Networks on Chips: Scalable Interconnects for Future Systems on Chips

Muhammad Ali, Michael Welzl, Martin Zwicknagl
Institute of Computer Science
University of Innsbruck, Austria
{Muhammad.Ali, Michael.Welzl, Martin.Zwicknagl}@uibk.ac.at

Abstract

According to the International Technology Roadmap for Semiconductors (ITRS), before the end of this decade we will be entering the era of a billion transistors on a single chip. It is being stated that soon we will have a chip of 50-100 nm comprising around 4 billion transistors operating at a frequency of 10 Ghz. Such a development means that in the near future we probably have devices with such complex functions ranging from mere mobile phones to mobile devices controlling satellite functions. But developing such kind of chips is not an easy task as the number of transistors increases on-chip, and so does the complexity of integrating them. Today's SoCs use shared or dedicated buses to interconnect the communicating on-chip resources. However, these buses are not scalable beyond a certain limit. In this case, the current interconnect infrastructure will become a bottleneck for the development of billion transistor chips. Hence, in this tutorial, we will try to highlight a new design paradigm that has been proposed to counter the inefficiency of buses in future SoCs. This new design paradigm has been termed with a variety of titles, but the most common and agreed upon one is Networks on Chips (NoCs). We will show that how this paradigm shift from ordinary buses to networks on chips can make the kind of SoCs mentioned above very much possible.

Keywords

SoC, Network on Chips, Design Challenges

1. Introduction

Chip integration has reached a stage where a complete system can be placed in a single chip. When we say complete system, we mean all the required ingredients that make up a specialized kind of application on a single silicon substrate. This integration has been made possible because of the rapid developments in the field of VLSI designs; this is primarily used in embedded systems.

Thus, in simple terms an SoC can be defined as "an IC, designed by stitching together multiple stand-alone VLSI designs to provide full functionality for an application [1]." While designing an SoC, a vendor may use a library of cores designed by external designers in addition to using cores from in-house libraries. Cores are basically pre-designed models of complex functions termed as Intellectual Property Blocks (IP Blocks), Virtual Components (VC) or simply micros. Since the design of an SoC comprises cores from different sources /vendors, we can say that an SoC is completely heterogeneous, and that is one of the key issues which complicates its design process.

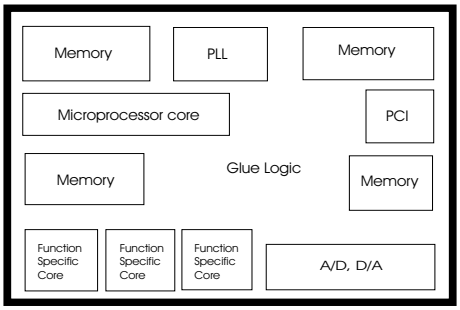

Figure 1. SoC – [1]

A generalized form of today's SoC architectures is depicted in figure 1. This figure shows the common components used in current SoCs; SRAMS, DRAMS, Flash memory, ROM, DSPs, 2D/3D graphics, and interface cores such as PCI, USB and UART. It should be noted that all these components may belong to different libraries of cores and may belong to different vendors. Also, their organization on the chip depends upon the application they are designed for – Application Specific Integrated Circuit (ASIC). A few examples of today's core based SoCs include GSM mobile phones, single chip digital/videocams, GPS controllers, smart pager ASICs etc.

However, the present SoC architecture doesn't suffice for the future needs, particularly in the terms of their interconnect design due to their poor scalability and inefficiency for

handling large number of partners (we will elaborate on this in section 2). Hence, from here we move on toward our actual topic of discussion, that is, Network on chips or more commonly called NoCs.

A NoC is percieved as a collection of computational, storage and I/O resources on-chip that are connected with each other via a network of routers or switches instead of being connected with point to point wires. These resources communicate with each other using data packets that are routed through the network in the same manner as is done in traditional networks [2]. It is clear from the definition that we need to employ highly sophisticated and researched methodologies from traditional computer networks and implement them on chip. *But why?* In order to elaborate on this question, we have to explore the motivating factors that are compelling the researchers and designers to move toward the adoption of NoC architectures for future SoCs.

The area of NoC is still in its infancy, which is one of the reasons why there are various names for the same thing; some call it on-chip networks, some networks on silicon, but the majority agrees upon "Networks on Chips" (NoCs). However, we will be using these terminologies interchangeably throughout our tutorial.

2 Motivations

As projected by ITRS [3], around four billion transistors will be accommodated by the end of this decade. Although it sounds incredible, a number of factors are posing hindrance to achieve billion transistor chip in future. In the following we discuss some of the issues that need to be overcome before we can have a real chip with billions of transistors.

