Middle-East Journal of Scientific Research 15 (7): 998-1004, 2013
ISSN 1990-9233
© IDOSI Publications, 2013
DOI: 10.5829/idosi.mejsr.2013.15.7.1829

Performance Enhancements in Reconfigurable Instruction Set Processors Using Tightly Coupled Configuration Units

M. Aqeel Iqbal and Shoab Ahmed Khan

Department of Computer Engineering, College of Electrical and Mechanical Engineering,
National University of Sciences and Technology (NUST), Islamabad, Pakistan

Abstract: Reconfigurable computing is the next generation approach to compute the algorithms with as much high speed as that of application specific integrated circuits along with the as much flexibility as that of programmable processors. The reconfigurable instruction set processors being most prominent execution platforms for reconfigurable computing have been evolved through many design alternatives. The evolution has explored that the execution performance of a typical reconfigurable processor is greatly dependant on the basic design of its configuration unit. Configuration updation of reconfigurable processor is the dynamically controlled by this unit and hence this unit plays a vital role in the performance enhancement of reconfigurable processor. In this research paper an efficient configuration unit design has been presented which has the capability of loading the most optimal configurations by making the maximum reusability of the existing configuration streams. The simulated results have shown that the proposed configuration unit always demonstrates most optimal configuration overhead for reconfigurable processors and hence can be used in high speed reconfigurable instruction set processors.

Key words: Configurable Logic Blocks · Configuration Streams · Reconfigurable Logic · RISPs · Reconfigurable Computing · Reconfigurable Functional Units

INTRODUCTION

The Reconfigurable Processor which is commonly known as Reconfigurable Instruction Set Processor (RISP) consists of a microprocessor core that has been extended with the reconfigurable logic as shown in Fig. 1. It is similar to an Application Specific Instruction Set Processor (ASIP) but instead of specialized functional units, it contains reconfigurable functional units. The reconfigurable functional units provide the adaptation of the processor to the application, while the processor core provides software programmability [1, 2]. The RISPs execute the instructions, just as normal processors and ASIPs, though the main difference is that the instruction set of a RISP is divided in two main sets: first is the fixed instruction set which is implemented in fixed hardware and second is the reconfigurable instruction set which is implemented in the reconfigurable logic and can be changed during the program execution.

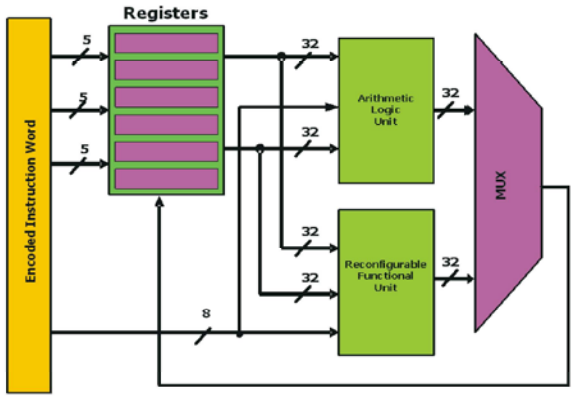

Fig. 1: Reconfigurable Processor Data-path Design

This reconfigurable instruction set is equivalent to the specialized instruction set of the ASIP but with the ability to be modified after the processor has been manufactured [3]. The cost of designing a microprocessor core is reduced by reusing it in many different

Corresponding Author: M. Aqeel Iqbal, Department of Computer Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Islamabad, Pakistan.

