DJ Journal of Advances in Electronics and Communication Engineering, Vol. 4(2) 2018, pp. 1-10

RESEARCH ARTICLE

FPGA Implementation of On-Chip Network

[*]N Murali Krishna[1]
[1]Department of ECE Sreyas Institute of Engineering &Technology, Hyderabad, India.

Received- 20 February 2018, Revised- 17 April 2018, Accepted- 23 April 2018, Published- 27 April 2018

ABSTRACT

This paper presents the design of 32 bit UART (Universal Asynchronous Receiver Transmitter) RISC (Reduced Instruction Set Computing) processor with dynamic power management system to minimize power consumption and transmission cost. Coarse grained architecture is suggested due to its innumerable advantages over fine grained architecture. Coarse Grained Arrays (CGAs) with run-time re-configurability play a challenging task to design Network on-Chip (NoC) communication systems satisfying the power and area of embedded system. The proposed architecture is implemented on FPGA (Field Programmable Gate Array) using VHDL (VHSIC Hardware Description Language), and the obtained comparison power graph signifies that it consumes less power when compared to BETA RISC processor.

Keywords: Coarse grained architecture, RISC processor, UART, Network on-chip, FPGA implementation.

1. INTRODUCTION

Reconfigurable computing architecture plays a significant role in embedded and high performance computing system. When compared to Application Specific Integrated Circuits (ASIC's) and microprocessor, reconfigurable architecture is commonly employed because of its advantages such as high throughput, low cost etc. Most of the architectures face limitations based on placement, routing and granularities which are overcome by dynamic compilation [1-4]. Fine grained architecture provides the advantage of high flexibility but is not used because of its inefficiency applications. Coarse grained reconfigurable system consists of elementary blocks to provide efficient application without providing gate level mapping. Several dynamic setup schemes are also incorporated in coarse grained array to overcome such limitations. Due to certain advantages such as improved performance, logic density and power efficiency, FPGA is widely employed in digital applications. Initially, optimizing programmable logic and routing architecture provides significant improvements in FPGA. Nowadays, the coarse grained elements including memories, processor, etc. are incorporated into fine grained programmable logic to provide better performance and high efficiency [5].

Coarse Grained Reconfigurable Array (CGRA) architecture accelerates the computation of algorithms in several scientific domains by reducing energy consumption. CGRA consists of number of interconnected reconfigurable processing units to perform arithmetic functions and logic and conditional operations, etc. [6]. Parallel processing adds advantage to coarse grained array, which operates the computational resources in parallel hence suitable for processing multiple parallel streams. These reconfigurable parallel architecture contains number of parts to execute the processing element concurrently, but drawbacks arises while realizing these

*Corresponding author. Tel.: +919885837535
Email address: omkrish29@gmail.com (N.M.Krishna)
Double blind peer review under responsibility of DJ Publications
https://dx.doi.org/10.18831/djece.org/2018021001

dependencies at compiling time which limits the exploitation parallelism. Hence network on-chip models are devised to develop scalable and reusable applications. CGRA operates with RISC processor to benefit general and special purpose processing. The processor minimizes the resource utilization and dissipated power by satisfying all the requirements of FPGA device [7, 8].

2. LITERATURE SURVEY

32-bit processor based on the open-source RISC-V (RV32I) ISA (Instruction Set Architecture) is proposed to target low cost embedded devices. The processor core is divided into logical modules such as fetch, decode and control logic responsible for fetching the instruction from memory, decoding and resolving jump and target address accordingly. The advanced version of this processor and software paves way for the application in future embedded systems [9]. SHA-3 Instruction Set Extension (ISE) is proposed to improve the performance of 32-bit MIPS processor. Two ISE approaches are proposed such as native data path and coprocessor-based ISE in which the utilization factor tends to increase with high execution speed [10]. A 32 bit re-configurable RISC processor is designed based on BETA ISA. Introduction of matrix multiplier enables the processor to be used in digital signal and image processing. This processer is considered as a prominent one due to its fault tolerant ability but it suffers from power consumption problem [11].

[12] has proposed an embedded RISC processor (RISC 32-bit) with dynamic power management capability which is designed based on Direct Memory Access (DMA), UART and timer and memory controllers. It operates at a maximum speed of 401.881 MHz with power usage of 1440 mW resulting in improved performance. Epiphany RISC array processor offers high computational energy-efficiency and parallel scalability. In spite of these advantages, it faces challenges due to parallel programming model. To overcome this, Message Passing Interface (MPI) standard is proposed, which supports larger problem size greater than the available local memory. The cost associated with this standard prevents the use of large buffers in the global memory [13].

