ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 14, April 2018

PROCESSORS ALLOCATION FOR MPSoCs WITH SINGLE ISA HETEROGENEOUS MULTI CORE ARCHITECTURE

Sundararajan.R

Department of Electronics and Communication Engineering

Anna University Regional Campus, Madurai

Madurai, India

rpssundara@gmail.com

Arun Prasath.R

Department of Electronics and Communication Engineering

Anna University Regional Campus, Madurai

Madurai, India

prasath2k6@gmail.com

*Abstract*__ **Single instruction set architecture (ISA) heterogeneous multi-processor architecture is promising for developing multi-processor system-on-chips (MPSoCs). In this architecture, all processors execute the same instruction set, yet with various performance and power behavior, since processors may have various micro-architectures. Therefore, systems with this architecture have the advantages of easy to develop new functions as the homogeneous architecture, and easy to customize the resource allocation to achieve high energy efficiency as the heterogeneous architecture. However, for an MPSoC utilizing the target architecture, a key design issue is how to select the set of processors so that the target system can achieve good performance while the cost of the chip is constrained to the expected value. To solve this, in this paper, we propose a processor allocation method for MPSoCs with single ISA heterogeneous multi core architecture. The goal of the proposed method is to automatically synthesize the allocation of cores for the given workload so that the performance is optimized while the resource constraint is met. To the best of our knowledge, this is the first work that tackles the processor allocation problem for MPSoCs with the target architecture. To bring out the best performance of a hardware configuration, the proposed algorithm also synthesizes the software design of task mapping for a selected hardware configuration. The experimental results show that compared with the homogeneous architecture with the least cost and lowest performance cores only, even if the number of core is set to the maximum parallelism degree of the target workload, the proposed method achieve up to 8.25% of performance improvement among all the cases we evaluated while the area constraint is met. Compared with the architecture with all high performance but large cores, when the number of cores is also set to the same as the maximum parallelism degree of the target workload, the proposed method has at most 11.5% of performance degradation, while the area cost is reduced by 60.7%.**

I.INTRODUCTION

Multiprocessor Systems-On-Chips (MPSOCs) are the latest incarnation of Very Large Scale Integration (VLSI) technology. A single integrated circuit can contain over 100 million transistors, and the International Technology Roadmap (ITR) for semiconductors predicts that chips with a billion transistors are within reach. Harnessing all this raw computing power requires designers to move beyond logic design into computer architecture. The demands placed on these chips by applications require designers to face problems not confronted by traditional computer architecture: real-time deadlines, very low-power operation, and so on. These opportunities and challenges make MPSOC design an important field of research.

The MPSOC platforms that provide high-level of adaptability, performance, reliability and energy efficiency. The MPSOC is a SOC which uses multiple processors, usually targeted for embedded applications. It is used by platforms that contain multiple,

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

usually heterogeneous, processing elements with specific functionalities reflecting the need of the expected application domain, a memory hierarchy and I/O components. All these components are linked to each other by an on-chip interconnect. Its main challenge is to combining energy efficiency and its performance. The main components of the System-on-Chip are: the processing elements responsible for executing the applications, the on-chip memories responsible for storing the application data and instructions, I/O components responsible for communicating with the outside world and, generally, the on-chip interconnect structure responsible for linking the processing elements with the memories and the I/O components.

The tasks are executed on Processing Elements (PEs). Depending on the type of PEs, two different categories of MPSOCs are distinguished:

- Heterogeneous MPSOCs which are composed of different types of PEs designed to perform their specific tasks (dedicated accelerators).
- Homogeneous MPSOCs which are composed of one type of PE instantiated multiple times where all the elements necessary for the targeted application are embedded inside the PE.

Comparing both MPSOC heterogeneous MPSOCs; provide good energy efficiency and performance. Heterogeneous MPSOCs target high-performance applications with low-power requirement.

II. EXISTING SYSTEM

Using Instruction level score boarding algorithm both software based static and dynamic implementations on top of a heterogeneous MPSOC prototyped on a field-programmable gate array fabric was introduced**.** The prototyped approaches on an FPGA platform with two scenarios. The first scenario entails four manually composed test cases with different dependence degrees, and the second scenario entails the JPEG encoding as a real-life application.

