On-chip Interconnect Schemes for Reconfigurable System-on-Chip

Andy S. Lee, Neil W. Bergmann.
School of ITEE, The University of Queensland, Brisbane Australia
{andy, n.bergmann} @itee.uq.edu.au

ABSTRACT

On-chip communication architectures can have a great influence on the speed and area of System-on-Chip designs, and this influence is expected to be even more pronounced on reconfigurable System-on-Chip (rSoC) designs. To date, little research has been conducted on the performance implications of different on-chip communication architectures for rSoC designs. This paper motivates the need for such research and analyses current and proposed interconnect technologies for rSoC design. The paper also describes work in progress on implementation of a simple serial bus and a packet-switched network, as well as a methodology for quantitatively evaluating the performance of these interconnection structures in comparison to conventional buses.

Keywords: FPGAs, Reconfigurable Logic, System-on-Chip

1. INTRODUCTION

System-on-chip (SoC) technology has evolved as the predominant circuit design methodology for custom ASICs. SoC technology moves design from the circuit level to the system level, concentrating on the selection of appropriate pre-designed IP Blocks, and their interconnection into a complete system.

However, modern ASIC design and fabrication are expensive. Design tools may cost many hundreds of thousands of dollars, while tooling and mask costs for large SoC designs now approach $1million. For low volume applications, and especially for research and development projects in universities, reconfigurable System-on-Chip (rSoC) technology is more cost effective. Like conventional SoC design, rSoC involves the assembly of predefined IP blocks (such as processors and peripherals) and their interconnection. However, here the fabrication technology uses mega-gate FPGAs, rather than custom ASICs. First generation commercial rSoC products are now offered by most FPGA vendors.

rSoC has re-programmability that allows designers to implement and test theories in real hardware many times on a single device. The turnaround time is also relatively short, allowing almost instantaneous hardware implementation, with the lengthy delay for silicon fabrication eliminated from the design cycle.

However, simply mapping ASIC designs on to reprogrammable devices will not yield efficient and optimized results. Due to the underlying architectural differences between reconfigurable devices and ASICs[1], such as much more constrained wiring channels, conventional SoC design methods may not always be appropriate for rSoC.

There is, therefore a need to revisit conventional SoC design techniques, and analyze their applicability for rSoC, and also to examine new rSoC design techniques that can make good use of the special characteristics of rSoC.

A key component in any SoC design is the interconnection fabric that is used for inter-module communication. The most common interconnection strategy is a parallel system bus. A number of different “standard” buses are already being used in custom SoC designs. Other interconnection strategies for SoC have also been proposed, such as cross-bar switches, packet-switch network, and on-chip LANs.

For rSoC, a number of different bus “standards” have also been proposed. It is not clear that these bus structures, which have been used more or less unchanged from custom SoC designs, are the most appropriate for rSoC. To date, little work has been done on alternate rSoC interconnection networks.

This paper presents our early work on evaluating the applicability of existing and proposed interconnection structures for rSoC designs. We first examine existing rSoC bus structures, viz. the IBM CoreConnect bus used by Xilinx, the ARM AMBA bus used by Altera, and the “Open Source” Wishbone standard. We compare these buses and evaluate their applicability to rSoC designs.

Next we look at VSIA's Virtual Component Interface, which aims to provide a "virtual" bus standard that combined with appropriate bus wrappers allows IP-block re-use across different busses.

Finally we describe some alternate interconnection structures that may be better suited to rSoC technology, such as serial buses and communication-network approaches and outline our future methodology for quantitatively evaluating the relative performance of these different approaches.

2. THE BUS ARCHITECTURE

The bus architecture is one of the most popular integration choices for SoC designs today. It is derived from traditional computer buses that can be seen in desktop workstations. Workstation buses are organized in a hierarchical fashion with each descending level less frequently accessed by the central processing unit and requiring less bandwidth. This allows the faster devices to work at their peak performance, and for slower components to be interconnected at lower cost.

A common feature of the bus architecture is its flexibility and extensibility. Components from different vendors can be used to build up a system as long as they adhere to the standard design specification. Having a large selection of compatible IP blocks is an attractive attribute for system integrators. This has encouraged standards such as the CoreConnect and AMBA busses for SoC development, which we describe in more detail below.

2.1. IBM CoreConnect Bus

The IBM CoreConnect [2] bus is a hierarchical, multi-bus based architecture. It provides three different buses for connecting IP cores. A typical system, shown in Figure 1, uses a high speed bus for connecting processors and controllers, a peripheral bus for lower bandwidth cores and a data control register bus.

