Efficient SOPC-Based Multicore System Design Using NOC

Vignesh Veerapandian, Xingguo Xiong
Department of Electrical and Computer Engineering, University of Bridgeport, Bridgeport, CT 06604, USA
Email: vveerapa@bridgeport.edu, xxiong@bridgeport.edu

***Abstract* — Due to the advancement of VLSI (Very Large Scale Integrated Circuits) technologies, we can put more cores on a chip, resulting in the emergence of a multicore embedded system. This also brings great challenges to the traditional parallel processing as to how we can improve the performance of the system with increased number of cores. In this paper, we meet the new challenges using a novel approach. Specifically, we propose a SOPC (System on a Programmable Chip) design based on multicore embedded system. Under our proposed scheme, in addition to conventional processor cores, we introduce dynamically reconfigurable accelerator cores to boost the performance of the system. We have built the prototype of the system using FPGAs (Field-Programmable Gate Arrays). Simulation results demonstrate significant system efficiency of the proposed system in terms of computation and power consumption. Our approach is to develop a highly flexible and scalable network design that easily accommodates the various needs. This paper presents the design of our NOC (Network on Chip) which is a part of the platform that we are developing for a reconfigurable system. The major drawback of SOPC based systems lies in the routing of the various on-chip cores. Since it is technically difficult to integrate more than one core on a single chip, we come across several routing problems which lead to inefficient functioning. Thus we implemented several NOC based routing algorithms which considerably improve accessing speed and enhance the system efficiency.**

Keywords: Multicore system, System on a Programmable Chip (SOPC), Network on Chip (NOC), Multiprocessor System-on-Chip (MPSOC).

I. INTRODUCTION

During the 1990s more and more processor cores and large reusable components have been integrated on a single silicon die, which has become known under the label of System-on-Chip (SOC). Buses and point-to-point connections were the main means to connect the components, Hence they can be used very cost efficiently. As silicon technology advances further, several problems related to buses have appeared. [1][5] Buses can efficiently connect 3-10 communication partners but they do not scale to higher numbers.

As a result, around 1999 several research groups have started to investigate systematic approaches to the design of the communication part of SOCs. It soon turned out that the Problem has to be addressed at all levels from the physical to the architectural to the operating system and application level. Hence, the term Network on Chip (NOC) is today used mostly in a very broad meaning, encompassing the hardware communication infra-structure, the middleware and operating system communication services and a design methodology and tools to map applications onto a NOC. All this together can be called a NOC platform. [4] Networks on Chip (NOCs) have emerged as a viable option for designing scalable communication architectures. For multiprocessor System-on-Chips (MPSOCs), on-chip micro networks are used to interconnect the various cores. The main idea with NOCs, besides the solutions to the physical issues, is the possibility for more cores to communicate simultaneously, leading to larger on-chip bandwidths. The adoption of NOC architecture is driven by several forces: from a physical design viewpoint, in nanometer CMOS technology interconnects dominate both performance and dynamic power dissipation, as signal propagation in wires across the chip requires 2 multiple clock cycles. NOC links can reduce the complexity of designing wires for predictable speed, power, noise, reliability, etc., thanks to their regular, well controlled structure. From a system design viewpoint, with the advent of multi-core processor systems, a network is a natural architectural choice.

NOC can provide separation between computation and communication; support modularity and IP reuse via standard interfaces; handle Synchronization issues; serve as a platform for system test and hence increase engineering productivity.

II. DIFFERENT NOC TOPOLOGIES

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.

