Fuzzy Logic Baised Task Out-Of-Order Execution In Mpsocs

Sreejith sp, PG student,
Department of ECE,
Anna university Regional office
, Madurai.

Arunprasath.R,
Assistant Professor,
Department of ECE ,
Anna university Regional office,
Madurai.

Dr. Ganesh Kumar.P,
Professor, Department of IT,
K.L.N college of Engineering,
pottapalayam, Sivagangi.

***Abstract*—Most of embedded system consisted of MPSoC. The complexity of today's Multi-Processors System-on-Chip (MPSoC) requires new design methodologies to solve time-to-market and design cost problems. So many subsystems are connected together to form SoCs, it is notice that lots of design time is wasted on solving the inter-subsystem (global) communication problem. In this paper propose a novel communication exploration method based on different abstraction levels .In that levels several subsystem are available and intercommunication structure can be optimized in the design process by using fuzzy logic scheduling. Here design a hierarchical middleware with an automatic task level OoO execution engine.**

***Index Terms*—FPGA, fuzzy logic, inter-subsystem, middleware, Task out of order,**

I.INTRODUCTION

The multiprocessor System-on-Chip (MPSoC) is a system-on-a-chip (SoC) in which several processors are used. Most of the embedded applications consists of MPSoCs. It is used by platforms that contain multiple, usually heterogeneous, processing elements with specific functionalities reflecting the need of the expected application domain, a memory hierarchy (often using scratchpad RAM and DMA) and I/O components. All these components are linked to each other by an on-chip inter connect. These architectures meet the performance needs of multimedia applications, telecommunication architectures, network security and other application domains while limiting the power consumption through the use of specialized processing elements and architecture.

The MPSoC architectures have significant limitations with respect to safety-critical applications. These limitations include difficulties in the certification process due to the high complexity of MPSoCs These limitations is over [10] come by different criticality levels have to be integrated on the same computational platform. Due to the adaptability and efficiency of multi core platforms they are used in the embedded application. A new level parallelism for sequential application presents in [13] , in heterogeneous multi core platform integer linear programming (ILP) based parallelization approach is used .

For efficient map application to devices and high performance processing platform by developers through run time mapping. this allowed the system to do scheduling technique techniques , from that automatic task level assignment for each processer [12] .[12] produces a light weight run time software frame work for reconfigurable shared memory MPSoCs .for that here using two slave cores

. A typical MPSoC architecture is consists of several RISC processors connected by an interconnect network. Each processor is denoted as a “node processor” or simply referred as a “node”. Each node has its own CPU/FPU and cache hierarchy (one level or two levels of cache). A read miss in L1 cache will create a L2 cache access, and a miss in L2 cache will then need a memory access. Both L1 and L2 may use write-through or write-back for cache updates. The MPSoC uses shared memories, the memories are associated with each node, but they can be physically placed into one big memory block. The memories are globally addressed and accessible by the memory directory. When there is a miss in L2 cache, L2 cache will send a request packet across the network asking for memory access. The memory with the requested address will return a reply packet containing the data to the requesting node. the Path Dispatcher: the packet context (i.e., flags and header fields), which is generated by the Pre-Processor, is stored in the CTX Memory, a triple port memory with one write and two read ports. At the center of the design, the Tree Classifier is located. It computes the Boolean variables required by the current tree node in two ALUs. While most of the past research in the field of Multiprocessor Systems-on-Chip (MPSoC) has been dedicated to increasing the available processing power on a chip, less has been dedicated to analyze their system-level performance, or to predict their behavior. This paper introduces PAM-SoC, a performance predictor for MPSoCs system-level performance, based on adapting Pamela, a performance prediction tool for parallel applications, In section 1 describes introduction also it gives an idea on MPSoCs In section 2 describes presents the problem statements in that described about existing system . in section 3 describes the proposed work of existing system. In section 4 deal with the results and last section consisted of concussion and future work

II.EXISTING SYSTEM

The motivation of out of order (OoO) task execution is illustrated[1] , by using how tasks are takes are running on the MPSoCs hardware architecture in parallel. In this paper token based description of data flow execution model is extended to general heterogeneous multicore computing scenario Generally, dataflow execution model handles inter-task dependences using tokens to represent the functional I/O parameters. Based on this token based technique, we make two essential enhancements . First token are mapped with parameter to match abstraction for functional source and destination . second function is arranged with multiple read token and a single write token for production and consumption of parameters. For data flow execution Consider a task to be executed. The task is requests read (write) tokens in the task read (write) set, and it is ready for execution only after it has acquired all its tokens needed. Similarly, once on completion, it releases the tokens which are issued to the waiting tasks accordingly. When a pending task has acquired its requisite tokens, it can be offloaded