The execution of multiple OoO tasks poses significant challenges to schedule them and map them on the processing elements. At this level, the major drawback of current programming models is that programmers need to handle the task assignments manually, which would seriously increase the burden of programmers. The OoO execution of instructions is dealt with by score boarding and the Tomasulo algorithm at the instruction level. Both approaches provide techniques for OoO instruction execution when there are sufficient computational resources and no data hazards among instructions. The Tomasulo algorithm is more complex as it can also detect Write-After-Write (WAW) and Write-After-Read (WAR) hazards through register renaming. Consequently, as programming models require more experience from the programmers, the extension from the instruction-level scheduling algorithm to task level provides an alternate methodology to utilize MPSOC platform effectively.

The reasons why we choose score boarding instead of the Tomasulo algorithm are as follows.

- Score boarding provides a light-weight task hazards engine for OoO execution. The architecture is simpler, which brings smaller scheduling overheads.
- For task-level parallelization, WAW and WAR hazards do not happen as much as at the instruction level. Most programmers intend to use different parameters in the case of WAR and WAW hazards. Therefore, introducing a mechanism as complex as the Tomasulo algorithm is not necessary.

III. PROPOSED SYSTEM

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

Reconfigurable computing has made major inroads with the advent of large-scale Field-Programmable Gate Arrays (FPGAs), which are capable of incorporating (multiple) complex application-specific computing elements on a single chip. It is predicted that MPSOCs will greatly improve computational capabilities of heterogeneous platforms in the near future therefore, we expect great promise by combining heterogeneous MPSOCs with Reconfigurable computing. In this paper, we study more hardware extensions, investigate how the score boarding approach can be applied to high-performance DSP or GPU cores where multiple functional units or vectors may execute in massive parallel, and investigate how the score boarding scheme works under Reconfigurable fabric conditions to further improve the performance. To improve the performance power swarm optimization technique is combined with the instruction level score boarding algorithm.

A. *Block Diagram*

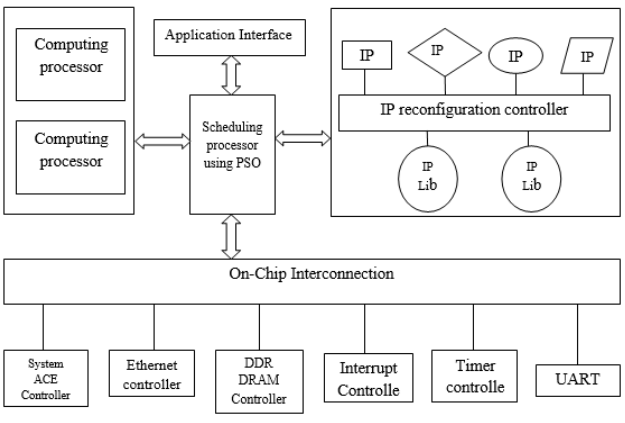

Figure 1 Block Diagram of PSO Algorithm

PSO is used to improve the performance of the task level circuit. The usual aim of the PSO algorithm is to solve an unconstrained continuous minimization problem. Similarly to GA, it is a population-based method, that is, it represents the state of the algorithm by a population, which is iteratively modified until a termination criterion is satisfied. The Particle swarm optimization maintains multiple potential solutions at one time. All the particles have a fitness value. The fitness value can be calculated using an objective function. All the particles preserve their individual performance. They also have known as the best performance of their group.

B. *Design Specification*

Peripheral parts are connected through the bus-based interconnect, such as universal asynchronous receiver/transmitter, double data rate dynamic RAM (DDR DRAM), System ACE controller, Ethernet controller, and time and interrupt controller.

Ethernet

A chip is commonly found on Ethernet devices. Its purpose is to provide analog signal physical access to the link. It is usually used in conjunction with a Media Independent Interface (MII) chip or interfaced to a microcontroller that takes care of the higher layer controller.