2.1 Poor scalability of standard buses

The primary interconnection mechanism behind today's SoCs are shared buses which help to time-share wires among the communicating partners and lead to reduction of I/O pins in cores, hence leading to a simplified wiring scheme. Previously, direct pin connections were used to connect various cores on a chip; this lead to a large number of pins for each core. Moreover, as the number of cores on-chip increased, so did the pins, thus, leading large routing time and area, and unpredictable delays in signals and signal quality. To simplify the structure, buses were introduced which proved to be a better solution than their predecessors in terms of reduced signal delays and signal quality, and controlled routing time. However, it has been observed that buses cannot be shared beyond 5 - 10 partners, hence, making scalability of the communication paradigm in SoCs a major concern[4].

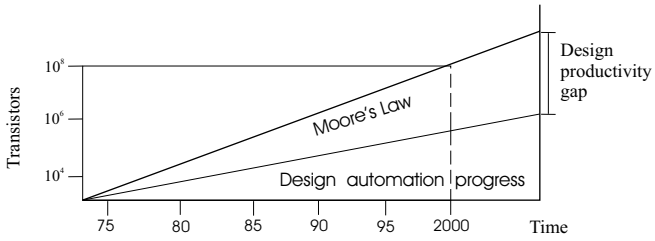

Figure 2. Moore's Law

2.2 Design productivity gap

It was in 1965 that Gordon Moore, co-founder of Intel published his all-famous paper in which he predicted that the capacity of integrated circuits will be doubled every 18 - 20 months (also called Moore's Law) [5]. It has been observed over the past years that current technology is not keeping pace with Moore's prediction resulting in a "design productivity gap" which is increasing at a pace of approximately 20% every year. This effect is shown in figure 2. This design productivity gap is not only because of more gates, and functionality and testability of the chips which were the only issues in the beginning, but many other factors like wire delay, power management, embedded software, more design choices, and signal integrity which are making the entire design process more time consuming and complex. In order to cope with the productivity gap, we need exponentially growing design teams or/and design time to design and implement systems which fit into a single IC; this is very unrealistic and rather impractical.

2.3 Difficult to maintain global synchrony

One of the major problems of growing chips is the global clock. It is becoming increasingly difficult to synchronize the clock signal traveling across the entire chip. This, in turn, is not only increasing the clock skew problem but also affecting the power consumption which is reaching unacceptable limits. One remedy of the problem is to adopt "Globally Asynchronous and Locally Synchronous (GALS)" paradigm [6]. However, in such a case, there remains no coordination among the on-chip communicating partners, hence, making chip a collection of distributed system.

2.4 Heterogeneity

A significant characteristic of SoCs is heterogeneity – components from different vendors lay on the same chip. Following the prediction of ITRS that the silicon substrate is becoming capable of absorbing more and more components, chips of the future are going to be more complex than

they are. Components having different functions and completely novel features will be integrated on the same silicon die, even though they are designed by different design teams on a variety of platforms. Finally, all these heterogeneous components of totally distinct characteristics (importantly, even analog devices can be included in addition to digital ones) have to be placed on a single chip, which makes the task quite complex.

3 Networks on Chips (NoCs)

After realizing the inefficiency of traditional buses in SoCs, in conjunction with so many other factors, the designers of SoCs have come to a cross-road where they meet the computer architecture designers who are always interested in finding dynamic and scalable architectures for building microprocessors. The scalability and wide success of the Internet has attracted the attention of computer architecture as well as SoC designers and influenced them to borrow the idea of using packet based switching networks for the design of future SoC communication infrastructure.

It is an understood fact that the actual reason behind success of the Internet and its scalability lies in a well defined protocol stack; the idea was to decouple communication from computation. Packet switched communication not only provides high scalability, but also facilitates reuse of the communication architecture. The two major problems faced by SoC designers – *re-usability and scalability* – can, therefore well be addressed by the adoption of packet switched communication infrastructure for SoC interconnects. Also, from a business point of view, it is important to reduce the design time by adopting reuse not only at the computational level but also reuse of the communication structure. This will in turn lower the time to market new products with ease. Keeping in view this idea of the Internet, many researchers have proposed communication architectures based upon packet switched on chip networks for connecting components in the future SoCs [7], [8], [9], [10].

Another important aspect of NoCs is that they decouple computation from communication, which is essential for chips that contain billions of transistors. Again, the idea comes from traditional networks such as the Internet, where the communication system is based upon a protocol stack irrespective of the number of the communicating partners. Likewise, the communication infrastructure in NoCs will be designed using a protocol stack which provides well defined interfaces separating communication service usage from service implementation. This means that instead of connecting high level modules (like processors, DSPs, controllers etc.) by routing dedicated wires, they are connected to a network that routes packets between them – as captioned in [7] *"Route Packets not wires"*.

