ISSN: 2229-6956(ONLINE)
DOI: 10.21917/ijsc.2012.0051

PERFORMANCE EVALUATION OF OR1200 PROCESSOR WITH EVOLUTIONARY PARALLEL HPRC USING GEP

R. Maheswari [1] and V. Pattabiraman [2]
School of Computing Science and Engineering, Vellore Institute of Technology, Chennai, India
E-mail: [1]maheswari.r@vit.ac.in and [2]pattabiraman.v@vit.ac.in

Abstract

In this fast computing era, most of the embedded system requires more computing power to complete the complex function/ task at the lesser amount of time. One way to achieve this is by boosting up the processor performance which allows processor core to run faster. This paper presents a novel technique of increasing the performance by parallel HPRC (High Performance Reconfigurable Computing) in the CPU/DSP (Digital Signal Processor) unit of OR1200 (Open Reduced Instruction Set Computer (RISC) 1200) using Gene Expression Programming (GEP) an evolutionary programming model. OR1200 is a soft-core RISC processor of the Intellectual Property cores that can efficiently run any modern operating system. In the manufacturing process of OR1200 a parallel HPRC is placed internally in the Integer Execution Pipeline unit of the CPU/DSP core to increase the performance. The GEP Parallel HPRC is activated /deactivated by triggering the signals i) HPRC_Gene_Start ii) HPRC_Gene_End. A Verilog HDL(Hardware Description language) functional code for Gene Expression Programming parallel HPRC is developed and synthesised using XILINX ISE in the former part of the work and a CoreMark processor core benchmark is used to test the performance of the OR1200 soft core in the later part of the work. The result of the implementation ensures the overall speed-up increased to 20.59% by GEP based parallel HPRC in the execution unit of OR1200.

Keywords:

GEP, Gene, Crossover, Mutation, CoreMark

1. INTRODUCTION

A soft- core processor is a reconfigurable component which can be synthesized into programmable FPGA (Field Programmable Gate Array) or ASIC (Application Specific Integrated Circuit) using HDL. This soft core processor comes under licensed and open source [1]. OR1200 is an open source soft core 32-bit RISC Harvard micro-architecture with 5 stage pipeline.OR1200 finds its application in most of the embedded area like embedded networking, automobile, internet, telecommunication and wireless application. This OR1200 RISC processor can efficiently run on any modern operating system. In any system, the processor performance can be increased by code optimization, parallel processing, high throughput, short response time, low utilization of resource, fast encoding /decoding data, high bandwidth etc. To build high performance computing system, it may require better understanding of the overall behaviour of the system in target. Huge number of researches focussed on the performance issues targeting understanding, evaluating, measuring and improving system performance. This paper focussed on improving the core performance by internally placing an evolutionary GEP parallel HPRC in the Integer Execution Pipeline unit of the CPU/DSP OR1200 core processor. GEP is an evolutionary devised system for encoding expression for applications which need fast/complex system using mutation and cross-breeding techniques which runs very faster than traditional genetic algorithm [2]. This proposed work has segregated into two part: in the former part, a Verilog HDL functional code for GEP parallel HPRC is developed and synthesised using XILINX ISE and in the later part, a CoreMark processor core benchmark is used to test the performance of the OR1200 soft core.

2. MATERIAL AND METHODS

In this paper a parallelized HPRC is implemented as "Full Generational Reconfigure" GEP as an evolutionary strategy. Parallel HPRC is activated/deactivated by two signals HPRC_Gene_Start and HPRC_Gene_end, controlled by a FDC (D flip-flop with Clock signal). Parallel processing, wherein each gene or a group of gene is hosted by a separate Processing Element (PE), is a feasible method to speed up the runs. In the parallel programming the selection is done at the global population. Then the selected string will subjected to undergo crossover or mutation in parallel. At the beginning of the process, the population of string is initialized with random chromosomes. When a fitness value is assigned to every set of parameters in the generation, the subroutine returns to the main program to perform the genetic operations to reproduce offspring. The process continues for a fixed number of generations, chosen to maximize the success rate of the core processor OR1200.

