A NOVEL MRPSoC PROCESSOR FOR DISPATCH TIME CURTAILMENT

Parvathy Asokan[1], Kavitha.V[2], K.V. Ramakrishnan[3]

[1]*Mtech, CMRIT, Bangalore, India*
[2]*Ph.D. Scholar, Jain University, India*
[3]*Professor*

Abstract

This paper describes a platform which consists of Multi Reconfigurable Instruction Set Processor System on Chip (MRPSOC). Reconfigurable Instruction set processor (RISP) consists of a microprocessor core that can be extended with reconfigurable logic. RISP and MPSOC are the two methods to improve the performance. By combining both, better results can be achieved. MRPSOC can run applications in parallel and accelerate the performance due to its reconfigurable functional unit (RFU) and at the same time it retains programmability. In this paper, Dynamic Critical Path algorithm is used to extract the custom instructions. In this platform, the critical portions of the code can be executed on RFU. For verifying the efficiency of MRPSoC, a set of instructions are executed in both MRPSoC and MPSoC. Finally, from the experimental results, it is concluded that the dispatch time of executing a set of instructions are curtailed as compared to MPSOC.

Keywords- *Reconfigurable Instruction set Processors (RISP), Multi-processor System on Chip (MPSoC), Reconfigurable Processing Unit (RPU), and Reconfigurable Functional Unit (RFU)*

---***---

1. INTRODUCTION

General Purpose Processors (GPP) are designed for general purpose computers such as PCs, workstations etc. The most important concern of GPP is the computation speed. The cost of the GPP is much higher than DSPs and microcontrollers. All techniques which increases the CPU speed can be applied to GPPs. For example, GPPs usually include on-chip cache and on-chip DMAs. GPPs provide hardware circuits for commonly used math and logic operations. GPPs have a balanced instruction set. They are not designed for fast real-time applications. The secondary concern of GPP is the cost. Most of the GPPs can execute Digital Signal Processing (DSP) algorithms. But they are not used for portable devices such as mobile phones. In such cases, specialized digital signal processors are used. They have almost the same integration level and clock frequencies as GPPs. But DSPs have better performance and lower latency. So DSPs are lower-cost alternative to GPPs.

An Application Specific Integrated Circuit *(*ASIC) is an integrated circuit customized for a particular application. For example, a chip designed to run in a digital voice recorder. As the feature size decreases, the maximum complexity possible in an ASIC has grown from 5000 gates to over 1 million. Modern ASICs include entire microprocessors, memory blocks including ROM, RAM, EEPROM etc and other large building blocks. Such an ASIC is often named as SoC (system-on-chip). Hardware description languages (HDLs) can be used to describe the functionality of ASIC. The non-recurring engineering (NRE) cost of an ASIC is very high.

An ASIP is a programmable architecture that is designed in a specific way to perform certain tasks more efficiently. ASIPs are having less production costs, simplified manufacturing process and less power consumption. An ASIP instruction is usually different than a normal instruction. It doesn't have to be composed of a mnemonic and register/memory operands. The application of ASIP determines the instruction-set format. An ASIP uses either general purpose registers or configuration registers. Instructions set in ASIPs can be divided into two parts- Static and Configurable. Static logic defines a minimum ISA and configurable logic can be used to design new instructions. In Application Specific Instruction Set Processors (ASIP), the critical portions of an application can be executed using Custom Function Unit (CFU). They provide high programmability and high flexibility. The major drawbacks of ASIP are increasing algorithm complexity and high NRE cost. As algorithm complexity increases, the completion time increases. So it is required to intend a design which reduces the dispatch time without compromising the performance.

Reconfigurable Instruction Set Processors (RISP) and Multi Processors System on Chip (MPSoC) can be used to increase the performance. So by amalgamating both in a single platform, better results can be achieved. Such a platform is entitled as Multi Reconfigurable Instruction Set Processor System on Chip (MRPSoC). In this paper, an MRPSoC with two RISP processors has been designed.

2. RELATED WORK

The researchers "F. Barat, R. Lauwereins and G. Deconinck" developed Dynamically Reconfigurable RISP which can be reconfigured during runtime [3]. The authors Giovanni Ansaloni, Paolo Bonzini, Laura Pozzi, described an Expression Grained Reconfigurable array (EGRA) in which the processor is stalled during reconfiguration [4]. EGRA is inspired by the Configurable Computation Acceleration (CCA) structure proposed by Clark [5]. The CCA is used as standalone acceleration. The authors A. Lodi, M.Toma, F.Campi, A.Capelli, R. Canegallo and R.Guerrieri described a VLIW processor with reconfigurable instruction set for embedded applications [6].