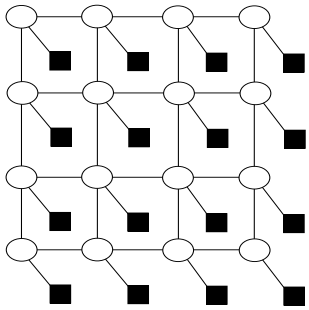

Figure 3. 2D mesh based NoC

4 NoC model

Now, since we already know that NoCs is the most appropriate design choice to develop the future SoCs, the next step is to discuss the design of the NoC itself. Since the area of NoCs is really new, it provides us with an opportunity to create things on a clean slate in order to obtain an optimal design. The immense amount of research that is already being conducted in the Internet has been considerably used in defining the structure of NoCs by the researchers. In the following passages we will discuss proposed topologies, protocols, switching and routing mechanisms for NoCs. In the following passages we will discuss proposed topologies, protocols, switching and routing mechanisms for NoCs.

4.1 Typical NoC topology

There are quite a few topologies proposed for NoCs including fat tree, honeycomb, 2D mesh etc.; we will discuss the most common and agreed upon topology – 2D mesh – in our tutorial because of its simplicity. Consider Figure 3 which shows a simple mesh topology where *circles* represent *switches* while *squares* are *resources*. A resource is a computational unit; it can be a processor, memory, DSP core etc, whereas switches route and buffer messages between resources. It can be seen from the figure that almost each switch is directly connected to neighboring four switches (except for the ones at the edges). The communication channel consists of two one-directional point-to-point buses between two neighboring switches or a switch and a resource. It is expected that, as the technology grows with time, the number and size of resources will also grow, resulting in growth of bandwidth of switch-to-switch or switch-to-resource links, but network wide communication will remain unaffected.

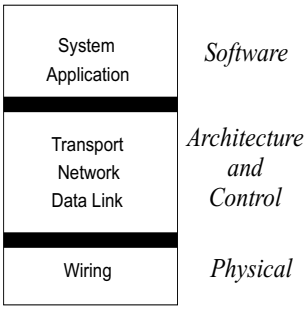

Figure 4. Protocol stack for NoC's – [11]

5 Proposed protocol stack for NoCs

In order to achieve a scalable communication paradigm for NoCs, a protocol stack in comparison to OSI model has been proposed in [11]. This model is shown in figure 4. It can be observed from the figure that the proposed protocol stack is mainly composed of three layers; *Physical, Architecture and Control*, and *Software*. The *Physical layer* deals with signal voltages, slopes and wire sizing in terms of SoCs. Wires are the physical realization of communication in the SoCs. Then comes the *Architecture and Control layer*, which is the most important layer in SoC stack as it encompasses Data Link, Network and Transport layers. In this part, the *architecture* defines the physical layout of the network resources, whereas the *control protocols* define the ways in which these network architectures can be used during system operations. Most of the research is happening at this level in SoCs. Finally, we have the *software layer* which takes care of the system and application software.

5.1 Switching techniques / Flow control

There are different techniques that are used to switch packets between nodes in a network. The most popular ones include *store-and-forward*, *virtual cut-through* and *wormhole* switching. In what follows, we will discuss these techniques in brief and see which one is more appropriate for NoCs based on mesh.

- *Store-and-Forward switching*: This is the most popular packet switching technique in computer networks. Here, parts of the entire packet are stored at the receiving router until the entire packet is received, after which it is forwarded to the next router in the path. In this case, enough buffer space is required at each routers to accommodate the entire packet. Also, for large packets, this method introduces extra packet delays in router. Since buffer resources on-chip are quite expensive, besides the fact that this technique needs more power consumption which is undesirable for NoCs, Store-and-forward is infeasible for on-chip networks.

- *Virtual Cut-through Switching*: This technique is proposed to reduce the packet delay caused by the store-and-forward switching. In this case, the packet is not stored in its entirety in the router, but can be forwarded to the next hop as soon as it is received by the current router. However, if the next hop router is not available, then the current router has to store it in complete form.

- *Wormhole switching*: This switching mechanism was basically developed for parallel processors. The advantage of wormhole flow control is that it achieves minimum network delay and needs less buffer space. In this technique, the packets are further split into small units called *flits* which are immediately forwarded upon arrival. The flits of a packet do not need to be stored in a single router, hence reducing the need for large buffer space. It is for this reason termed as the best candidate for on-chip interconnection networks.