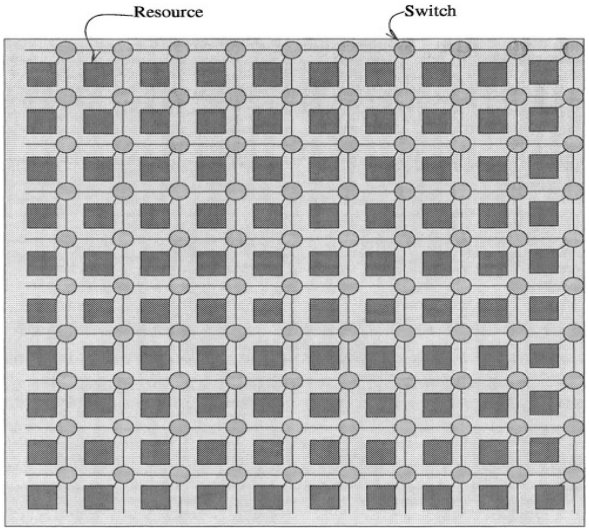

Figure.1. Network on chip [7]

A number of different NOC topologies have been proposed. They all have in common that they connect resources to each other through networks and that information is sent as packets over the networks [7]. Network on Chip (NOC) has evolved as an important research topic during the last few years. The idea is that scalable switched networks are used for on-chip communication among processing units, in order to cope with design of continuously growing systems. Design complexity promotes reuse of investment in earlier designs as well as purchase of outside intellectual property (IP). However, in larger designs, communication among components will become a bottleneck using traditional techniques like common buses. NOC is one solution to address this issue because packet switched communication can provide higher flexibility, throughput and reusability. To gain full advantage when using this concept in NOC architecture design, the size of resources should be similar and the communication facilities should be homogeneous.

2.1 Honey Comb Technology

In NOC design, the resources communicate with each other by sending addressed packets of data and routing them to the destinations by the network of switches [7]. Though many topologies are possible, we will first discuss about Honey comb topology. The overall organization is in the form of a honeycomb, as shown in Figure.2. The resources - computational, storage and I/O - are organized as nodes of the hexagon with a local switch at the centre that interconnects these resources. Hexagons at the periphery would be primarily for I/O, whereas the ones in the core would have storage and computational resource. To further improve the connectivity, switches are directly connected to their next nearest neighbors, as shown in Figure 2, allowing any resource to reach 27 additional resources with two hops. As a last measure to further improve connectivity, every alternate switch is directly connected making each resource element reach a lot more elements with minimal number of hops.

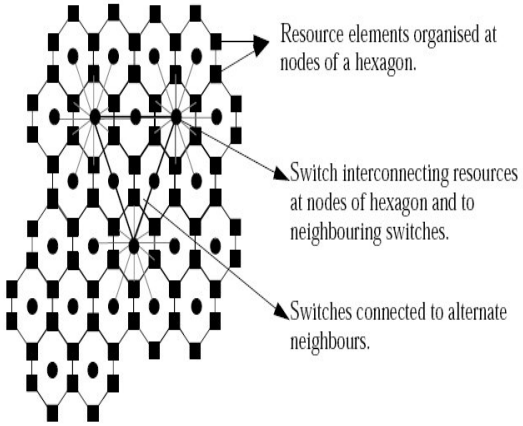

Figure.2. A honey comb structure for NOC [7]

2.2 Mesh Topology

NOC is a scalable packet switched communication platform for single chip systems. The NOC architecture consists of a mesh of switches together with some resources which are placed on slots formed by the switches.[2] Figure 3 shows NOC architecture with 16 resources. Each switch is connected to four neighboring switches and one resource. Resources are heterogeneous. A resource can be a processor core, a memory block, a FPGA, custom hardware block or any other intellectual property (IP) block, which fits into the available slot and complies with the interface with the NOC switch. We assume switches in NOC have buffers to manage data traffic.

