ISSN 2395-1621

Development of Multi-Processor System on chip using Soft core: A Review

Prashant S. Titare, D. G. Khairnar

[1]Research Scholar, E&TC Department, D Y Patil College of Engineering, Akurdi, Pune, India

[1]pstitare@dypcoeakurdi.ac.in

[2]Professor, E&TC, D Y Patil College of Engineering, Akurdi, Pune, India

[2]dgk@ee.iitb.ac.in

ABSTRACT

Review of multiprocessor systems on a chip (MPSoC) which provides parallel computing of multiple processes is presented in this paper. Today's world demand for high performance with low power consuming embedded devices. The demand to increase speed and to satisfy the requirements of modern embedded application, like image processing, audio or video encoding or decoding, network application, and many more, has encouraged the development of MPSoC. The solution to increase speed with minimal increase in a computational power, is to use several processors and execute parallel tasks on them. Implementation of such architecture on a single chip (MPSoC) is feasible and appealing, due to the advancements in FPGA Technology. This paper provides a survey of implementation of MPSoC using soft-core processors like MicroBlaze, NIOS-II, Leon, Xtensa in addition to their comparison. Also it presents the speed improvement method along with review of different topologies.

Keywords— VLSI, FPGA, MPSoC, soft core processors.

ARTICLE INFO

Article History

Received: 8th March 2020

Received in revised form :

8th March 2020

Accepted: 10th March 2020

Published online :

11th March 2020

I. INTRODUCTION

Embedded systems, initially, were being used as simple controllers for specific application. Such system requires more computational power to execute the need of modern application, like encoding or decoding of audio or video information, image processing, etc. and hence, the multiprocessors system on a chip (MPSoC) is one of the alternatives to deal with such escalating computational needs. [1][15][16].

An MPSoC is a multi-processor on a single silicon-chip which may include about two or more number of processor-memory modules (PMMs). Considering ten PMMs, if the memory modules are grouped together into a single first level cache (L1) that is shared between all the available cores then it is called a multi-core processor. The term multi core architecture is used if two or more number of cores is developed on the chips that are interconnected together by appropriate resources. [22].

MPSoC have become the important necessities of fast growing technology. Faster speed along with efficiency is the ultimate requirement of embedded consumers. All these requirements are possible in small integrated chips through MPSoC. Thus MPSoCs have emerged as an important class in the field of micro-electronics and VLSI (Very Large Scale Integration). MPSoC shows an absolute method to incorporate multiple processing cores on a single silicon chip. It is promising to recognize following five major steps for proper designing of an MPSoC: (a) to develop an application; (b) to configure platform accurately; (c) To program a code; (d) To map an application on to the platform; (e) And to debug.

Most of the MPSoC development work is centered at one of the steps like platform configuration or application mapping [2].

According to the system architecture model, MPSoCs are classified as Homogeneous and Heterogeneous. The Homogeneous MPSoC has processors of similar architecture while Heterogeneous type has different processor architecture on same platform.

Following are the certain levels of design space exploration, enabled by MPSoC:

- Network on chip (NoC): It explores various topologies, routing algorithms, priority schemes, etc;
- Operating System (OS): It is a program to regulate different scheduling algorithm. For example, Task migration, Dynamic Voltage and Frequency Scaling (DVFS), distributed memory architecture,

monitoring their respective parameters are some of the tasks controlled by OS.

- Network Interface (NI): Exploration to NI is used to guarantee QoS (Quality of Service) for injection control.
- Processing Elements (PEs): It is used to examine various processor architectures, according to the application requirement.
- Power dissipation: There are different types of power dissipation at processor level, static or dynamic based on energy dissipated and frequency.
- Inter-board communication: Central system monitors and coordinates hardware operations through MPSoC. [3]

This paper provides a review on establishing the environment for MPSoC using Micro-blaze, NIOS-II, Leon, and Xtensa. The communication architecture along with different topologies is explained.

This paper is organized in following sequence, section II: Soft processor core architecture is explained along with feature; Section III: Comparison of different soft core processors; Section IV: Communication architecture; Section V: MPSoC Architecture; Section VI: Communication topologies; Section VII: Mapping for MPSoC; Section VIII: The complete system and Section IX: Conclusion.

II. SOFT PROCESSOR CORE ARCHITECTURE

A. Micro-Blaze

The embedded processor named Micro-Blaze is a 32-bit soft core reduced instruction set computer (RISC) which is optimized for implementation on Xilinx Field Programmable Gate Array (FPGA).

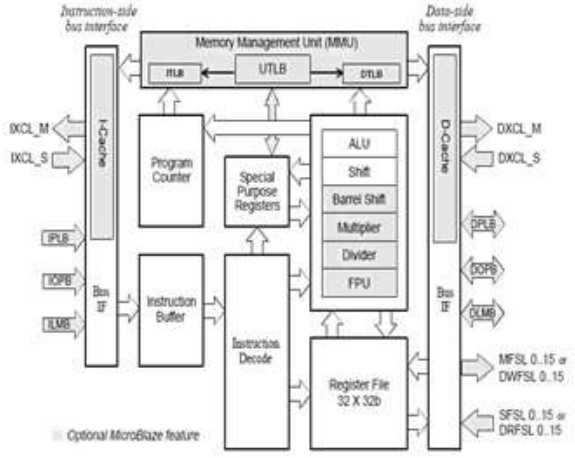

Fig. 1 Microblaze Core Block Diagram [9]