Having inspected the three popular switching techniques, we can now easily say that due to the memory and buffer constraints on-chip, wormhole switching seems to be the best option for NoCs. Here, it is important to clarify the difference between packet switching (forwarding) and routing (which is discussed in section 5.2) – these two terms are sometimes intermixed, thus creating a confused picture. In traditional computer networks, packet switching/forwarding is mainly concerned with moving a packet from an input port of a router to the output. Routing deals with determining the entire path a packet may take from source to destination. Confusingly, the term "routing function" sometimes denotes the packet forwarding method in the context of NoCs.

5.2 Routing in NoCs

Routing is the process of moving information from a source to a designated target. This term is very common in the Internet. Routing can be static or dynamic. Static routing is managed by an administrator manually, and is suitable for networks where network traffic is predictable and relatively simple, which is a rare case in the Internet. Dynamic routing, as the name suggests, is used to dynamically discover routes in case of path changes. Due to the regular structure and on-chip memory constraints static routing is more feasible for NoCs. However, in case of path failures, adaptive routing can be introduced but special care must be taken as to avoid excessive use of buffers or logic on-chip.

A contention based hot potato routing method has been suggested for NoCs [12]. This technique can predict about

contention in the forthcoming stages by using direct connections with the adjacent node. Here packets are divided into small units – *flits*, so they can be easily handled using limited buffers. Routes that lead towards the destination are termed as profitable routes. Alternatively, a route that leads a packet away from the destination is a misroute. Ideally, a packet should follow the profitable route to reach a destination. However, in cases when profitable routes are congested and/or their queues are too long, following a misroute might offer less delay in reaching the destination.

6 Future of NoCs

We can see from our previous discussion that a lot of research has already been done in the Architecture and Control layer of NoCs. This provides an opportunity to extend the amount of research to those areas which are not addressed very frequently but can prove to be vital for designing viable NoCs for future applications. In the following, we will see some important issues that will have a significant impact on the future of NoCs.

6.1 Reliability

SoCs are mainly designed for consumer products – the main issue related to these products is reliability. As the number of transistors increase on a chip, so does the probability of faults, making reliability a major issue [13]. Failures can occur due to a variety of reasons, for example, crosstalk faults can lead to permanent or transient failures of the communication links [14]. In addition to this, implementing packet-based communication on-chip brings new reliability related challenges along with it. A transient fault may cause a bit-flip n the packet header due to which packet get routed to a wrong destination. Similarly, in case of permanent faults, one or many links may go down, causing congestion in the alternate paths. Thus, it is extremely important to deploy mechanisms in NoCs that can handle both permanent and transient errors to ensure reliable packet delivery over shared communication channels. In [15], various reasons affecting the links and routers on-chip are discussed and a model of dynamic routing for NoCs is proposed. We have provided some preliminary results to reroute packets on alternate paths in case of link failures in [16].

6.2 Quality of Service

We know that on-chip networks are designed for a preknown set of computing resources with pre-defined traffic patterns, as compared to traditional networks which are built for future growth and expansion. From an ordinary user's perspective, behavior of any application must be predictable. Although to guarantee the highest level of predictability is minimal, some degree of fitness for purpose is always assumed. Also, in terms of NoCs, which are made for main stream consumer products, such a degree of expectation becomes inevitable. For example, a mobile phone should provide a better voice and video quality to its user than contemporaries.

From the Internet, we learned that a service can theoretically be "guaranteed" if a commitment is made, otherwise it is termed as "best effort". In case of SoCs both kinds of services are essential. An SoC can accommodate traffic ranging from real time data which needs to be delivered in a stipulated amount of time with no or minimum distortion to regular data streams which follow no such constraints. However, providing separate infrastructure for both the services would mean to waste precious resources on-chip. Instead, a combined best effort and guaranteed services architecture has been proposed in [17] for NoCs. This seems to be quite a feasible solution considering the efficient utilization of resources on-chip.

6.3 Software model

As discussed in section 5, above the architecture and control layer, we have the software layer that encompasses the application and system software categories. Programming model gives an abstract view of the hardware to application developers. In parallel programming, there are two programming models – *shared memory* and *message passing*. In the *shared memory model* communication is implicit with shared address space, whereas in the *message passing model*, processors have private memories and communication occurs through explicit messages [18]. In context of NoCs, where computational resources represent a variety of Intellectual Property (IP) blocks, message passing seems to be a choice of programming model for *NoC application software* [11]. This model, despite being harder to write, is more efficient in terms of scalability and performance in heterogeneous environments.