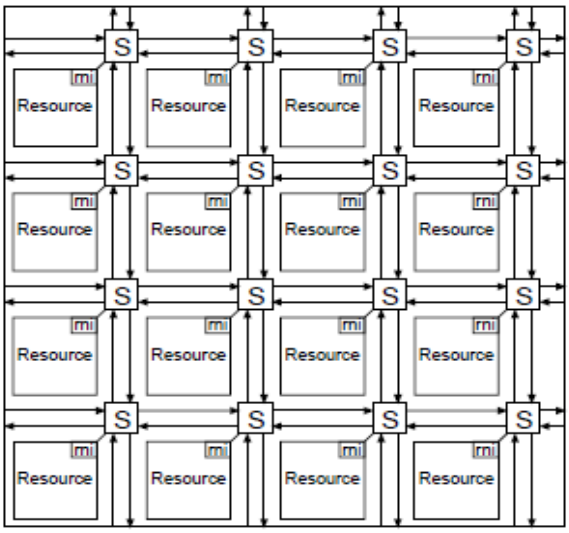

Figure.3. 4×4 NOC switch [2]

Every resource has a unique address and is connected to a switch in the network via a resource network interface (RNI). The NOC platform defines four protocol layers: the physical layer, the data link layer, the network layer, and the transport layer. The RNI implements all the four layers, whereas every switch to switch interface implements the three of four layers except physical layer. The NOC architecture also has a concept of region allows us to handle physically larger resources and can be used to provide fault tolerance. A typical NOC architecture will provide a scalable communication infrastructure for interconnecting cores. The area of multi-media is a very suitable candidate for using this high computing capacity of NOCs. NOC is a general paradigm and one needs to specialize a NOC based architecture for every application area.

III. SOPC BUILDER

SOPC Builder is a powerful system development tool. SOPC Builder enables us to define and generate a complete system-on-a-Programmable-chip (SOPC) in much less time than using traditional, manual integration methods. SOPC Builder is included as part of the Quartus II software (www.Altera.com). We used SOPC Builder to create systems based on the Nios® II processor. [3]

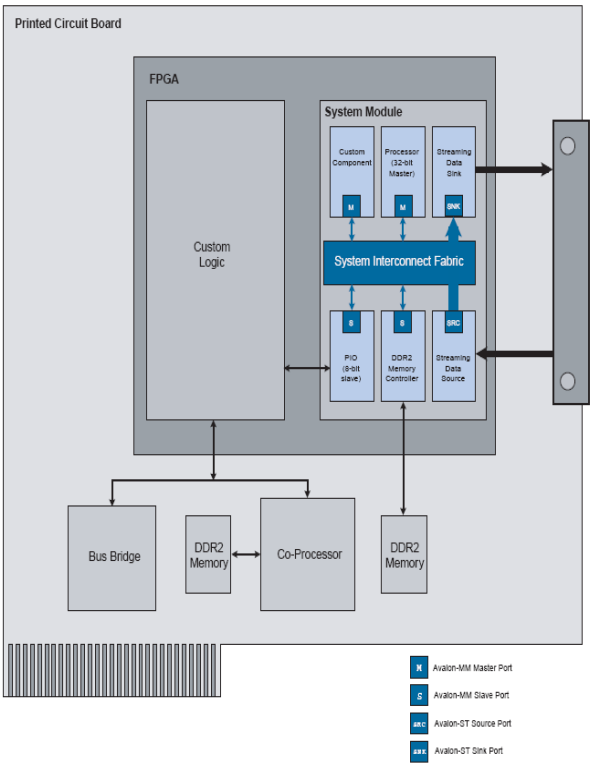

Figure.4. System example [3]

Figure.4 shows an FPGA design that includes an SOPC Builder system and custom logic modules. We can integrate custom logic inside or outside the SOPC Builder system. In this example, the custom component inside the SOPC Builder system communicates with other modules through an Avalon-MM master interface. The custom logic outside of the SOPC Builder system is connected to the SOPC Builder system through a PIO interface. The SOPC Builder system includes two SOPC Builder components with Avalon-ST source and sinks interfaces. The system interconnect fabric shown below in Figure.5 connects all of the SOPC Builder components using the Avalon-MM or Avalon-ST system interconnects as appropriate.