RISC soft-core called low RISC is proposed based on RISC-V ISA to estimate the reliability of this processor. Several fault errors usually occur in this processor. Therefore modification has to be done in processor design to improve its fault tolerant system thereby mitigating single event upset [14]. BTWC (Better Than Worst Case) RISC processor is designed to avoid worst case extra delay in critical path without providing significant improvement in performance. The results can be obtained by integrating latency-insensitive design and Variable-Latency (VL) unit. If error correction or two cycle execution is infrequent, then BTWC possesses high execution speed [15]. Due to large size bit stream and inherent complexity, dynamic reconfiguration of FPGA device becomes a non-feasible approach. CGRA offers a solution, however it has difficulties in implementation. Thus scalable CGRA is proposed to ease the implementation of algorithms on FPGA platforms. This proposed method is advantageous with respect to FPGA technology and standard CGRA [16].

[17] has proposed an integration phase of Memory Management Unit (MMU) to COFFEE (Core For FREE) RISC processor and has provided virtual memory to run operating system without degradation in operating frequency. COFFEE processor finds suitable application in embedded and system-on-chip due to its important characteristics such as reusability and configurability. It has the capability to execute 66 instructions in 6-stage pipeline. [18] has discussed an OpenSHMEM implementation of Adapteva Epiphany RISC array processor, that provides excellent one-sided communication routines. The OpenSHMEM routines enable compact implementation thereby saving memory resources. One-sided communication and weaker synchronization requirements minimize the code size of an application when compared to MPI routines.

3. COARSE GRAINED ARCHITECTURE

Depending on granularity i.e. number of bits manipulated by the programmer, reconfigurable architecture is classified as fine grained and coarse grained whereas fine grained architecture that is commonly employed in FPGA allow data manipulation in

bit level consisting of processing elements with least number of significant input and output. Coarse grained architecture overcomes the limitations of fine grained structure, and hence suitable for all related applications by providing operator level functional blocks, efficient routing switches etc. These advantages experience reduction in configuration time and memory as well as placement allocation and routing. Table 1 shows the properties exhibited by fine and coarse grained architecture based on granularity [19].

Table 1.Fine and coarse grained based on granularity

Properties	Granularity	
	Fine grained	Coarse grained
Operation	Bit level	Word level
Reconfiguration information	High	less
Performance	Medium	High
Flexibility	High	Medium/High
Configuration time	Long	Short

Due to several advantages, coarse grained offer better performance when compared to fine grained hence commonly employed in many applications. The architecture of coarse grained structure shown in figure 1 consists of a RISC processor, DMA controller, external memory block and reconfigurable computing module interconnected through data bus [20].

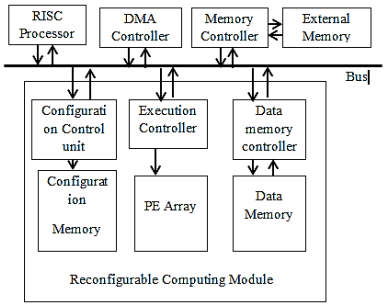

Figure1.Coarse grained architecture

The function of coarse grained architecture relies mainly on RISC processor, which controls the other components during execution. It supports single cycle operation with maximum memory speed [21]. RISC processor increases the speed of computer and reduces the time needed to accomplish each operation. Also this processor consists of a flexible architecture thus minimizing dynamic power consumption. Direct Memory Access controller is used between Reconfigurable Computing module (RCM) and main memory to provide efficient communication [22]. DMA controller acts as co-processor to handle the data managed by RISC processor. It is important to use this controller to perform data movement task.

RCM plays a vital role in accelerating data intensive code blocks depending on application. This module operates faster when compared to accessing the context from external memory. It consists of elements such as configuration control unit, execution controller, data memory controller, data memory, Processing Element (PE) array and configuration memory. The configuration control unit controls the configuration memory whereas data memory controller controls the data memory. Execution controller supervises the overall action of RCM [23]. Data memory stores the instruction of the application unit. Configuration memory is an important block of RCM, providing data flow diagram codes (DFM) based on applications. The codes will be loaded in the hardware to configure the specific application and it also specifies interconnected gates as well as programming element to be used. Configuration memory also provides hardware mapping to its application [24]. PEs are the processing blocks said to be homogenous. Based on the designer requirement, the number of processing element used is decided. Each element is associated with corresponding row and column identified by index number, which addresses the element while data transferring. It is a small processor without instruction fetch and branch units. PE is configured by configuration memory and configuration control unit. Figure 2 shows the interconnected topology in PE array.