The Micro-Blaze processor is exceedingly configurable, as it allows selecting a definite set of features required by design. It implements Harvard architecture. It signifies that it has different interfacing units for data bus and instruction bus access. Each bus unit is further divided into a Local Memory bus (LMB) and On-Chip Peripheral Bus (OPB). LMB provides an access to on-chip dual port block RAM. On-chip and off-chip memory and peripherals, both are interfaced using OPB. The Micro-Blaze core also facilitates with 8 input and 8 output interfaces to the Fast Simplex Link (FSL) buses. These FSL buses are unidirectional dedicated communication channels, detail explanation is in section IV. [1] [9]. Micro-blaze is a soft core specifically designed for Xilinx FPGAs.

B. NIOS-II

Nios-II is a soft core processor explicitly designed for Altera FPGA devices. It is an instruction set architecture (ISA). Advantage of having ISA is that it implements the instructions for certain functional units. It is a hardware design which implements the Nios II instruction set. The processor core only includes the required circuit to implement the NIOS II architecture. It does not comprise of any peripheral interface unit or any external connection logic. [12]

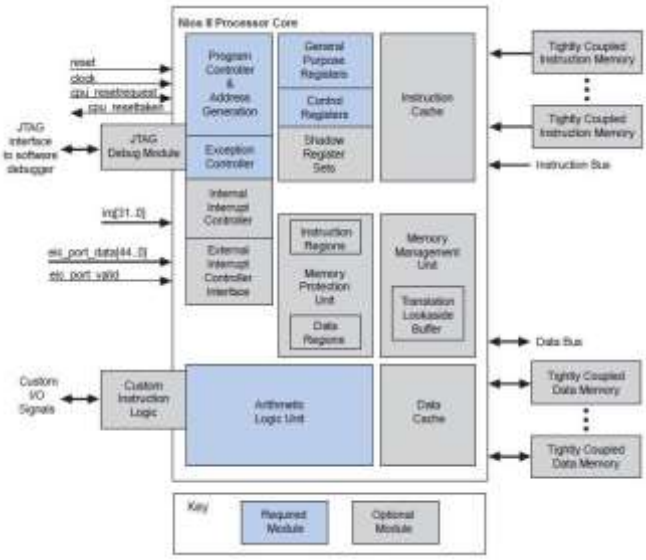

Fig. 2 NIOS II Core Block Diagram [12]

Depending on the memory size, data-width and peripherals required, NIOS II architecture can be configured. NIOS supports 6-stage pipeline and Harvard memory architecture. Customization of instruction is possible with this architecture to improve performance. [23]

There are many other soft-core processors available like Leon by Gaisler research and Xtensa series from Tensilica. Leon3 supports maximum operating frequency of around 400 MHz by using ASIC implementation while MicroBlaze and NIOS-II has 200 MHz on their respective FPGA platform. However, Xtensa has the highest design flexibility since unlimited custom instructions and execution units can be implemented on the processor's core. Comparison of such soft cores is explained in further section. [25]

III. COMPARISON OF SOFT-CORE PROCESSOR

Following table shows the differences between different soft-core processors.

Table 1: Soft-core processor comparison [25]

Properties	Micro - Blaze	NIOS-II	LEON3	Xtensa XL
FPGA/ASIC Technology	Virtex (Xilinx Based)	Stratix (Altera based)	0.13 micron technology	0.13 micron technology
Speed MHz (ASIC / FPGA)	200 MHz (FPGA)	200 MHz (FPGA)	125 MHz/ 400 MHz (FPGA/ASIC)	350 MHz (ASIC)
Reported Speed in DMIPS	166	150	85	N/A
Standard for Floating Point Unit (optional)	IEEE 754	IEEE 754	IEEE 754	IEEE 754
Cache Memory	64 KB	64 KB	64 KB	64 KB
Pipeline	3 stage	6 stage	7 stage	5 stage
Custom Instruction	None	256 instruction approximately	None	Unlimited
Size of Register File	32	32	2-32	32 / 64
Area	1269 LUTs	700-1800 LEs	3500 LUTs	0.26mm^2
Implementation on hardware	FPGA	FPGA	FPGA/ASIC	FPGA, ASIC

IV. COMMUNICATION ARCHITECTURE

A. Fast Simplex Link (FSL) Overview

FSL buses are used as a communication link between MicroBlaze cores. MicroBlaze has eight input and eight output FSL interfaces. These FSL channels are dedicated, unidirectional, point to point data streaming interfaces. The width of FSL interface is 32 bit. Data and control words can be exchanged through FSL channel. The FSL interface can have maximum speed up to 300 MB/sec depending on the target device. The FSL bus system is an ideal choice for inter processor communication (like MicroBlaze processor to another MicroBlaze processor) or Input-Output streaming communications [11].

The features of the FSL bus interface are:

- Unidirectional and Non-arbitrated communication
- Dedicated, point-to-point communication
- Support for data and instruction communication
- 600 MHz standalone operation
- Configurable data size
- FIFO based communication [1]

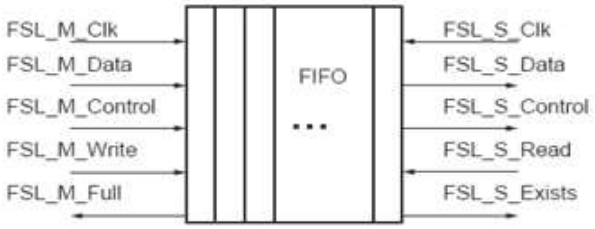

Fig. 3 FSL bus signal [11]

One Master drives FSL bus and in turn FSL drives one Slave. Above figure signifies the working principle of the FSL bus system and its signals.