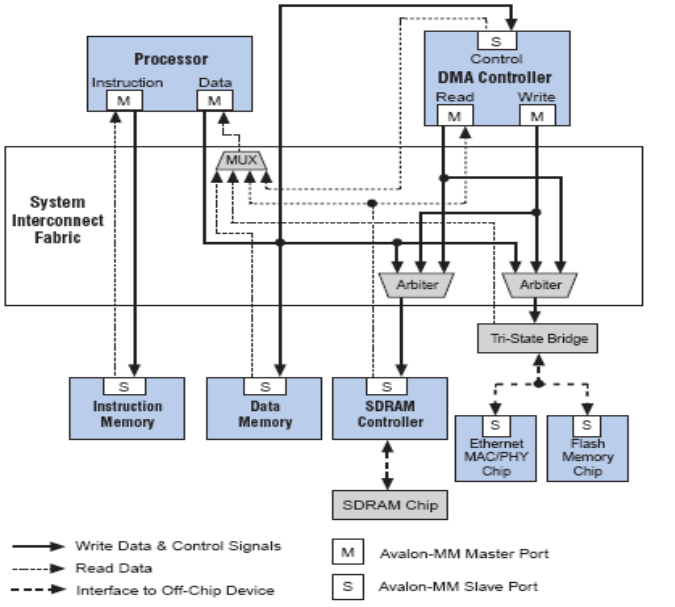

Figure.5. System interconnect fabric

The systems interconnect fabric [3] for memory-mapped interfaces are a high-bandwidth interconnects structure for connecting components that use the Avalon® Memory-Mapped (Avalon-MM) interface. The system interconnect fabric consumes minimal logic resources and provides greater flexibility than a typical shared system bus. It is a cross-connect fabric and not a tri-stated or time domain multiplexed bus. Here we describe the functions of system interconnect fabric for memory-mapped interfaces and the implementation of those functions.

3.1. Chip Planner

The Chip Planner provides a visual display of chip resources. It can show logic placement, Logic Lock and custom regions, relative resource usage, detailed routing information, fan-in and fan-out paths between registers, and delay estimates for paths.

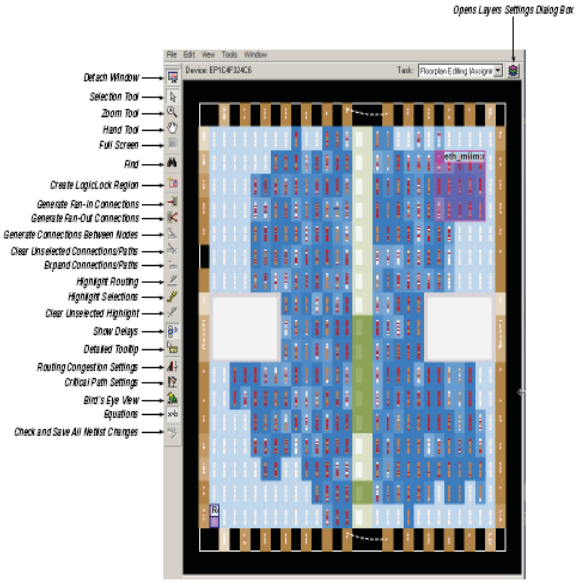

Figure.6. Chip planner tool bar from Quartus II Software

With the Chip Planner, we can view critical path information, physical timing estimates, and routing congestion. We can also perform assignment changes with the Chip Planner, such as creating and deleting resource assignments, and post-compilation changes like creating, moving, and deleting logic cells and I/O atoms. By using the Chip Planner in conjunction with the Resource Property Editor, we can change connections between resources and make post-compilation changes to the properties of logic cells, I/O elements, PLLs, and RAM and digital signal processing (DSP) blocks. With the Chip Planner, we can view and create assignments for a design floor plan, perform power and design analyses, and implement ECOs in a single tool.

