Networks on Chip

Axel Jantsch

Laboratory for Electronics and Computer Systems, Royal Institute of Technology, E-mail: axel@imit.kth.se

Abstract

Future single chip systems will resemble more traditional computer networks than traditional central processors. The main reasons for this trend are (A) the infeasibility of global synchrony on a single chip, (B) the necessity of reuse of existing hardware and software components as much as possible, and (C) the heterogeneity of system functions and features. The consequences of this trend are far reaching and imply the shift in concern from computation and sequential algorithms to concurrency, communication and interaction in every aspect of design and development of hardware and software. A concrete example of this shift is the expected replacement of purely sequential computer languages by languages that contain concurrency as a first order object.

1 Motivation

IC manufacturing technology will provide us with a a few billion transistors on a single chip within a few years. Assuming that these predictions hold and that the market will continue to absorb ever higher volumes of ICs, the key questions are: how will the future chips be organized and how will future systems, which include these chips, be designed? One possible answer is that single CPUs will still occupy entire chips and will exhibit correspondingly higher performance. The instruction set will essentially remain unchanged to provide backwards compatibility and the will still be implemented in C or C++. However, there are a few factors which make this scenario unlikely:

1. Physical effects of deep sub-micron technology make it increasingly difficult to maintain global synchrony among all parts of the chip. The clock signal will soon need several clock cycles to travel across the chip and the clock distribution tree is already today a major source of power consumption and cost. The trends of scaling to smaller geometric dimension and higher clock frequency make these problems more significant every year.

2. Synthesis and compiler technology development do not keep pace with IC manufacturing technology development. As a consequence, which is called the *design productivity gap*, we need either exponentially growing design teams or design time to design and implement systems which fit onto a single IC. Since both alternatives are unrealistic we have in the past escaped from the problem by using ever more complex components as primitive design units. These primitive design units have evolved from individual transistors to logic gates to entire ALUs, multipliers and finite state machines. This trend will likely continue with CPU and DSP cores and blocks for compression, encryption and similar functions being the primitive design units. These design units have however asynchronous interfaces to the outside and vastly different internal clocking regimes. As a result a *globally asynchronous and locally synchronous (GALS)* design style emerges already today.

3. Obviously, systems that can be implemented on a single chip become increasingly more complex. As a result different functions and features with vastly different characteristics and history reside on the same chip. signal processing algorithms which recover and generate radio signals will coexist with global control, maintenance and accounting functions as well as with natural language comprehension and generation functions. These functions are developed in different contexts, by different teams, with different design languages and tools. However, they need to be integrated into a single chip.

Taking these current trends and facts together it is natural to contemplate a design paradigm where a set of interacting functions and features are implemented on a set of asynchronously communicating resources, such as CPU cores and specialized hardware blocks. In this scenario the mapping and implementation of system functions onto resources is covered by traditional design and synthesis methods. It may even an integral part of the reuse of a given system function and a resource. For instance, the purchasing of a bluetooth protocol stack may include its implementation on a combination of a custom hardware block and an ARM processor core. However, providing a chip level communication infra-structure and mapping of system level interactions onto the communication infrastructure is not covered by any traditional design methodology and will become the focus of research and tool development in the near future. In fact, in the last two years we have seen several concrete proposals for on-chip network architectures.

in 2000 Hemani and al. [1] have proposed a packet switched architecture with switches surrounded by six resources and connected to 6 neighboring switches. The architecture is called a Honeycomb due to the hexagon based pattern of switches and resources. The concept of packet switching re-appears in other consecutive approaches but the topology simplifies in most proposals to a mesh of resources and switches. In 2001 Dally and Towles [2] propose a mesh based packet switched network with very simple switches which require less than 10% area overhead. MicroNetworks proposed by Drew Wingard [3] is another packet switched on-chip interconnection mechanism proposed recently. Keutzer et al. [4] and Sgroi et al. [5] provide general discussions and motivations for communication centric on-chip architectures and platforms and for the strict conceptual separation of computation from communication. More recently, Kumar et al. [6] put forward a detailed packet switched, mesh based on-chip communication infra-structure together with a design methodology. The proposed concept of a region breaks the strict mesh-based geometry. A region can cover an arbitrary number of switches and resources and allows to accommodate larger resources such as FPGA areas and memory banks in a flexible way. This architecture we will also outline in the consecutive sections of this paper. Valtonen et al. [7, 8] propose an on-chip interconnected network of resources or cells but put the main emphasis on fault tolerance and unlimited scalability.