Ethernet Physical Transceiver

The Ethernet PHY is a component that operates at the physical layer of the OSI network model. An Ethernet physical transceiver can also be referred to as a physical layer transmitter and/or receiver, a physical layer transceiver, a PHY transceiver, a PHY receiver, or simply a PHY.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

DDR Control

Dynamic random-access memory (DRAM) is a type of random-access memory that stores each bit of data in a separate capacitor within an integrated circuit.

DRAM is widely used in digital electronics where low-cost and high-capacity memory is required.

An asynchronous DRAM chip has power connections, some number of address inputs (typically 12), and a few (typically one or four) bidirectional data lines. There are four active-low control signals:

- *RAS,* the Row Address Strobe. The address inputs are captured on the falling edge of RAS, and select a row to open. The row is held open as long as RAS is low.
- *CAS,* the Column Address Strobe. The address inputs are captured on the falling edge of CAS, and select a column from the currently open row to read or write.
- *WE*, Write Enable. This signal determines whether a given falling edge of CAS is a read (if high) or writes (if low). If low, the data inputs are also captured on the falling edge of CAS.

OE, Output Enable. This is an additional signal that controls output to the data I/O pins. The data pins are driven by the DRAM chip if RAS and CAS are low, WE is high, and OE is low.

UART

UART is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are conFigureurable. The electric signaling levels and methods are handled by a device circuit external to the UART. The UART takes bytes of data and transmits the individual bits in a sequential fashion. Serial transmission of bits is less costly than parallel transmission.

ACE Controller

Advanced ConFigureuration Environment (System ACE) to address the need for a space-efficient, pre-engineered, high-density conFigureuration solution for systems with multiple FPGAs. System ACE technology is a ground-breaking in-system programmable conFigureuration solution that provides substantial savings in development effort and cost per bit over traditional PROM and embedded solutions for high-capacity FPGA systems. The System ACE CF solution combines Xilinx expertise in conFigureuration control with industry expertise in commodity memories.

System ACE Functions and Requirements

System ACE CF is a two-chip conFigureuration solution. It requires the System ACE CF Controller, and either a Compact Flash card or one-inch Micro drive disk drive technology as the storage medium. System ACE CF uses Boundary-Scan protocol to conFigure devices connected to the System ACE CF controller.

XPS SYSACE Interface Controller

The XPS System ACE Interface Controller (or, interchangeably, the XPS SYSACE) is the interface between the Processor Local Bus (PLB) and the Microprocessor Interface (MPI) of the System ACE.

Computing Processor

There are two kinds of source files, first is Main.C, which is the main program of the application, and the others are Apps.C, which are the application library on each computing processor.

The regular source codes shall be further processed by the Xilinx software tool chain to generate the executable files. The executable file of main program will be executed on scheduling processor and the executable files of application libraries will be loaded into each computing processor. A certain

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

function will be executed when requested by the main program.

One general-purpose processor serves as a scheduling processor, which is employed to perform task scheduling in dynamic scheme or to do preprocessing in a static way.

Application Interface

The original task sequence will be sent to the scheduling processor through application interfaces.

Scheduling Processor

Several general-purpose processors connected with the scheduling processor provide a run-time environment for software tasks. They can execute different types of software tasks.

The on-chip interconnect is used for data transmission between scheduler and other hardware blocks (computing processors, IP cores). Considering the frequent data exchanges between the scheduling processor and other computing elements, we select star topology with the scheduler-centric interconnect structure

IP ReconFigureuration

At the beginning of the program, each functional unit is initialized by uploading the application binary into its private memory. Meanwhile, each IP core is initialized with the downloaded hardware bit streams. Several heterogeneous IP cores serve as functional units (indicated with different shapes) are responsible for specific tasks to achieve acceleration. In particular, the IP cores can be reconFigured from IP library to fit indifferent applications.

C. Particle Swarm Optimization

PSO algorithm for optimizing the mapping and scheduling problem on the heterogeneous multiprocessor architectures. In this paper, it present an algorithm based on the meta-heuristic optimization technique, Set-Based discrete Particle Swarm Optimization (S-PSO), which efficiently solves scheduling and mapping problems on the target platform. This algorithm can simultaneously addressed the mapping and scheduling problems on a complex and heterogeneous MPSOC and it has better performance than other algorithms in dealing with large scale problems. This algorithm also reduces the execution time. This approach was able to reach the optimal solutions much faster than OoO.