III.PROPOSED WORK

In MPSoCs design interconnection network are available, it plays an important role in the memory share or shared memory. Packed data flow is usually used to transport between the networks, this will affected the MPSoC performance and its power consumption. In on chip communication power models patatization are introduced . so that due to this impact over all power consumption and MPSoCs performance. we propose a quantitative analysis method to evaluate the relationship between different design options (cache, memory, packetization scheme, etc.) at the architectural level. From the benchmark experiments, we show that optimal performance and power tradeoff can be achieved by the selection of appropriate packet sizes. Shared memory multi-processor systems-on-chips (MPSoCs) have been widely used in today’s high performance embedded systems, such as network processors and parallel media processors (PMP). They combine the advantages of data processing parallelism of multi-processors (MP) and the high level integration of systems-on-chip (SoC). The MPSoC performance is not only determined by the capacity of the node processors (e.g. CPU speed, cache size, etc.), but it is also limited by the interconnect network that connects the processors and memories. Design and optimization of such interconnect network are critical for MPSoC performance. MPSoC uses shared memories to exchange data between processors, and the exchanged data are transported from one processor to the other through the interconnect network. Dataflows are first packetized and then routed to their destinations. Packetized on-chip communication has many advantages over the ad-hoc wire routing in ASICs . The signal integrity and interference are more controllable, multiple high-bandwidth parallel dataflows can be supported concurrently, and the systems are more modularized for IP reuse. Performance and power consumption are the two most critical issues in MPSoC interconnect network design.. Particularly, on-chip network traffic is coming from the following sources:

1. *Cache and memory transactions*. Every cache miss needs to fetch data from the shared memories, and consequently creates traffic on the interconnect.
2. *Cache coherence operations*. In MPSoC, one data may have multiple copies in the caches of different node processors. When the data in memory is updated, its cache copies also need to be updated or invalidated. This synchronization operation will create traffic on the interconnect as well.
3. *Packet segmentation overheads.* When dataflows are segmented into packets, traffic on the interconnect will carry additional overhead. The overhead is dependent on the packet size and header/tail size.
4. *Contentions between packets*. When there is contention between packets on the interconnect, the packets need to be re-routed to another data path or buffered temporarily. This effect will again change the traffic pattern on interconnect. The above factors are not independent. Instead, the performance and power trade-off is determined by the interactions of all factors dynamically, and the variation of one factor will impact other factors.

A)Fuzzy logic;

Fuzzy logic is a form of many-valued logic. Which deals with reasoning that is approximate rather than fixed and exact. Compared to traditional binary sets in which variables may take on true or false values, but fuzzy logic variables may have a truth value that ranges in between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, but in the case of traditional binary sets the truth value may range between completely true and completely false. Further more, we use linguistic variables.

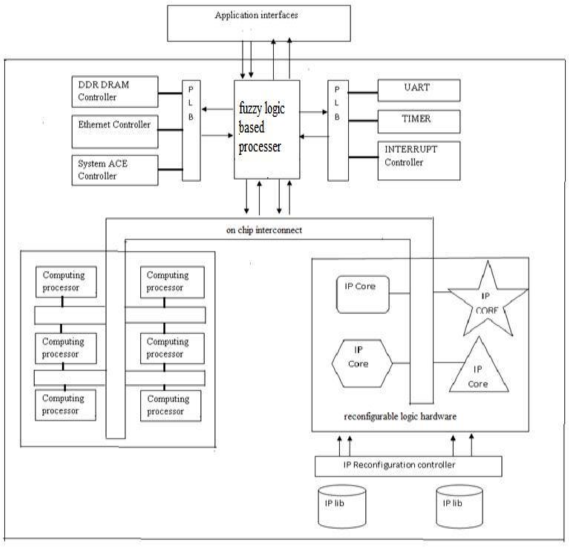
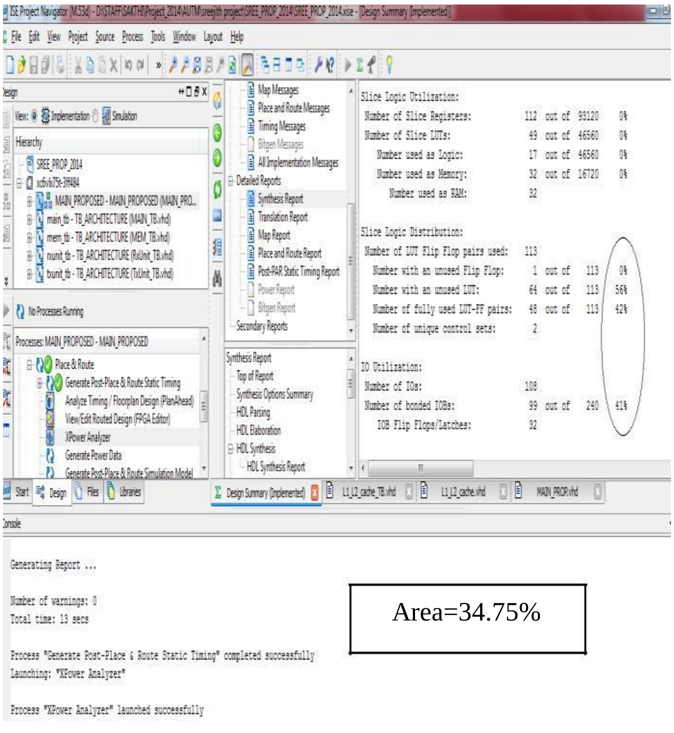