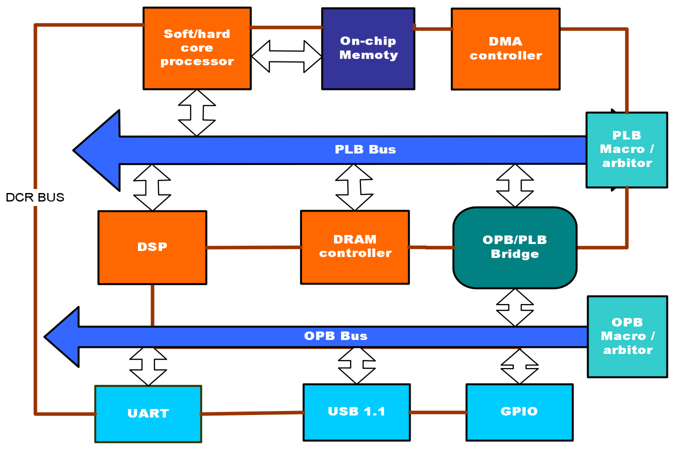

Figure 1: IBM CoreConnect Bus System

- **Processor Local Bus (PLB)**

The PLB is used for high speed, high bandwidth and low latency communication, such as between the processors and DMA controllers. Separate address, read and write data bus (extendable to 256 bits) allows simultaneous transfer requests and a maximum of two transfers per cycle. The system is fully synchronous, supporting split transactions and multiple master/slaves.

A typical system, shown in Figure 2, contains three main components: the PLB bus master interface, PLB Arbiter and the PLB slave interface. All PLB bus masters connects to the PLB Arbiter via separate address, read and write data buses. The Arbiter provides a programmable arbitration scheme and provides central control between all masters and slaves.

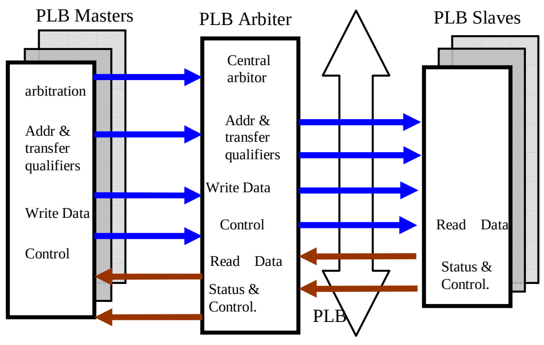

Figure 2: Example of a PLB Bus

- **On-chip Peripheral Bus (OPB)**

The OPB is used mainly for low data-bandwidth peripherals to alleviate traffic load on the PLB bus. UARTs, GPIO, and timers are some of the typical applications used on the OPB. The OPB supports a fully synchronous protocol with separate 32bit address and data busses.

Communication between the PLB and OPB devices is handled by an OPB bridge. The OPB Bridge acts as a slave device on the PLB and as a master device on the OPB. It is capable of dynamic bus sizing and split transactions, allowing devices with different data widths to communicate effectively.

- **Device Control Register Bus (DCR)**

Lower performance and configurable registers are typically read and written through the Device Control Register bus. The DCR bus provides a maximum throughput of one read or write transfer per every two cycle. The DCR bus removes configuration registers from the memory address map, reduces loading and improves bandwidth of the processor local bus. It also allows data to be transferred among OPB peripherals to occur independently from, and concurrent with, data transfers between processor and other PLB devices.

2.2. ARM AMBA

The Advanced Microcontroller Bus Architecture (AMBA)[3] from ARM is another on-chip communication standard based on a traditional bus architecture. There are two levels of hierarchy with a higher bandwidth Advanced System Bus for processors and more demanding controllers and a narrower Advanced Peripheral Bus. A revision of the specification in 1999 also added the Advanced High-performance Bus to allow more complex protocols and wider buses. Figure 3 shows a typical AMBA bus system:

