International Journal For Technological Research In Engineering
Volume 3, Issue 8, April-2016 *ISSN (Online): 2347 - 4718*

IMPLEMENTATION OF GENETIC ALGORITHM FOR PERFORMANCE IMPROVEMENT IN MPSoC

Nimmi Prakash[1], Mrs. Jenitha. A[2]
[2]Associate Professor, Dept. of Electronics and Communication
East Point College of Engineering and Technology, Bangalore, India

ABSTRACT: Multicore designs are the viable replacement to single core designs, but still the new architecture has some sort of problems associated with it. The main problem is lack of necessary tools and techniques to take the complete advantage of multiple processors and in turn it adds burden to software and tool developers to devise new techniques that produce powerful architecture. To overcome the above problems, one of the must needed technique is to schedule tasks on the system processing it along with memory management. In this paper we introduced an algorithm namely genetic algorithm. Genetic algorithm finds the best suited solution by performing three operations: mutation, crossover and fitness function to schedule the different tasks that get executed by multiple processor with proper utilization of memory. It reduces the delay (clock cycles) to produce the output and results in increased throughput. The Xilinx programming is utilized for the reproduction of this framework. This paper presents a method for the design and implementation of powerful architecture with high throughput using genetic algorithm on Field Programmable Gate Array (FPGA). The reason for the hardware implementation on FPGA is that it provides faster prototyping and can be reprogrammed quickly. Also they are inexpensive and easily testable. This feature enables the faster implementation and verification of the new design.

Key terms: memory management, task scheduling, multiprocessor (MPSoC), genetic algorithm, mutation, chromosome, crossover, fitness function.

I. INTRODUCTION

In the era of high performance computers to personal computers and to embedded systems, the use of multiple processors is infiltrating system architecture. The earlier method of increasing clock speed of single core is not a viable solution due the power requirements. Hence, computer architects deployed multiple processing as a great source to increase clock speed. In embedded system, if multiple processor is embedding on a single chip it is sometimes referred as multiprocessors system on chip(MPSoC). An MPSoC has few advantages such as it has required power to run complex embedded applications with less power consumption. It usually consists of heterogeneous processing elements, memory along with complex communication platforms (network-on-chip and set of input/output). Multiple processor designs have the high performance potential but the challenge is to find the way that can extract this potential. If embedded applications are scheduled properly on MPSoC we can optimize the utilization of system resources. System resources means processing elements and on-chip memory. an embedded applications can be divided into multiple tasks. A task can exhibit different times depending upon the processor executing it and the memory assigned to it. Access to data elements by a task in off-chip memory is slow with respect to on-chip memory. Due to this reason memory partitioning and task scheduling must be handled in integrated manner.so, an effective technique to handle this is task scheduling with memory management based on genetic algorithm.

ANATOMY OF MPSOC:

MPSoC is a system-on-chip with multiple instruction-set processors(CPU). These are typical heterogeneous multiprocessor, which consists of:

- different types of PEs.
- memory which is heterogeneously distributed over the entire machine.
- interconnection between Pes and memory and require large memory.

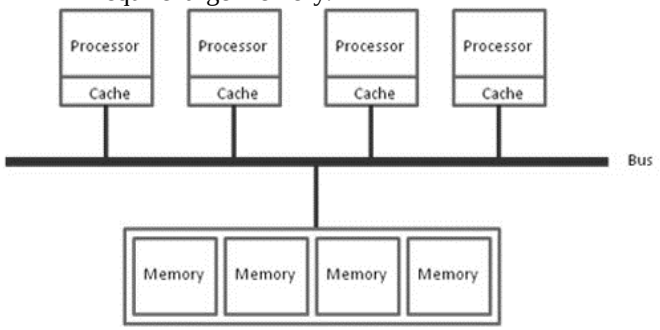

Fig 1.1: generic shared memory multiprocessor

In the above figure shared memory multiprocessor is shown with pool of processors and memory that are connected by an interconnection network. They are regularly structured and mostly preferred because it is easy to program. Multiple processor system balance specialization and programmability. Programmable processors can allow SoC to be programmed after it is being fabricated. MPSoC is sometimes called as platforms because many implementation of a given type system is allowed in it. the advantages of programmability are:

- reusable chip.
- reduced product cost.
- Compartmentalized design tasks.
- Longer shelf life than specialized SoC.
- MPSoC systems uses software-controlled memory

which is known as scratchpad memories(SPM).it allows to predict execution time with accuracy. SPM consists of data array along with memory decoder.