2 Network on Chip Architecture

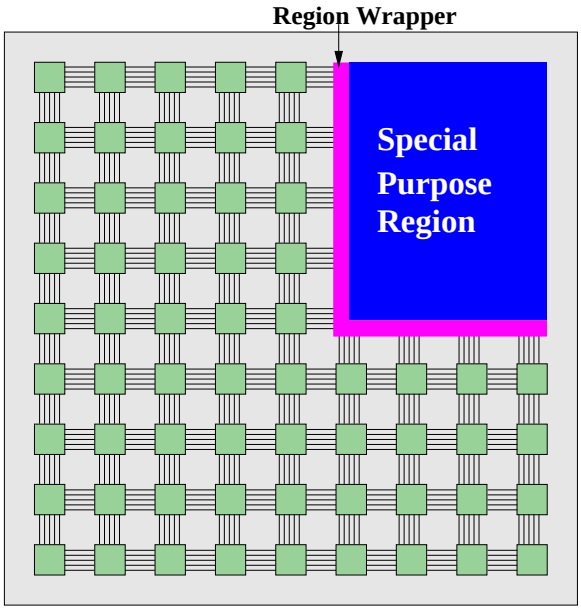

Figure 1: Each node in the mesh contains a switch and a resource.

The Network-on-Chip (NOC) architecture, as outlined in figure 1, provides the communication infrastructure for the resources. In this way it is possible to develop the hardware of resources independently as stand-alone blocks and create the NOC by connecting the blocks as elements in the network. Moreover, the scalable and configurable network is a flexible platform that can be adapted to the needs of different workloads, while maintaining the generality of application development methods and practices.

A simple mesh interconnection topology is simplest from a layout perspective and the local interconnections between resources and switches are independent of the size of the network. Moreover,routing in a two-dimensional mesh is easy resulting in potentially small switches, high bandwidth, short clock cycle, and overall scalability. A NoC consists of resources and switches that are connected using channels as a mesh (Manhattan- like structure) so that they are able to communicate with each other by sending messages. A resource is a computation or storage unit. Switches route and buffer messages between resources. Each switch is connected to four other neighboring switches through input and output channels. A channel consists of two one-directional point-to-point buses between two switches or a resource and a switch. Switches may have internal queues to handle congestion. The precise layout and geometry depends on the technology generation. We expect that the area of a resource is the maximal synchronous region in a given technology. It is expected to shrink with every new technology generation. Consequently the number of resources will grow, the switch-to-switch and the switch-to-resource bandwidth will grow, but the network wide communication protocols will be unaffected. Figure 2 illustrates the principles of the physical floor plan within the NOC. Consider a 60nm CMOS technology expected in 2008, a 22mm 22mm chip size, a resource size of 2mm 2mm and a minimum wire pitch of 300nm. A NoC would accommodate 10 10 resources, each switch would occupy $30\mu m \times 30\mu m$ and the channels would be can use 3 metal layers for we have space for 300 wires. Since we need control, handshaking and signaling, this scenario would yield an effective data bus width of 256 wires.

A *Region*, as illustrated in figure 1, is an area inside the NoC which is insulated from the network by a wrapper. The internal architecture and organization of a region is invisible to the network and can be arbitrary. The network is not even aware of the presence of the region because the region wrapper maintains the illusion of an undisturbed network to the outside by, for instance routing packets around rather than through the region if the packet's destination is on the other side. This concept allows to accommodate arbitrary sized resources in a flexible way.

3 NoC Implications

There are many compelling reasons for the emergence of NoCs although the architecture and organization may be different from everything proposed today. However, the

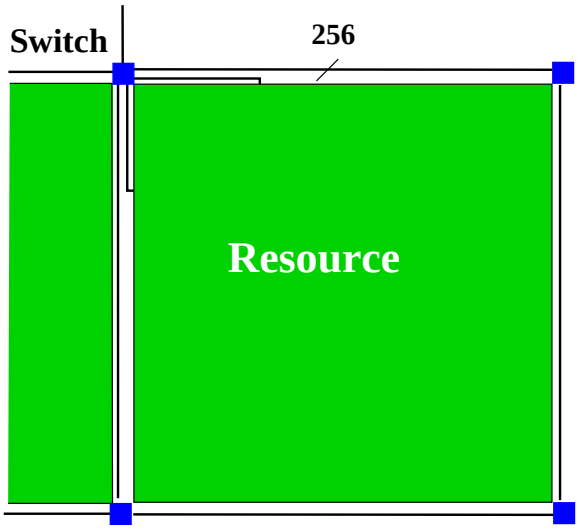

Figure 2: The expected footprints of resource, switch and channels in 60nm CMOS technology.

precise application area and deployment mechanism is still very uncertain. Two scenarios can be envisioned. Scenario (A) holds that there will be a rather big number of NoC based platforms specialized for a particular product range. For instance, the NoC platform for cellular phones will be optimized for power efficiency and equipped with a limited but highly targeted set of functions and features. The NoC platform for base station will be optimized for optimized for performance and scalability to allow painless growth when higher performance or extended functionality is requested. NoC platforms for automotive and medical applications will be highly fault tolerance and safety critical while multi-media consumer devices will provide a variety of compression/decompression and encryption/decryption built-in functions. Development and verification cost for the platform, the communication infra structure, operating system services and design methodology and tools will be shared among all the products in an application domain. Conceptually, a NoC platform will be similar to dedicated embedded systems of today.