2.1 OR1200

The block diagram for an embedded parallel HPRC using evolutionary GEP in OR1200 RISC processor Integer Execution pipeline unit is shown in Fig.1. The performance of HRPC unit inside the OR1200 soft core is achieved by dynamic/reconfigurable hardware unit [1].

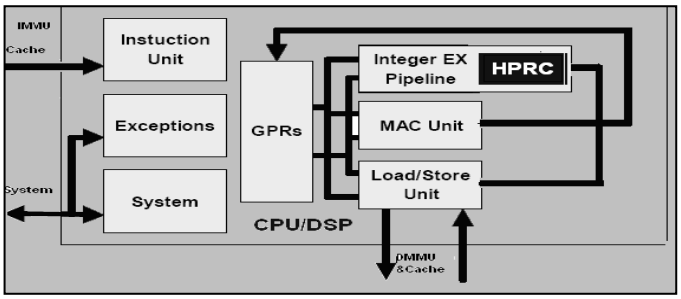

Fig.1. Block diagram with parallel HPRC in Execution unit

HPRC in OR1200 is a collection of interconnected programmable PE. The active PE's are identified from the

HPRC unit and GEP is executed in each active PE. The sample active PE is shown in the Fig.2.

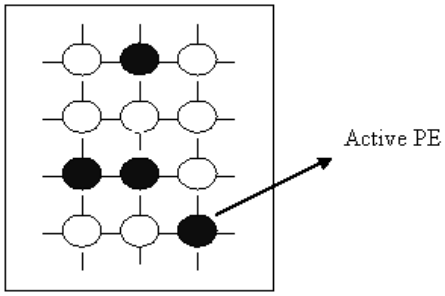

Fig.2. Active Processing Elements (PE) in HPRC (GEP)

2.2 GENE EXPRESSION PROGRAMMING

GEP is an evolution technique of genetic family proposed by C. Ferreira in 2001 [3, 4]. This evolutionary model based GEP adopts the merits of both the traditional genetic concepts of *Genetic Algorithm* (GA) and *Genetic Programming* (GP). GEP due to its large-space search capability, it performs well in global optimization of the system [5]. Parallelism in GEP classified into two ways i) Genetic operation parallelism (Mutation, selection , transposition, Inversion etc) ii) Fitness Evaluation parallelism[6].

2.2.1 Fundamentals of GEP:

The two basic blocks of GEP are *chromosome* (*single* or multi-genic) and *expression tree* [7]. Based on this, the language of GEP is classified into two ways i) language of the genes ii) language of the expression tree. The gene of the GEP contains two sections a *head* and a *tail*. The head (H) of the gene contains functions and terminals (variables and constants) and the tail (T) contains only terminals. In GEP the gene symbols of the chromosomes are encoded into mutate valid expression using *Karva* language called *K-Expression*. *K-Expression* is an efficient way of representing the complex program as simple, extremely compact, symbolic strings in any language [5].

2.3 PROPOSED ALGORITHM

In this implementation at first initial population of chromosome is created for the active PE and *heuristic* function $h(n)$ for the node (gene) is calculated with the $H(n)$ and $T(n)$, the number of symbols in head and tail respectively. Using this heuristic function, the *fitness score* (f_s) of the individual is identified.

Based on this fitness score the entire population is segregated into three types:

i. *ideal* gene (f_s=1)
ii. *viable* gene ($0< f_s <1$)
iii. *unviable* (f_s<0)

According to this fitness value, the individuals are selected by Binary Roulette wheel sampling which return Boolean value 0 or 1 (1 for ideal and viable gene and 0 for unviable gene). After selection, new generation of gene is created by any one of the reproduction genetic operator functions like cross over and mutation. These offspring are put into an intermediate population or also called as *gene pool*. To avoid the disruption of genetic information when performing genetic operators on the parents it is necessary to copy them into an intermediate population [9].