The author G.Martin described a Multi-Processor SoC – Based Design Methodologies using Configurable and Extensible Processors [7]. Several algorithms can be used to generate the custom instructions. One such algorithm is the synthesis algorithm [8]. It performs scheduling and customization procedure simultaneously. Authors in [9] demonstrate that combination of multicore processor and reconfigurable instruction set extensions creates multi- level parallelism for high performance. ReMAP (Reconfigurable Multicore Acceleration and Parallelization) [10] uses common RFU between heterogeneous cores. RFU is loosely coupled with cores with a fine granularity. The methodology in [11] customizes a MPSoC platform by a repetitive procedure. Initially it assigns the tasks to processors and then adds CIs for the tasks which are on critical path until the selected path is no longer critical.

3. MRPSoC

MRPSoC is an amalgamation of RISP and MPSoC in a single platform. In this paper, an MRPSoC with two RISP processors has been designed as shown in Figure.1.

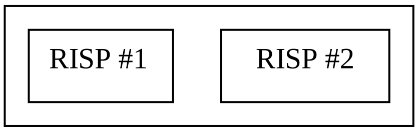

Fig-1 MRPSoC with two RISP processors

A Reconfigurable processor is a microprocessor with an obliterate hardware and can be rewired dynamically. These processors amalgamate a microprocessor with reconfigurable logic. The reconfigurable logic will furnish the hardware specialization of the application to be executed. Reconfigurable processors can be revamped after design in the same manner as programmable processors. The Reconfigurable Instruction Set Processor (RISP) is a subgroup of reconfigurable processors. RISP consists of a microprocessor and a Reconfigurable Processing Unit (RPU). The orientation of the RPU with respect to the microprocessor determines the performance of RISP. The advantage acquired from enacting a piece of code in the RPU relies upon the communication and execution costs. The time required for an operation execution in RPU is the aggregate of the time required to send the processed data and the time needed to process it. If this time is less that the time it would take with the processor alone, an up gradation is required.

The RPU can be situated in three positions with respect to the processor: Attached processor, Coprocessor, Functional unit. In attached processor, the RPU is attached on the I/O bus. An example for the attached processor is PRISM- 1. In coprocessor, the RPU is kept besides the processor. An example for coprocessor is GARP. If the RPU is placed inside the processor, then it is said to be Reconfigurable Functional Unit (RFU). An example for the RFU is One Chip98.

The coupling of RPU with respect to the microprocessor is shown in Figure.2. The attached processor and the coprocessor lashing schemes are called loosely coupled. The Reconfigurable functional unit is tightly coupled. The RFU takes less time for data transfer between the processor and the reconfigurable logic as compared to Attached processor and Coprocessor. The first reconfigurable processor used Attached processor or Coprocessor type processing unit. In now a days, as the gate capacity is growing, the processors are designed with RFU.

