ESP-IJCEET
ESP International Journal of Communication Engineering & Electronics Technology
ISSN: 2583-9217 / Volume 1 Issue 2 September 2023 / Page No: 25-32
Paper Id: IJCEET-V1I2P105 / Doi: 10.56472/25839217/ IJCEET-V1I2P105

Original Article

On-Chip Permutations Network Design with a Three Dimensional Mesh Network

Reena Kumari[1]**, Mohammad Ismail Khattab**[2]

[1]*Research Scholar, Department of Electronics, University of Ottawa, Canada.*
[2]*Supervisor, Department of Electronics, University of Ottawa, Canada.*

Received Date: 13 July 2023 Revised Date: 02 September 2023 Accepted Date: 08 September 2023

Abstract: *The network on chip (NoC) is being investigated as an on-chip communication fabric for future multiprocessor system on chips (MPSOCs) because of its adaptability, scalability, and high bandwidth. My abstract describes the methodology behind the construction of a network-on-chip that can sustain a specified throughput. Using three layers of 4-port switches in a closed network configuration. For system-on-chip applications involving multiple processors, this network is built to guarantee a specific traffic permutation. The proposed network utilizes a dynamic path-setup mechanism in conjunction with a pipelined circuit-switching methodology to allow for path arrangement for arbitrary traffic permutations at runtime. It is common for IP cores in a SoC to communicate with one another via a "network on chip," or NoC.*

Keywords: *NoC, MPSOCs, Pipelined circuit.*

I. INTRODUCTION

Over the past decade, MPSoCs have become a prominent type of VLSI system due to their ability to integrate several processors onto a single chip. The term "multi-programmable system-on-chip" (MPSoC) refers to a very large scale integration (VLSI) system that integrates most or all of the components required for an application that uses many programmable processors. The multi-purpose nature of MPSoCs has led to their adoption in a variety of fields, including networking, communications, signal processing, and multimedia. For parallel processing, scientific computing, and other uses, a trend is emerging toward MPSoC designs that incorporate on-chip networks to facilitate communication between the chip's many processors. Technology advancements allow for ever more transistors, but Moore's empirical rule describes much more than that. It also brings about fresh necessities and difficulties. The complexity of systems grows proportionally quickly. The methods used to create systems 20 years ago are completely inapplicable to those used today. Constant innovation in architectural design is essential. There have been three major revolutions made possible by Moore's law over the past two decades. Arbitration mechanisms were a major flaw in the old system. Each switch is equipped with a run-time programmable arbiter to help us combat these issues. New Arbitration schemes, which improve upon their predecessors, are built into the Arbiter's code. The effectiveness has increased. The primary benefit of circuit switching is the ability to manually construct a dedicated channel for data transmission with a predetermined delay.

II. LITERATURE REVIEW

The progress of technology has made it possible to fit billions of transistors onto a single chip. This paves the way for the optimal use of resources and the development of ever-more-complex and diversified architecture. As a result, a new paradigm, known as Multiprocessor Systems on Chip (MPSoC), has arisen. Processors (Scalar, Risk), memory, hardware accelerators (DMA, DCT, FFT), and peripheral input/output are all examples of heterogeneous components that may be included. Network on Chip (NoC) refers to the interconnection network that allows for communication between all of these individual components. The performance of the MPSOC itself, and more specifically its communication architecture, must be evaluated in order to make informed design decisions and trade-offs. In this study, we take a look at what has already been done to investigate and practice Design Space Exploration (DSE) for MPSoCs. Today's MPSoC advancements are still focused on shrinking the size of the device while packing more processing power into a single die.

The approaches utilized to reduce chip size have allowed for tremendous progress in the field of integrated circuits. Because of its high computational demand, high performance, flexibility, high energy efficiency, and low cost design, MPSoC has been seen as the best contender for applications like networking, telephony, multimedia, etc. Designers face a challenging task when tasked with creating MPSoCs for such applications due to the large number of potential design choices available to them in terms of Processing Elements (PEs), micro-architectural features, interconnects, etc., all of which must be balanced against application-specific requirements.

Research on multi-core processors was conducted by Geoffrey Blake, Ronald G. Dreslinski, and Trevor Mudge. The benefits of employing multicore systems are discussed, as are the architectural classifications that allow for performance to be improved by increasing the number of cores rather than the frequency. As commercial multi-core architectures proliferate across a wide range of use cases, it's crucial to have a firm grasp on the fundamental features shared by all of them. The survey identified five key characteristics shared by modern multi-core designs, and the benefits and drawbacks of each were explained in the context of real-world devices. These included the processing elements, memory, accelerators/integrated peripherals, memory, power/performance, and the realm of applications.

The MPSoC multistage interconnect network was surveyed by B. Neji, Y. Aydi, R. Ben-atitallah, S. Meftaly, M. Abid, and J-L. Dykeyser. In addition, this demonstrates the superiority of multistage networks for MPSoC. Area, latency, and power consumption estimates are also included. Topology, routing algorithms, and technical arbitration are also covered. The performance of several types of networks is compared using an FPGA prototype.