2.3.1 Reproduction in GEP:

Crossover is a genetic operator in which reproduction is achieved by swapping the chromosome between the individuals. In mutation offspring is produced by randomly changing the part of the string chromosome [8][9]. In GEP the mutation may be performed in one of the following various ways like i) Simple Mutation, ii) Mutation by inversion, iii) Mutation by transposing symbol in gene, iv) Mutation by transposing gene in chromosome, v) Mutation by recombine chromosomes. Mutation by chance produces an extra ordinary gene called *intellectual* gene or simply *genius*.

2.3.2 Pseudo Code for GEP Parallel HPRC:

Begin
Identify the active PE
HPRC_Gene_Start
While (PE)

1. Initialize initial population
   a. Encode K-expression
   b. Calculate $H(n)= \sum_{i=0}^{i<n-1} P_i$

      (where, P_i is the chromosomes parameter for analysis)
   c. Calculate $T(n) = H(n) + (MaxA(n) - 1) + 1$

      where, $MaxA(n)$ is the maximum number of the argument
2. Convert the K-Expression into expression tree and calculate the heuristic function $h(n)$ as,

   $$h(n) = \delta t(p) + H(n) \cup T(n)$$

   where, δ is the rate of change of PE, p is active PE
   $p \in \{1, 2, . ., n\}$,
   n is the number of PE.
3. Calculate the fitness function to produce fitness score,

   $$f_s = \sum_{i=0}^{T} E * m_i * \log_2(1000/i) + h(n)$$

   where, f_s is the Fitness Score, E is the K-Expression of the PE, m is the scaling factor, T total number of K-Expression.
4. Identify the workable fitness node using the f_s

   $$f(n) = \begin{cases} 0 < f_s < 1 & \text{viable fitness} \\ f_s < 0 & \text{unviable fitness} \\ f_s = 1 & \text{ideal fitness} \end{cases}$$

5. Transfer the best fit chromosome into next generation using Binary Roulette-wheel sampling R_{wb}. Depending upon the $f(n)$ it returns binary value for selection.

   $$R_{wb} = \begin{cases} 1 & \text{if } 0 < f_s < 1 \;\&\&\; f_s = 1 \\ 0 & \text{if } f_s < 0 \end{cases}$$

6. Create new population by reproduction within the PE

a. If f_s <0 exist further perform crossover to produce offspring

$$C = Gen(k) \cup Gen(k+1)$$

b. Depending upon the K-Expression and requirement, execute any one of the mutation function to produce offspring.

7. Place the generated offspring in the gene pool
8. Repeat step 2 to 8 for next evolution cycle

End While

HPRC_Gene_End

End

3. THEORY/CALCULATION

The gate level net-list is generated by synthesizing the Verilog HDL code for parallel HPRC with an evolutionary GEP in OR1200.The Fig.3 below shows the RTL (Register-Transfer Level) schematic for parallel HPRC using GEP generated in XILINX ISE for SPARTAN3 family, XC3S200 device configuration. The efficiency of the core (OR1200) in the parallel computation using GEP is calculated using the formula,

$$S = \frac{S_t}{PES_t \times P_t} \quad (1)$$

In Eq.(1) S is the speed-up of the system, S_t is the serial execution time, PES_t is serial execution time of total active PE's, $PES_t \in \{1, 2, \ldots, n\}$, n is the number of active PE and P_t is the parallel execution time. To increase the performance, various genetic operators will be applied to the initial/intermediate population of behaviours. One cycle of testing all of the competing behaviour is defined as a generation and is repeated until a good behaviour is evolved. The good behaviour is then applied to the real problem.

The efficiency of the core OR1200 is evaluated by running the Matrix manipulation of CoreMark processor core bench mark. Speed-up is thus calculated by using the above formula by considering four active PE. It has observed that the values for the variables are S_t = 189.24 ns, PES_t = 756.96 ns and P_t = 82.36 ns.

Thus the result of the implementation ensures the overall increase in speed-up achieved is 20.59% by GEP based parallel HPRC in the execution unit of OR1200.