3.2. Viewing Routing Congestion

The Routing Congestion view allows us to determine the percentage of routing resources used after a compilation. This feature identifies where there is a lack of routing resources. This information helps us to make decisions about design changes that might be necessary to ease the routing congestion and thus meet design requirements. The congestion is visually represented by the color and shading of logic resources. The darker shading represents greater routing resource utilization. We can set a routing congestion threshold to identify areas of high routing congestion. After selecting the Routing Utilization layer setting, click on the Routing Congestion icon on the taskbar.

3.3. Viewing I/O Banks

The Chip Planner can show all of the I/O banks of the device. To see the I/O bank map of the device, click the Layers icon located next to the Task menu. Under Background Color Map, select I/O Banks.

3.4. Generating fan-in and fan-out Connections

This feature enables us to view the immediate resource that is the fan-in or fan-out connection for the selected atom. For example, selecting a logic resource and choosing to view the immediate fan-in enables us to see the routing resource that drives the logic resource. We can generate immediate fan-in and fan-outs for all logic resources and routing resources. To remove the connections that are displayed, click the “Clear Connections” icon in the toolbar.

3.5. Highlight Routing

This feature enables us to highlight the routing resources used for a selected path or connection.

3.6. Delay Calculation

We can view the timing delays for the highlighted connections when generating connections between elements. For example, you can view the delay between two logic resources or between a logic resource and a routing resource.

3.7. Viewing Assignments in the Chip Planner

Location assignments can be viewed by selecting the appropriate layer set from the tool. To view location assignments in the Chip Planner, select the Floor plan Editing (Assignment) task or any custom task with Assignment editing mode. The Chip Planner shows location assignments graphically, by displaying assigned resources in a particular color (gray, by default). We can create or move an assignment by dragging the selected resource to a new location.

IV. RESULTS

Using SOPC Builder in Quartus II tool, we designed and simulated efficient SOPC-based Multicore System, and the results are listed in Figure 7-11.

Figure 7 shows the screenshot of the developed SOPC builder system. It is done using SOPC builder, this build system has a design of multicore system with 2 CPU’s, Avalon tri state bridge, flash memory, LCD and PIO’s. It’s a FPGA design which includes SOPC builder system and custom logic modules. We can integrate custom logic inside or outside the SOPC builder system. In this design the custom logic modules inside the SOPC builder system communicates with other modules through an Avalon-MM-master interface. The custom logic modules outside of the

SOPC builder system is connected to SOPC system through a PIO interface.

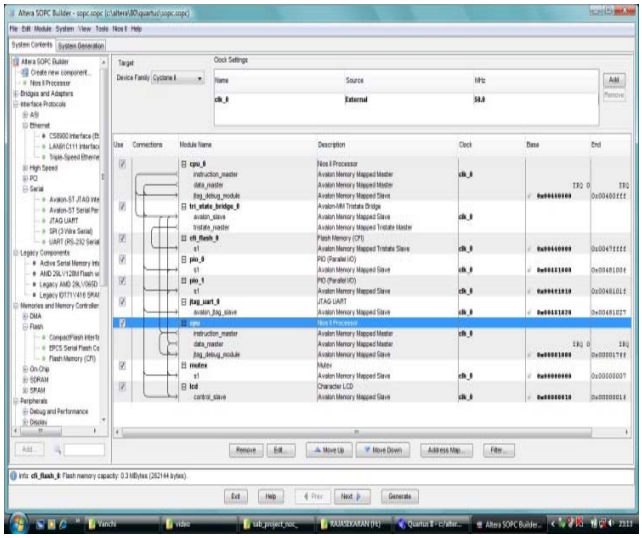

Figure.7. SOPC builder for system building

The block diagram of the symmetric file is shown in Figure 8. SOPC builder allows us to design the structure of a hardware system. The GUI allows adding components to a system configure the components and specify the connectivity. After adding and parameterize components, SOPC Builder generates the system interconnect fabric, outputs HDL files and .BDF during system generation. This .BDF file shown in Figure8 represents the top –level SOPC system for use in Quartus II.