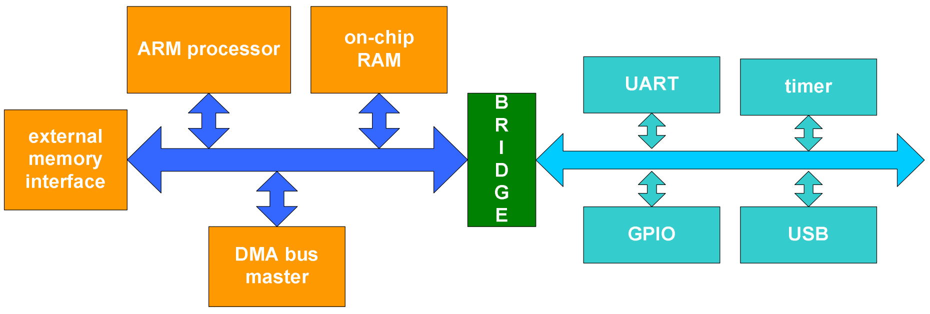

Figure 3: A typical AMBA system

- **Advanced High-performance Bus**

  The AHB is designed for wide bandwidth devices and it is similar to the CoreConnect PLB. Address, read and write buses are decoupled from each other and the specification recommends a minimum of 32/64 bit data operation, extendable to 256 bits. Multiple master/slave operation is supported and controlled by a central arbiter. Advanced bus protocols such as pipe-lined operation, burst and split transaction are supported.

- **Advanced System Bus**

  ASB bus is available for applications where high performance features of the AHB are not required. ASB supports 16/32 bit data bus operation and multiple master/slaves.

- **Advanced Peripheral Bus**

  Low bandwidth peripherals typically reside on the APB. This is similar to the CoreConnect OPB bus.

2.3. WISHBONE

The WISHBONE [4] SoC Interconnect architecture was developed by Silicore Corporation and is supported by the Opencores organization as a public domain license project that provides freely available designs and cores. The WISHBONE standard takes a simpler approach of having a single level bus. The data width is defined from 8 bit to 64 bit separate read/write ports with possible extension to 256 bits. The address bus supports up to 64 bits. Single read/write, block read/write and read-modify-write cycles are support.

The major difference that the WISHBONE architecture brings is the ability to provide customized interconnection schemes. System integrators choose among four types of interconnection between IP cores:

- Point-to-point - the simplest way of connecting two components: A master connects directly to a slave.
- Share Bus - the share bus architecture is much like traditional PC backplane bus. The share medium requires additional logic such as arbiter and address comparator.
- Data flow - For pipeline processing, the cores are connected in a chain in the direction the data is processed. Each IP core must have a master and a slave interface and this allows pipeline parallelism as shown in Figure 4.

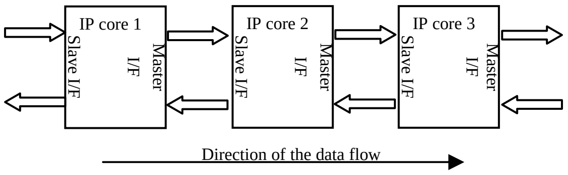

Figure 4: Wishbone Data Flow connection

- Cross bar switch - This interconnection scheme allows one or more master devices to communicate simultaneously with others, provided that they are not targeting the same slave device. This scheme offers much higher bandwidth, but comes at the cost of using more interconnection logic and routing resources.

2.4. Comparison of bus standards

In principle all three buses are similar to traditional backplane buses, with the only major difference being the number of wires utilized. Separate read, write data and address lines are an affordable option on-chip, compared to backplane buses where increased connector and wire cost is problematic.

Both CoreConnect and AMBA busses are quite similar in terms of features and structure. Both offer a two-level bus hierarchy for data communication and provide similar sets of protocols. In addition, IBM offers a separate DCR bus to reduce performance limiting status and control read/write transfers on the PLB.

Both IBM and ARM have provided full specifications with relatively strict guidelines. This will ensure consistent and compatible designs among different IP designers (for the same specification), but offers little choice in programmability. The rich set of features might be overkill for simple embedded systems, resulting in wasted silicon area.

The WISHBONE architecture provides a simpler approach, allowing designers to choose and customize a system. The customizable interconnection allows the system integrators to choose the best scheme for a given system, which might in term produce better performance with less interfacing logic. However, the single level bus approach might present problems with the co-existence of slow and fast components on a single bus.