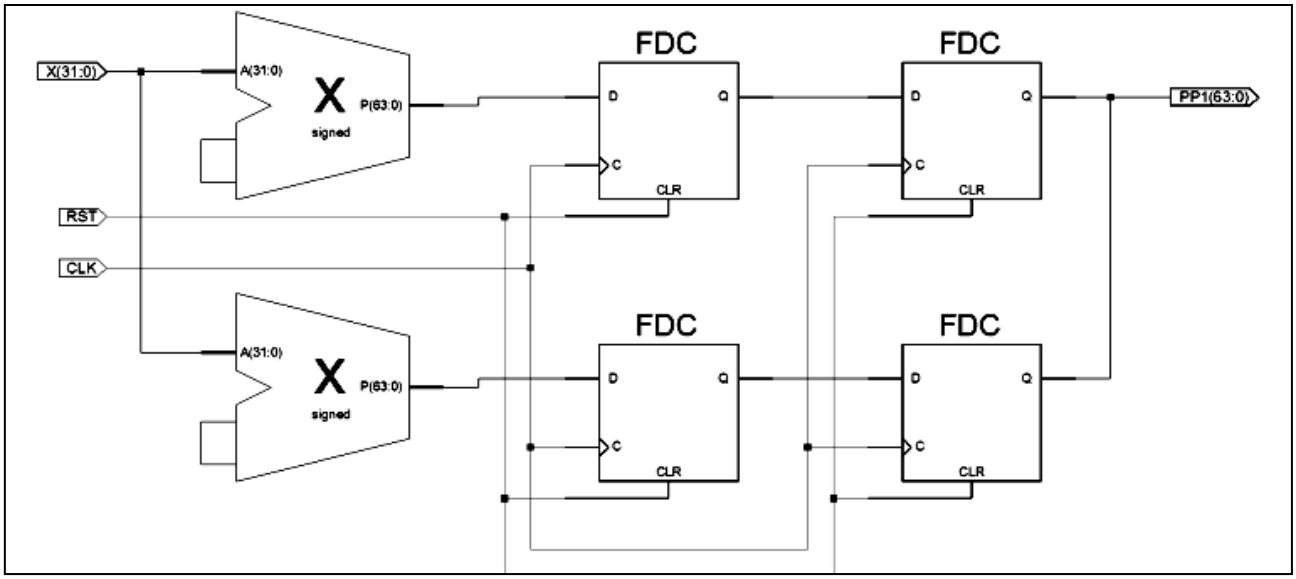

Fig.3. View RTL Schematic for parallel HPRC using GEP in Integer Execution unit

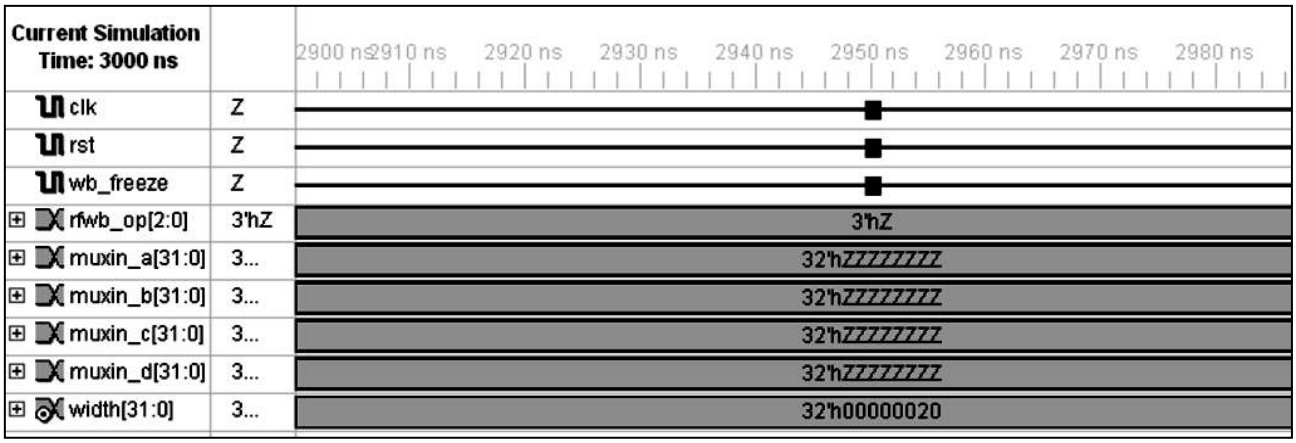

Fig.4. Behavioral Model Simulation in ISim

4. RESULTS

In the later part of the proposed work the developed functional code in HDL for parallel HPRC using GEP is simulated in XILINX ISE to analyse its OR1200 processor core performance.

4.1 BENCHMARK

To test the performance of the parallel HPRC using GEP in OR1200 processor core CoreMark bench mark is used rather than classical Dhrystone benchmark. CoreMark is a embedded generic benchmark developed by EEMBC (Embedded Microprocessor Benchmark Consortium) specially for processor's core features [11].

4.1.1 Output Simulations:

Fig.4 shows the behavioural model simulation of parallel HPRC with GEP tested for Matrix manipulation test bench [10] using CoreMark benchmark in XILINX ISE Simulator (ISim). Matrix manipulation forms the basis of many more complex algorithms used as test vector in many processor testing.

The test bench is executed for the specified time interval (Start Time = 0 ns and End Time = 3000 ns) measured between the marker properties between 697128.1 ps to 2850000.0 ps.

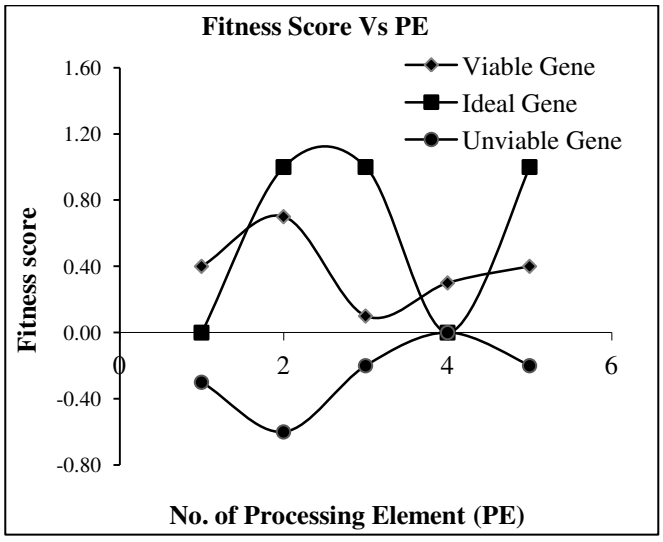