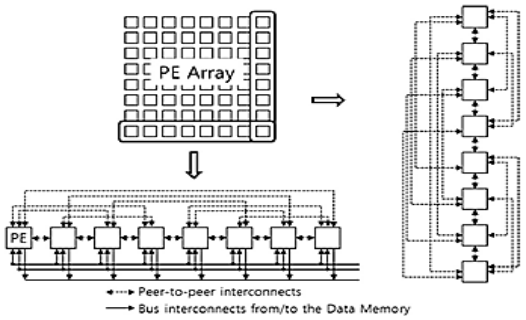

Figure 2.Topology interconnection in PE array

The topology to interconnect cells in PE array is different. There are different types of topology used based on the application requirements. Figure 3 shows the examples of interconnections and topology used in the processing element [25].

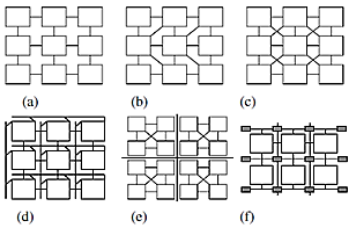

Figure 3.Various interconnections and topology (a) Grid topology (b) Hexagonal (c) Octal Global buses to connect I/O ports (e) Cluster topology (f) Grid topology using crossbars

4. NETWORK ON-CHIP

In order to meet the requirements of low power and high performance, the resources used in single chip has automatically increased. Interconnection between the resources plays a challenging task [26]. Shared interconnection bus topology is commonly employed in most of the system on-chip applications but it faces limitations due to its scalability. Only one master can use the bus at a time and other should wait until the process gets completed. In the situation when the bus request and required bandwidth is large, this cannot be considered. Hence NoC architecture is designed. NoC provides a methodology to connect hundreds of IP cores used for general purpose processors, application specific processors, digital signal processors and so on. Figure 4 shows the structure of NoC [27].

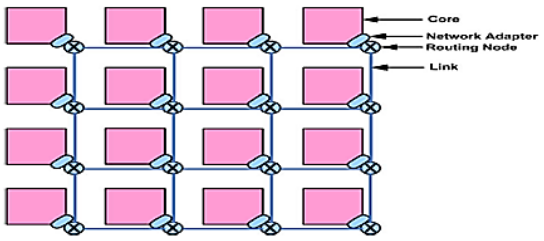

Figure 4.Structure of NoC

NoC is composed of three main components such as network adapter, routing node and link. The network adapter provides interface between processing core and communication network. Router executes the routing mechanism and provides the information through channels or links. Its performance depends on throughput and latency. This authorizes the parallel communication which results in high bandwidth. The reconfiguration time must be smaller than the computational blocks.

5. RISC PROCESSOR

The design block of 32-bit RISC processor incorporates Arithmetic Logic Unit (ALU), Accumulator, Program Counter (PC), Instruction Register (IR), memory, Control Unit (CU) and additional logic to handle 32 bit data, 28 bit address and uses fixed instruction format of length 32 bit. Size of opcode is of 4 bit which can handle 15 instructions with a 256 memory locations. Figure A1 presents the architecture of RISC processor [28]. ALU takes input either from memory, register bank or immediate data and perform arithmetic and logic operations. In addition to that, it also performs some bit operations like rotate or shift [29]. PC administers the fetched instruction counter. PC is a latch which contains the memory address from which the processor fetches the instructions. PC is the largest sub-block, and second to control unit based on complexity. It consists of 6-bit pointer to specify the instruction memory and additionally uses a 6-bit pointer to indicate the data memory, which is used only when a load/store instruction is executed. Instruction execution flow and logical operation flow of processor is controlled by PC. Generally it executes incrementing and loading operation. Conventional adder circuit is equipped in PC to perform increment operation.