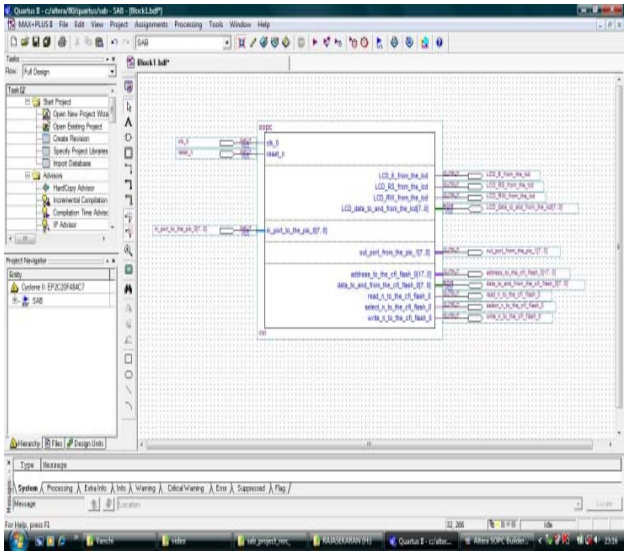

Figure.8. Block diagram of the symmetric file

The compilation result is shown in Figure 9. Once the system design is over it need to be verified whether the designed system has no errors so in order to test the system we compile our system. It shows 100% full compilation during synthesize of our SOPC system.

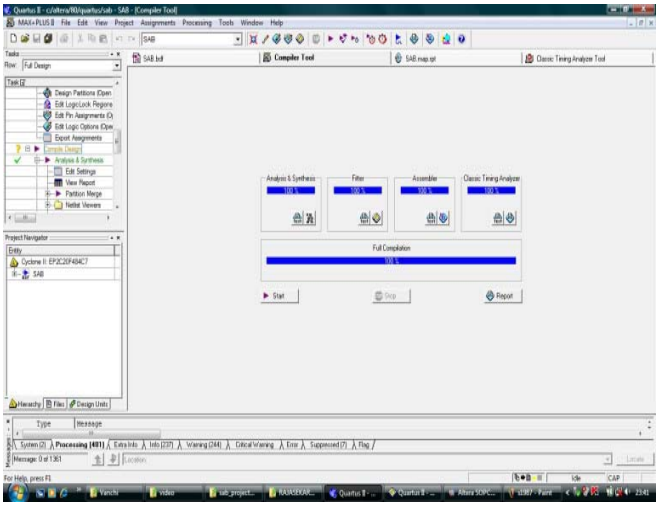

Figure.9. Compilation

The chip planner view of the compiled design is shown in Figure 10. In this screenshot we can see the build components placed in this chip planner. We can view critical path information, physical timing estimation and routing congestion. The Chip Planner uses a hierarchical zoom viewer that shows various abstraction levels of the targeted Altera device. As we increase the zoom level, the level of abstraction decreases, thus revealing more detail about our design.