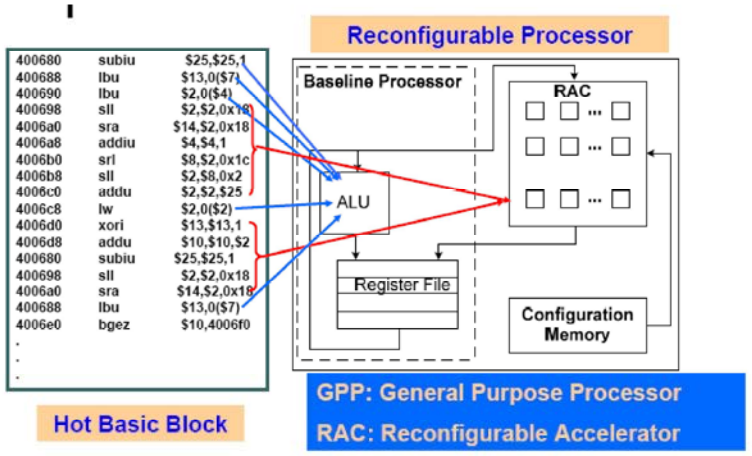
```
400680    subiu    $25,$25,1
400688    lbu      $13,0($7)
400690    lbu      $2,0($4)
400698    sll      $2,$2,0x18
4006a0    sra      $14,$2,0x18
4006a8    addiu    $4,$4,1
4006b0    srl      $8,$2,0x1c
4006b8    sll      $2,$8,0x2
4006c0    addu     $2,$2,$25
4006c8    lw       $2,0($2)
4006d0    xori     $13,$13,1
4006d8    addu     $10,$10,$2
400680    subiu    $25,$25,1
400698    sll      $2,$2,0x18
4006a0    sra      $14,$2,0x18
400688    lbu      $13,0($7)
4006e0    bgez     $10,4006f0
.
.
.
```

Fig. 2: Typical Reconfigurable Processor Architecture

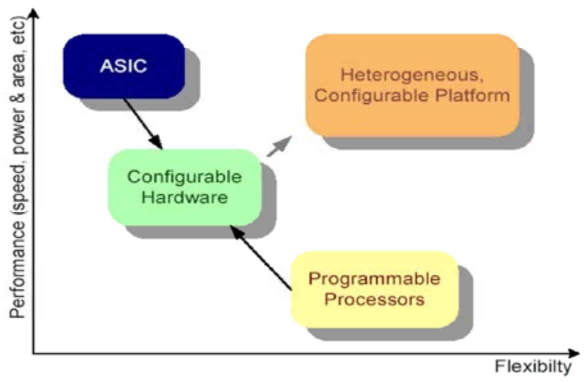
Performance (speed, power & area, etc)
Flexibilty
ASIC
Heterogeneous, Configurable Platform
Configurable Hardware
Programmable Processors

Fig. 3: Performance vs Flexibility for Computing Systems

applications. In ASIPs, the cost for each new generation comes from the re-design of the specialized functional units, which can be quite high [4, 5]. Rapid prototyping techniques are essential for reducing the cost and time required for the re-design. Reconfigurable Instruction Set Processor can be used as a prototyping device for a family of ASIP processors that share the same basic core or fixed instruction set. In this sense, it can be viewed as a field programmable gate array (FPGA) specialized for ASIP design [6].

Furthermore, in some applications the RISPs would be the processor of the choice. Evolving standards, unknown applications and a broad diversity of algorithms are cases where a fixed solution will eventually fail to deliver the required performance. Evolving standards and unknown applications make it very difficult to create specialized hardware for them. Applications with a broad diversity of algorithms require much specialized hardware which may be more expensive than a reconfigurable solution [7]. RISPs offer the flexibility that ASIPs lack. Consider the Figure 2 which represents the architecture for a typically available reconfigurable processor. *Figure Courtesy is for "Farhad Mehdipour", Kyushu University, Fukuoka, Japan.*

As with modern microprocessors, RISP cannot be designed focusing on the hardware alone. The success of RISP depends on the quality of the software development tools available [8]. RISPs are not easy to program without adequate tools. Without them the programmer has not only to program the sequence of instructions the processor will execute but also the instructions themselves. The hardware design of instructions is a new concept for programmers that can be simplified with tools [9]. Consider Figure 3 which represents the comparison for different computing platforms in reference with system flexibility vs. performance.

Related Work: A large no of different reconfigurable hardware based architectures have been proposed so far. Previously proposed reconfigurable processor architectures generally fit into one of two categories depending on the size of the computations they map onto the reconfigurable logic.

First category is *Fine-grained Reconfigurable Processors*, such as PRISC, OneChip and CHIMERAE integrate the small blocks of reconfigurable logic into superscalar processor architectures, treating the reconfigurable logic as programmable ALUs that can be configured to implement application-specific instructions. These systems can achieve better performance than conventional superscalar processors on a wide range of applications by mapping commonly-executed sequences of instructions onto their reconfigurable units, but the maximum speedup they can achieve is limited by the small amount of logic in their reconfigurable units [10].