IV. RESULTS AND DISCUSSION

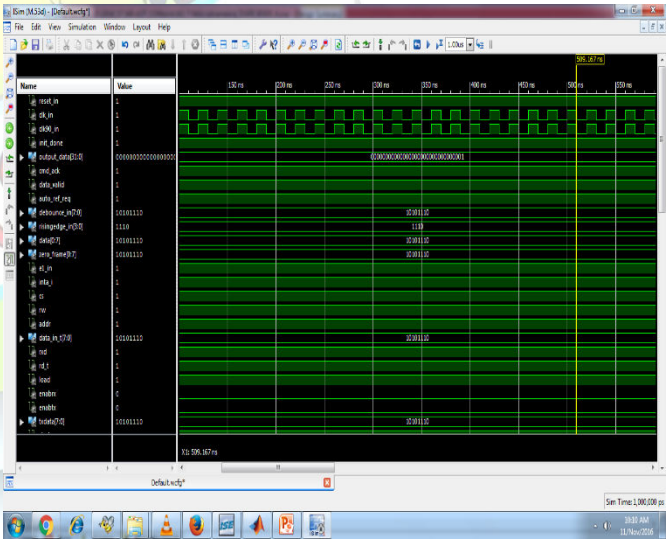

Figure 2 Simulated Result of Reconfigurable Heterogeneous MPSOC System

SYNTHESIS RESULT OF RECONFIGURABLE HETEROGENEOUS MPSOC SYSTEM

ISSN 2394-3777 (Print)

ISSN
2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

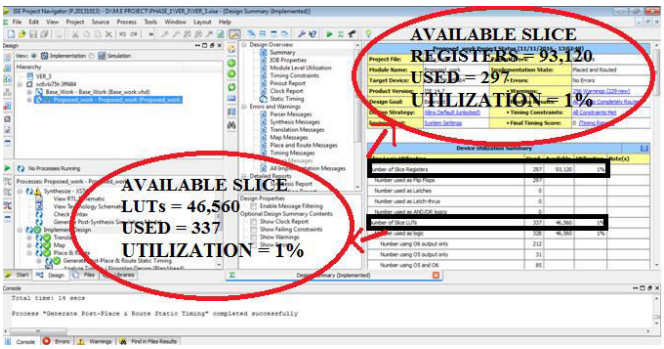

Figure 3 Area Analysis of Reconfigurable Heterogeneous

MPSOC System

The Area analysis of Reconfigurable Heterogeneous MPSOC system is shown in above Figure 3. This analysis mainly concentrated on the total number of slice registers and the slice LUTs in the entire system and the number of slice registers and the slice LUTs used to obtain the output. Hence, the available numbers of slice registers are 93,120 and the used numbers of slice registers are 297. The available slice LUTs are 46,560 and the used number of slice LUTs are 337. The utilization of slice register and the slice LUT is 1%.

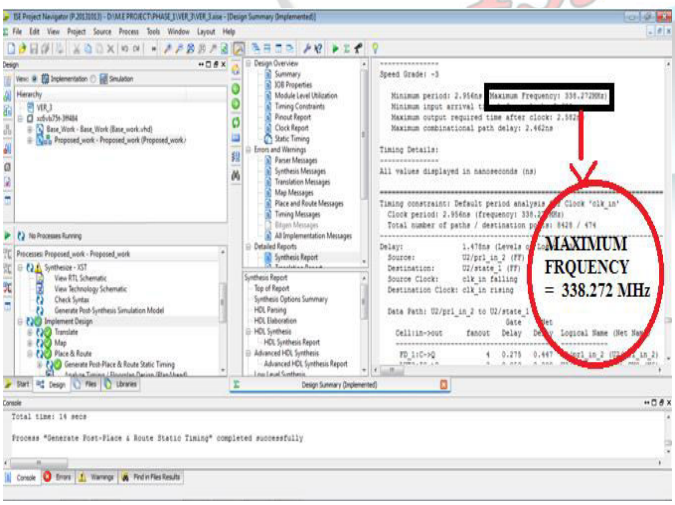

Figure 4 Speed Analysis of Reconfigurable Heterogeneous MPSOC System

The Speed analysis of Recongigurable Heterogeneous MPSOC is shown in the above Figure 4 This analysis mainly depends on the maximum range of frequency. Hence, the maximum range of frequency used to generate the output is 338.272 MHz.