III. MULTIPROCESSOR SYSTEM ON CHIP

Very large scale integration (VLSI) technology has recently taken form in the form of multiprocessor systems on a single chip (MPSoC). More than 100 million transistors can fit on a single Integrated Circuit, and chips with 1 billion transistors are expected to become a reality in the near future, according to a worldwide technical road map for semiconductors. To make use of all this raw processing power, designers must expand their skillsets beyond logic design and into computer architecture. Applications' high expectations for these devices provide challenges unseen in conventional computer design. Deadlines in real time, ultra-efficient power consumption, etc. Given these merits and drawbacks, MPSoC design is worthy of study.

The functionality of an entire electronic system can be implemented on a single integrated circuit called a system-on-chip (SoC). Complexity is the defining feature of anSoC. Although a memory chip may have numerous transistors, its deterministic design classifies it as an element rather than a system. The Soc's final configuration depends on the task at hand. Input/output (I/O) circuitry is often found in SoCs, and it is either analog or mixed-signal. Most of anSoc is digital because it is the only method to safely design such complicated functions, but some high-performance I/O applications require a separate analog interface device that serves as a companion to a digital Soc. Memory, instruction-processors (CPUs), specialized logic, buses, and other digital operations could all be present in the system. Instead of using generic CPUs, the system architecture is typically modified for each individual task.

IV. MULTISTAGE INTERCONNECTION NETWORK

Most of the connections between the aforementioned Processing Elements (PEs) are handled by a Network-on-Chip (NoC). Network Interfaces (NI), routing nodes, and connections make up a network operation center (NoC). The NI is in charge of enforcing the protocol used by the PE domain to talk to the interconnection environment. Functions of computation and communication are separated. The data between the source and destination PEs via the links is routed and arbitrated by nodes known as routing nodes. Many different configurations of networks have been analyzed.

A. An Introduction to Interconnection Networks:

The requirements of the communications sector, especially in the context of telephone switching, drove much of the earliest work on interconnection networks. As the computer industry expanded, more and more use cases emerged for internal computer networks. In addition, many networks were developed to connect processors to memory and to one another as interest in parallel processing grew. One such network was used to arrange sequences of numbers. Interconnection networks have gone full circle with the introduction of the fast packet switch, with many of the networks first proposed for parallel processing now being evaluated for use in fast packet switch designs. The first signs of mechanical aid appeared.

B. Control Mechanism:

The mechanism used to regulate links between inputs and outputs can further categorize interconnection networks. The routing decisions can be made by consulting the state of all current connections and all connection requests if the algorithm is centralized and implemented in a central processor. Circuit switching, when the holding period of a connection is significantly longer than the time required to establish connection, is a necessary consequence of using a centralized control mechanism. The majority of today's telephone switches are operated from a central location. The control mechanism for rapid packet switching applications must be decentralized over the switch fabric and work without full knowledge of the switch's state.

V. DESIGN APPROACH

A. Proposed on Chip Network Topology:

By including a dynamic path-setup technique into the architecture of the multistage switching on chip network topology, it is possible to configure the optimal path for any given traffic pattern at runtime. The data is guaranteed to be permuted when using the circuit-switching method. The suggested architecture includes setting up an Arbiter in a switch circuit and programming it.