A table of summary of different bus standard is shown below:

BUS standard	***bus type***	**data bandwidth**	**address width**	**features**
PCI	Single level, shared-medium	32	32, multiplexed	Multiple masters, fully-synchronize. Single, burst, delayed transaction. Pipelined operation
ARM AMBA	Hierarchical, shared-medium	Separate read/write data bus		Fully-synchronous. Single, burst, split transaction. Pipelined operation
	APB	8/16/32	32, separate	Single master
	ASB	8/16/32	32, separate	Multi master
	AHB	32/64/128	32, separate	Multi master
IBM CORECONNECT	Hierarchical, shared-medium	Separate read/write data bus		Fully-synchronous. Single, burst, split transaction. Deep Pipelined operation.
	OPB	32	32, separate	
	PLB	32/64/128/256	64, separate	
	DCR	32	10	Daisy chained,
Wishbone	Single level bus	Separate read/write data bus		Fully-synchronous, single, block, RMW cycle. Pipelined.
	Wishbone bus	8/16/32/64	64, separate	Point-to-point/data flow, crossbar/shared-medium

Table 1: Table of bus comparisons

2.5. Other I/O buses for rSoC

There is another category of buses that are often used on desktop workstations: the Input/Output peripheral serial buses (such as USB). Unlike parallel buses such as PCI, where multiple bytes are transferred over a single clock cycle, serial buses often requires several clocks cycles to complete a single transaction, resulting in much smaller bandwidth. However, serial buses are inexpensive to implement, therefore it is much more appropriate for applications that require to communicate over larger distances. Recent advances have also allowed the serial protocol to provide relatively high transfer rates (USB 2.0 can transfer at maximum of 480Mbps), making it a viable option to consider on reconfigurable SoCs.

2.6. Bus Wrappers

The lack of compatibility between different bus standards has limited the idea of IP core reuse. CoreConnect, ARMA, and Wishbone are just a few out of a growing list of commercial specifications, each with its own share of the market. Multiple standards reduce the capacity for IP reuse.

IP core designers often choose to concentrate on designs for a single standard, which is often incompatible with other standards. This means the IP cores can only be used by the system integrators that use the same bus specification, effectively reducing the available market opportunity. Vice versa, system integrators are limited in their choice of IP blocks by their choice of system bus. This problem has prompted the development of the bus wrapper solution.

The Virtual Socket Interface Alliance has developed a generic bus interface that is aimed to separate behavioral logic design from communication logic. The VCI standard, Virtual Component Interface, is a “bus wrapper” that allows easy interoperability between different bus standards on the market [5]. IP cores are designed to a set of standard bus interface signals that communicate to the VCI interface. A selection of VCI bus wrappers are then available for system integrators to convert from the VCI interface to their particular industry standard bus.

Similar work is also being carried out in a separate project within our group at University of Queensland [6]. This projects aims to develop an interface logic generation methodology which can be used across different communications architectures (not just parallel buses).

3. "COMMUNICATION BASED" ARCHITECTURES

The silicon industry is moving towards sub 0.1 micron technology in 2004 [7] allowing complex and larger SoCs. As the size of the chip grows relative to the size of the IP cores, on-chip buses will become relatively slow[8]. Larger chip sizes will also allow several complex processing units to exist in a single system, pushing the on-chip communication to exhibit a multi-processor network behavior. Thus another direction for SoC interconnection network research favours communication based architectures, borrowing ideas from the telecommunications and computer network areas.

A brief description of the OSI reference model will be presented below, followed by a look at current developments in this field.

3.1. OSI Reference Model

The communication network designs utilize a layered approach, derived from the well-known Open System Interconnect Reference Model [9]. The OSI model is constructed of layered protocols. Each layer is created on different abstraction level, providing well-defined functions within each layer. Each lower layer provides services to the next higher layer.

The layered approach of the OSI model provides a way of decomposing design issues. Designers can implement protocols within each layer as required. However, the complete OSI model is often deemed to be too complicated and often some adjacent layers are combined in the design process. Nevertheless, experience with the OSI model suggests that a layered communications approach can provide system-level advantages. One SoC example is given in the next section.

3.2. Network on-chip

Researchers at Stanford University have presented a SoC interconnect architecture that is similar to a tightly coupled multi-processor system [10]. The interconnect scheme is based on a 2-dimensional mesh topology, where each IP core is designed to fit into a rectangular area as a single tile. Each tile interconnects to one another in the mesh topology, allowing a structured network as shown in Figure 5. Inter-tile communication follows a packet/message based paradigm, with data packets routed through the network by a simple router interface which resides on each tile.