Instruction fetch unit fetches instruction from instruction memory using the current value of PC and increments the PC value for the next instruction. The logic elements used in instruction fetch unit are 8-bit PC register, an adder to increment PC by four, instruction memory, a multiplexor, and an AND gate to select the value of the next PC. Depending on memory address pointed by program counter, the instruction is fetched and placed in data bus which is called as called as instruction fetch cycle. From the data bus, the data is loaded into instruction register which is known to be instruction load. In this cycle, 4 MSB of the instruction are separated and

placed in the opcode register which are finally loaded to control unit and ALU. The remaining bits are sent out [30].

MUX connects the output of instruction memory and PC. During negative cycle, instruction register output is enabled while during positive cycle, PC output is enabled. The important function of instruction decode unit is to acquire the register data values from the instruction fetch. The logic elements included in decode unit are multiplexors and the register file [31]. Register file consists of eight general purpose registers of 32-bits capacity each, which are used while executing the arithmetic and data-centric instructions. The load instruction loads the value into the register file and store instruction retrieves the files back to memory. Control unit checks whether the entire processor operates correctly. It also generates various signals and co-ordinates the overall modules structure [32]. Data memory unit uses the load and store instructions. The load instruction asserts the MemRead signal and the obtained read output is written into register file. A store instruction asserts the MemWrite signal and writes the data to processor.

6. UART

UART consists of transmitter and receiver blocks. The transmitter converts the input data byte into bits and sends one by one serially while the receiver receives the serial bit data and converts them to data byte. Figure 5 shows the block diagram of UART [33].

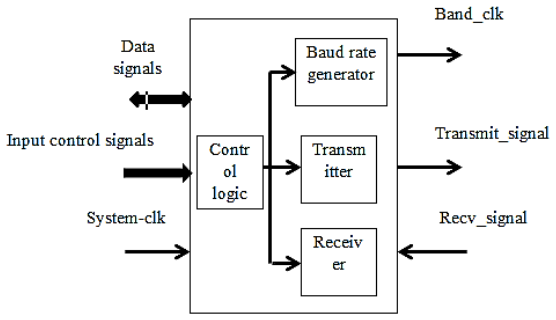

Figure 5.Block diagram of UART

UART transmits an individual bit in sequential fashion hence called as serially based device. Certain timing parameters should be followed by sender and receiver, and special bits are added to each word in order to synchronize the units. Asserting a "start bit" in UART initiates data transmission and also informs the receiver, that data byte is about to send. Each bit of data byte is sent serially and equal amount of time is needed for each bit transmission. After receiving the bits, the receiver samples the logic depending on the period assigned to it. The transmitter may add "parity bit" indicating the receiver to perform simple error checking. When data is transmitted, the transmitter sends "stop bit" pointing the completion of data transmission. For each byte of data, similar process is carried out by the transmitter. Since RISC processor processes parallel input, the serial output of UART is converted into parallel using SIPO (Serial In Parallel Out) register, thus producing parallel output. Since serial interface has several advantages over parallel interface with respect to long cable length, simple wiring etc., the RISC parallel output is converted to serial using PISO register [34].

7. LOW POWER TECHNIQUE

The traditional RISC processor consumes more power. Reduction of power is done at fabrication step, which is a complex process. Power reduction technique used in this method is clock gating, in which it prevents the clock signal from reaching various modules and hence the input to the circuit remains unchanged resulting in no switching activity. Leakage power i.e. quiescent power cannot be reduced but reduces the dynamic power consumption. This system is wireless and thus can be more acceptable and cost effective [35].

8. IMPLEMENTATION ON NoC

Coarse grained 32 bit RISC processor, with UART is implemented on NoC as shown in figure 6. [17] NoC system consists of RISC processor, memory, UART, network interfaces and routers. RISC consists of separate memory management units (MMUs) for instruction and data memory. Several low power design of RISC processor has certain limitations on NoC such as (i) linear increase in battery capacity (ii) need of expansive cooling system (iii) co-dependent voltage and frequency etc. These limitations degrade the system performance. Hence dynamic power management technique is introduced in RISC processor to adjust the power consumption by activating small number of devices to attain better performance.

RISC consists of relatively few instructions, addressing modes and formats. Due to these reductions advantages, designer can fit large number of CPU registers into chip thereby enhancing the throughput of the processor. UART model used in this system is 16550 UART [36].