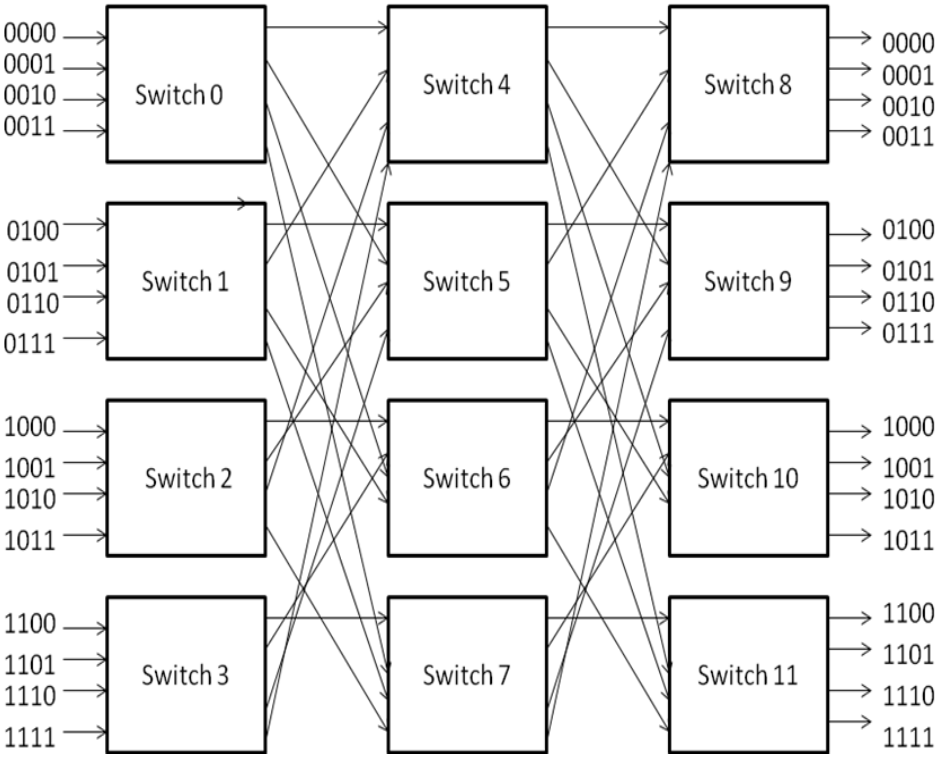


Figure 1: Proposed on-chip Network Topology with Port Addressing Scheme

VI. VERILOG HARDWARE DESCRIPTION LANGUAGE

A computer, or a part of a computer, can be described in detail using a language called a Hardware Description Language. There are various tiers at which a digital system can be described. The registers and inter-register transfers of information vectors are described at a higher level. Register Transfer Level (or RTL) describes this phenomenon. Each of these abstractions is supported by Verilog. While Verilog has many features, this guide will only cover those that are relevant at the RTL level.

Hardware designers in both business and academia commonly work in Verilog, one of two major HDLs. A second language is VHDL. Professionals are now divided over which is superior. Many people think Verilog is less complicated than VHDL. One hardware architect said, "I hope the competition uses VHDL." In 1987, IEEE officially recognized VHDL as a standard, but Verilog is currently being evaluated for standardization.

A. Popularity of Verilog HDL:

- Over time, Verilog HDL has become the de facto standard HDL for describing hardware. There are several helpful tools available in Verilog HDL.
- Verilog HDL is a high-level language for describing hardware that can be used for a variety of purposes. Its syntax is quite close to that of C. Designers that are familiar with C programming will have little trouble picking up. HDL Verilog.
- With Verilog HDL, you can combine models with different levels of abstraction. In this way, a hardware model can be defined in terms of switches, gates, RTL, or behavioral code by the designer. Not to mention, a designer can master stimuli and hierarchical design using the same language.

VII. SIMULATION AND SYNTHESIS TOOLS

A. ModelSim:

ModelSim is a verification and simulation tool for VHDL, Verilog, System Verilog, and mixed language designs. It is divided into four topics, which you will learn more about in subsequent lessons.

- Basic simulation flow.
- Project flow.
- Multiple library flow.
- Debugging tools.

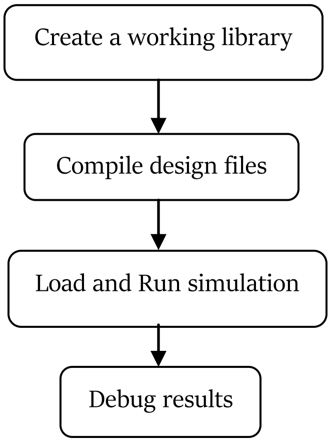


Figure 2: The Fundamentals of Running a Simulation in ModelSim

Simple Steps in a Simulation Figure.7.1 demonstrates the fundamentals of running a simulation in ModelSim. All ModelSim designs are assembled into a library during the process known as "Creating the Working Library." For most ModelSim simulations, you'll begin by making a new working library with the name "work," which is where the compiler sends newly-compiled design units by default. Putting Together Your Plan Once the library is functional; your design units can be compiled into it. All supported platforms share a common library format with ModelSim. Your design can be simulated on any platform without a recompile.

- To Run a Simulation, You Must First Load Your Design into the Simulator After the design has been constructed, it is loaded into the simulator by calling the simulator on the Verilog's top-level module. To start a simulation, you load the design, reset the simulation timer to zero, and then hit "run".
- Debugging Your Results if you don’t get the results you expect, you can use ModelSim’s robust debugging environment to track down the cause of the problem