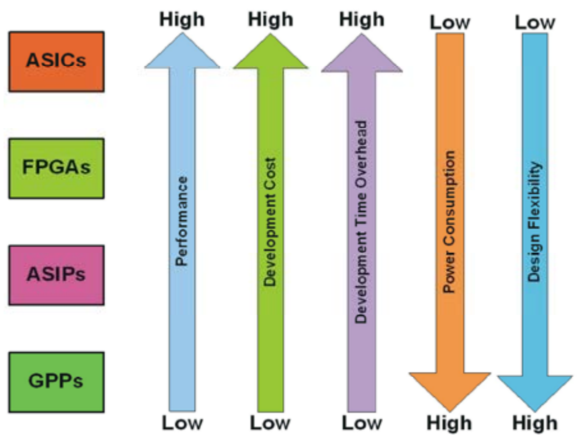

Fig. 4: Characteristics of Computing Systems

Consider the Figure 4 which represents the comparison for different computing platforms in reference with system characteristics.

Second category is *Coarse-grained Reconfigurable Processors,* such as REMARC, Garp, Napa, PipeRench, Rapid and RAW provide larger blocks of reconfigurable logic that are less tightly-coupled with the programmable portions of the processor. These architectures can achieve extremely good performance on applications that contain long-running nested loops that can be mapped onto the processor's reconfigurable arrays but perform less well on applications that require frequent communication between programmable and reconfigurable portions of the processor [11-13].

Proposed Configuration Unit: The performance of the reconfigurable processor is mainly dependant on the time overhead required by it to configure its reconfigurable function units (RFUs). Normally it has been observed that this configuration overhead negatively hits to the computational speed of any reconfigurable processor. Hence researchers are now focusing the issue of configuration overhead minimization for reconfigurable processors. In order to minimize the configuration overheads for the reconfigurable processors; an efficient hybrid design has been proposed for configuration unit of a typical VLIW based reconfigurable processor [3].

The configuration process of the proposed configuration unit is shown in Fig.5. The proposed design includes both the hardwired and the programmable logic modules. The reconfigurable processor being simulated in this research paper is a VLIW processor having a very long instruction word of eight instructions where each instruction is 32-bits instruction.

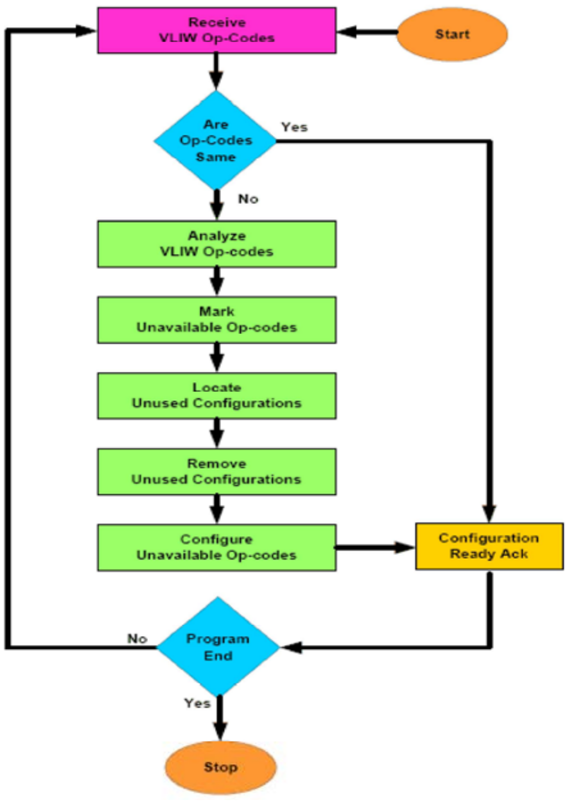

Fig. 5: RISP Configuration Process

Configuration unit plays a vital role in the performance enhancement of the reconfigurable processor [14, 15]. Hence in such a type of processor there is an extra hardware unit being known as configuration unit along with standard micro-programmed control unit whose job is to manage the configuration activities of the reconfigurable processor? The location and the interconnections of the configuration unit inside a typical VLIW based reconfigurable processor have been shown in Fig.6.