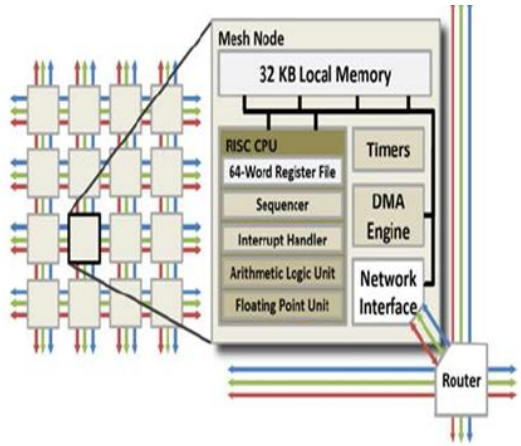

Figure 6.Implementation on NoC

Flash memory used in this system is Intel's smart voltage 28F016S2 flash with a capacity of 2MB.WISHBONE is a bus protocol standard used to interconnect IP cores using various interconnection schemes. NI is used to convert the wishbone signals from the IP cores to packets.

9. RESULTS AND DISCUSSION

32 bit UART RISC processor designed with dynamic power management is implemented on FPGA based NoC.

Figure 7.Simulated waveform of proposed architecture

FPGA, a dominant reconfigurable fabric well known by their flexibility finds applications in specific architecture like memory, processor etc. This technology promises to reshape the architecture by providing the best hardware resources which is implemented in VHDL and simulated using Xilinx ISE 12.3 [37]. Simulated waveform of the proposed architecture is shown in figure 7.

9.1 RTL schematic

After obtaining simulation results, RTL schematic representation is generated. RTL schematic refers to the representation of design on the basis of symbols. The symbol includes logic gates, counters, adders etc. Figure 8 shows the RTL schematic of this proposed scheme. In RTL view, clk is the input of NoC to synchronize the processor and peripheral devices. Reset is the input to synchronize the processor with clk and is used to reset the memory contents and all other system components.

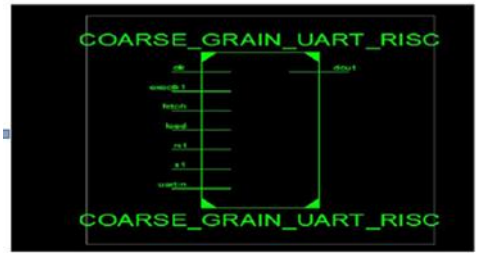

Figure 8.RTL schematic of proposed RISC processor.

9.2 Technology schematic

The internal design of the RTL is generated by technology schematic which is shown in figure 9.

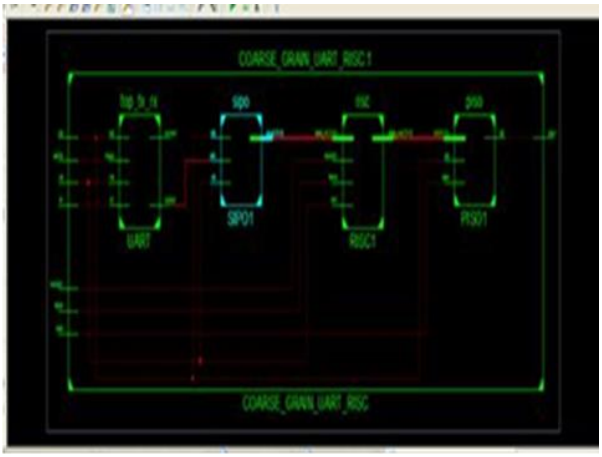

Figure 9.Technology schematic

Power consumption of proposed design is obtained using Xilinx XPower analyser, and performance comparison based on power is shown in figure 10. Existing method based on BETA RISC processor consumes 93.09 mW of power whereas this

proposed architecture consumes only 2.706 mW, thus resulting in better performance.

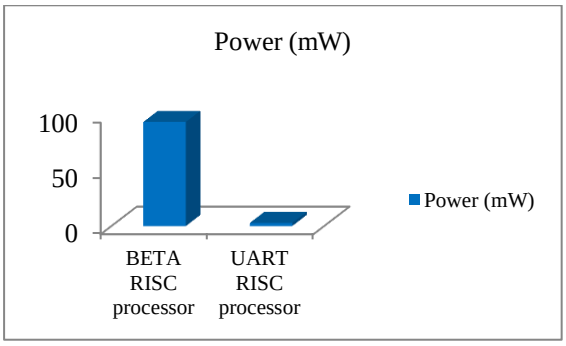

Figure 10.Power comparison between existing and proposed method

10. CONCLUSION

This paper presents the design and implementation of a 32 bit UART RISC processor intended for computer architecture. The proposed architecture is implemented in VHDL and simulated using Xilinx software. The simulated power graph result indicates that this method consumes less power when compared to existing method and hence finds application in certain area where power is considered as a main criterion.