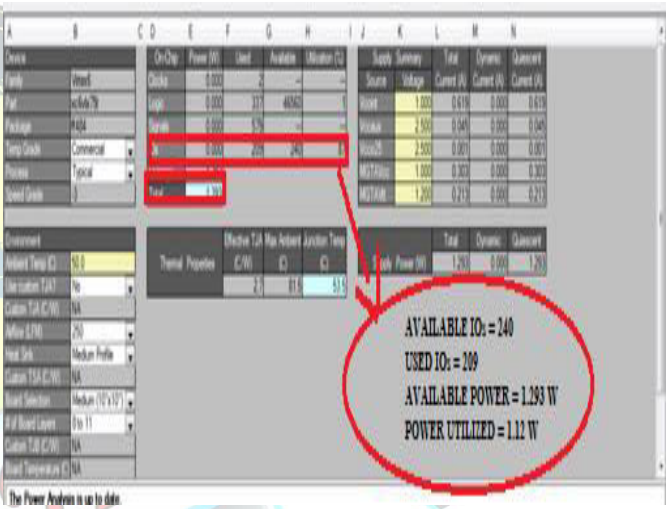

Figure 5 Power Analysis of Reconfigurable Heterogeneous MPSOC System

The Power analysis of Reconfigurable Heterogeneous MPSOC is shown in above Figure 5. This analysis mainly concentrated on the number IOs in the entire system to obtain the output. In the entire system, the available numbers of IOs are 240 and the IOs used to obtain the output are 209. The total available power in the system is 1.23 W and hence the utilization of power is 1.12 W.

ISSN 2394-3777 (Print)

ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

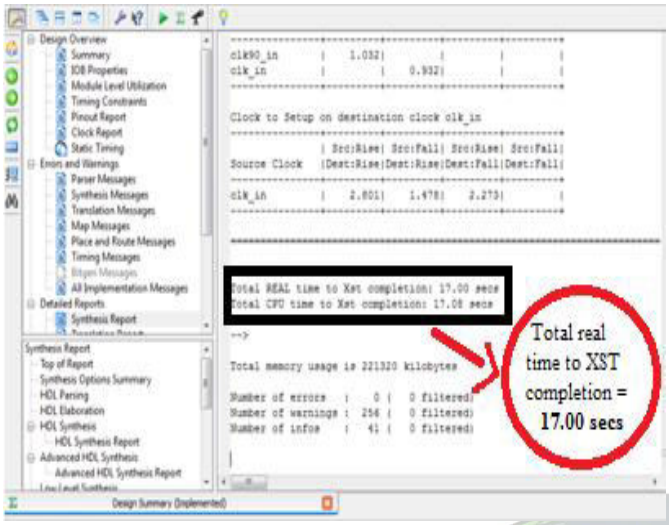

Figure 6 Delay Analysis of Reconfigurable Heterogeneous MPSOC System

The Delay analysis of Reconfigurable Heterogeneous MPSOC system is shown in above Figure 6. The Main concentration of delay analysis is based on the time taken to obtain the output from the time of applying input. Hence, the total time required to XST the completion is 17secs.

Comparison Results about Heterogeneous MPSoC and Reconfigurable Heterogeneous MPSoC Systems.