VIII. SIMULATION RESULTS

A. Top Module 3 Stage Switching Circuit:

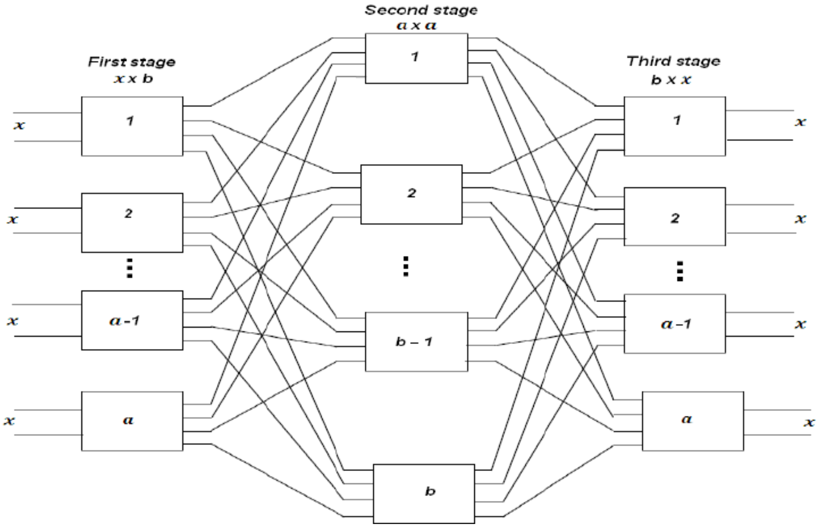
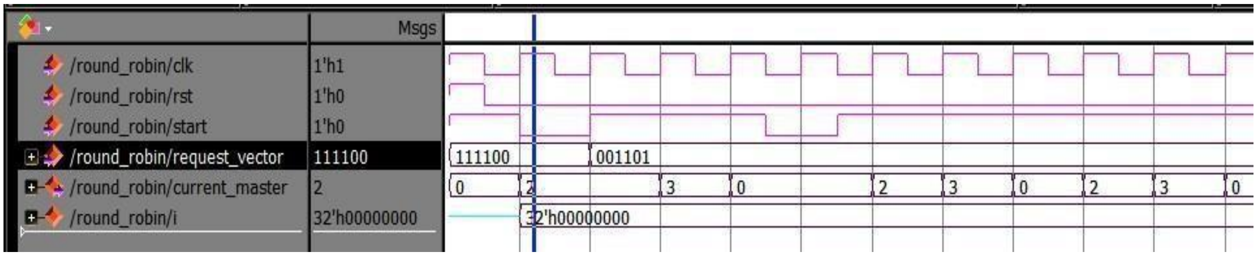


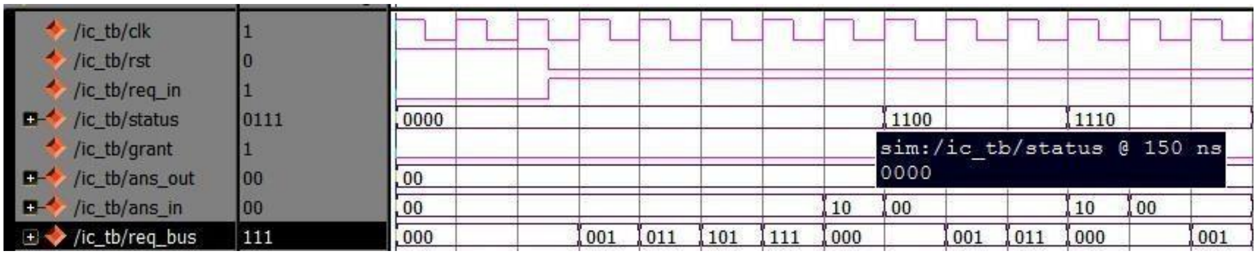
Figure 3: Top Module 3 Stage Switching Circuit

B. ARBITER:



```
Msgs
/round_robin/clk              1'h1
/round_robin/rst              1'h0
/round_robin/start            1'h0
/round_robin/request_vector   111100
/round_robin/current_master   2
/round_robin/i                32'h00000000
```

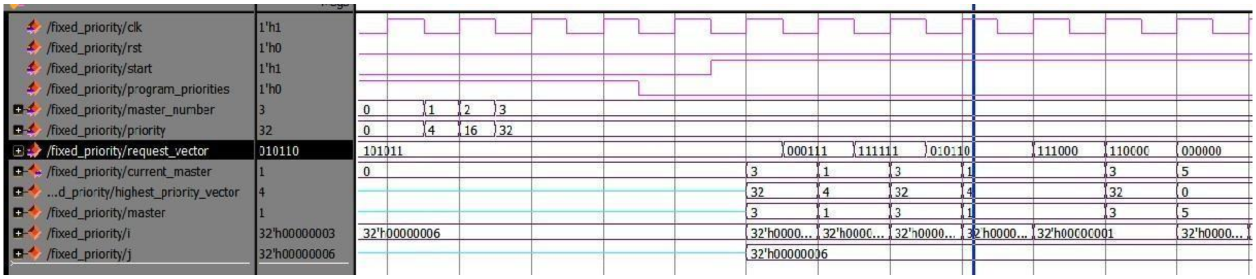
Figure 4: Simulation Results for Round Robin



```
/ic_tb/clk       1
/ic_tb/rst       0
/ic_tb/req_in    1
/ic_tb/status    0111
/ic_tb/grant     1
/ic_tb/ans_out   00
/ic_tb/ans_in    00
/ic_tb/req_bus   111
```