REFERENCES

[1] Ricardo S.Ferreira, Alex Damiany, Julio Vendramini and Tiago Teixeira, Fast Placement and Routing by Extending Coarse-Grained Reconfigurable Arrays with Omega Networks, Journal of Systems Architecture, Vol. 57, No. 8, 2011, pp. 761-777, https://dx.doi.org/10.1016/j.sysarc.2011.03.006.

[2] Seamas McGettrick and C.J.Bleakley, Rapid Functional Modelling and Simulation of Coarse Grained Reconfigurable Array Architectures, Journal of Systems Architecture, Vol. 57, No. 4, 2011, pp. 383-391.

[3] Giovanni Ansaloni, Paolo Bonzini and Laura Pozzi, EGRA: A Coarse Grained Reconfigurable Architectural Template, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 19, No. 6, 2011.

[4] A.K.Parvathi, A Network Architecture using Super Base Station for Communication in Energy-Efficient Fifth-Generation, DJ Journal of Advances in Electronics and Communication Engineering, Vol. 1, No. 2, 2015, pp. 1-11, http://dx.doi.org/10.18831/djece.org/2015021001.

[5] Chi Wai Yu, Julien Lamoureux, Steven J.E.Wilton, Philip H.W.Leong and Wayne Luk, The Coarse-Grained/Fine-Grained Logic Interface in FPGAs with Embedded Floating-Point Arithmetic Units, Southern Conference on Programmable logic, 2008.

[6] Grigorios Dimitroulakos, Stavros Georgiopoulos, Michalis D.Galanis and Costas E.Goutis, Resource Aware Mapping on Coarse Grained Reconfigurable Arrays, Microprocessors and Microsystems, Vol. 33, No. 2, 2009, pp. 91-105, https://dx.doi.org/10.1016/j.micpro.2008.07.002.

[7] Sajjad Nouri, Waqar Hussain and Jari Nurmi, Implementation of IEEE-802.11a/g Receiver Blocks on a Coarse-Grained Reconfigurable Array, Design and Architectures for Signal and Image Processing (DASIP), Poland, 2015.

[8] Zain-ul-Abdin and Bertil Svensson, Evolution in Architectures and Programming Methodologies of Coarse-Grained Reconfigurable Computing, Vol. 33, No. 3, 2009, pp. 161-178, https://doi.org/10.1016/j.micpro.2008.10.003.

[9] Don Kurian Dennis, Ayushi Priyam, Sukhpreet Singh Virk, Sajal Agrawal, Tanuj Sharma and Arijit Mondal, Single Cycle RISC-V Micro Architecture Processor and its FPGA Prototype, International Symposium on Embedded Computing and System Design, India, 2017.

[10] Ahmed S.Eissa, Mahmoud A.Elmohr, Mostafa A.Saleh, Khaled E.Ahmed and Mohammed M.Faraq, SHA-3 Instruction Set Extension for a 32-bit RISC Processor Architecture, IEEE International Conference on Application-Specific Systems,

Architectures and Processors, UK, 2016.

[11] Raj Prakash Singh, Ankit K.Vashishtha and R.Krishna, 32 Bit Re-configurable RISC Processor Design and Implementation for BETA ISA with Inbuilt Matrix Multiplier, International Symposium on Embedded Computing and System Design, India, 2016.

[12] Narender Kumar and Munish Rattan, Implementation of Embedded RISC Processor with Dynamic Power Management for Low-Power Embedded System on SOC, International Conference, India.

[13] James A.Ross, David A.Richie, Song J.Park and Dale R.Shires, Parallel Programming Model for the Epiphany Many-Core coprocessor using Threaded MPI, Microprocessors and Microsystems, Vol. 43, 2016, pp. 95-103, https://dx.doi.org/10.1016/j.micpro.2016.02.006.