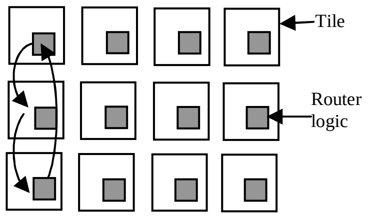

Figure 5: Network-On-chip overview

The advantage of this physically-structured design is to provide optimized and well-controlled electrical parameters. The interconnecting wires have a fixed distance between all tiles, producing predictable levels of noise and lower crosstalk by separating them from intra-tile wiring.

A simple router is implemented in each tile, with five input/output controllers: one controller for ports on each side and one input/output controller for the tile logic. This router implements functions at similar to the network and the transport layer, which routes flits (300 bit packet) in and out of the network and employs a virtual flow control mechanism[10].

4. ANALYSIS OF EXISTING SOC ARCHITECTURES

The wide acceptance of the bus architecture in the industry can be seen by the number of different bus specifications. This is perhaps due to the fact that the bus architecture is easy to adopt and well-known among the computer industry. Buses are also relatively inexpensive to implement. However, wide data paths buses are known for their power inefficiency with studies showing that wires accounts for 40-50% of the total energy consumption on-chip[11].

Another common criticism is the lack of scalability. This is common among all shared-medium type network channels: as the number of components increases the effective bandwidth is shared. One solution employed by many high-performance buses is to increase data width, but this inevitably leads to an increase in power consumption. Whilst wide buses are appropriate to the large available wiring resources on a custom ASIC, they are much less appropriate to wiring-constrained rSoC systems [12].

Network-on-chip presents a structure that models multiprocessor systems. By using a mesh topology, total system bandwidth increases when a new IP core is added, as number of paths for data transmission increases. This architecture is complimented by use of data packets, which allow concurrent communication among exiting IPs for better utilization of the bandwidth. The use of point-to-point connection between cores also eliminates the need for different interfaces for different bus widths, as needed by hierarchical buses topology.

One major disadvantage of the on-chip network is silicon usage. In order to guarantee more predictable electrical and performance (in-between cores) parameters, the system specifies fixed tile areas to allocate IP cores. This is easy to achieve for systems where multiprocessor or duplicated computational units are used, but inefficient for general systems. IP cores differ greatly in logic usage, and simple controllers will leads to unused silicon area within a fixed-area tile. This is particularly of concern in resource-limited rSoC systems.

Router design is another cost concern for the network architecture. Bus architectures cost little in terms of silicon area and employ a central arbiter. On the other hand, mesh networks need to implement distributed arbitration logic, also known as routers, in each tile space. The complexity of the router and the number of system components increases silicon usage in comparison to the bus architecture.

5. PROPOSED METHODOLOGY

Choosing the most appropriate interconnection scheme for a custom SoC is a major design issue. We suggest that interconnect scheme design may be even more important for rSoC designs, as interconnects and wirings have a proportionally larger effect on area and speed performance on reconfigurable devices. We have therefore started research into evaluating the relative performance of different interconnection strategies. This research is just at its beginning, but some early ideas are presented below.

Our methodology is composed of three parts. Firstly different architectures will be selected and evaluated as possible on-chip interconnection schemes. Secondly, the selected architectures will be logically divided into interface layer and interconnect physical layer for implementation. Finally, we plan to evaluate the performance of the implemented architectures with different system parameters and criteria's.

5.1. Architecture selection

Our study of the different interconnection technologies has found that interconnect schemes can be divided into three main categories: serial schemes, parallel buses and communication-based networks. The following list of the different categories present some of the possible options for on-chip communication implementation:

a) Serial buses:

I^2C –	A two wire serial bus system with transfer speeds up 400Kbit per second. The I^2C protocol uses one wire for data/address and one serial clock line for synchronizing half duplex data transmission. Multiple masters and slaves are supported on a shared-medium bus.
Dallas 1-wire –	Dallas 1-wire bus uses a single data/address line for communication. A single master is usually implemented in a system and the slave devices are self-timed for synchronized transmission. Half duplex communication of up to 115.3 Kbits/s is possible on a master/slave bus.
SPI –	4 wire interface. Two data and two control lines are implemented with separate read, write, serial clock and slave select line. Full duplex operation is allowed with the initializing component being

	the master, which also generates the serial clock and slave select signal. In a single-Master-multi-slave system, n select lines are required for n number of slaves. Multi-master system can be implemented. No error correction or flow control is specified .
USB -	Universal Serial Bus uses a 4 wire system, but essentially only uses a single communication channel. Two wires are used for power and ground with a differential signaling pair for data transmission. Speeds of up to 480 KB/s can be achieved.
IEEE1394-	Also known as Firewire, is a high speed bus that uses 2 differential signaling pairs for data transmission. Speeds of up to 400 KB/s can be achieved.