II. LITERATURE SURVEY

The main objective is to reduce the execution cycle time and conservation of power. The literature review is being provided based on the various surveys taken for task scheduling algorithm and partitioning methods in order to satisfy our main objective.

A. Integrated SPM Optimization and Task Scheduling

V.Suhendra, C.Raghavan, and T.Mitra proposed an ILP model which formulates the interaction between scheduling and partitioning of the system. ILP is an integration of scheduling, partitioning and allocation of tasks to the available processors of the system. In this the tasks are the applications taken and they are scheduled first. Then they are pipelined for simultaneous processing.

B. Compiler Based Strategy

M.Kandemir, J.Ramanujam, and A.Choudhury presentedthe compiler based technique. It is used to enhance the data accessing method in the multiprocessor system in order to increase the utility of the resources and reduce the energy consumption. In this the data are accessed by the use of arrays in a parallel manner. In focuses on two major issues namely:

- Extra accessing request to off-chip memory.
- Reducing energy delay.

This compiler based technique is being incorporated with the help of arrays and software oriented memories residing internal to the system in order to resolve the mentioned problems.

C. The Partitioning Strategy

The cross-interference occurring in cache as issue can be solved with the help of this partitioning strategy. The features related to the program code are considered and based on that they are partitioned. The features of this strategy are as follows:

- Variables and constants.
- Array size
- Cycle time.
- Computational cost.
- Loop conflicts.

The algorithm uses above features to map the data to SPM budget for the accomplishment of the task by the processors.

D. Variable Partitioning and Scheduling

Lei Zhang, Meikang Qiu, Wei-Che Tseng focuses on two major problems which are inter-dependent on each other. The issues are partitioning and scheduling which are resolved by the help of decoupled approach. The decoupled approach uses the following two methods to solve the above problems.

- High Access Frequency First (HAFF) variable partitioning.
- Global View Prediction (GVP) variable partitioning.

III. IMPLEMENTATION

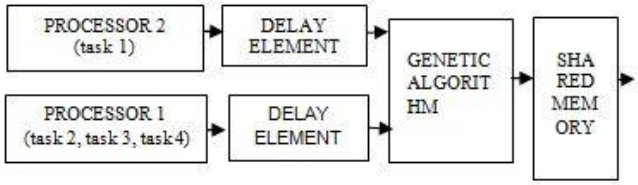

Fig 3.1 Block diagram of proposed design

Processor 2: in processor 2 we have created a task 1 that perform radix 2 FFT.

Processor 1:in processor 1 we have created three tasks that perform sipo, ripple carry adder and booth multiplier.

Delay Element: it calculates the number of clock cycles required by both the tasks in both the processor to get executed completely.

Genetic algorithm: it performs a mathematical calculation to determine the best suited solution by performing mutation, crossover and fitness function to schedule the task with proper utilization of memory. It reduces the delay in producing the output and results in high throughput. For our design the best suited fitness function is 2 clock cycles.

Shared Memory: once the task is completed the result is stored in shared memory in FIFO manner.