Typical RISP Modules: A typical RISP architecture using proposed configuration unit is a high speed VLIW based design using an intensive pipelined architecture. The computation pipeline contains a Fetch Unit (FU), Schedule Unit (SU), Dispatch Unit (DU), Execution Unit (EU) and Register Window (RW).

The FU of the pipeline is responsible to fetch a packet (Long Word) of eight instructions where each instruction is a 32-bits instruction. The FU is a State Machine (Mealy Machine or Moore Machine) based module. It fetches a long word from the instruction cache of processor and loads it into the SU of the pipeline.

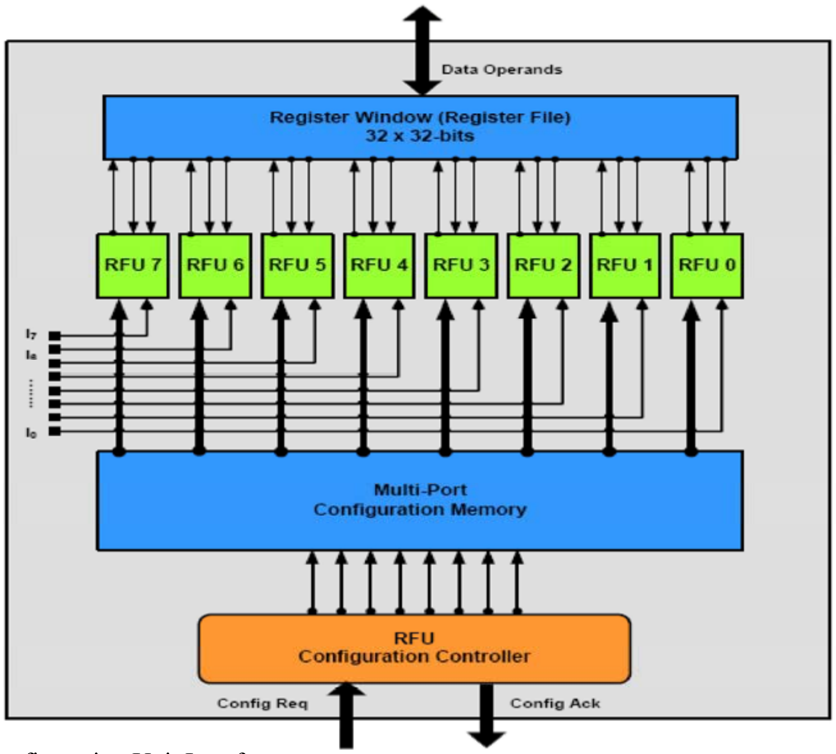

Fig. 6: Proposed Configuration Unit Interfaces

The SU of the pipeline is responsible for scheduling the received long word from the FU. SU loads the long word into the DU of the pipeline and Op-codes of all instructions of the long word are also transferred towards the configuration unit of the processor which updates the RFUs configurations and accordingly sends the dispatch signals to DU so that the instruction can be dispatched to their relevant RFUs. The DU of the pipeline is responsible for dispatching the instructions of long word into their relevant RFUs inside the EU for execution. The DU contains a layer of eight De-multiplexers whose control signals are received from the configuration unit of processor. Each De-multiplexer is a 1 x 8 DMUX of 32-bits size. It transfers one instruction of the long word to one of the eight RFUs which has been reconfigured for it by the configuration unit.

The EU of the pipeline is responsible for the execution of the instructions of the long word. The EU contains a layer of eight RFUs. Each RFU has been integrated with an FPGA core like provided by the Xilinx Corporation and a layer of common data buses. The FPGA core is configured by the configuration unit of the processor according to the execution requirements of running application program. The proposed design is a Register-Register Architecture in which the source operands required by each instruction are fetched from the register window of the processor and similarly the results generated after the execution of each instruction are stored back temporarily to same register window of processor. The register window of pipeline is responsible for providing the source data operands for the execution of eight instructions of the long word and temporarily storing their results. The register window contains a layer of thirty two registers where each register is a 32-bits register.

Proposed Configuration Unit: Consider the Fig. 7 for the detailed design of the proposed configuration unit and flow chart in Fig.5 for its configuration updation process. The main job of the configuration unit is to update the loaded configurations of the RFUs according to the changing requirements of the running application. The RFUs have been integrated with the high speed partially reconfigurable FPGA cores like those provided by the Xilinx Virtex series of FPGAs. RFUs make the actual execution layer of the EU being available inside the computational pipeline. The reconfigurable systems mostly require a lot of time to perform this configuration update process. Hence this configuration overhead greatly degrades the performance of such systems.