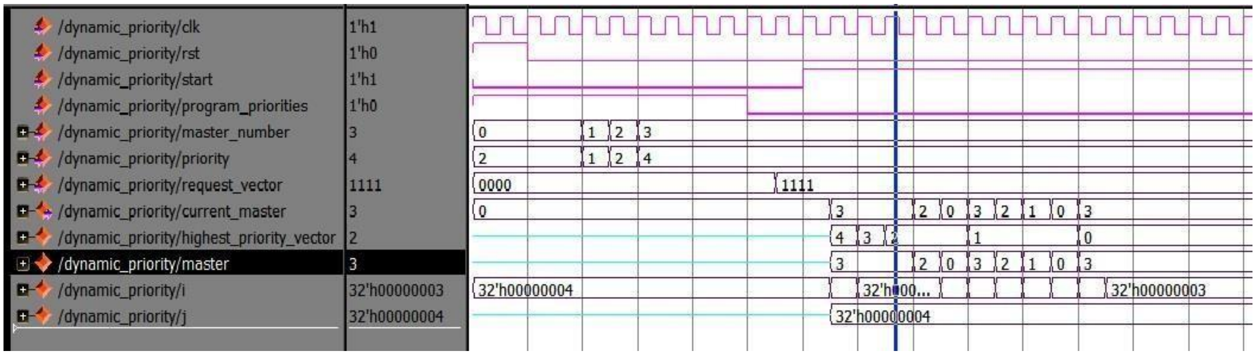
```
sim:/ic_tb/status @ 150 ns
0000
```

Figure 5: Simulation Results for Input Circuit



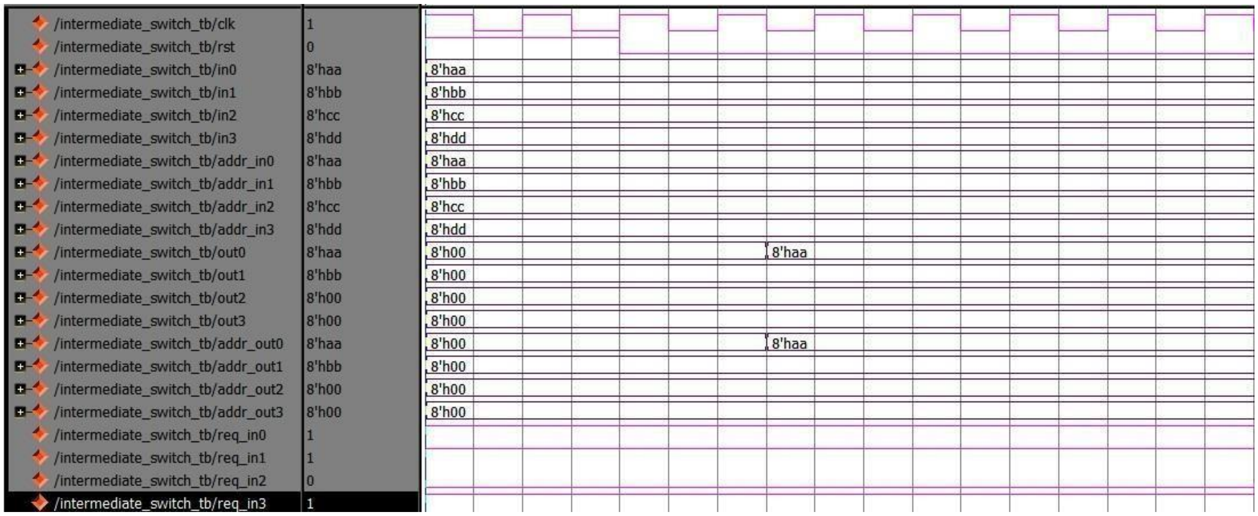
```
/fixed_priority/clk                        1'h1
/fixed_priority/rst                        1'h0
/fixed_priority/start                      1'h1
/fixed_priority/program_priorities         1'h0
/fixed_priority/master_number              3
/fixed_priority/priority                   32
/fixed_priority/request_vector             010110
/fixed_priority/current_master             1
...d_priority/highest_priority_vector      4
/fixed_priority/master                     1
/fixed_priority/i                          32'h00000003
/fixed_priority/j                          32'h00000006
```