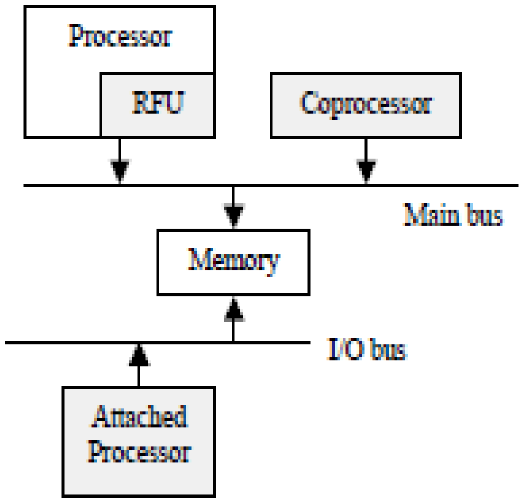

Fig-2.RPU coupling with respect to microprocessor

For designing a RISP, two main tasks are there: One is the interfacing between the microprocessor and the RFU and the second is the design of RFU. The interfacing between the microprocessor and the RFU rely on the instruction types that can be executed on the RFU. There are two types of instructions which can be executed on the RFU: Stream based or block based instruction and custom instruction. The block based instructions manage prodigious amounts of data in sequence or by blocks. Only some applications use this type of instructions. The custom instructions manage fewer amounts of data at a time and it also produces small amounts of data. As these instructions impose diminutive restrictions on the attributes of the application, they can be employed for most applications. The number of instructions and the type of instructions that will increase the performance of RISP is decided by the design of RFU. The primary facet in the RFU design is the granularity of RFU. The granularity elucidates the functionality of the basic building blocks and their interconnection paths. Depending on the granularity, RFUs can be classified into two: Fine grained RFU and Coarse grained RFU. For fine grained RFU, the building blocks are gates. So more information is required to express an instruction. The fine grained RFU is best suited for bit manipulation operation. For coarse grained RFU, the building blocks are larger such as ALU, multipliers, shifters etc. They operate on wider bus. It requires less information to express an instruction. The coarse grained RFU is best suited for bit parallel operations and computation intensive applications. As compared to fine grained RFU, coarse grained RFU provides high performance.

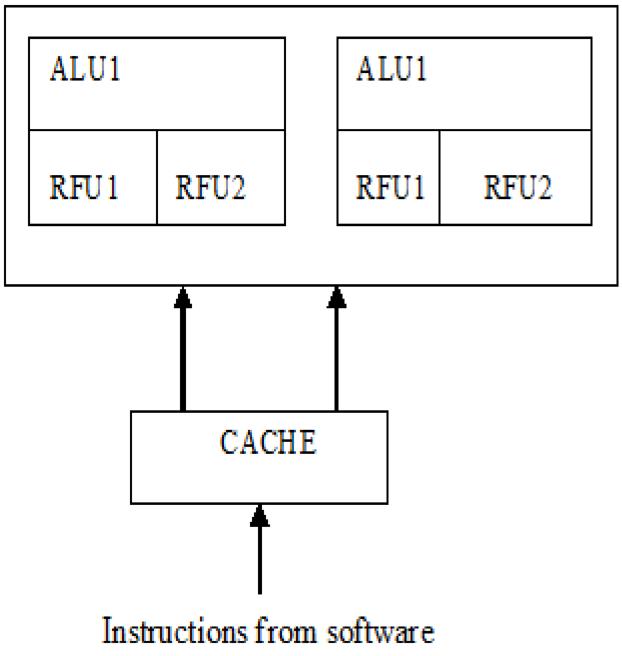

Fig-3.Implemented MRPSoC block diagram

The granularity also influences the size of the configuration stream and configuration time. As the size of the application data is compact when compared to the logic, the performance curtails because of unexploited power. If the size of the application data is prodigious when compared to the logic, the performance curtails because of reconfigurability overhead.

Multiprocessor System On chip (MPSOC) can be classified into two heterogeneous and homogeneous. Heterogeneous MPSOC is more suitable to meet the desired performance but it does not provide enough scalability and feasibility as compared to homogeneous MPSOC. Homogeneous MPSOC are flexible, fault tolerant and scalable but they do not have desired performance. So we need to design a middle solution that uses both features of homogeneous and heterogeneous MPSOC.

The MRPSoC is a combination of RISP and MPSoC in a common platform. This platform curtails the dispatch time of instruction execution by executing the complex or critical instructions on the RFU and executing the instructions concurrently on the ALU and RFU. The block diagram of the implemented MRPSoC is shown in Figure.3.

The MRPSoC curtails the dispatch time of instruction execution by executing the complex or critical instructions on the RFU and executing the instructions concurrently on the ALU and RFU. The implemented MRPSoC consists of two RFUs. One RFU consists of four input, three output and four levels with four, three, one and one function units in each level respectively. The second RFU consists of three inputs, two outputs and three levels with three, two and one function units in each level

4. MRPSoC FLOW DIAGRAM

The MRPSoC flow diagram is shown in Figure.4. Instructions from software are converted into binary by using the Perl Compiler. Then it is accumulated in the cache memory. Each instruction is analyzed to determine whether it is critical or not. This can be done by using Dynamic Critical Path (DCP) algorithm. For implementing the DCP algorithm the Absolute Earliest Start Time (AEST) and the Absolute Latest Start Time (ALST) of the instruction is calculated. If AEST and ALST are equal, that particular instruction is critical. Such instructions are entitled as Custom Instructions (CI).