Fig.5. Statistical Fitness Score

The Fig.5 shows the statistical fitness score generated while running the test bench.

4.2 SAMPLE RESULT DATA

The Table.1 shows the sample result data generated while running the GEP based parallel HPRC with Population size of 150 with 1510 runs.

Table.1. Sample Result Data Set

No of Runs	: 1510
No of Generations	: 545
Population size	: 150
Absolute error	: 0.02
Maximum fitness	: 15100

Fitness cases:

terminal a	f(a)
2.81	95.2425
6	1554
7.043	2866.55
8	4680
10	11110
11.38	18386
12	22620
14	41370
15	54240
20	168420

Function and terminal set of Gene:

Head * + - / a b c

Tail a b c

Solution with maximum fitness 15100:

Expression Tree:

```
+
++
***
*---
//*--
*+-b*+a*-
*+c//ba*/**bbaa
*accb//c*/**bca
*+aa//aa*/**aa
aaaabbaaaaaaaccaaaa
bbbbbbbabbaaaccccccc
aaacccccbbbbaaaaaabbbbbbbbaaaaa
```

Success rate: 0.987

5. DISCUSSION

The OR1200 core processor performance was increased by placing parallel HPRC using multigrain parallelism in the Integer Pipeline Unit of CPU/DSP [1]. In this paper the performance of the core processor will be increased with the help of proposed evolutionary parallel HPRC using GEP model by 20.59% and this performance analysis is justified in the below section.