Figure 6: Simulation Results for Fixed Priority



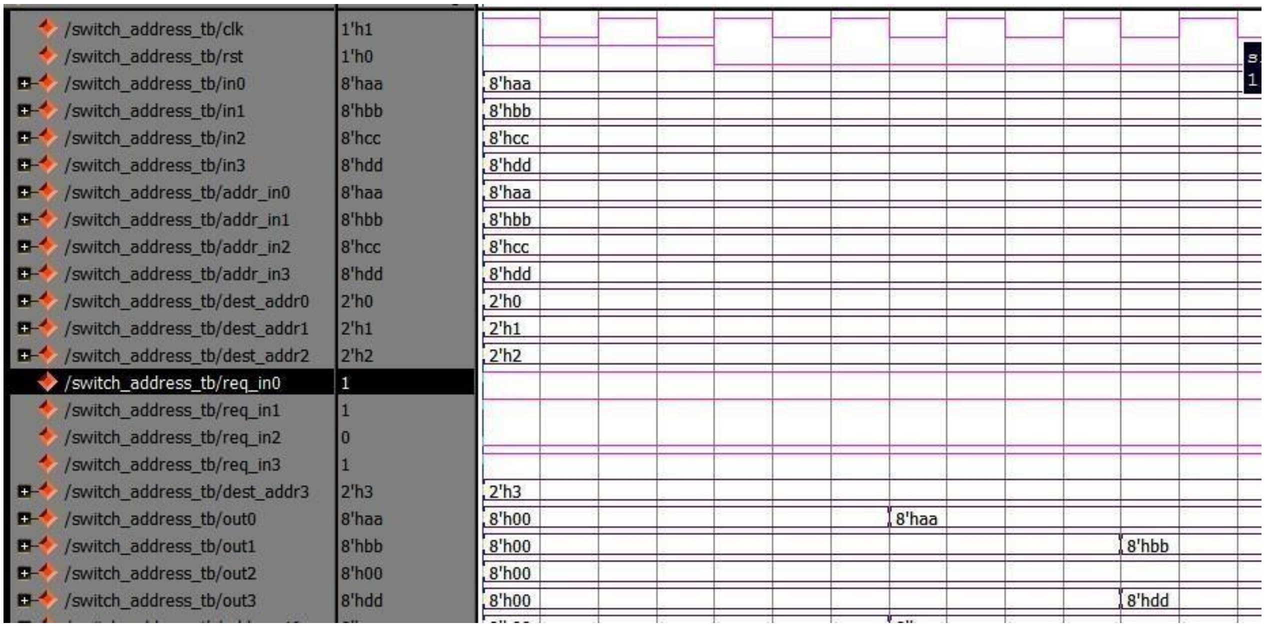
```
/dynamic_priority/clk                      1'h1
/dynamic_priority/rst                      1'h0
/dynamic_priority/start                    1'h1
/dynamic_priority/program_priorities       1'h0
/dynamic_priority/master_number            3
/dynamic_priority/priority                 4
/dynamic_priority/request_vector           1111
/dynamic_priority/current_master           3
/dynamic_priority/highest_priority_vector  2
/dynamic_priority/master                   3
/dynamic_priority/i                        32'h00000003
/dynamic_priority/j                        32'h00000004
```

Figure 7: Simulation Results for Dynamic Priority



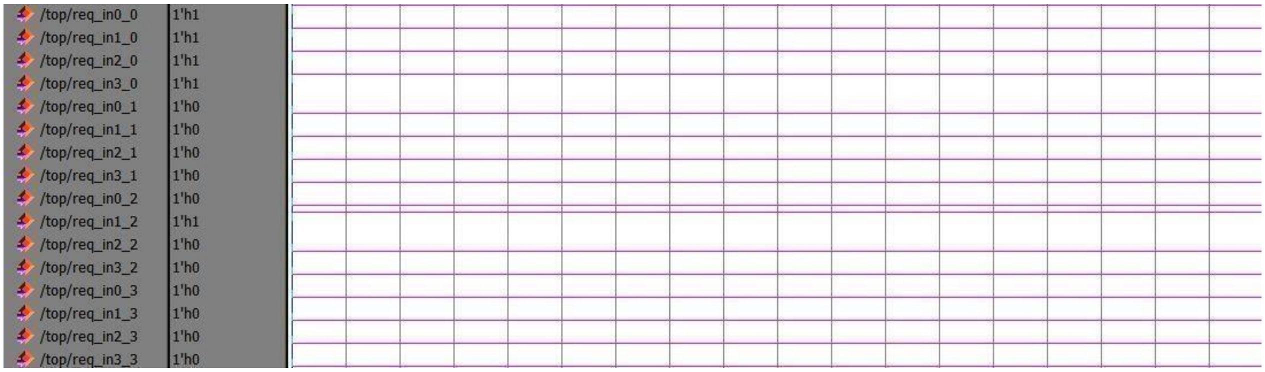
```
/intermediate_switch_tb/clk        1
/intermediate_switch_tb/rst        0
/intermediate_switch_tb/in0        8'haa
/intermediate_switch_tb/in1        8'hbb
/intermediate_switch_tb/in2        8'hcc
/intermediate_switch_tb/in3        8'hdd
/intermediate_switch_tb/addr_in0   8'haa
/intermediate_switch_tb/addr_in1   8'hbb
/intermediate_switch_tb/addr_in2   8'hcc
/intermediate_switch_tb/addr_in3   8'hdd
/intermediate_switch_tb/out0       8'haa
/intermediate_switch_tb/out1       8'hbb
/intermediate_switch_tb/out2       8'h00
/intermediate_switch_tb/out3       8'h00
/intermediate_switch_tb/addr_out0  8'haa
/intermediate_switch_tb/addr_out1  8'hbb
/intermediate_switch_tb/addr_out2  8'h00
/intermediate_switch_tb/addr_out3  8'h00
/intermediate_switch_tb/req_in0    1
/intermediate_switch_tb/req_in1    1
/intermediate_switch_tb/req_in2    0
/intermediate_switch_tb/req_in3    1
```