Similarly, in terms of *system software*, a standardized set of operating system services and interfaces needs to be developed. However, a few questions are still open – like having a purely distributed kind of OS versus centralized one. In a distributed system, each resource has an independent OS running; such a system is just like LAN on a chip, but it has very high overhead. On the other hand, in case of a centralized OS, a special processor is meant to run the OS services, but here the problem of scalability comes into play: will the running processor be enough when the system grows or will we need more processors for it?

7 Conclusion

The NoC methodology will likely be the best solution counter the increasing complexity of future SoCs. Form the above discussion it can also be concluded that future SoCs will be platform-based because of short-time-to-market constraints. Development of NoCs will be a huge effort as it involves reuse at all levels; reuse of architecture, hardware and software. Also, it includes reuse of different languages, methods, tools and practices during development.

Although the potential of NoCs is tremendous, it would be rather unlikely to fulfill all its promises before the development of some of its key components like a reliable NoC architecture, assurance of quality of service, and a viable NoC software model. Clearly, it can be concluded that there is a lot of need and scope for research in this area.

References

[1] Rochit Rajsumman, "System-on-a-chip: Design and Test", *Artech House Publishers*, 2000

[2] Érika Cota, Luigi Carro, Flávio Wagner, Marcelo Lubaszewski, "Power Aware NoC reuse on the testing of core based systems", *Proceedings, ITC*, Sept 30 - Oct 02, 2003, Charlotte NC USA.

[3] http://www.itrs.net/Common/2003ITRS/Home2003.htm

[4] Axel Jantesh and Hannu Tenhumen, "Networks on chips", *Kluwer Academic Publications*, 2003, Boston, USA.

[5] Gordon E. Moore, "Cramming more componenets onto integrated circuits", *Electronics*, Volume 38, Number 8, April 19, 1965.

[6] Daniel M. Chapiro "Globally-Asynchronous Locally-Synchronous Systems" *PhD Thesis*, Stanford University, Oct. 1984

[7] William J. Dally and Brian Towles, "Route packets, not wires: on-chip interconnection networks," *Proceedings, Design Automation Conference (DAC)*, pp. 684-689, Las Vegas, NV, June 2001.

[8] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg Johny Oberg, Kari Tiensyrja and Ahmed Himani, "A network on chip Architecture and Design Methodology", *Proceedings, IEEE computer society annual symposium on VLSI*, 2002.

[9] Pierre Guerrier, Allen Greiner, "A generic architecture for on-chip packet switched interconnections", *Proceedings, DATE*, 2000, pp. 250-256.

[10] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johny Oberg, Mikael Millberg, Dan Lindqvist, "Networks on a chip: An Architecture for billion transistor era", *Proceedings, IEEE NorChip Conference*, November 2000.

[11] L. Benini and G. De Micheli, "Networks on Chip: A new paradigm for component-based MPSoC design", in *A. Jerrraya and W. Wolf Editors, Multiprocessors Systems on Chips,* Morgan Kaufman, 2004, pp. 49-80.

[12] Terry Tao Ye, Luca Benini, Giavanni De Michelli, "Packetization and Routing Analysis of On-chip Multiprocessor Networks", *Journal of System Architecture*, Vol 50, Feb 2004, pp. 81 - 104.

[13] Partha Pratim Pande, Cristian Grecu, Andre Ivanov, Resve Saleh, and Giovanni De Michelli, "Design, Synthesis, and Test of Networks on Chips", *IEEE Design and Test*, September/October, 2005 (Vol. 22, No. 5), pp. 404-413.

[14] Ming Shae Wu and Chung Len Lee, "Using a Periodic Square Wave Test Signal to Detect Cross Talk Faults", *Journal, IEEE Design & Test of Computers*, Volume 22, Issue 2, March-April 2005, pp. 160-169.

[15] Muhammad Ali, Micheal Welzl, Martin Zwicknagl, Sybille Hellbrand, "Considerations for fault tolerant Network on chips", *Proceedings, 17th IEEE ICM,* Islamabad, Pakistan, 13-15 Dec. 2005.

[16] Muhammad Ali, Michael Welzl, Sybille Hellebrand, "A dynamic routing mechanism for network on chip", *Proceedings, IEEE NORCHIP*, Oulu, Finland, 21-22 November 2005

[17] E. Rijpkema and K. Goossens and A. Radulescu, J. Dielissen and van Meerbergen, J. and P. Wielage and E. Waterlander "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip" , *Proceedings, IEEE: Computers and Digital Technique*, September 2003.

[18] Steven Brawer, "Introduction to Parallel Programming", *Academic Printers* (July 1989).