FSL link allows data transfer through group of macro supported by Xilinx EDK (Embedded Development Kit). Reading and writing of FSL allows communication between processors. [1]

B. Heterogeneous IP Block Interconnection (HIBI)

The platform includes multiple Altera Nios II soft-core processors and custom hardware accelerators, which are interfaced using Heterogeneous IP Block Interconnection (HIBI) communication architecture. HIBI allows multiple NIOS II processors to interconnect.

Maximum efficiency with minimum energy per transmitted bit together with quality of service (QoS) is the main objective of HIBI interconnects. [5]

An automated synthesis method for generating hierarchical bus structures was presented by Ryu and Mooney[27][5]. Bus topology can improve performance and speedup by 2.4 times, as presented by authors, Ryu et al., They also proved that number of gates required is reduced by 37% [17]. Lahiri et al., shows the utilization of automated design space exploration with two application examples. They concluded that ideal communication architecture could achieve 3.2 to 4.9 times speedup over the single shared bus whereas multiple buses could achieve speedup of 1.8 to 2.7 times. However, the area of a network was compromised with speedup [18].

Over a single shared bus, HIBI interconnect use hierarchical structures in order to improve performance and scalability. Zeferino et al., concluded that mesh based NoC is having better switching of tasks as compared to simple bus [5][19]. Network topologies are detailed in Section VI. They estimated that mesh topology performance is seen improved if number of processor is around 16 to 25, assuming that same frequency is present over an entire network.

If signal propagation delay on wires is more, then mesh topology demonstrates higher efficiency due to shorter links between the nodes. They demonstrated an MPSoC system of 8 processors using HIBI on FPGA. This system used 36400 logical elements, which is approximately 88% of logical resources available on Altera FPGA (Stratix 1840). [5][20]

As the HIBI interconnection can be scaled upwards to certain topologies, as explored in Section VI, they are referred as NoC (Network on Chip). Its hierarchical nature provides freedom of choice for selection of capacity for data transfer and also the properties of programming become simpler due to this. Developing

such architecture with multiple processors is termed as MPSoC, as explained further.

V. MPSOC ARCHITECTURE

VLSI, the branch of microelectronics starts evolving since 1990. The single processor technique developed for an embedded system provides application from communication domain, networking domain, etc. However, multimedia application requires system having multiple processors to compute the given task in shorter stipulated time as compared to single processor time. Processors involved in multiple processing systems should have the capacity of parallel programming to improve its performance. Very Long Instruction Word (VLIW) based processors are the example for the same, having parallelism approach for the execution of a task. On the other hand, ASIC architecture has specific blocks and is not preferred for the design of general purpose application. Following block diagram corresponds to multiple processor architecture.

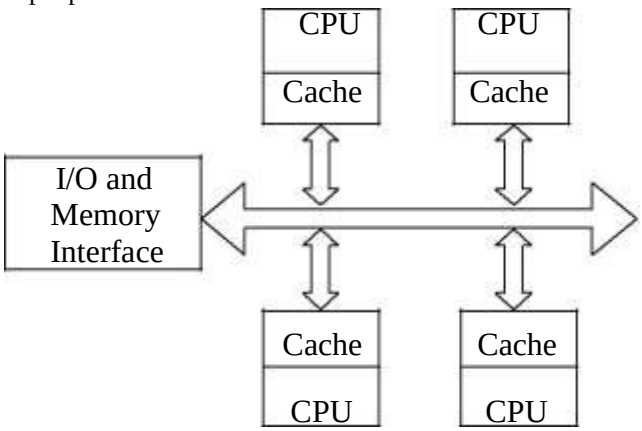

Fig. 4 Block diagram of MPSoC platform [7]

MPSoC architecture includes multiple processors which collectively has CPU and memory together, to control all the peripherals (I/O or Input /Output devices) over a network. Memory sharing, data transfer and interconnection are main concern of multi-processing system. [7] Almasi and Gottlieb, et al., defined multiprocessors as parallel processing elements that collectively cooperate and communicate to compute complex problem at faster rate. [24][60]

Designing an environment for MPSoC architecture comprises of CPU, cache memory, I/O units and memory interface, as shown in figure 4. CPU interconnection supports higher level of component integration which will reduce the design area and design time that too without sufficient loss in efficiency. MPSoC environment can be designed with hardware-software co-design, including synthesizable hardware interfaces, hardware accelerators, operating systems (OS) and device drivers. All these units are controlled by OS & Application Programming Interface (API) [8][63]. Thus all the above mentioned units along with communication link (like FSL, HIBI) complete the MPSoC architecture. Subsequent section will elaborate on methods to interconnect multiple processors in a network.