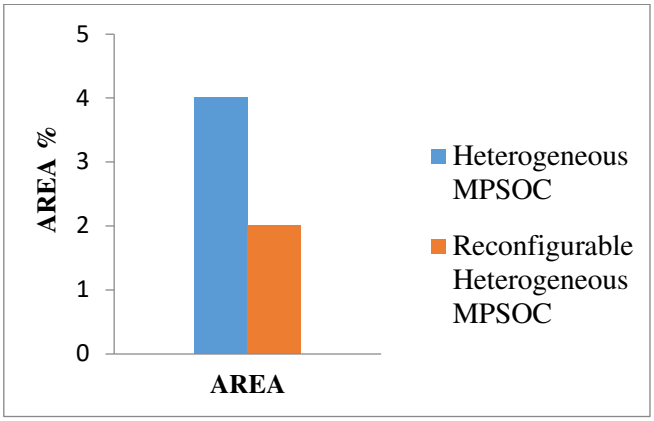

Figure 7 Area Analysis for Heterogeneous MPSOC

And Reconfigurable Heterogeneous MPSOC

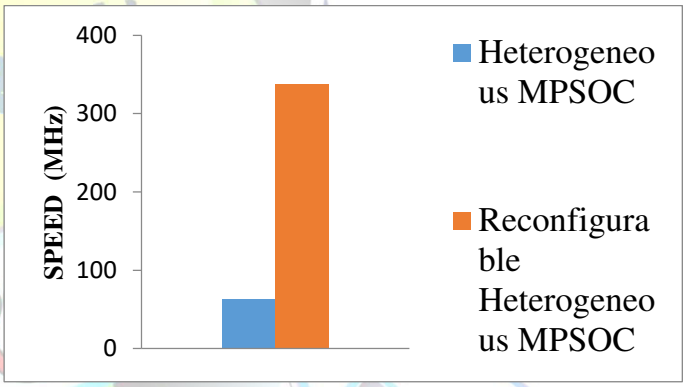

Figure 8 Speed Analysis for Heterogeneous MPSOC

And Reconfigurable Heterogeneous MPSOC

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET) Vol. 5, Special Issue 14, April 2018

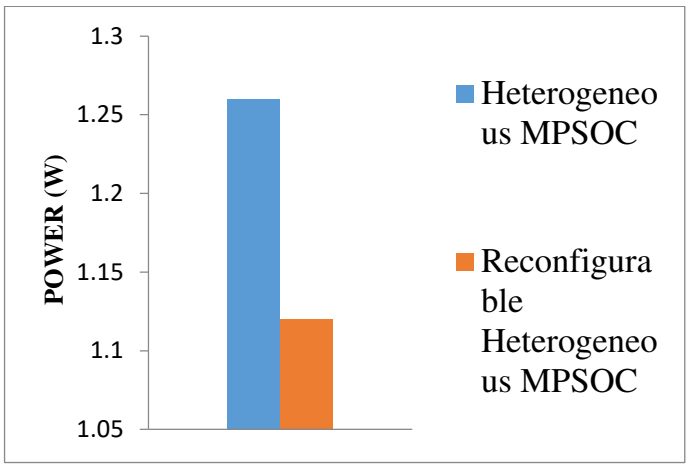

Figure 9 Power Analysis for heterogeneous MPSOC And Reconfigurable Heterogeneous MPSOC

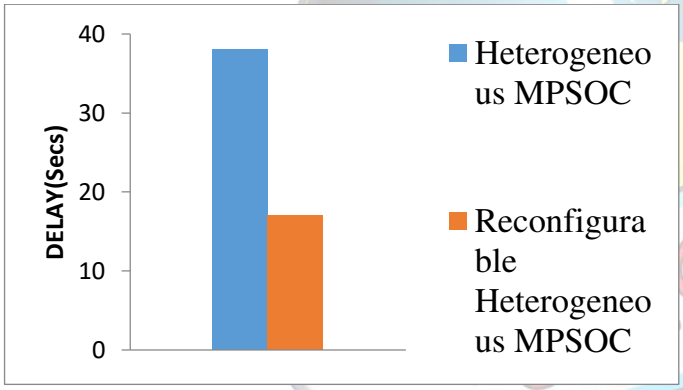

Figure 10 Delay Analysis for Heterogeneous MPSOC and Reconfigurable Heterogeneous MPSOC

Table 1 Performance Analysis Comparison Table

	AREA (%)	SPEED (MHz)	POWER CONSUMPTION (W)	DELAY (secs)
HETEROGENEOUS MPSOC	4	63.485	52.2	38
RECONFIGURABLE HETEROGENEOUS MPSOC	1	338.272	1.12	17