The instructions are fetched in the order ALU1, RFU1 & RFU2, ALU2, and RFU3 & RFU4. If an instruction is non-critical, it is fetched by the fetch unit of ALU1. Then it is decoded and executed. If an instruction is critical, it is fetched by the fetch unit of RFU1 or RFU2 depending on the availabilities of RFUs. If both RFUs of one core are available, they both will fetch the critical instructions at the same time and then decoded and executed. So a total of three instructions can be executed concurrently. Since the resources are shared by the two cores, if one core operates the other will be in idle state.

The programming language used is System C and the operating system is CYGWIN.

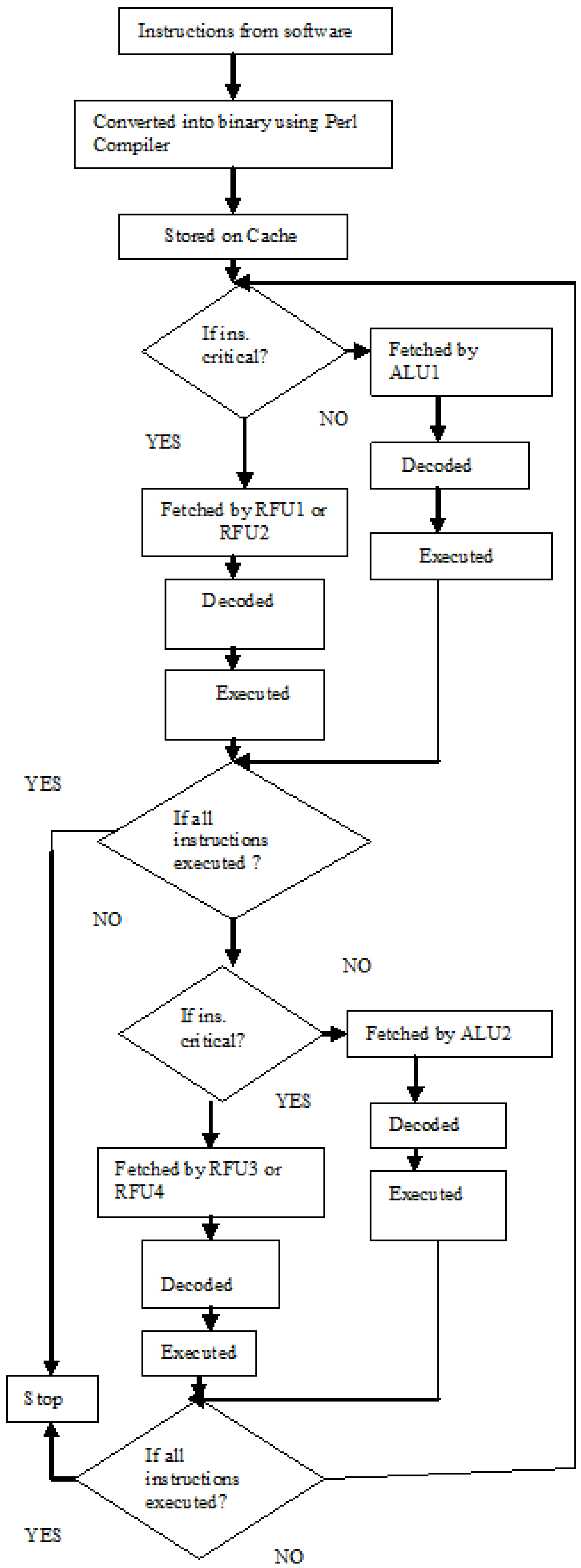

Fig-.4. MRPSoC flow diagram

5. RESULTS

In this paper, the performance of MRPSoC has been examined by executing a set of instructions. These instructions are also executed by using MPSoC to certify that MRPSoC curtails the dispatch time of instruction execution. This is shown in Table.1 and Figure.5.