VI. COMMUNICATION TOPOLOGIES

Multiple processors can be inter-connected to each other using different topologies. Selection of topology depends on the type of soft-core to be used. Some of the network topology used for connecting various processors (CPU) in a cluster through the FSL link or HIBI communication architecture for point to point data transfers is discussed as follows: [1]

- *Bus topology:* In this topology, all the computers are interconnected using a single line through trans-receivers. All lines should be closed with matched termination. Speed of operation and network performance depends on the number of CPUs available on a network. If one CPU transmits a message, then rest all the CPUs will be waiting to transmit their data. At a time only one can send data. The complete communication network will fail if there is even a single break in the main line. Such bus networks are also called passive topology as the computers or CPUs on the bus only responds (or listen) to data transmitted. They do not transfer data from CPU to CPU.

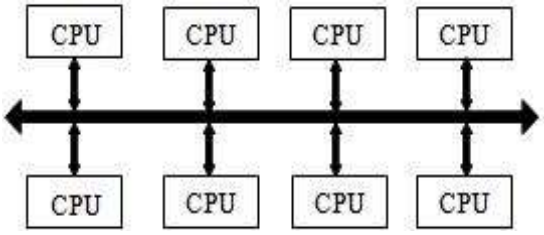

Fig. 5 Bus network

- *Hierarchical bus network:* It is also termed as split-bus architecture. It is a network that connects two or more buses using Bus Bridge. This bus bridge is a controllable connection point, which when enabled makes the connection otherwise disconnect it.

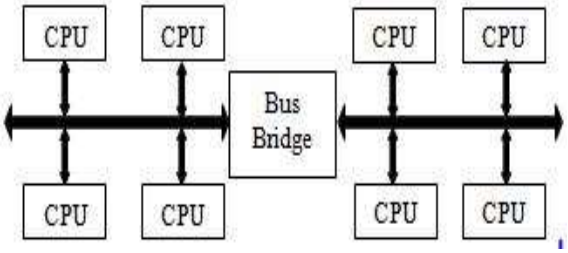

Fig. 6 Hierarchical Bus network

- *Ring Topology:* A Network in which one CPU is connected to next CPU, in continuous manner and in turn last CPU is connected to first CPU, forming a ring pattern. The data is transferred from one node (CPU) to another until it reaches the last destination node present in a network. The main drawback of this topology is that the data propagation delay will be longer if transmitting CPU and receiving CPU are present at longer distance in a ring. More the number of nodes, longer will be the propagation delay.

However, it has the advantage lesser circuit complexity as the number of interconnections is less.

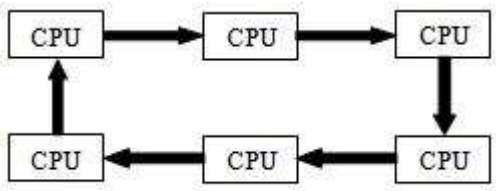

Fig. 7 Ring network

- *Mesh Network:* It is a network where each and every node (CPU) is interconnected to each other. This type of topology has least travelling time for data transfer between any nodes over a network. This is because of direct connection between transmitting node and receiving node. The major drawback of such topology is that the circuit complexity increases as the number of interconnection increases due to increase in the number of nodes in a network.

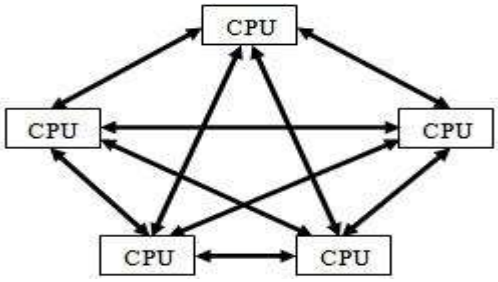

Fig. 8 Mesh network

- *Star Topology or network:* This topology has a central node connected to each and every node present in a network. Its structure seems to be like a star and hence the name. One centralized controlled called Master CPU and multiple slave type of configuration can be achieved in such topology. Due to master slave architecture, central CPU decides which task has to be allocated to which CPU, and can also collect results from all these CPUs. The main drawback is if the central node fails, then entire control is lost and the complete system fails. As all the operations are regulated through central node, then communication bottleneck can occur for the large bulk of data operations.

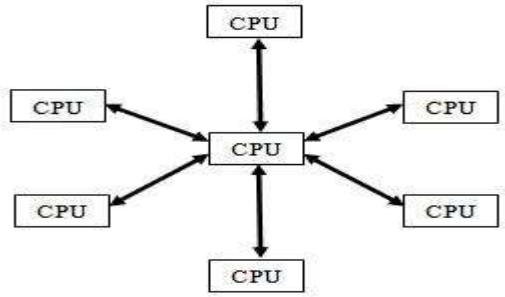

Fig. 9 Star network

Multi-Star network can be created by grouping individual star network. The central star network is responsible for control of multiple star networks attached to it.

All these topologies have some advantages and disadvantages. Depending on the application requirements, a particular type of topology can be selected and mapped on MPSoC.

VII. MAPPING FOR MPSOC

Mapping of tasks for MPSoC means tasks can be mapped to different CPUs as per topology selected. Previous literature shows the availability of multiple task mapping algorithms. However, the proper approach has to be followed while mapping the tasks on MPSoC. Different types of approach for task mapping are mentioned below which are classified on the basis of timing instants when these tasks are mapped:

- *During design time:* if the static or offline mapping approach is preferred then MPSoC resources can be explored at better level using complex process. The only major issue with static mapping is to handle dynamic tasks.
- *During run time:* the dynamic or also called online mapping approach needs simple but faster processes as it is handling application when it is in execution mode.
  There are two different dynamic mapping approaches for designing an MPSoC, as mentioned below:
- *Mapping with resources reservation:* This method helps to verify whether enough MPSoC resources are available or not before mapping their respective application tasks on system.
- *Mapping without resources reservation:* This approach helps to map the initial task of the application keeping the remaining tasks in waiting state. Task belonging to wait-state are mapped whenever required. Hence in this case, application execution will start faster, but may wait for other resources to get into ready state.

