	-	9*	10*	×	×	×	×	n/a	n/a	12*
Wishbone	×	×	×	-	×	S	3*	3*	3*	3*	3*	3*	8,16,32,64	1-64	×	n/a	-	×	n/a	n/a	11*
Silicon Backplane	-	-	-	-	×	S	-	6*	-	6*	-	-	8,16,32,64	n/a	×	×	×	×	×	×	n/a
Core Frame	14*	-	-	-	-	S	3*	3*	3*	3*	3*	3*	n/a	n/a	2*	-	n/a	×	×	n/a	n/a
Marble	-	-	-	×	-	A	×	-	-	-	-	-	n/a	n/a	×	×	×	×	×	n/a	n/a
PI bus	-	-	×	-	-	S	3*	3*	3*	3*	3*	3*	1-32	1-32	×	-	×	-	-	-	n/a
OCP	×	-	-	-	-	S	-	-	-	-	-	-	n/a	n/a	×	-	×	×	×	-	n/a
VCI	n/a	n/a	n/a	n/a	n/a	S	3*	3*	3*	3*	3*	3*	n/a	n/a	×	×	×	n/a	-	n/a	n/a
Lotterybus	-	-	-	×	-	S	-	-	×	-	-	-	n/a	n/a	n/a	n/a	n/a	×	n/a	n/a	n/a

Exceptions for Table:1* Data lines shared, control lines point-to-point ring; 2* Palmbus uses handshaking, Mbus does not; 3* Application specific, arbiter can be designed regarding to the application requirements; 4* Programmable priority fairness; 5* Two level arbitration, first level TDMA, second level static priority; 6* Two level arbitration, first TDMA, second round-robin token passing; 7* Application specific except for APB which requires no arbitration; 8* For AHB and ASB bus width is 32, 64, 128 or 256 byte, for APB 8, 16 or 32 byte; 9* For PLB bus width is 32, 64, 128 or 256 byte, for OPB 8, 16 or 32 byte and for DCR 32 byte; 10* For PLB and OPB bus width is 32 byte, and for DCR 10 byte; 11* User defined operating frequency; 12* Operating frequency depending on PLB width; 13* Slave side arbitration; 14* System of buses, Palmbus and Mbus, both are point-to-point;

The considered SoC buses support various data transfer types. Almost all, with exception of Avalon, support handshaking data transfer procedures. Pipelined data transfer, excluding Wishbone, supports almost all SoC buses. Burst data transfer is not typical for PI bus, only. AMBA, Core Connect, SiliconBackplane, Marble and VCI support split transfer. SiliconBackplane, Core Frame, Marble and OCP support broadcast. SiliconBackplane supports multicast.

4 Conclusion

Complex VLSI IC design has been revolutionized by the widespread adoption of the SoC paradigm. The benefits of the SoC approaches are numerous, including improvements in system performance, cost, size, power dissipation, and design turnaround time. Many SoC designs consist of one or more IPs, designed for a single or narrow set of applications with highly characterize-able communication. As the level of chip integration continues to advances at a fast pace, the desire for efficient interconnects rapidly increase. Currently on-chip interconnections networks are mostly implemented using traditional interconnects like buses. The wide variety of buses used in SoC designs presents the major problem for reusable-design. A number of companies and standards committees have attempted to standardize buses and interfaces with mixed results. In this paper we have discussed some of the issues facing SoC designers in determining which bus architecture to use in order to provide flexible and high-bandwidth between IPs.

References

[1] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs, 2/E*. Boston: Kluwer Academic Publishers, 1999.

[2] W. Ho and T. Pinkston, "A design methodology for efficient application-specific on-chip interconnects," *IEEE Trans. On Parallel and Distributed SystemsFebruary*, vol. 17, no. 2, pp. 174–190, Feb. 2006.

[3] N. Horspool and P. Gorman, *The ASIC Handbook*. Upperside River, NJ: Prentice Hall, 2001.

[4] L. Bernini and G. D. Micheli, "Networks on chips: A new paradigm for component-based mpsoc design," in *Miltiprocessor Systems-on-Chips*, A. A. Jerraya and W. Wolf, Eds. Amsterdam: Elsevier, 2005, pp. 49–80.