Table.1. Comparison of MPSoC and MRPSoC based on the dispatch time

Processor/ Dispatch Time(ns)	For pc=5	For pc=7	For pc=9
MRPSoC	37	46	48
MPSoC	37	52	65

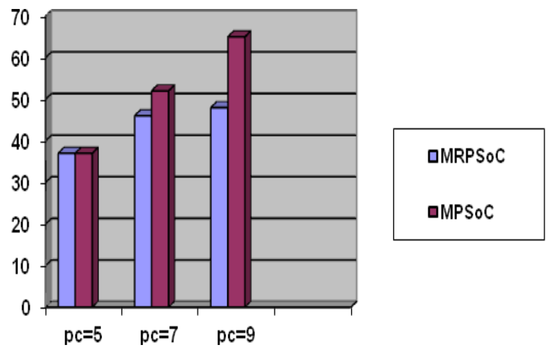

Fig.5. Comparison of MPSoC and MRPSoC based on the dispatch time

6. CONCLUSIONS AND FUTURE SCOPE

In this paper, an MRPSoC is designed with two reconfigurable instruction set processors. Each RISP processors accommodates a special function unit called Reconfigurable Functional Unit (RFU) to reduce the dispatch time of a set of instructions. In order to recognize the curtailment of dispatch time of a set of instructions by using MRPSoC, a set of instructions has been executed in MPSoC also provided the same initial fetch time. After the successful execution of both MPSoC and MRPSoC, it is concluded that MRPSoC curtails the dispatch time of a set of instructions. The MRPSoC curtails the dispatch time of instruction execution by two ways: executing the complex or critical instructions on the RFU and executing the instructions concurrently on the ALU and RFU. For distinguishing the critical instructions, the DCP algorithm is used.

In this paper, a set of instructions has been executed. But in future, real time applications can be implemented using MRPSoC. In this paper, two cores are implemented in a single chip. Later, it can be extended for more than two cores and to apply synchronization between the cores.

Further MRPSoC can be configured for different traffic loads.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my Guide **Mrs. Kavitha.V**, Ph.D. Scholar, Jain University for her timely advice on the technical support and regular assistance throughout the paper work.

REFERENCES

[1]. R.Soleymanpour, Siamak Mohammadi, “A Platform for Multi Reconfigurable Instruction Set Processor System On Chip”, in CSI International Symposium, 2013, pp.99-104

[2]. F. Barat and R.Lauwereins, **“**Reconfigurable Instruction Set Processors: A Survey”, Rapid system prototyping, 11th international workshop, 2000, pp.168- 173.

[3]. F. Barat, R. Lauwereins and G. Deconinck, “Reconfigurable instruction set processors from a hardware/ software perspective”, Software engineering, IEEE Transactions on, vol.28, no.9, pp.847-862, 2002.

[4]. Giovanni Ansaloni, Paolo Bonzini, Laura Pozzi, “EGRA: A Coarse Grained Reconfigurable Architectural Template”, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on Vol.19, no. 6, pp 1062 – 1074, 2010.-

[5]. N. Clark, M. Kudlur, H. Park, S. Mahlke, and K. Flautner, “Application-specific processing on a general-purpose core via transparent instruction set customization,” in *Proc. 37th Ann. Int. Symp. Micro arch.(MICRO’37)*, Washington, DC, Dec. 2004, pp. 30–40.

[6]. A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo, and R.Guerrieri, "A VLIW processor with reconfigurable instruction set for embedded applications," *Solid-State Circuits, IEEE Journal of*, vol. 38,no. 11,pp. 1876-1886,2003.

[7]. G. Martin, "Overview of the MPSoC design challenge," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 274-279.

[8]. R. Soleymanpour, S. Mohammadi, and H. Rajabi, "A synthesis algorithm for customized heterogeneous multi-processors," in *SoC Design Conference (ISOCC), 2012 International*, 2012, pp. 151 -154.

[9]. Z. Chen, R. N. Pittman, and A. Forin, "Combining multicore and reconfigurable instruction set extensions," in *Proceedings of the 18th annual ACMISIGDA international symposium on Field programmable gate arrays*, 2010, pp. 33-36.

[10]. M. A. Watkins and D. H. Albonesi, "ReMAP: A reconfigurable heterogeneous multicore architecture," in *Micro architecture (MICRO),2010 43rd Annual IEEEIACM International Symposium on*, 2010, pp.497-508.

[11]. F. Sun, S. Ravi, A. Raghunathan, and N. Jha, "A Framework for Extensible Processor Based MPSoC Design," *Designing Embedded Processors*, pp. 65-95, 2007

[12]. https://www.ida.liu.se/~TDDI08/labs/lab1.pdf