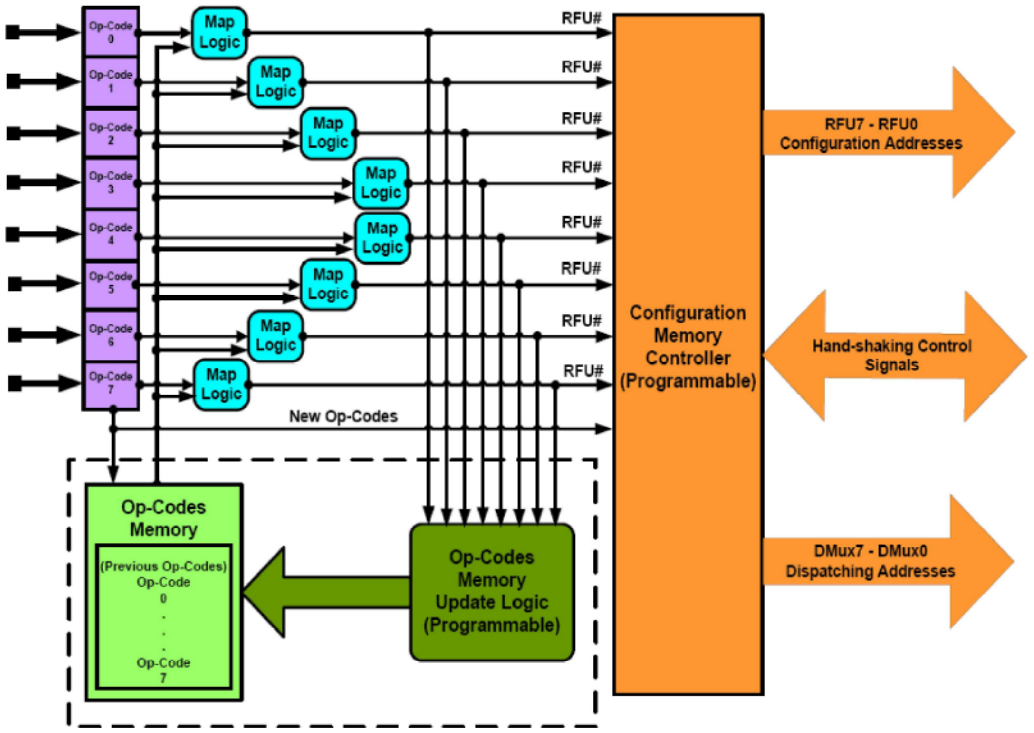
Op-Code 0
Op-Code 1
Op-Code 2
Op-Code 3
Op-Code 4
Op-Code 5
Op-Code 6
Op-Code 7
Map Logic
RFU#
New Op-Codes
Configuration Memory Controller (Programmable)
RFU7 - RFU0 Configuration Addresses
Hand-shaking Control Signals
DMux7 - DMux0 Dispatching Addresses
Op-Codes Memory
(Previous Op-Codes) Op-Code 0 . . . Op-Code 7
Op-Codes Memory Update Logic (Programmable)

Fig. 7: Proposed Configuration Unit Design

In order to optimize the configuration overhead of a typical reconfigurable processor based on VLIW architecture; a unique idea of RFU Partial Configuration has been introduced. In this technique the configuration unit constantly keeps on monitoring the currently loaded configurations and newly demanded configurations. It maps the newly demanded configurations on the currently available configurations and maximizes the reusability of the already available configurations and loads only those configurations which are no more currently available in any RFU of the reconfigurable processor [3].

The configuration unit contains a layer of *Op-code Map Logic (OML)* which actually compares each Op-code of the incoming long word with all Op-codes of the currently configured long word in RFUs as shown in Fig. 7. This mapping process is performed concurrently with a high speed ASIC circuit which contains a parallel network of comparators. All those op-codes who have been compared with any of the existing op-codes are then allocated their respective RFU no, where they will be executed. If an op-code has not been compared with any one of the existing op-codes then it is not allocated any RFU no. This information of RFU allocation or not allocation is sent to two programmable logic controllers. One is known as the *Configuration Memory Controller (CMC)* and is responsible to generate the RFU Configuration addresses for only those RFUs who really need configuration updation and at the same time it calculates and sends the sufficient control signals to DU of computational pipeline [3].