IV. RESULT

Xilinx results:

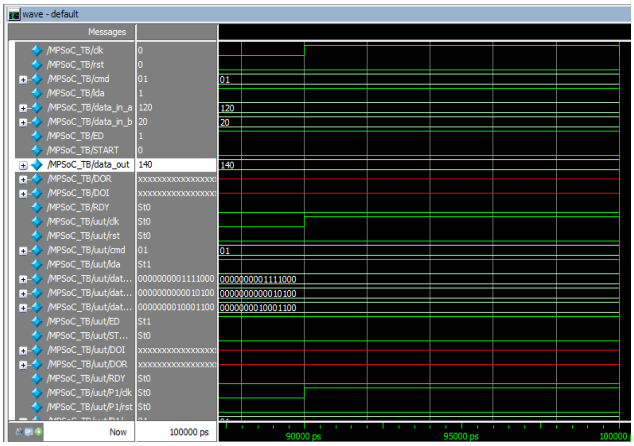

Fig 4.1 simulation result of MPSoC

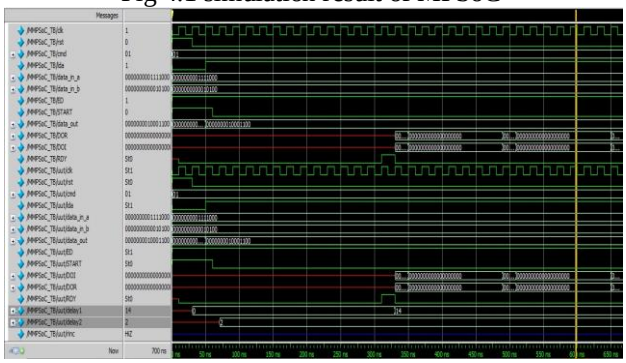

Fig 4.2 simulation result of modified MPSOC

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	923	960	96%
Number of Slice Flip Flops	981	1920	51%
Number of 4 input LUTs	1369	1920	71%
Number of bonded IOBs	93	66	140%
Number of MULT18X18SIOs	4	4	100%
Number of GCLKs	1	24	4%

Fig 4.3 device utilization memory for MPSoC

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	923	960	96%
Number of Slice Flip Flops	981	1920	51%
Number of 4 input LUTs	1369	1920	71%
Number of bonded IOBs	93	66	140%
Number of MULT18X18SIOs	4	4	100%
Number of GCLKs	1	24	4%

Fig 4.4 device utilization memory for modified MPSoC

V. CONCLUSION

A genetic algorithm method is used to find out the best solution to schedule the task with proper utilization of memory. this implementation leads to reduction in delay to produce the output and results in high throughput. And in our design we have done a comparison study between normal MPSoC and modified MPSoC, even though the normal MPSoC covers less area, the delay is more which is approximately 14 clock cycles to get output whereas modified MPSoC require 2 clock cycles.

REFERENCES

[1] Hassan salamy, semih aslan,'A genetic algorithm based approach to pipelined memory-aware scheduling on an MPSoC,'in proc.IEEE2015.(base paper).

[2] Poorani.A, anuradha.B, Dr.C.vivekanadhan,"An Effectual Elucidation of Task Scheduling and Memory Partitioning for MPSoC,"in proc. International conference on intelligent systems and control(ISCO),2014.

[3] L. Benini, D. Bertozzi, A. Guerri, and M. Milano, "Allocation and scheduling for MPSoC via decomposition and no-good generation," in Proc. International Joint conférences on Artificiel Intelligence (IJCAI),2005.

[4] Suhendra, C. Raghavan, and T. Mitra, "Integrated scratchpad memory optimization and task scheduling for MPSoC architecture," in proc. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), 2006.

[5] Anuradha.B, hemalatha M, vivekanadhan C," task allocation and memory partitioning for MPSOC in embedded systems, "in proc. international journal of engineering science and innovation technology(IJESIT),2013.

[6] Kanoun, N. Mastronarde, D. Atienza, and M. V. D. Schaar, "Online energy-efficient task-graph scheduling for multicore platforms," IEEE Transactions on Computer Aided Design, vol. 33, no. 8, 2014.

[7] P.-H. Tseng, P.-C. Hsiu, C.-C. Pan, and T.-W. Kuo, "User-centric energy-efficient scheduling on multi-core mobile devices," in Design Automation Conference