5.1 PERFORMANCE ANALYSIS

The CoreMark Benchmark for Matrix multiplication is executed in the OR1200 core processor. Performance analysis is done with the comparison study of OR1200 by running the processor in three different scenarios such as traditional OR1200 processor, OR1200 with parallel HPRC using multigrain and OR1200 with parallel HPRC using evolutionary GEP. The simulated model for the test bench program is analyzed for the

performance of the core OR1200. The sample matrix functional code in verilog (HDL) is:

```
or1200_gep matrix (
//Matrix Dimension
 .hprc_gene dm,
//Resultant Matrix
 .res_gene *rg,
//Input Matrix
  .in_gene *in,
//Matrix operator
 .in_gene *op);
```

The time required to execute (time complexity) the NXN matrix multiplication in OR1200 without HPRC is $O(N^3)$, when N=10, the processor requires $1.4x10^{-6}$ seconds to complete the execution. While running the matrix code in the processor OR1200 with parallel HPRC using GEP, it requires $1.13x10^{-7}$ seconds to complete the execution. Therefore the overall time complexity for the core OR1200 with HPRC using GEP is reduced to $O(N^2)$.

To review on the processor performance on the analyzer, the two event ratios CPI (Cycle per Instruction) and Parallelization Ratio are focused here. CPI value indicates the number of clock cycles required by the instruction and Parallelization ratio signifies the amount of parallelization used. In a multi-threaded application this Parallelization ratio value should close to numeric one.

Table.2. Event Ratios of OR1200

	Clocks per Instructions	Parallelization Ratio
OR1200	6.920	0.89
OR1200 with HPRC using GEP	3.178	1.00

The optimization analysis graph (Fig.6) for the processor OR1200 with HPRC using GEP is drawn for the event ratio tabulated in Table.2.

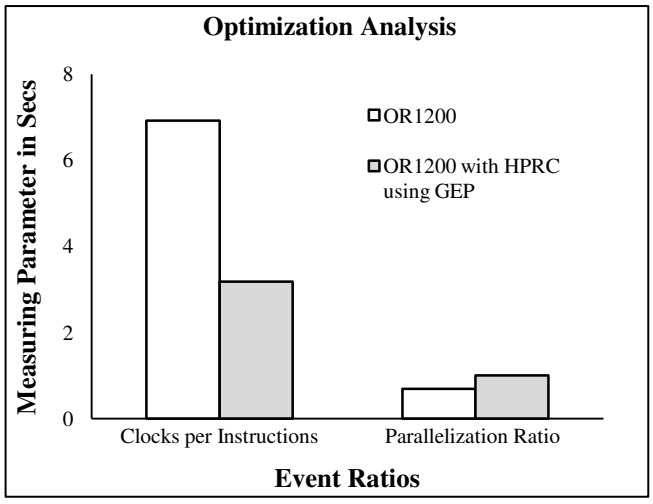

Fig.6. Optimization Analysis of OR1200

The Fig.7 shows the performance analysis graph drawn for the below configuration table Table.3 of OR1200.

Table.3. Configuration details of OR1200

OR1200 (Secs)	Existing (Traditional OR1200)	Existing (with Multigrain)	Proposed (with GEP)
Total ticks	25875	25875	20766
Total time (secs)	25.875	12.875	8.37628
Iterations/Sec	3864.7343	1468.6003	1209.08
Iterations	100000	70990	50345

