ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2009. No. 6(94)
ELEKTRONIKA IR ELEKTROTECHNIKA

TELECOMMUNICATIONS ENGINEERING
T180
TELEKOMUNIKACIJŲ INŽINERIJA

Ant System Implementation using Microblaze: Some Preliminary Results on Efficiency Study

R. Laptik, V. Arminas, D. Navakauskas
Department of Electronic Systems, Vilnius Gediminas Technical University,
Naugarduko str. 41-422, LT-03227 Vilnius, Lithuania, phone: +370 5 2744765, e-mail: raimond.laptik@el.vgtu.lt

Introduction

Various optimization techniques are widely used to solve real world problems. Field of image processing is not an exception – optimization techniques were successfully applied in terrestrial imagining [1], growing field of computer vision [2] and biomedical image processing [3], to name a few. As performance and complexity of the digital circuits constantly increase, complete system-on-chip solutions [4] that are market appealing become a reality.

Field-Programmable Gate Array (FPGA) is a semiconductor device with programmable logic. FPGAs are primarily used for implementation of complex logic schemes. They can be programmed developing logic circuit diagram in software supplied by device manufacturer, or writing code in a Hardware Description Language. Performance of FPGA is usually lower than that of Application Specific Integrated Circuits however they are getting more powerful and cheaper as integrated circuit production technology improves. Because of flexible development and quick upgrade, FPGAs are widely used in industry.

Internal structure of FPGA can be represented by a set of programmable logic blocks that can carry out basic logic functions and by the help of reconfigurable interconnections, can implement various complex logic circuits. FPGAs logic blocks also include D-type flip-flops that can implement various memory elements. Large quantity of programmable blocks makes FPGA a perfect choice for system-on-chip applications.

Xilinx – one of producers of FPGAs – also develops a program code for implementation (on Spartan and Virtex FPGA families) a software processor called Microblaze [5]. The availability of such software processor enforces quicker system development, moreover, enables to implement sophisticated system-on-chip architectures [6]. However the use of Microblaze usually demands FPGA of larger size and reduces advantage of parallel computing. Thus successful application of Microblaze inheritably requires a good balance between system complexity, functionality and speed.

In this paper we present initial results on Microblaze efficiency study, while implementing Ant System on Virtex family FPGAs. Ant System [7] is a predecessor of less aggressive search algorithm – Max-Min Ant System [2]. Both paradigms are the part of Ant Colony Optimization theory [8] and multi-agent evolutionary optimization technique. This field is relatively new, nevertheless Ant Colony Optimization already proved to be suitable for solving real world combinatorial optimization problems [2, 9].

Most typical and widely known in combinatorial optimization [9] is Traveling Salesman Problem (TSP). TSP is usually included as a part of a testing set for evaluation of performance of optimization algorithms. Thus, in this work we present comparative results on implementation of Ant System and Brute Force algorithms that solve TSP.

We start with the setup of Microblaze for Ant System implementation. First we experimentally ground the best use of Pseudo-Random Number Generator testing it with Floating Point Unit and single or double precision arithmetic. Then we experimentally reason the usefulness of eight supplemental Microblaze core units. After determination of suitable Microblaze configuration, we experimentally investigate Ant System implementation performance on the TSP size.

Setup of Pseudo-Random Number Generator

Ant System uses random numbers for selection of movement direction [7], thus random number generator, implementable on different platforms, is a necessary part of the system. For the implementation of Pseudo-Random Number Generator we select *Multiply-With-Carry algorithm* [10, 11]. This algorithm, proposed by George Marsaglia, appears to be fast algorithm with uniform distribution of generated pseudo-random numbers.

Multiply-With-Carry algorithm implementation on FPGA is not unique, thus resulting different Pseudo-Random Number Generators needs to be evaluated. As a test framework, Monte-Carlo [12] method for number π calculation was implemented. It is based on calculation of the ratio of areas of quarter circle with radius r and square with side length r. So, π can be expressed by:

$$\pi = 4\frac{T_C}{T_S}, \tag{1}$$

here T_C is a number of random generated coordinates belonging to quarter circle; T_S is a number of generated coordinates belonging to square.

This test framework let us to evaluate performance of different Pseudo-Random Number Generators when single or double precision data are used.

In brief, there are two types of a floating point numbers available in Microblaze: single precision and double precision. Single precision number occupies 32 bits, where 24 bits are the precision bits. Double precision number occupies 64 bits, where 53 bits are the precision bits. Thus, single precision type has 7 significant digits, while double precision – 16 significant digits. However, implementation of Multiply-With-Carry algorithm can provide up to 9 significant digits. Results of performed experiments with different implementations are summarized in Table 1.