Run time mapping of tasks in MPSoC is a need for the dynamic mapping approach. Such mapping can be controlled using:

- *Centralized:* One processing element will be centralized, which will control, regulate and combine the mapping process. This central (or single) master will manage the mapping of resources. Due to single centralized control, bottleneck is observed with respect to scalability and performance.
- *Distributed:* Multi-master or multi-cluster approach is preferred. One processing element in each network or in each cluster is responsible for mapping of tasks. It is complicated but better approach with respect to performance and speed. [6]

Thus such mapping of tasks on MPSoC with communication link forms the complete system as discussed in further section.

VIII. THE COMPLETE SYSTEM

MPSoC system is collectively a combination of multiple processing cores, having tasks mapped onto it with the respective algorithms. These cores are interconnected to each other using communication link like FSL, etc depending on the type of FPGA. A Single Processor system on a chip (SPSoC) consists of one CPU, program and data memory, timer and an application based peripheral. The major difference between SPSoC and MPSoC is that in MPSoC, the number of CPU is not limited to one. Depending on an application requirement, program or data memory can be configured. Additional program memory can be used to operate them in parallel. So, MPSOC provides an advantage of speed through parallelism, thus improves the overall performance. [1]

P. Huerta et al., have defined parameters named 'Speedup' and 'Efficiency' to check MPSoC performance as elaborated in the next section. [1]

A. Speedup and efficiency

Speedup is the ratio of time taken by the single processor (ts) to the time taken by the 'p' number of processor (tp) executing in parallel for MPSoC. Ideally, this ratio should have the number equal to the number of processor (p). It indicates as the number of processor increases, speedup increases. However, speedup never remains equal to p, as there is always a communication overhead. It means certain time is consumed during task mapping and inter-communication, because of which it is lower than p. [1]

$$\text{Speedup} = \frac{ts}{tp}$$

For parallel operating processors in MPSoC, performance can also be measured through efficiency. Efficiency is defined as the ratio of speedup to the number of processor (p). An ideal value of efficiency should be one. [1]

$$\text{Efficiency} = \frac{\text{speedup}}{p}$$

B. Matrix Multiplication Application

The Matrix multiplication of integers and floating point numbers were performed by P.Huerta, et al., [1]. An application of matrix multiplication was tested by them to check the parallel execution of the task. The demonstration involves a technique or an algorithm for computing matrix multiplication application. This technique was implemented by transmitting rows of first matrix and entire second matrix to each processor present in a network. Then each processor computes the matrix multiplication for that particular row and corresponding second matrix received. Later after computation, this processor returns the result back to main central processor acting as master processor. Master or central processor is responsible to collect results from each processor and produce output in desired form.

By using this application, authors [1] performed different test by varying different parameters like matrix size, data types, number of processors, etc. Such tests were performed to check the application execution time. The total time consumption includes time taken for the collection of data (matrix values) from memory, transferring data to each processor, processing (multiplication) of the collected data, the collection of results from each processor and storing final results back to memory. [1]

They concluded that the results were not as good as it could be expected since the efficiency gets affected as the number of processors increases. However, this effect is less as compared to speed improvement obtained from parallelism. For example, if the matrix multiplication by single processor takes 10 micro second (µs) to conclude with results, then ideally dual processor should compute it in 5µs. But this time is more than 5µs. This is due to communication overhead. Time taken for distributing the data in multiple processors and collecting it back gets added in total time. However, due to parallelism, the overall time consumed gets reduced as compared to the single processor system.

P. Huerta et al., [1] performed the matrix multiplication on floating point values as well. As compared to integer value, the floating point matrix multiplication consumes more time as the information contents present in the data are larger than the integer values. However, the time required to transfer data will remain same. This extra time consumed in the floating point operation as compared to integer matrix multiplication can be reduced using floating point processors. As the processing time for floating point multiplication is longer, improvement in efficiency and speedup for MPSoC is observed. However, in floating point as well, there will be communication overhead for the transfer of matrix on multiple processors and collecting results back. To overcome this problem of overhead, the common message can be broadcasted to all the processors, was suggested. [1] This will reduce the time required to transfer the same message to each processor individually. With respect to the previous application of matrix multiplication, second matrix is common to all processors and hence can be broadcasted to all the processors. This technique helps to reduce the sufficient amount of time.

IX. CONCLUSION

This paper presents the review on MPSoC architecture for different soft-core processors like Microblaze and NIOS-II along with the link for communication between the multiple processors, as per survey from referred papers. As per review and to the best of our knowledge, it is observed that FSL inter-processor link is better for Microblaze processor due to its high data rates capacity and HIBI interface for NIOS-II processor because of its scalable and programmable property.