[14] Alexis Ramos Juan and Antonio Maestro Pedro Reviriego, Characterizing a RISC-V SRAM-based FPGA Implementation Against Single Event Upsets using Fault Injection, Microelectronics Reliability, Vol. 78, 2017, pp. 205-211 https://dx.doi.org/10.1016/j.microrel.2017.09.007.

[15] Mario R.Casu and Paolo Mantovani, A Synchronous Latency-Insensitive RISC for Better than Worst-Case Design, Integration, The VLSI Journal, Vol. 48, 2015, pp. 72-82, https://dx.doi.org/10.1016/j.vlsi.2014.01.003.

[16] Claudio Brunelli, Fabio Garzia, Davide Rossi and Jari Nurmi, A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Supporting Subword and Floating-Point Calculations, Journal of Systems Architecture, Vol. 56, No. 1, 2010, pp. 38-47, https://dx.doi.org/10.1016/j.sysarc.2009.11.003.

[17] Farid Shamani, Vida Fakour, Sevom Tapani and Ahonen Jari Nurmi, Integration Issues of a Run-Time Configurable Mmanagement Unit to a RISC processor on FPGA, Microprocessors and Microsystems, Vol. 49, 2017, pp. 179-191, https://dx.doi.org/10.1016/j.micpro.2016.12.001.

[18] James A.Ross and David A.Richie, Implementing OpenSHMEM for the Adapteva Epiphany RISC Array Processor, Procedia Computer Science, Vol. 80, 2016, pp. 2353-2356, https://dx.doi.org/10.1016/j.procs.2016.05.439.

[19] Rahul K.Hiware and Dinesh Padole,Configuration Memory Based Dynamic Coarse Grained Reconfigurable Multicore Architecture for 8 Point FFT, International Conference on Emerging Trends in Engineering and Technology, Japan, 2015.

[20] Manhwee Jo, Dongwook Lee, Kyuseung Han and KiyoungChoi, Design of a Coarse-Grained Reconfigurable Architecture with Floating-Point Support and Comparative Study, Integration, The VLSI Journal, Vol. 47, No. 2, 2014, pp. 232-241, https://dx.doi.org/10.1016/j.vlsi.2013.08.003.

[21] Samiappa Sakthikumaran, S.Salivahanan and V.S.Kanchana Bhaaskaran, 16-Bit RISC Processor Design for Convolution Application, International Conference on Recent Trends in Information Technology, India, 2011.

[22] Priyanka Trivedi and Rajan Prasad Tripathi, Design & Analysis of 16 bit RISC Processor using Low Power Pipelining, International Conference on Computing, Communication & Automation, India, 2015.

[23] Yinhui Wang, Teng Wang, Pan Zhou and Xinan Wang, Design and Implementation of a Flexible DMA Controller in Video Codec System, International Conference on Digital Signal Processing (DSP), China, 2014.

[24] Dinesh Padole and Rahul Hiware, Configuration Memory Based Dynamic Coarse Grained Reconfigurable Multicore

Architecture, IEEE Region 10 Conference TENCON, China, 2013.

[25] Vinicius Montenegro Silva, Ricardo S.Ferreira and Alisson Garcia, Mesh Mapping Exploration for Coarse-Grained Reconfigurable Array Architectures, IEEE International Conference on Reconfigurable Computing and FPGA's, Mexico, 2006.

[26] Yalavarthi Ramakrishna Paramahamsa, An Efficient Implementation of Power and Area Optimized On-Chip Network Coarse-Grained Processor, International Journal of Engineering Research and Science & Technology, Vol. 4, No. 2, 2015.

[27] V.Veera Prathap, N.Nagaraja and M.Z.Kurian, Network on Chip Design and Implementation on FPGA with Advanced Hardware and Networking Functionalities, International Conference on Computing, Communications and Networking Technologies, India, 2013.

[28] Suyog V.Pande and Prashant D.Bhirange, An Efficient High Speed RISC Processor for Convolution, IEEE International Conference on Intelligent Systems and Control, India, 2015.

[29] B.Rajesh Kumar, Ravisaketh and Santha Kumar, Implimentation of a 16-bit RISC Processor for Convolution Application, Advance in Electronic and Electric Engineering, Vol. 4, No. 5, 2014, pp. 441-446.

[30] Sangeetha Palwkar and Nitin Narkhede, 32-Bit RISC Processor with Floating Point Unit for DSP Applications, IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology, India, 2016.