Figure 1 MPSoC hardware platform

IV RESULTS AND DISCUSSION

4.1PROPSED SYSTEM

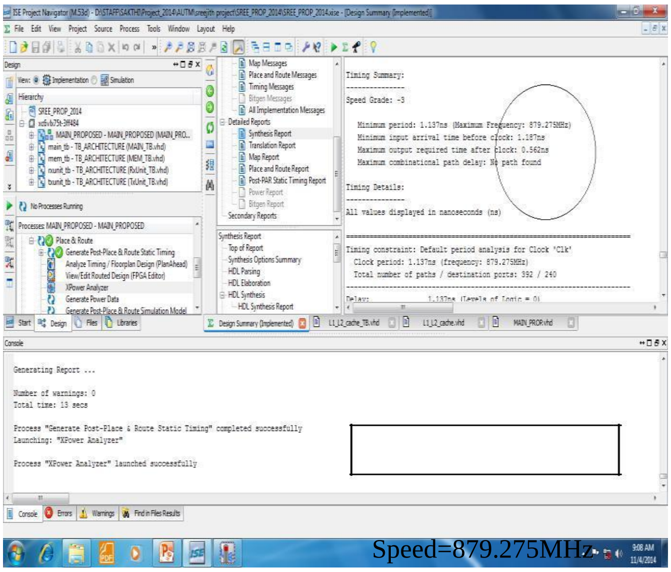

ISE Project Navigator (M.53d) - D:\STAFF\SAKTHI\Project_2014\AUTM\sreejith project\SREE_PROP_2014\SREE_PROP_2014.xise - [Design Summary (Implemented)]
File Edit View Project Source Process Tools Window Layout Help
Design Summary (Implemented)
L1_L2_cache_TB.vhd
L1_L2_cache.vhd
MAIN_PROP.vhd
Console

```
Slice Logic Utilization:
 Number of Slice Registers:              112  out of  93120    0%
 Number of Slice LUTs:                    49  out of  46560    0%
   Number used as Logic:                  17  out of  46560    0%
   Number used as Memory:                 32  out of  16720    0%
      Number used as RAM:                 32

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:     113
  Number with an unused Flip Flop:         1  out of    113    0%
  Number with an unused LUT:              64  out of    113   56%
  Number of fully used LUT-FF pairs:      48  out of    113   42%
  Number of unique control sets:           2

IO Utilization:
 Number of IOs:                          108
 Number of bonded IOBs:                   99  out of    240   41%
   IOB Flip Flops/Latches:                32
```

```
Generating Report ...

Number of warnings: 0
Total time: 13 secs

Process "Generate Post-Place & Route Static Timing" completed successfully
Launching: "XPower Analyzer"

Process "XPower Analyzer" launched successfully
```

Area=34.75%

Figure 2 area report of proposed system


ISE Project Navigator (M.53d) - D:\STAFF\SAKTHI\Project_2014\AUTM\sreejith project\SREE_PROP_2014\SREE_PROP_2014.xise - [Design Summary (Implemented)]
File Edit View Project Source Process Tools Window Layout Help
Design Summary (Implemented)
L1_L2_cache_TB.vhd
L1_L2_cache.vhd
MAIN_PROP.vhd
Console
Errors
Warnings
Find in Files Results
9:08 AM
11/4/2014

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 1.137ns (Maximum Frequency: 879.275MHz)
   Minimum input arrival time before clock: 1.187ns
   Maximum output required time after clock: 0.562ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'Clk'
  Clock period: 1.137ns (frequency: 879.275MHz)
  Total number of paths / destination ports: 392 / 240
-------------------------------------------------------------------------
Delay:               1.137ns (Levels of Logic = 0)
```

```
Generating Report ...