The different types of communication topologies are studied and the reason why particular topology is to be selected is elaborated from the survey. From review, it can be concluded that star topology is better for Microblaze as master processor, or the central node decides the task distribution and overall processing. Entire control is with master Microblaze which provides

final results of application. Also from investigation it is observed that Microblaze can have 8 interconnections. From this it can be concluded that 8 subsystems having 8 processors in it can be connected to master Microblaze, forming multi-star network. As per review obtained from the study of MPSoC for NIOS II architecture, it is observed that mesh topology provides better performance because of its point to point dedicated interconnection through HIBI interface.

Study on MPSoC also involves different algorithm implementation to improve application speed, to reduce the power consumption, without contributing external hardware into it. Also clocking strategies to improve speed, porting of different OS on multiple processors will also contribute some advancement on this.

REFERENCES

[1] P.Huerta, J.Castillo, J.I.Mártinez, V.López, "A MicroBlaze based Multiprocessor SoC" HW/SW Codesign Group, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid Spain.' ResearchGate – 29226 7954, pg 423-430, v'2005.

[2] Eduardo W. Wächter, Carlo Lucas, Everton A. Carara, Fernando G. Moraes, "An Open-source Framework for Heterogeneous MPSoC Generation" FACIN - PUCRS - Av. Ipiranga 6681-Porto Alegre - 90619-900 – Brazil, IEEE '2012

[3] Eduardo Weber Wächter, Adelcio Biazi, Fernando G. Moraes, "HeMPS-S: A Homogeneous NoC-Based MPSoCs Framework Prototyped in FPGAs," PUCRS – FACIN – Av. Ipiranga 6681 – Porto Alegre – 90619-900 – Brazil., IEEE 2011

[4] Tero Arpinen, Petri Kukkala, Erno Salminen, Marko Hännikäinen, and Timo D. Hämäläinen, "Configurable Multiprocessor Platform with RTOS for Distributed Execution of UML 2.0 Designed Applications," Tampere University of Technology, Institute of Digital and Computer Systems, Korkeakoulunkatu 1, FI-33720 Tampere, Finland., ISSN: 1558-1101 IEEE July 2006.

[5] Erno Salminen, Tero Kangas and Timo D. Hamalainen, "HIBI Communication Network for System-on-Chip," Tampere University of Technology, P.O. Box 553, FIN-33101, Tampere, Finland. Journal of VLSI Signal Processing 43, 185–205, Springer Science & Business Media, LLC, Netherlands. DOI: 10.1007/s11265-006-7270-6, May 2006.

[6] Marcelo Mandelli1, Alexandre Amory1, Luciano Ost2, Fernando G. Moraes1, "Multi-Task Dynamic Mapping onto NoC-based MPSoCs," 1PUCRS – FACIN – Av. Ipiranga 6681 – Porto Alegre – 90619-900 – Brazil, 2LIRMM – 161 rue Ada, Cedex 05 – Montpellier – 34095 – France, DOI: 10.1145/2020876.2020920, August 2011.

[7] Wayne Wolf, Ahmed Amine Jerraya, and Grant Martin, "MPSoC Technology". IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 27, No. 10, DOI 10.1109/Tcad.2008.923415, pg no 1101 to 1113, October 2008.

[8] Wander O. Cesário, Damien Lyonnard, Gabriela Nicolescu, Yanick Paviot, Sungjoo Yoo, and Ahmed A. Jerraya, "Multiprocessor SoC Platforms: A Component- Based Design Approach," TIMA Laboratory, IEEE Design & Test of Computers, IEEE Design and Test of Computers 19(6):52 - 63 · December 2002, DOI: 10.1109/MDT.2002.1047744 (2002 IEEE).

[9] MicroBlaze Processor Refrence Guide, Embedded Devel;opment kit 13.1, Xilinx, http://www.xilinx.com, September 2018

[10] EDK Concepts, Tools, and Techniques, Xilinx, http://www.xilinx.com, July 2018

[11] LogiCORE IP Fast Simplex Link (FSL) V20 Bus (v2.11f), Xilinx, http://www.xilinx.com, November 2018

[12] NIOS II Processor Refrence Handbook, Altera, http://www.altera.com, August 2017.

[13] Nios II Integrated Development Environment, Altera Corporation May 2007, http://www.altera.com, January 2017

[14] Nios II Performance Benchmarks, Altera Corporation July 2013, http://www.altera.com, August 2017

[15] Wolf, W: "The Future of Multiprocessor Systems-on-chip" Proceedings of the 41st annual Design Automation Conference (DAC'04)SanDiego, New York, USA ©2004 ISBN:1-58113-828-8 DOI 10.1145/996566.996753, Pages 681-685, June 07 - 11, 2004.

[16] Wolf, W: "Multimedia Applications of Multiprocessor Systems-on-Chip". Proceedings of the Design, Automation and Test in Europe Conference (DATE'05). IEEE Computer Society Washington, DC, USA ©2005, Vol 3, pages 86-89, ISBN:0-7695-2288-2 DOI 10.1109/DATE.2005.217 March 2005.

[17] K. K. Ryu and V. J. Mooney III, "Automated bus generation for multiprocessor SoC design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 11, pp 1531–1549, November 2004.

[18] K. Lahiri, A. Raghunathan, and S. Dey, "Design space exploration for optimizing on-chip communication architectures," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 6, pp. 952–961, 2004.

[19] C. A. Zeferino, M. E. Kreutz, L. Carro, and A. A. Susin, "A study on communication issues for system-on-chip," Proceedings of the 15th Symposium on Integrated Circuits and System Design (SBCCI), SBCCI, Porto Alegre, Brazil, December 2002, pp. 121–126.