Figure 8: Simulation Results for Intermediate Switch



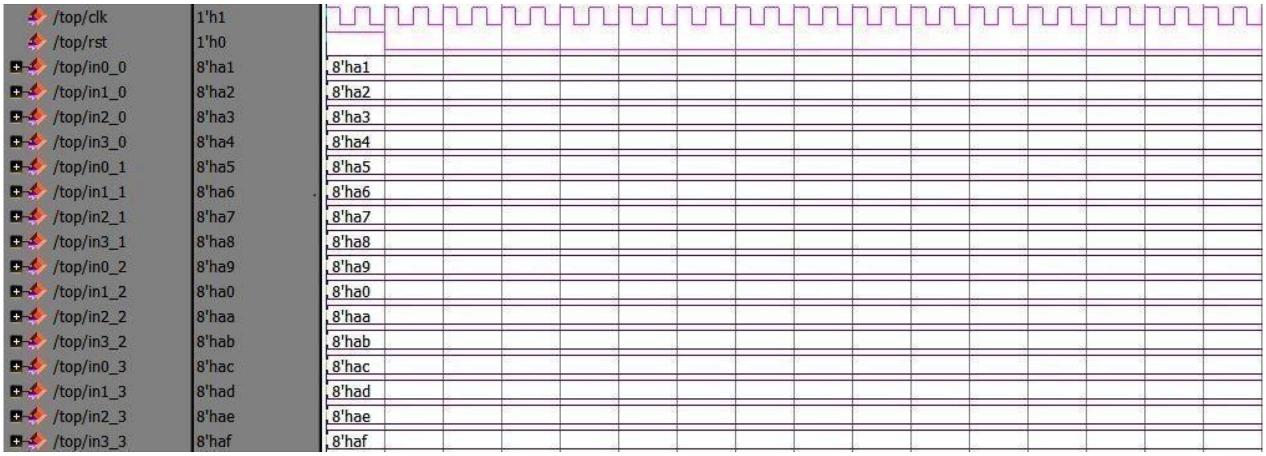
```
/switch_address_tb/clk         1'h1
/switch_address_tb/rst         1'h0
/switch_address_tb/in0         8'haa
/switch_address_tb/in1         8'hbb
/switch_address_tb/in2         8'hcc
/switch_address_tb/in3         8'hdd
/switch_address_tb/addr_in0    8'haa
/switch_address_tb/addr_in1    8'hbb
/switch_address_tb/addr_in2    8'hcc
/switch_address_tb/addr_in3    8'hdd
/switch_address_tb/dest_addr0  2'h0
/switch_address_tb/dest_addr1  2'h1
/switch_address_tb/dest_addr2  2'h2
/switch_address_tb/req_in0     1
/switch_address_tb/req_in1     1
/switch_address_tb/req_in2     0
/switch_address_tb/req_in3     1
/switch_address_tb/dest_addr3  2'h3
/switch_address_tb/out0        8'haa
/switch_address_tb/out1        8'hbb
/switch_address_tb/out2        8'h00
/switch_address_tb/out3        8'hdd
```

Figure 9: Simulation Results for Switch Address



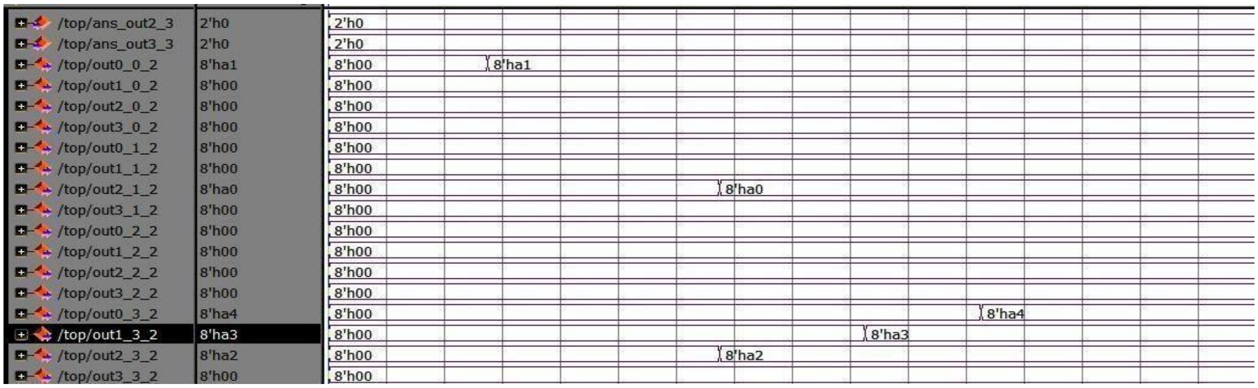
```
/top/req_in0_0  1'h1
/top/req_in1_0  1'h1
/top/req_in2_0  1'h1
/top/req_in3_0  1'h1
/top/req_in0_1  1'h0
/top/req_in1_1  1'h0
/top/req_in2_1  1'h0
/top/req_in3_1  1'h0
/top/req_in0_2  1'h0
/top/req_in1_2  1'h1
/top/req_in2_2  1'h0
/top/req_in3_2  1'h0
/top/req_in0_3  1'h0
/top/req_in1_3  1'h0
/top/req_in2_3  1'h0
/top/req_in3_3  1'h0
```