Number of warnings: 0
Total time: 13 secs

Process "Generate Post-Place & Route Static Timing" completed successfully
Launching: "XPower Analyzer"

Process "XPower Analyzer" launched successfully
```

Speed=879.275MHz

Figure 3 speed report of proposed system The above figure 2shows the area of the proposed systen in which it consisted of no of LUTs , no of FFs ,no LUT-FF pairs and no of bounded IOBs. The over all area concumed in vetex 6 is 34.75% . The figure 3 shows the speed of the system it is MHz .

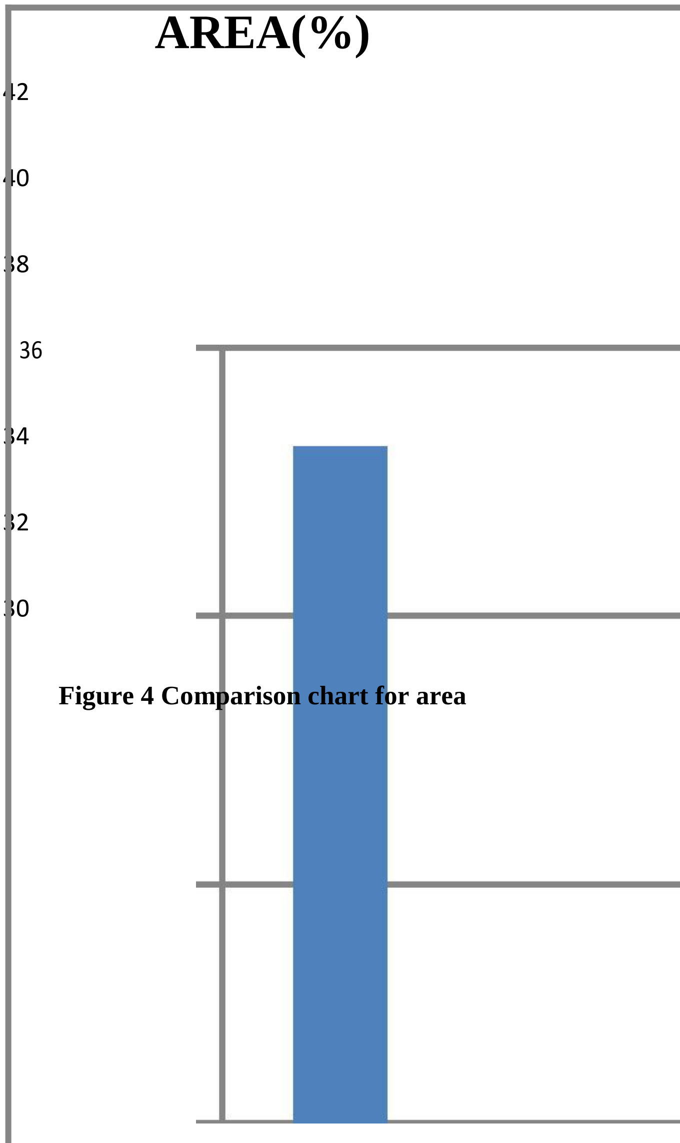
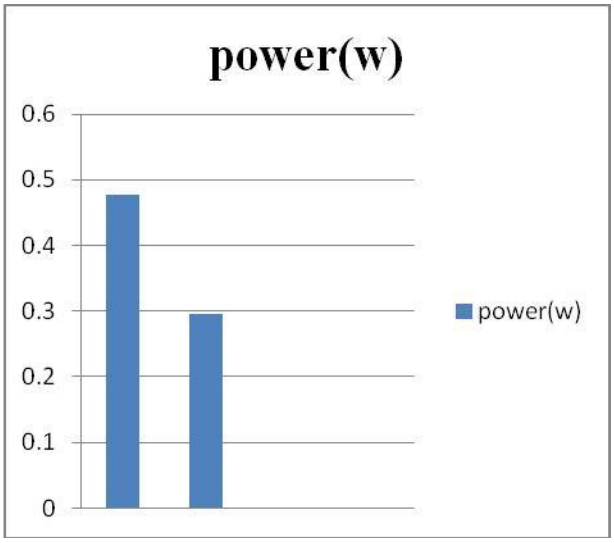


Figure 4 Comparison chart for area

The above figure 4 shows the comparison chat. In that chart y axis shows area and x axis shows no of system In which 1st bar shows the base paper work and 2nd bar shows the proposed work . we can see that area consumed in proposed system is low

Figure 5 Comparison chart for power



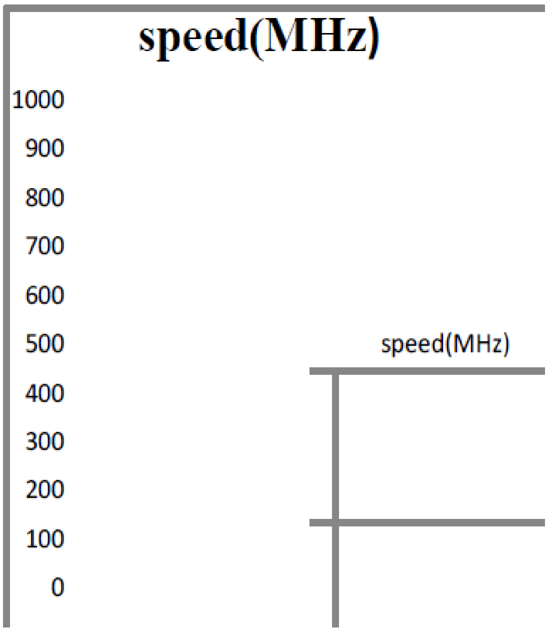


Figure 6 Comparison chart for area

The above figure 5 shows the comparison chat for power. In that chart Y axis shows power and X axis shows no of system In which 1st bar shows the base paper work and 2nd bar shows the proposed work. We can see that power consumed in proposed system is low in vertex 6

The above figure 6 shows the comparison chat for speed . In that chart Y axis shows speed in MHz and X axis shows no of system In which 1st bar shows the base paper speed and 2nd bar shows the proposed speed. From that we can say that speed of proposed system is high .

The below table shows the compassion table between the base paper work with proposed work. Main difference is fuzzy logic is introduced in the proposed system .mainly here we take three parameters for comparison, they are area, speed and power . Existing system and proposed system are done in vertex 6 FPGA. Area is represented in overall percentage of no of LUTs, no of FFs, no of LUT-FF pairs and no of bounded IOBs used. Speed is determine the delay or it is help to fine delay in the device , here speed is in MHz,the power can represented in Watts or mW. here used is Watts.

From the table overall performance of proposed system is comparatively higher than the base paper work that is shown below table. From that we can say that in proposed system different applications is mapped to heterogeneous MPSoCs by using fuzzy logic is more effective.

TABLE 1

PERFOMANCE AND ANALYSIS TABLE FOR BASE SYSTEM AND FUZZY BASED SYSTEM

ITEAM	AREA	SPEED	POWER CONSUMPTI ON
BASE SYSTEM	41.25%	273.973MHz	476.32mW
PROPOSED SYSTEM	34.75%	879.275MHz	294.76mW