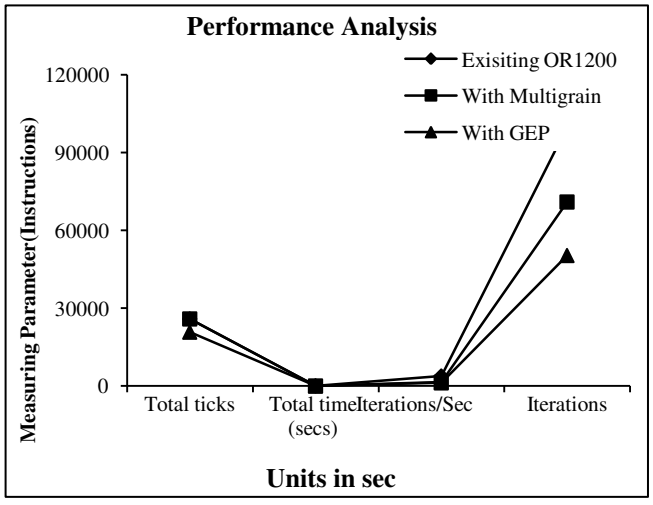

Fig.7. Performance Analysis of OR1200 using CoreMark Benchmark

6. CONCLUSION

A novel technique of parallel HPRC using GEP an evolutionary model is proposed in this paper to increase the performance of OR1200 RISC processor in the embedded system. Using the evolutionary reproduction genetic operator of

the chromosome such as crossover and mutation the offspring are produced in the PE and resulted in the optimized code with parallel thread with increased speed-up.

A Verilog HDL code is developed and implemented using XILINX ISE to synthesize HPRC in Integer execution unit of OR1200 in the former part of the working model. A behavioral model simulation is generated for CoreMark benchmark matrix manipulation and the performance of the reconfigured architecture is identified using ISim in the later part of the working model. Thus the result of the implementation for matrix manipulation ensures the overall speed-up increased to 20.59% by optimizing the execution unit of OR1200.

Using the FPGA gate logic the proposed working model based on the evolutionary GEP algorithm can be synthesized. This derived GEP technique can be incorporated in the various open source soft core IP cores such as Microblaze, PacoBlaze, LatticeMico32 etc to increase the speed-up of the core with very less overhead in hardware.

REFERENCES

[1] Maheswari. R and Pattabiraman. V, "A New Technique Of Embedding Multigrain Parallel HPRC In OR1200 A Soft-Core Processor", *Proceedings of the 11th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, Vol. 25, pp. 92-97, 2012.

[2] Candida Ferreira, "Gene Expression Programming, A New Adaptive Algorithm for Solving Problems", *Transactions on Complex Systems*, Vol. 13, No 2, pp. 87-129, 2001.

[3] Candida Ferreira, "*Gene Expression Programming: Mathematical Modelling by an Artificial Intelligence*", Springer, 2004.

[4] Yu Chen, Chang Jie Tang, Rui Li, Ming Fang Zhu, Chuan Li and Jie Zuo, "Reduced-GEP: Improving Gene Expression Programming by Gene Reduction", *Proceedings of Second International Conference on Intelligent Human-Machine Systems and Cybernetics*, Vol. 2, pp. 176-179, 2010.

[5] Jiang Wu, Taiyong Li Bing Fang, Yue Jiang , Zili Li, and Yangyang Liu, "Parallel Niche Gene Expression Programming Based On General Multi-CoreProcessor", *Proceedings on Artificial Intelligence and Computing Intelligence*, Vol. 3, pp. 75-79, 2010.

[6] Heitor S. Lopes and Wagner R. Weinert, "Egipsys: An Enhanced Gene Expression Programming Approach For Symbolic Regression Problems", *International Journal of Applied Mathematics and Computer Science*, Vol. 14, No. 3, pp. 375–384, 2004.

[7] Irina Ciornei and Elias Kyriakides, "Hybrid Ant Colony-Genetic Algorithm (GAAPI) for Global Continuous Optimization", *IEEE Transactions on Systems, Man and Cybernetics-Cybernetics*, Vol. 42, No. 1, pp. 234-245, 2012.

[8] Laurens Jan Pit, "Parallel Genetic Algorithms", Master's Thesis, Dept. Computer Science, Leiden University pp.19-22, 1995.

[9] L. Zhuo, and V. K. Prasanna, "Scalable and Modular Algorithms for Floating-Point Matrix Multiplication of Reconfigurable Computing Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 4, pp. 433-448, 2007.

[10] CoreMark http://www.coremark.org.

[11] http://www.dtreg.com/gep.htm.