Figure 10: Simulation Results for Part1 of OCP



```
/top/clk    1'h1
/top/rst    1'h0
/top/in0_0  8'ha1   8'ha1
/top/in1_0  8'ha2   8'ha2
/top/in2_0  8'ha3   8'ha3
/top/in3_0  8'ha4   8'ha4
/top/in0_1  8'ha5   8'ha5
/top/in1_1  8'ha6   8'ha6
/top/in2_1  8'ha7   8'ha7
/top/in3_1  8'ha8   8'ha8
/top/in0_2  8'ha9   8'ha9
/top/in1_2  8'ha0   8'ha0
/top/in2_2  8'haa   8'haa
/top/in3_2  8'hab   8'hab
/top/in0_3  8'hac   8'hac
/top/in1_3  8'had   8'had
/top/in2_3  8'hae   8'hae
/top/in3_3  8'haf   8'haf
```

Figure 11: Simulation Results for Part2 of OCP



```
/top/ans_out2_3  2'h0    2'h0
/top/ans_out3_3  2'h0    2'h0
/top/out0_0_2    8'ha1   8'h00   8'ha1
/top/out1_0_2    8'h00   8'h00
/top/out2_0_2    8'h00   8'h00
/top/out3_0_2    8'h00   8'h00
/top/out0_1_2    8'h00   8'h00
/top/out1_1_2    8'h00   8'h00
/top/out2_1_2    8'ha0   8'h00   8'ha0
/top/out3_1_2    8'h00   8'h00
/top/out0_2_2    8'h00   8'h00
/top/out1_2_2    8'h00   8'h00
/top/out2_2_2    8'h00   8'h00
/top/out3_2_2    8'h00   8'h00
/top/out0_3_2    8'ha4   8'h00   8'ha4
/top/out1_3_2    8'ha3   8'h00   8'ha3
/top/out2_3_2    8'ha2   8'h00   8'ha2
/top/out3_3_2    8'h00   8'h00
```

Figure 12: Simulation Results for Part3 of OCP

IX. CONCLUSION

On-chip network architecture was developed to accommodate various traffic scenarios for MPSoC-based projects. The suggested solution allows for arbitrary traffic permutation at runtime with little implementation complexity by employing a circuit-switching strategy in conjunction with a dynamic path-setup scheme under a closed network topology. Networks with a Closed Topology can function at frequencies up to 100 MHz and data rates up to 30 Gbps. We can have a constant latency from one node to the next by employing circuit switching. Once a connection is made, it is maintained until the final payment has been made. Absurdly Prosperous When establishing a data communication link, the backtracking routing approach is utilized. In probing, dynamic path construction is crucial for establishing link.

X. FUTURE SCOPE

Power is wasted because of the self-switching and cross-coupling capacitance nets in a Network-On-a-Chip link. We employ a Bus- Invert method encoding technique to reduce the amount of dynamic power needed.

XI. REFERENCES

[1] De Micheli, Giovanni, and Luca Benini. "Networks on chips: 15 years later." Computer 50.05 (2017): 10-11.

[2] Kumar, Shashi, et al. "A network on chip architecture and design methodology." Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002. IEEE, 2002.

[3] Fu, Yaosheng, and David Wentzlaff. "PriME: A parallel and distributed simulator for thousand-core chips." 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2014.

[4] Thul, Michael J., Frank Gilbert, and Norbert Wehn. "Concurrent interleaving architectures for high-throughput channel coding." 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).. Vol. 2. IEEE, 2003.

[5] Hu, Wen-Hsiang, Jun Ho Bahn, and Nader Bagherzadeh. "Parallel LDPC decoding on a network-on-chip based multiprocessor platform." 2009 21st International Symposium on Computer Architecture and High Performance Computing. IEEE, 2009.

[6] Moussa, Hazem, Amer Baghdadi, and Michel Jezequel. "On-chip communication network for flexible multiprocessor turbo decoding." 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications. IEEE, 2008.

[7] Martina, Maurizio, et al. "On chip interconnects for multiprocessor turbo decoding architectures." Microprocessors and Microsystems 35.2 (2011): 167-181.

[8] Yin, Shihui, and Jae-Sun Seo. "A 2.6 TOPS/W 16-bit fixed-point convolutional neural network learning processor in 65-nm CMOS." IEEE Solid-State Circuits Letters 3 (2019): 13-16.

[9] Yasrebi, Shahrouz, et al. "A fuzzy integrated congestion-aware routing algorithm for network on chip." Frontiers of Information Technology & Electronic Engineering 22.5 (2021): 741-755.