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an architecture support for OoO task execution with middleware on FPGA basedMPSoCs. . In this project a fuzzy based logic to access the MPSoC which is present in the middleware Compared to base paper the proposed system achieved some goals. The goal are increased the speed of operation also reduce the area and power consumed.Using of fuzzy logic the Speed is incremented by 220.934%

by 38.134% compare to base work and area is reduced by 15.7575% Several optimization goals and metrics for constructing a decision tree have been introduced and the effects on the overall result were discussed. Heterogeneous but relatively static parts of such a rule base are dealt with in the tree structure, while homogeneous and quickly changing parts of the rule base are evaluated by a table lookup or other specialized classifiers. Isomorphic sub-trees are merged into a single instance transforming the decision tree into a DAG—to save memory.

Next level of enhancement is FPGA based implementation of the fuzzy based system. and using of manhattans packing strategies for accelerating run-time task mapping in NoC-based heterogeneous MPSoCs.

REFERENCES

[1] Chao Wang, Xi Li, Junneng Zhang, Peng Chen, Yunj Chen, Xuehai Zhou, RAY C.C. Cheung” Architecture Support for Task Out-of-order Execution in MPSoCs” IEEE transactions on computers January 2014.

[2] Scott Hauck , Thomas W. Fry, Matthew M. Hosler, and Jeffrey P. Kao “The Chimaera Reconfigurable Functional Unit” *IEEE TRANSACTIONS* on very large scale integration (VLSI) systems, Vol 12, No. 2, February 2004.

[3Georgi Kuzmanov, Georgi Gaydadjiev, Stamatis Vassiliadi“The MOLEN Processor Prototype” *IEEE conference of embedded computer system.,* No. 296-299 , April 2004.

[4] Cesare Ferri, Tali Moreshet, R.Iris Bahar, Luca Benini, and Maurice Herlihy “A Hardware/Software Framework for supporting Transactional Memory in a MPSoC Environment ” *IEEE conference* No. 39-48 , Oct. 2007

[5] Amit Kumar Singh, Wu Jigang,Alok Prakash Thambipillai Srikanthan“ Mapping Algorithms for NoC-based heterog-eneous MPSoC Platforms” *12th Euromicro Conference on Digital System Design / Architectures Methods and Tools* 2009

[6] N. Ventroux, T. Sassolas, R. David, G. Blanc, A. Guerre and C. Bechara “Sesam extension for fast mpsoc architectural exploration and dynamic streaming applications”*IEEE conference* No.341-346 Sept. 2010

[7] Daniel Y. Deng, Daniel Lo, Greg Malysa, Skyler Schneider, and G. Edward Suh “Flexible and Efficient Instruction-Grained Run-Time Monitoring Using On-Chip Reconfigurable Fabric” *43rd Annual IEEE/ACM International Symposium on Microarchitecture 2010.*