b) Parallel buses:

ISA-	Computer backplane bus with 16 bit data, 20 bit address and other sideband bus signals. Synchronous operation at 8 MHz.
PCI -	Peripheral Component Interconnect uses 32/64 bit multiplex data/address line. Synchronous operation at 33/66 MHz.
SCSI -	Peripheral bus interface for I/O devices with multiplexed 16 bit data/address line. Supports up to 320MB/s operation using LVD technology (low voltage differential signaling).

c) Communication network architecture:

Network and protocols:-

Ethernet protocol-	Data link layer protocol with Carrier Sense Multiple Access/Collision Detection (CSMA/CD).
TCP/IP protocol –	Transport/network layer protocol
ATM network -	Packet-switched Asynchronous Transfer Mode network.

Our research plans to investigate possible candidates from each category for on-chip communication. From the three categories above, only parallel buses have already been implemented and used extensively for rSoC. On the other hand, there has been little use of both serial and communication network architectures for SoC. Therefore, our research will implement new schemes based on serial and communication-based design for testing and evaluation in comparison to existing bus-based schemes.

5.2. Interface and interconnect design

In order to achieve modularity in the interface and interconnect design, we separate the IP core's communication design aspects from its functionality. This will provide easier integration with other interconnect structures and better IP reuse. The logical design of the interconnect scheme will be divided into two main parts: The communication interface layer and the physical interconnect (as shown in Figure 6 below)

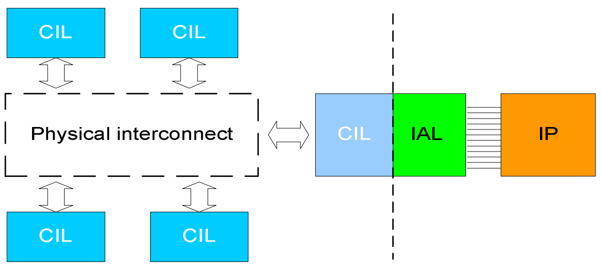

Figure 6: Proposed methodology system overview

The interface layer will be implemented as the Communication Interface Logic (CIL). The CIL should provide reliable data communication with other CILs. Ideally, only one type of CIL will be required for a particular scheme. An Interface Adapter Logic (IAL) will automatically generate the required logic to match a generic IP block to a particular CIL.

Each CIL acts as an edge node of the network – the rest of the network (such as routers, arbiters, wires and switches) is lumped together as the "Physical Interconnect" box in Figure 6.

5.3. System analysis

Ryu and Mooney's study of different multiprocessor systems with different bus topologies has shown that the performance of a particular topology is very much application dependent in ASIC design[13]. Thus, this project aims to show such dependency in rSoC designs.

Our investigation involves quantitative analysis of the architectures in particular rSoC applications. The analysis will be based on cost and performance metrics relevant to rSoC platform design. Cost is measured in terms of the logic resources used and power consumption. The performance can be calculated in terms of bandwidth, throughput and latency.

The performance analysis will be applied to rSoCs with different system performance requirements. For example, in a real-time system, the parameter of latency might outweigh the maximum throughput.

The initial analysis will be performed using standard EDA development tools. The synthesis tools will provide logic resource usage estimation while hardware simulators can be used to provide preliminary performance measurements.

One of the advantages of rSoC is that designs can be implemented and tested in real hardware. Tests will also be conducted on the reconfigurable devices to investigate the performance measures under real world conditions. Different system applications will be applied to test how different architectures perform under different requirements.

6. DESIGN IMPLEMENTATION

6.1. Design platform

The Microblaze softcore processor [14] will be used as the main processor for integration. This processor is supported by Xilinx ISE EDA tools, and we will use the Xilinx Virtex II FPGA device. The Modelsim simulation program will be used for design verification and simulation.

The IBM Core Connect bus, provided as part of Xilinx's Microblaze development system, will be used as the example bus architecture.

6.2. Architecture implementation: Serial Bus

It is proposed that a modified version Dallas One-wire bus [15] is implemented as an on-chip serial bus. The One-wire architecture presents a simple and reliable bus and protocol that is able transfers data, address and control information all over a single wire. We have chosen this bus for our initial exploration because of its simplicity and because its single wire provides a significant contrast to wide parallel buses.