[5] Core connect bus architecture. IBM Microelectronics. [Online]. Available: http://www.ibm.com/chips/products/coreconnect

[6] (1999) Arm. amba specifications v2.0. ARM. [Online]. Available: http://www.arm.com

[7] (2002, Jan.) Sonics network technical overview. Sonics.inc. [Online]. Available: http://www.sonicsinc.com

[8] K. Lahiri, S. Dey, and A. Raghunathan, "Design of communication architectures for high-performance and energy-efficient systems-on-chip," in *Multiprocessor Systems-on-Chips*, A. A. Jerraya and W. Wolf, Eds. Amsterdam: Elsevier, 2005, pp. 187–222.

[9] (1994) Draft standard omi 324: Pi-bus, rev. 0.3d, open microprocessor systems initiative. Siemens AC. Munich. [Online]. Available: http://www.cordis.lu/esprit/src/omi-home.htm

[10] W. J. Dally and B. Towel, *Principles and Practices of Interconnection Networks.* Amsterdam: Elsevier, 2004.

[11] N. Shandhag, "Reliable and efficient system-on-chip design," *IEEE Computer*, vol. 37, no. 3, pp. 42–50, Mar. 2004.

[12] J. Hennessy and C. A. A. Q. A. D. Petterson, *Computer Architecture: A Quantitative Approach*. Amsterdam: Elsevier, 2003.

[13] A. Radulescu and K. Goossens, "Communication services for networks on chip," in *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*, S. Bhattacharyya, E. Deprettere, and J. Teich, Eds. New York: Marcel Dekker Inc., 2004, pp. 193– 213.

[14] Overview of open core protocol (ocp-2001-9-26). OCP International Partnership Association. Portland, OR 97221, USA. [Online]. Available: http://www.ocpip.org

[15] C. Rowen, *Engineering the Complex SoC: Facts, Flexible Design with Configurable Processors.* Upperside River, NJ: Prentice Hall, 2004.

[16] G. Strano, S. Tiralongo, and C. Pistrito. (2006, Jan.) Cp/stbus plug-in methodology. http://ww.techoline.com/community/tech_group/com/tech_paper/37923.

[17] (2001, Apr.) Virtual component interface standard version 2 (ocb 2 2.0). VCI Alliance. [Online]. Available: http://www.vsi.org

[18] J. L. A. et all, "State-of-the-art soc communication architectures," in *Embedded System Handbook*, R. Zurawski, Ed. Boca Raton: CRC Taylor & Francis Group, 2006, pp. 20.1–20.2.

[19] J. Ayala, M. Lopez-Vellejo, D.Bertozzi, and L. Benini, "State-of-the-art soc communication architectures," in *Embedded Systems Handbook*, R. Zurawski, Ed. Boca Raton: CRC Press, 2006, pp. 20.1–20.22.

[20] (2001) Arm. amba multi-layer ahb overview. ARM. [Online]. Available: http://www.arm.com

[21] (2003) Arm. amba axi protocol specifications. ARM. [Online]. Available: http://www.arm.com

[22] (2003, July) Altera, avalon bus specification: Reference manual. Altera Corporation. [Online]. Available: http://www.altera.com

[23] (2005, Apr.) Altera, avalon interface specification. Altera corporation. [Online]. Available: http://www.altera.com

[24] G. Pelissier, R. Hersemeule, G. Cambon, L. Torres, and M. Robert. (2006, Jan.) Bus analysis and performance evaluation on a soc platform at the system level design. [Online]. Available: http://www.ra.informatik.uni.stutgart.de/∼pricopin/noc03/paper_44.pdf

[25] (2002, Sept.) Wishbone system-on-chip (soc) interconnection architecture for portable ip cores, revision: B.3. [Online]. Available: http://www.opencores.org/projects.cgi/web/wishbone/wishbone

[26] (2002, Jan.) Overview of the coreframe architecture. Palmchip Corporation. [Online]. Available: http://www.palmchip.com

[27] Palmchip, overview of coreframe architecture, white paper. Palmchip Corporation. [Online]. Available: http://www.palmchip.com

[28] B. C. P. Corporation. A bus architecture for system-on-chip designs. Palmchip Corporation. [Online]. Available: http://www.palmchip.com

[29] W. Bainbridge, “Asynchronous system-on-chip interconnect,” Ph.D. dissertation, University of Manchester, Mar. 2003.

[30] W. Bainbridge and S. Furber. Asynchronous macrocell interconnect using marble, technical report. [Online]. Available: http://www.cs.manchester.ac.uk/apt/projects/interconnect/