[8] Yoav Etsion, Felipe Cabarcas, Alejandro Rico, Alex Ramirez, Rosa M Badia “Task Superscalar: An Out-of- Order Task Pipeline” *43rd Annual IEEE/ACM International Symposium on Microarchitecture 2010*

[9]Aparna Kotha, Kapil Anand, Matthew Smithson, Greeshma Yellareddy and Rajeev Barua “Automatic Parallelization in a Binary Rewriter” *43rd Annual IEEE/ACM International Symposium on Microarchitecture* 2010

[10]Christian El Salloum, Martin Elshuber, Oliver Höftberger, Haris Isakovic, Armin Wasicek “The ACROSS MPSoC – A New Generation of Multi-Core Processors designed for Safety-Critical Embedded Systems” *IEEE conference on didital system design (DSD).* Sept. 2012

[11]Fernando G. Moraes, Guilherme A. Madalozzo, Guilherme M. Castilhos, Everton A. Carara “Proposal and Evaluation of a Task Migration Protocol for NoC-based MPSoCs” *IEEE conference on circuits and systems,* may 2012

[12]Dimitris Theodoropoulos , Polyvios Pratikakis ,DionisiosPnevmatikatos “Efficient runtime support for embedded MPSoCs” *IEEE conference on embedded computer system.,*No. 164-171 July 2013

[13]Christian El Salloum, Martin Elshuber, Oliver Höftberger, Haris Isakovic, Armin Wasicek “Automatic Extraction of Task-Level Parallelism for Heterogeneous MPSoCs” *42nd International Conference on Parallel Processing,* 2013

[14].Gupta, G. and G.S. Sohi. “Dataflow execution of sequential imperative programs on multicore architectures. in Proceedings” *ofthe 44th Annual IEEE/ACM International Symposium on Microarchitecture.Porto Alegre, Brazil:* ACM.2011

[15].Hayashi, A., Y. Wada, T. Watanabe, et al. “Parallelizing compilerframework and API for power reduction and software productivity of real-time heterogeneous multicores.” *in Proceedings of the 23rd international conference on Languages and compilers for parallel computing. Houston, TX: Springer-Verlag.*2010

[16].J.A.Kahle, M.N.Day, H.P.Hofstee, et al., “Introduction to the Cell multiprocessor”. *IBM Journal of Research and Development, 2005. 49.*

[17].Taylor, M.B., J. Kim, J. Miller, et al., “The Raw microprocessor: a computational fabric for software circuits and general-purpose programs.” *Micro, IEEE, 2002. 22(2): p. 25-35.*

[18].Hammond, L., B.A. Hubbert, M. Siu, et al., “The Stanford Hydra CMP”. *IEEE Micro, 2000. 20: p. 71-84.*

[19].Swanson, S., K. Michelson, A. Schwerin, et al. WaveScalar. In Intl. Symp. on Microarch. 2003.

[20].Sankaralingam, K., R. Nagarajan, R. McDonald, et al. “Distributed microarchitectural protocols in the TRIPS prototype processor”. *in Intl. Symp. on Microarch. 2006.*