V.CONCLUSION

The RST processor allocation method for MPSoCs with single-ISA heterogeneous multi-core architecture, which is considered to be a promising platform for developing MPSoCs. The goal of the proposed method is to end a proper processor allocation and task mapping configuration such that the target workload's execution time is optimized while the given resource/area constrain is met. Since the solution space of the target synthesis problem grows exponentially with the number of tasks in the target workload, and the number of cores, we proposed a heuristic-based method that RST decides the groups of tasks that should be mapped to the same processor, and then perform processor allocation and task mapping. The experimental results show that, the proposed method effectively reduced the solution space, and synthesized a good quality configuration in a reasonable time. Under the same area constraint, the results synthesized by our method achieved up to

8.25% of performance improvement over the performance of the system with all simple cores, which provide the maximum execution parallelism in the system. The results also showed that, the proposed method synthesized a configuration for a workload with up to 36 tasks and 53 edges within one second.

REFERENCES

[1] Alb, M., Alotto, P., Magele, C., Renhart, W., Preis, K., & Trapp, B. (2016). Firefly algorithm for finding optimal shapes of electromagnetic devices. *IEEE Transactions on Magnetics*, Vol.*52, No.3*, PP.1-4.

[2] Alrashidi, M. R., & El-Hawary, M. E. (2009). A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, Vol.13*, No.4*, PP.913-918.

[3] Campos, M., Krohling, R. A., & Enriquez, I. (2014). Bare bones particle swarm optimization with scale matrix adaptation. *IEEE transactions on cybernetics*, Vol.44*, No.9*, PP.1567-1578.

[4] Chauhan, N. C., Kartikeyan, M. V., & Mittal, A. (2009). CAD of RF windows using multiobjective particle swarm optimization. *IEEE Transactions on Plasma Science*, Vol.*37, No.6*, PP.1104-1109.

[5] Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures*, Vol.*89, No.23*, PP.2325-2336.

[6] Gohringer, D., Hubner, M., Schatz, V., & Becker, J. (2008, April). Runtime adaptive multi-processor system-on-chip: RAMPSoC. *IEEE Transactions on* Parallel *and Distributed Processing, 2008. IPDPS 2008. on* (pp. 1-7). IEEE.

[7] Guo, Q., Li, X., Wang, C., & Zhou, X. (2016). Evaluation and Tradeoffs for Out-of-Order Execution on Reconfigurable Heterogeneous MPSoC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.*24, No.1*, PP.79-91.

[8] Huang, S. J., Liu, X. Z., Su, W. F., & Yang, S. H. (2013). Application of hybrid firefly algorithm for sheath loss reduction of underground transmission systems. *IEEE transactions on power delivery*, Vol.*28, No.4*, PP.2085-2092.

[9] Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014). A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. *IEEE transactions on evolutionary computation*, Vol.18*, No.2*, PP.301-305.

[10] Pal, S. K., Rai, C. S., & Singh, A. P. (2012). Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems. *International Journal of Intelligent Systems and Applications*, Vol.*4, No.10*, PP.50.

[11] Silva, L. I., Belati, E. A., & Junior, I. C. S. (2016). Heuristic Algorithm for Electrical Distribution Systems Reconfiguration Based on Firefly Movement Equation. *IEEE Latin American Transactions*, Vol.*14, No.2*, PP.752-758.

[12] Sundareswaran, K., Peddapati, S., & Palani, S. (2014). MPPT of PV systems under partial shaded conditions through a colony of flashing fireflies. *IEEE transactions on energy conversion*, Vol.*29, No.2*, PP.463-472.

[13] Yang, X. S. (2010). Firefly algorithm, stochastic test functions and design optimization. *IEEE Transactions on Bio-Inspired Computation*, Vol*2, No.*2, PP.78-84.

[14] Yang, X. S., Hosseini, S. S. S., & Gandomi, A. H. (2012). Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. *IEEE Transactions on Applied soft computing*, Vol.*12, No.*3, PP.1180-1186.

[15] Yang, X. S. (2013). Multiobjective firefly algorithm for continuous optimization. *IEEE Transactions on* Engineering *with Computers*, Vol.*29, No.*2, PP.175-184.