[31] Neenu Joseph, S.Sabarinath and K.Sankarapandiammal, FPGA Based Implementation of High Performance Architectural Level Low Power 32-bit RISC Core, International Conference on Advances in Recent Technologies in Communication and Computing, India, 2009, pp. 53-57.

[32] Byreddy Swetha and Fazal Noor Basha, A Low Power Controlling Processor Implementing in SOC, International Journal of Engineering Trends and Technology, Vol. 4, No. 7, 2013.

[33] Liakot Ali, Roslina Sidek, Ishak Aris, Alauddin Mohd.Ali and Bambang Sunaryo Suparjo, Design of a Micro-UART for SoC Application, Computers & Electrical Engineering Vol. 30, No. 4, 2004, pp. 257-268, https://dx.doi.org/10.1016/j.compeleceng.2003.01.002.

[34] S.P.Singh, S.Bhoj, D.Balasubramanian, T.Nagda, D.Bhatia and P.Balsara, Network interface for NoC based architectures, International Journal of Electronics, Vol. 94, No. 5, 2007, 531–547, https://dx.doi.org/10.1080/00207210701306462.

[35] Surendra Bajia, Power and Delay Optimization of Customized 16-Bit Low Power RISC Processor Using VHDL, International Journal of Latest Technology in Engineering, Management & Applied Science, Vol. 2, No. 10, 2013, pp. 53-60.

[36] Vinayak Pai, Swapnil S.Lotlikar and Deepthi Dasari, NoC Based Interconnection of Open RISC Processors, pp. 1-8.

[37] Konstantin Berestizshevsky, Guy Even Yaniv Fais and Jonatan Ostrometzky, Software Defined Network on a Chip, Microprocessors and Microsystems, Vol. 50, 2017, pp. 138-153, https://dx.doi.org/10.1016/j.micpro.2017.03.005.

APPENDIX

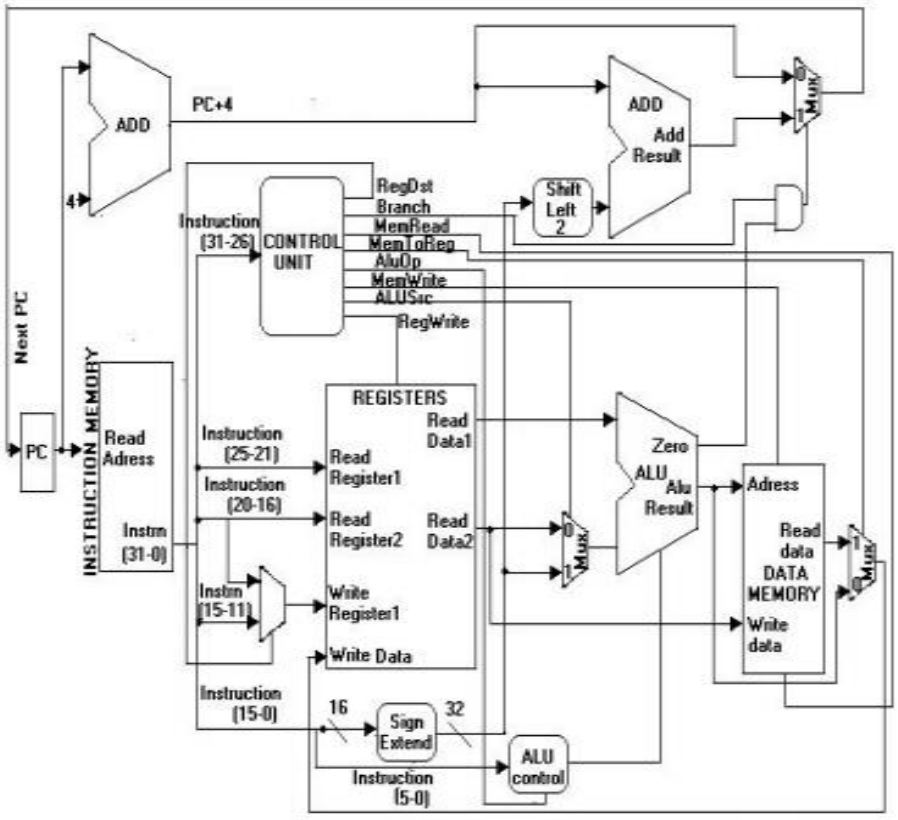

Figure A1.RISC processor architecture