[20] Erno Salminen, Ari Kulmala, and Timo D. Hämäläinen "HIBI-based Multiprocessor SoC on FPGA" Published in IEEE International Symposium on Circuits and system. IEEE 2005 Tampere, Finland. DOI:10.1109/ISCAS.2005.1465346 0-7803-8834-8/05/$20.00 ©2005 IEEE

[21] Carara, E.; et al. "HeMPS - a Framework for NoC-based MPSoC Generation". In: PUCRS – FACIN – Av. Ipiranga 6681 Porto Alegre – 90619-900 – Brazil, ISCAS'09, 978-1-4244-3828-0/09/$25.00 ©2009 IEEE pp. 1345 – 1348, 2009.

[22] Stefan Aust, Harald Richter, "Energy aware MPSoC with Space-sharing for real time apllication" in The Fifth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2011), Lisbon, Portugal, www.esearchgate.net/publication/216680154, November 2011.

[23] Siegfried Brandstätter And Mario Huemer, (Senior Member, IEEE) "A Novel MPSoC Interface and Control Architecture for Multistandard RF Transceivers", IEEE Access Journal of Digital Object Identifier 10.1109/ACCESS.2014.2345194, Vol 2, pp no 771 to 787.

[24] G, S. Almasi and A. Gottlieb, *Highly Parallel Computing*, 2nd edition, Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA, pp 670-678, ISBN: 0-8053-0443-6, ©1994.

[25] Jason G. Tong, and Ian D.L. Anderson "Soft core processors for embedded systems". Research centre for integrated micro-systems, University of Windsor.18th international conference on micro-electronics-2006, ISSN : 1-4244-0765-6/06/$20.00 @2006 IEEE

[26] Cheng-Min Lien, Ya-Shu Chen, Chi-Sheng Shih, "On-Chip Bus Architecture Optimization for Multi-core SoC Systems", Published in Software Technologies for Embedded and Ubiquitous Systems. SEUS 2007. Lecture Notes in Computer Science, vol 4761. Springer, Berlin, Heidelberg, ISBN 978-3-540-75663-7, pp 301-310, '2007.

[27] Kyeong Keol Ryu, Eung S. Shin, Vincent John Mooney, " A comparison of five different multiprocessor SoC bus architectures", Published in IEEE Proceedings Euromicro Symposium on Digital Systems Design, ISBN: 0-7695-1239-9, August 2002, DOI:10.1109/DSD .2001.952283.

[28] Mohamed M. Sabry ; David Atienza "Temperature-Aware Design and Management for 3D Multi-Core Architectures", Publisher: IEEE conference on Now Foundations and Trends, Print ISBN: 9781601987747, DOI: 10.1561/1000000032, pp 96, Mrach 2014.

[29] Jacob Murray Paul Wettin Partha Pande Behrooz Shirazi, "Sustainable Wireless Network-on-Chip Architectures" 1st Edition, Published in Elsevier, eBook ISBN: 97801 280 36518, pg count 162, March 2016

[30] T. Selvameena ; R. Arun Prasath "Out-of-order execution on reconfigurable heterogeneous MPSOC using particle swarm optimization", 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), DOI 978-1-5090-3294-5/17@2017 IEEE Pages: 1 - 6, Year: 2017.

[31] Ali Hurson Hamid Sarbazi-Azad, "Dark Silicon and Future On-chip Systems", Published in Elsevier & Science Direct as Dark Silicon and Future On-chip Systems, Volume 110, 1st Edition, eBook ISBN: 9780128153598, pg count 304, July 2018.

[32] A. P. Johnson, R. S. Chakraborty, D. Mukhopadyay, "An improved DCM based Tunable Random Generator for Xilinx FPGA", *IEEE Trans. on Circuit & Systems, vol. 64, Issue 4, ISSN:1549-7747,pg no 452 - 456, April 2017.*