The implementation will initially follow the protocol defined by the one-wire standards. Four layers, physical, link, network and transport layer function will be implemented with some modification. The physical layer structure will be a simple shared bus with slaves coupled on to the one-wire bus. The link layer protocol provides the basic communication functions. The network layer implements the ROM commands used to transfer addresses. Finally the transport layer functions are responsible for data transfers.

During the first phase, a single master/slave system will be implemented followed by a modification to support multiple masters, which is a feature not specified in the original one-wire bus standard.

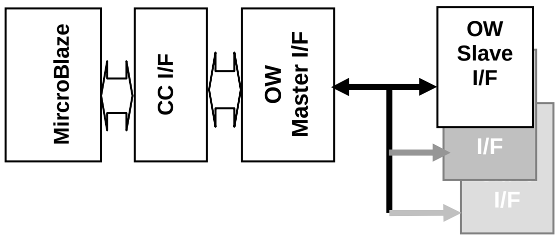

Figure 7: Modified 1-Wire bus with single master

In the first phase of the design, a system supporting a single master and multiple slaves will be implemented, as shown in Figure 7. The master interface will be connected to the Microblaze softcore processor using the IBM CoreConnect specification. This will allowed for more rapid development, as the Microblaze system is designed to the CoreConnect specification. The slave logic will interface with both the master device and the IP core that it connects to.

The master interface will be able to address and initialize transfers to a slave device connected on the one-wire bus. The basic reset, presence pulse detect, read-one, write-one time slots will be implemented as well as the more complex ROM reading/writing functionalities. The ROM function will be used for data transfer from the slave device to the microcontroller.

The slave interfaces listen on the one-wire bus and send out the presence pulse when a reset pulse is issued by the master. When addressed by the master device, the slave will then respond to the commands issued and performs required operations.

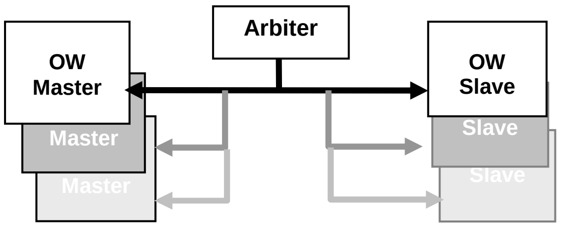

Figure 8: Modified 1-Wire bus with multiple masters

In the second phase of the one-wire protocol design, support for multiple slaves and masters will be implemented. A simple arbiter will be implemented so that the effect of different arbitration schemes can be analyzed.

Several modifications to the original specification will be made to improve the performance for on-chip rSoC communication. Firstly, the speed of the protocol will be modified to operate at higher speeds (depending on the final system speed) with higher on-chip clock frequency and better signal integrity.

Secondly, the ROM address bits will be reduced to eliminate overheads. As the original intentions for 64 bit address is for identification of unique devices, such measure is not required for on-chip addressing.

6.3. Architecture Implementation: Packet-Switched Network

ATM networks have high performance characteristics that are suited to high speed applications such as teleconferencing. ATM includes a connection-orientated protocol aimed to provide guaranteed bandwidth. However, complex mechanisms are required for achieving QoS. Therefore we plan to design and implement a modified "ATM-like" packet-switched network for on-chip communication.

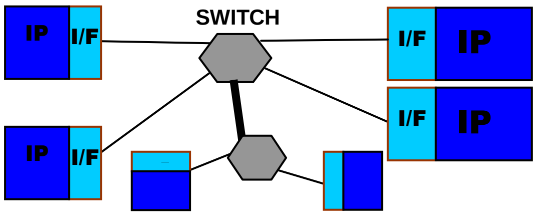

Figure 9: ATM-like packet-switched network

The ATM-like interface should implement simple flow control mechanisms that will allow the co-existence of slow and fast system components. Switch/Routing control logic should be used to connect the different components, implementing a packet-switched architecture. Finally, a protocol should implement establishment of virtual circuits to provide guaranteed bandwidth and QoS.

The packet-switched interface will be setup to test for different topologies. The serial bus system described in the last section will be used to form the base of this packet-switched scheme with initial goal of designing a network with a two switches each supporting up to six connections.

Alternative consideration

Although the one-wire bus has been chosen at present for the simplicity of the design, the suitability will be reviewed once the first stage of the design is completed. Alternative studies will also be conducted into the feasibility of implementing faster serial protocols such as IEEE 1394 and USB standard. Both standards have much more complex protocol layers which will cost more in terms of logic fabric on the FPGAs.