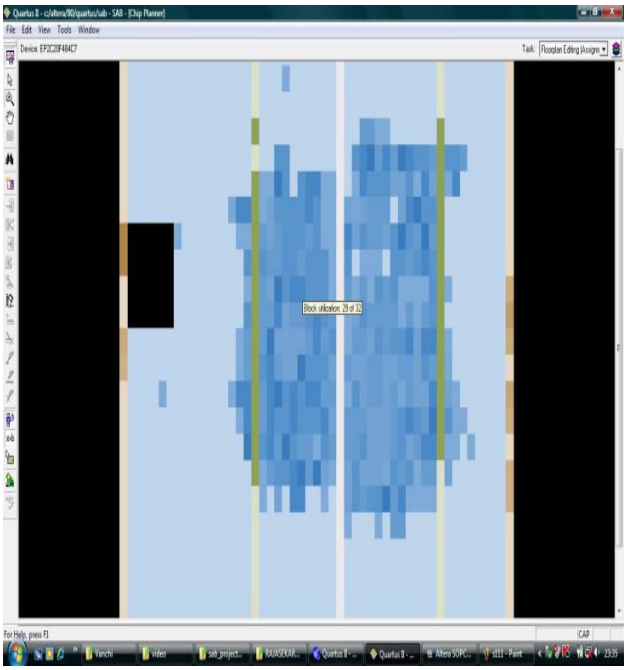

Figure.10. Chip planner view of compiled design

The routing utilization is shown in Figure 11. It allows us to determine the percentage of routing resources used after compilation. This information helps us to make decisions about design changes which is necessary to ease the routing congestion to meet the design requirements. The routing congestion is visually represented by the color and shading of logic resources. In Figure 11 we can see some areas are dark and some areas are bright. The dark regions represent greater routing resource utilization and the bright regions represent no routing congestion.

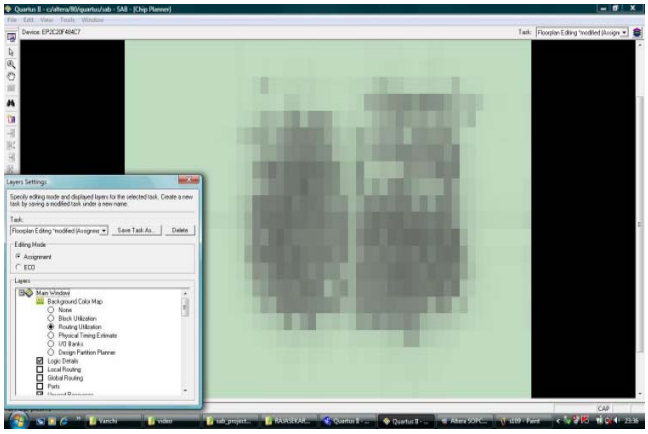

Figure.11. Routing utilization (Darker areas Show dense routing connections)

V. CONCLUSIONS AND FUTURE WORK

In this paper, a simple multicore embedded system was developed and synthesized using Altera SOPC Builder. The synthesize results demonstrate that routing will be the greater issue when it comes to the chip design. Using network-on-chip architecture a prototype system based on networking was developed to overcome the routing issue. As a result, network-on-chip micro networks are used in multicore processors to interconnect the various cores to communicate simultaneously. This leads to larger on-chip bandwidth and reduces routing congestion so that the system efficiency is enhanced. Our designed multicore system has two CPU cores which are individually optimized to the particular computational characteristics of different application fields, complementing each other to deliver high performance levels with high flexibility at reduced cost.

The research focus is shifting from implementation of NOC to investigation of its optimal use. The research problems in NOC design are identified as synthesis of communication infrastructure, choice of communication paradigm, application mapping and optimization. In the future, we will continue to design an efficient multicore SOPC with optimized timing constraints, reduced latency and improved programmability. We will also develop highly embedded, multi-core systems with more number of cores which in turn increases the system performance and many applications can run at the same time.

REFERENCES

[1] Luca Benini, Giovanni De Micheli, “Networks on Chips: A New SoC Paradigm”, *Computer*, v.35 n.1, p.70-78, January 2002.

[2] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, “A network on chip architecture and design methodology”, *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pp. 105-112, April 2002.

[3] Quartus II Handbook, SOPC Builcer, Version 9.0, Volume 4, URL: www.altera.com/literature/hb/qts/qts_qii5v4.pdf

[4] Axel Jantesh and Hannu Tenhumen, “Networks on Chip”, *Kluwer Academic Publications*, 2003, Boston, USA.

[5] Muhammad Ali, Michael Welzl, Sybille Hellebrand, “A dynamic routing mechanism for network on chip”, *Proceedings of IEEE NORCHIP*, Oulu, Finland, 21-22, Nov. 2005.

[6] International Technology Roadmap for Semiconductors 2003, URL: http://public.itrs.net

[7] H. Tenhunen and A. Jantsch, "Networks on Chip", Springer, 1st edition, ISBN: 1402073925, Jan. 2003.