In contrast, scenario (B) holds that the volumes in individual application areas such as base stations, mobile phones, automotive devises, etc. are not sufficient to amortize the significant investment in platform development and manufacturing sites. Only the combined volumes of several of these application areas will justify the costs of 60nm-10nm CMOS fabs and the development and maintenance of a NoC platform architecture and design tools. The result would be a general purpose NoC assuming the role of today's CPUs as general computing engines for a wide range if not for all application areas including the PC. The consequences would be far reaching. Many of today's computer science and computer engineering disciplines have a sequentially operating single CPU as implicit or explicit fundamental assumption. These disciplines had to broaden their scope considerably and entire curricula had to be rewritten. The bulk of theory and practice in disciplines such as computer languages, compiler construction, operating systems, programming, algorithm design, simulation, etc. are geared towards single CPU implementations. This is not to say that there is not a significant amount of knowledge and expertise on parallelism and concurrency in all these disciplines. But replacement of current CPUs by NoC based computing engines would bring concurrency to the mainstream in all disciplines and downgrade sequentiality to a limited and special case.

Several key developments are a prerequisite for scenario (B) to become a viable alternative.

NoC Architecture Since the resources in a NoC can be of different types from CPU cores, DSP cores, FPGA blocks to dedicated hardware blocks, a right mixture has to be found which suits all the different application areas sufficiently well. Furthermore, the communication network must have a sufficiently high bandwidth in addition to be able to accommodate various traffic types such as real-time traffic, regular data streams and irregular control messages.

NoC Assembler Language A standardized way to configure and program a NoC has to be developed which is independent from the particular NoC instance with a fixed number of resources. It has also to be independent from the various ways to specify and model the functionality of systems at a high level. In addition to capture the computation of resources it has to capture the acts of communication between resources.

NoC Operating System A standardized set of operating system services and interfaces has to be developed. A NoC operating system is a generalization of traditional operating systems and will include, among other features, the internal and external communication, run-time error diagnostics and recovery, load balancing and dynamic task migration.

NoC Design Methodology A sufficiently robust and predictable design methodology, including the detailed methods and tools, has to include the mapping and verification of the functionality as well as of all the relevant non-functional requirements and constraints.

The great potential of NoC based platforms notwithstanding, it is far from obvious that this potential can be tapped. For this list illustrates clearly that if even one of the necessary components of a NoC platform is not developed well enough, the application of NoC concepts will be limited to rather small niches and not become mainstream.

4 Conclusion

We have briefly discussed the emerging concept of Networks-on-Chip (NoC) and we have described one concrete NoC

architecture. This led us to speculate on the future and potential impact of NoC platforms. We have concluded this discussion with the observation that, even though the potential of NoC concepts may be tremendous, it is not likely that NoC can fulfill its promises due to the significant number of key components that have to be developed before NoC products can be deployed on a larger scale.

References

[1] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johnny Öberg, Mikael Millberg, and Dan Lindqvist. Network on chip: An architecture for billion transistor era. In *Proceeding of the IEEE NorChip Conference*, November 2000.

[2] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, June 2001.

[3] Drew Wingard. MicroNetwork-based integration of SOCs. In *Proceedings of the 38th Design Automation Conference*, June 2001.

[4] Kurt Keutzer, Sharad Malik, Richard Newton, Jan Rabaey, and Alberto Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trasnactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1523–1543, Decmber 2000.

[5] Marco Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proceedings of the 38th Design Automation Conference*, June 2001.

[6] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johnny Öberg, Kari Tiensyrjä, and Ahmed Hemani. A network on chip architecture and design methodology. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, April 2002.

[7] T. Valtonen et al. Interconnection of autonomous error-tolerant cells. In *Proceedings of the International Symposium on Circuits and Systems*, Scottsdale, AZ, USA, 2002.

[8] T. Valtonen et al. An atonomous error-tolerant cell for scalable network-on-chip architectures. In *Proceedings of the 19th IEEE NorChip Conference*, Kista, Sweden, November 2001.