6.4. System analysis

As mentioned in section 5.3, the major costs are in terms of area, power and performance:

*Area Cost***:** The area cost will be measured in terms of number of CLB slices used. The EDA tools provided by the Xilinx ISE development kit will provide an accurate report of the resources that will be used to implement a rSoC design on the Virtex II FPGA device.

Power Cost: Power estimation can be calculated using commercial software programs. Programs such as XPower from Xilinx use post-route netlist and physical constraint files to provide power estimation of FPGA designs.

Performance analysis: Simulation and validation of the design will be performed using Modelsim HDL simulation software. This will provide initial results that can be used to calculate the performance of the system.

Hardware performance analysis will also be conducted after the system is implemented into the target FPGA device. A test vector generating hardware will be used to provide input to the system. Several hardware counters can be implemented into the system to record the communication activities. The output from the system will then be used to calculate the latency and throughput of the system.

Different system will be implemented using the different interconnect schemes: the One-wire Bus, packet-switched network and the IBM CoreConnect bus. In particular, the real-time system and computational-intensive systems will be used to benchmark the interconnect schemes.

7. CONCLUSION

We have presented our initial motivation and methodology for investigating efficient on-chip architectures for rSoC. Although our work is still at an early stage, we note the following outcomes to date:

- SoC interconnection fabrics are not necessarily the most appropriate for rSoC systems, because of the relatively higher cost of communication in rSoC systems.
- There has been little research into alternative rSoC interconnection architectures.
- Use of alternate interconnection strategies will depend on the availability of generic IP block designs.
- Serial busses provide a potentially attractive interconnection fabric for low speed peripherals.
- Packet-based connection networks provide a more flexible and extensible communications fabric for complex multi-processor systems, but the cost of routers and switches may be problematic.
- More research, design and testing is planned to provide quantitative results about the relative communication network performance.

REFERENCES

[1] D. May, "Advanced High-level HDL Design Techniques for Programmable Logic," [online] 1999, www.synopsis.com (Accessed: 22nd-March 2003).

[2] IBM Corporation, "The Coreconnect Bus Architecture," [online] 1999, www.chips.ibm.com/products/coreconnect (Accessed: 2002).

[3] ARM Corporation, "AMBA specification Rev. 2.0," [online] 1999, www.arm.com (Accessed: 6 June 2002).

[4] Silicore Corporation, "WISHBONE System-on-Chip Interconnect Architecture for Portable IP Cores," [online] 2001, www.silicore.net/wishbone.htm (Accessed: March 2003).

[5] VSIA Alliance, "Architecture document," [online] 1997, www.vsi.org (Accessed: 4 April 2003).

[6] T. Lee and N. Bergmann, "An Interface Methodology for Retargettable FPGA Peripherals," in *Engineering of Reconfigurable System and Algorithms*, Les Vegas, Nevada, 2003, pp. to be published.

[7] International SEMATECH, "International Technology Roadmap for Semiconductors 2002 Update," [online] 2002, http://public.itrs.net/Files/2002Update/2002Update.pdf (Accessed: 21st April 2003).

[8] H. Chen, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, *Surviving the SOC Revolution: a guide to platform-based design*, vol. 1. Massachusetts: Kluwer Academic Publishers, 1999.

[9] A. S. Tanenbaum, *Computer Networks*, vol. 1, 3rd ed. Upper Saddke River, N.J.: Prentice Hall, 1996.

[10] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th conference on Design automation*, Las Vegas, Nevada, US, 2001, pp. 648--689.

[11] D. Liu and C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips'," *IEEE Journal of Solid-State Circuits*, vol. Vol 29, pp. 663-669, 1994.

[12] J. Oldfield and R. Dorf, "Introduction of FPGA architecture," in *Field-programmable gate arrays : reconfigurable logic for rapid prototyping and implementation of digital systems*. New York: John Wiley & Sons, 1993.

[13] K. Ryu, E. Shin, and V. Mooney, "A Comparison of Five Different Multiprocessor SoC Bus Architectures," in *Proceedings of the EUROMICRO Symposium on Digital Systems Design*, Warsaw, Poland, 2001, pp. 202--209.

[14] Xilinx Corporation, "Microblaze Soft processor," [online] 2003, http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=microblaze (Accessed: July 2003).

[15] Maxim IC, "Book of iButton Standard," [online] 2002, www.ibutton.com (Accessed: June 2002).