On the basis of these signals the DU dispatches the instructions into their relevant RFUs. Second is known as the *Op-codes Memory Update Logic (OMUL)* and is responsible to update the op-codes memory contents according to the newly arrived op-codes of the long word. The time taken by the Map Logic to compare the all incoming op-codes with the all existing op-codes is always constant and is equal to 1-Cycle. But the time taken by the CMC and OMUL are variable and are dependent on the no of the newly arrived op-codes that are not matched with the existing op-codes and it may vary from 0-Cycles to 8-Cycles. If all op-codes are matched then its latency is 0-Cycles and if none of them is matched with the existing op-codes then its latency will be 8-Cycles and so on. For those applications where the same operation is repeated again and again like the operation of convolution in conventional DSPs; they will always be given 0-Cycle latency and hence it dramatically enhances the computation speed of the system by minimizing the configuration overhead to 0-Cycles [3].

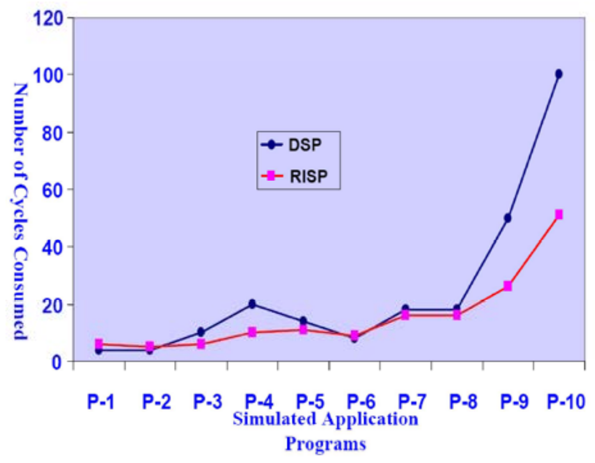

Fig. 8: DSP vs RISP Using Proposed Unit

Table 1: Model Parameters

Parameters	Possible Values
No of Fetched Packets, N_{FP}	1, 2, 3, ...N / Program
Packet Fetch Time, T_{PFT}	1-Cycle / Fetched Packet
Operand Fetch Time, T_{OFT}	1-Cycle / Fetched Packet
Execute Packets, E_N	1 / Fetched Packet
Delay Slots, D_N	0 - to -1 Cycle
Functional Unit Latency, F_N	1-Cycle / Execute Packet
Configuration Time, T_C	0 - to -1 Cycle
Dispatching Time, T_D	1-Cycle / Execute Packet

Such kind of drastic performance revolutions that have been observed are shown in the performance graph of RISP in Fig.8 which has been obtained by using the proposed configuration unit in a typical VLIW based reconfigurable processor and benchmarking its performance with a DSP (TMS320C6X). The configuration unit has a RFU Configuration Controller and a Multi-port Configuration Memory as shown in Fig.6. RFU configuration controller is responsible for providing optimal configuration overhead. The multi-port configuration memory contains a set of most frequently used configurations that can be dynamically changed during the execution of the application by loading them externally through Configuration EPROM.

Mathematical Model: Following is the mathematical formula being formulated for the calculations of total number of cycles (T_{Total}), consumed for the execution of an application program. Consider the model parameters in Table 1.

$$T_{total} = \sqcap(N_{FP}, \beta_{FP}) + \Sigma\, E_N\,(F_N + D_N) + T_C + T_D \text{ Cycles}$$

where $\beta_{FP} = \Sigma\,(T_{PFT}, T_{OFT})$

Simulated Performance: In order to measure the performance gain achieved by the use of the proposed configuration unit, a typical RISP has been integrated with it and the performance of the RISP has been analyzed and benchmarked with a typical DSP processor by executing a variety of application programs on both of them. Performance statistics have been measured in terms of the no of clock cycles being consumed by each of them for program execution. It has been observed that the segments of code of an application program containing repeated operations or loops of operations will be drastically boasted as shown in the graph in Fig. 8.