[33] Logue J. D, Percey A K, Goetting F. Erich "Sychronized Multi-output Digital Clock Manager", Europeon Patent International, Publication no WO 2002/029974, Appl no., PCT/US2001/031251, ISSN : 0916-8524, vol E81-C, no.2, pgs 277-283, Feb 2013
[34] X. Iturbe, K. Benkrid, C. Hong, A. Ebrahim, R. Torrego, I. Martinez, T. Arslan, J. Perez, "R3TOS : A novel Reliable Reconfigurable Real Time Operating system for Highly Adaptive, Efficient, & Depandable computing on FPGAs", *IEEE* Trans on Computers, Vol 62, no. 8, pgs 1542 – 1556, August 2013.
[35] Chao Wang, Xi li, J. Zhang, P. Chen, Yunji Chen, X. Zhou, Ray C, C. Cheung, "Architecture support for task out of order Execution in MPSoCs", *IEEE* Trans on Computers, DOI.10.1109/TC.2014.2315628, pgs 1296-1310, August 2013.
[36] Slim Ben Othman, Ahmed Karim Ben Salem, Hedi Abdelkrim, Slim Ben Saouo. "MPSoC Design Approach of FPGA-based Controller for Induction Motor Drive" *L.E.C.A.P.*-E.P.T./I.N.S.A.T.B.P. 676,1080 Tunis Cedex, Tunisia. pp. 134-139 2012 IEEE
[37] Roberta Piscitelli and Andy D. : "A High-Level Power Model for MPSoC on FPGA" Pimentel Computer Systems Architecture group Informatics Institute, University of Amsterdam, The Netherlands. International Parallel & Distributed Processing Symposium.pg 259-272, 2011 IEEE.
[38] G. Almeida, Everton Carara, Rémi Busseuil, Nicolas Hebert, "Predictive Dynamic Frequency Scaling for Multi Processor System on Chip", IEEE Conference on Programmable Logic, id no. 978-1-4244-9472-9,pgs 1500-1503, year : 2011
[39] He Chen, Liang Yin, G. Peng, "Implementation of Multi-core Embedded System on Compound Guiding System", Beijing China, id no.978-1-4577-0321-8/11, pg4348-4352, IEEE 2011
[40] F. Anjam, S. Wong, F. Nadeem, "A Shared Reconfigurable VLIW MultiProcessor System", Delft, Natherlands, id no. 978-1-4244-6534-7/10, ISBN: 978-1-4244-6533-0 pp: 1-8 IEEE 2010
[41] Hristo Nikolov, Todor Stefanov, Ed Deprettere. "Efficient external memory interface for Multi-processor platforms realized on FPGA chips" LIACS, Leiden University, The Netherlands. ISSN: 1946-147, elec: 1946-1488, IEEE 2007.
[42] V B Chandra, V Sharma, Dr. M. Chaudhari, "Issues with designing a dual core processor with a shared L2 Cache on a xilinx FPGA board", Project report of researcher at Indian Institute of Technology, Kanpur. (IIT Kanpur), Y3383,Y3393, May 2007
[43] G. Beltrame, L. Fossati, D. Sciuto, "High level Modeling & Exploration of Reconfigurable MPSoCs", NASA Conference on Adaptive Hardware Systems, 978-0-7695-3166-3/08, pgs 324-337, IEEE Conference, October 2008
[44] D. Gohringer, M. Hubner, E. N. Zeutebouo, J. Becker, "Operating systems for runtime reconfigurable MPSoCs", Research article at International Journal of Reconfigurable Computing, Hindawi publication, vol. 2011, id. 121353, pgs : 16, KIT, Germany, Feb 2011
[45] T. Kangas, P. Kukkala, H. Orsila, E. Salminen, "UML based Multi-Processor SoC Design Framework", ACM transactions on Embedded Computing Systems, (Nokia Research Centre) Vol. 5, no. 2, pgs 281-320, May 2006
[46] S. pawar, J. Zalke, "Efficient Implementation of Ogg Vorbis Decoder using Soft core Processor", IOSR Journal of Engineering, Vol. 2,(4), pg: 928-931, April 2012
[47] Gaughan W., Embry-Riddle Aeronautics university, USA, "Using an FPGA Digital Clock Manager to generate sub nanosecond phase shifts for LIDAR (Light detection & ranging) applications", Conference on Programmable Logic, ISBN : 978-1-4244-6309-1, pgs:163-166, March 2010.
[48] Wikipedia, (www. Wikipedia.org) Free Encyclopedia for DCM, Peloton & MPSoC
[49] Xilinx White Paper on "Designing multi processor systems in Xilinx Platform Studio", WP262 (v2.0), Nov 2007
[50] Xilinx White Paper on "Digital Clock Manager (DCM) Module", DS485, Apr 2009.
[51] OpenRisc 1000, www.opencores.org, 2005
[52] Gaissler, J: The LEON processor. www.gaissler.com, 2005
[53] Xilinx: MicroBlaze Processor Reference Guide. (v 4.0). 2004
[54] Altera: Nios 3.0 CPU Data Sheet, 2004
[55] Altera: Nios II Processor Referente Handbook, 2005
[56] Xilinx: XAPP529: Connecting Customized IP to the MicroBlaze Soft Processor core using the Fast Simplex Link (FSL) Channel. (v 1.3). 2004
[57] FIPS, "Advanced Encryption Standard", Nov, 2001
[58] Xilinx White Paper on "Embedded Design with The Xilinx Embedded Developer Kit", Apr 2015.
[59] Xilinx White Paper on "Getting started with the Microblaze Development kit – Spartan 3E 1600E", Nov 2007.
[60] Kyungho Ryu, Jiwan Jung, Dong-Hoon Jung, Jin Hyuk Kim, and Seong-Ook Jung, Senior Member, IEEE, "High Speed, Low power and Highly reliable Frequency multiplier for DLL based Clock Generator" IEEE Transactions on very large scale integration (vlsi) systems, ISSN no. 1063-8210, pgs 1484-1492, DOI: 10.1109/TVLSI.2015.2453366 © 2015 IEEE.
[61] ML505/6/7 Virtex-5 Evaluation Platform, ML505 Schematic, Xilinx 1280415.
[62] Tarek Darwish, Sany Kabbani, Acile Sleiman "Multi-Processor System Design on FPGA" Final Year Project Report at The American University of Beirut, Faculty of Engineering & Architecture, Spring 2005-2006.
[63] Siegfried Brandstätter1 And Mario Huemer2, (Senior Member, IEEE) "A Novel MPSoC Interface and Control Architecture for Multi-standard RF Transceivers", IEEE. Translations, Volume 2, ISSN 2169-3536, pp: 771-787, 2014.