CONCLUSION

Reconfigurable Instruction Set Processors (RISPs) are the best computing platforms which can be used as an alternative to Application Specific Instruction Set Processors (ASIPs) as well as an emerging prototyping platform for Application Specific Integrated Circuits (ASICs). The performance of a typical RISP is dependant on different parameters including the design of its configuration unit. The configuration unit is the main module of processor which is responsible for managing all of the activities relevant to the configuration updation of reconfigurable functional units (RFUs). Hence it plays a vital role in the enhancement of the performance of a typical RISP. The configuration overhead of RFUs can be drastically optimized by partially loading only unavailable configuration streams and making the maximum reusability of the already existing configurations. The proposed configuration unit always loads the most optimal configurations and hence always demonstrates the most optimal configuration overhead. This loading of the most optimal configuration streams by intelligently reusing the existing configurations explores many new directions of the research in the field of the reconfigurable computing.

REFERENCES

1. Aqeel Iqbal, M., Farooque Azam, Uzma Saeed Awan and Saif ullah Hammad, 2011. Performance enhancement techniques for modern reconfigurable computing systems, International Journal of Computer Applications (IJCA), 27(09): 33-38.
2. Todman, T.J., G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk and P.Y.K. Cheung, 2005. Reconfigurable computing: architectures and design methods, Proceedings of IEE Computers and Digital Techniques, 152(02): 193-207.

3. Aqeel Iqbal, M. and Uzma Saeed Awan, 2009. RISP Configuration Overhead Optimization Using An Efficient Configuration Unit, Proceedings of ASME International Conference on Advanced Computer Theory and Engineering (ICACTE-2009), pp: 15-24, Cairo, Egypt.
4. Wilton, S.J.E., 2002. Implementing logic in FPGA memory arrays: heterogeneous memory architectures, Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM' 02), pp: 142-147, Napa Valley, Calif, USA.
5. Plessl, C. and M. Platzner, 2005. Zippy, A coarse-grained reconfigurable array with support for hardware virtualization, Proceedings of the 16th International Conference on Application-Specific Systems, Architecture and Processors (ICASAP' 05), pp: 213-218.
6. Compton K. and S. Hauck, 2000. An introduction to reconfigurable computing. IEEE Computer.
7. Francisco Barat, R. and Lauwereins G. Deconinck, 2002. Reconfigurable Instruction Set Processors from a Hardware/Software Perspective; IEEE Transactions on Software Engineering, 28(9): 847-862.
8. Hartenstein, R., 2001. A decade of reconfigurable computing: a visionary retrospective. Proceedings of Design, Automation and Test in Europe Conference (DATEC' 01), pp: 642-649, Munich, Germany, 2001.
9. Azween Bin Abdullah, 2009. Survivability Using Adaptive Reconfigurable Systems, IJCSNS International Journal of Computer Science and Network Security, 9(1).
10. Kuon J. Rose, 2006. Measuring the gap between FPGAs and ASICs, Proceedings of the ACM/SIGDA 14th International Symposium on Field-Programmable Gate Arrays (FPGA' 06), pp: 21-30, Monterey, Calif, USA, 2006.
11. Benkrid Khaled, 2008. High performance reconfigurable computing: from applications to hardware, IAENG International Journal of Computer Science (IJCS), 35(01).
12. Aqeel Iqbal, M., Shoab A. Khan and Uzma Saeed Awan, 2009. Reconfigurable computing systems related hardware and software perspectives, International Journal of Intelligent Information Technology Application (IJIITA), 02(05): 209-217.
13. Compton, K. and S. Hauck, 2002. Reconfigurable computing: a survey of systems and software, ACM Computing Surveys, 34(02): 171-210.
14. Aqeel Iqbal, M., Asia Khannum, Saleem Iqbal and M. Asif, 2010. Emerging requirements of reconfigurable computing systems for performance enhancement, International Journal on Computer Science and Engineering (IJCSE), 02(05): 1572-1579.
15. Aqeel Iqbal, M., Uzma Saeed Awan and Shoab A. Khan, 2010. Reconfigurable computing technology used for modern scientific applications, Proceedings of 2nd IEEE International Conference on Education Technology and Computer (ICETC' 10), pp: 36-41, Shanghai, China, 2010.