Table 1. Evaluation of Pseudo-Random Number Generators

Number of iterations	Computing time, s			
	Without FPU		With FPU	
	Single[1]	Double[1]	Single[1]	Double[1]
1 000	0.15	0.20	0.13	0.20
10 000	1.56	1.96	1.28	1.90
100 000	15.10	19.57	12.18	19.40
1 000 000	150.00	194.00	123.60	193.00

[1] – precision.

Based on these results one can state, that computing time of Pseudo-Random Number Generator linearly depends on number of iterations, moreover:

- The use of single against double precision requires about 1.3 times less time (without FPU) and about 1.5 times less time (with FPU);
- The use of floating point unit is advantageous; nevertheless it only slightly reduces the computing time of Monte-Carlo method.

Aiming to check the precision requirements, Ant System algorithm was simulated. For that purpose standard PC was used. It was found that for a TSP with n cities, required precision for Pseudo-Random Number Generator was $\log_{10}(n-1)$ significant digits, and for path calculation maximum used precision (even after 10^9 iterations) was less than 6 significant digits. For the further investigation of Ant System as Pseudo-Random Number Generator implementation Multiply-With-Carry algorithm in a single precision floating point numbers was considered.

Setup of Microblaze

Core of software processor Microblaze has supplemental units that can be used in order to speed up calculations. Implementation of each core unit takes some area on FPGA however it introduces additional commands or functions advantageous for specific applications:

- *Basic Floating Point Unit* (FPU) provides single precision floating point arithmetic operations;
- *Extended Floating Point Unit* add few more commands, e. g., square root operation;
- *Integer Multiplier Units* are of two types: 32 bits and 64 bits, and are used for integer multiplication;
- *Barrel Shifter Unit* provides additional shift by bits operations;
- *Integer Divider Unit* enables to perform division of integer numbers;
- *Machine Status Register Unit* adds two commands for setting and clearing machine status register;
- *Pattern Compare Unit* compares provided words and improves performance of string and pattern matching.

The usefulness of inclusion of additional core units (for summarized results see Table 2) was tested comparing time necessary to reach solution by two algorithms: Ant System and Brute Force method. As a testing framework, TSP with 14 cities was selected. This particular number of cities was chosen because search time for the solution by both approaches was almost the same.

Table 2. Evaluation of Microblaze with supplemental core units

Microblaze with supplemental core units	Computing time, s	
	Brute Force	Ant System
Only core	54.0	57.0
Basic FPU	53.0	33.4
Extended FPU	53.0	–
32 bits Multiplier	53.5	43.0
64 bits Multiplier	–	–
Barrel Shifter	–	–
Integer Divider	54.0	57.0
Machine Status Register	54.0	57.0
Pattern Compare	54.0	57.0
Basic FPU with 32 bits Multiplier	52.0	23.0

Results presented in Table 2 indicate, that the best improvement in performance of Ant System implementation is achieved when Basic Floating Point Unit together with 32 bits Integer Multiplier are used. The usefulness of these units can also be grounded by the nature of Ant System algorithm, where a lot of operations with floating point data need to be performed in order to calculate distance probability and pheromone deposition. Collective use of Basic Floating Point Unit and 32 bit Integer Multiplier in comparison with no use of supplemental core units enables to reduce the computing time about 2.5 times. Performance results of Brute Force algorithm implementation also confirm advantage of the same units use, however reduction of computing time is comparatively small (only 4 %).

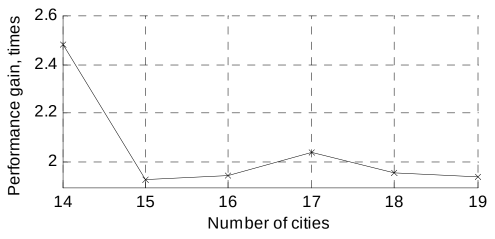

Fig. 1. Performance gain of Ant System

These results may be justified by the fact that Brute Force algorithm is of extensively iterative nature with only few calculations per iteration. During extended investigation of collective use of both units (see Fig. 1) appeared that when the total number of cities increases performance gain decreases and stabilizes at 1.9 times (cache memory limit is reached).

Preliminary results of Ant System implementation

Implementation of Ant System algorithm [7] was done on a multi FPGA DN8000K10PSX board produced by Dini Group [13], featuring three Xilinx Virtex-4 FPGAs. For implementation Microblaze V9.0 was used, provided together with Xilinx EDK 10.1 software.

In solving TSP it is important to choose good initial parameters for Ant System as it increases the probability to find better solution. Parameters for probability of ant movement expressed by

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{l}\left[\tau_{il}(t)\right]^{\alpha} \cdot \left[\eta_{il}\right]^{\beta}}, \qquad (2)$$

with pheromone level expressed by

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot \rho + \Delta\tau_{ij}^{\text{best}}, \qquad (3)$$

are chosen according to recommendations [7]:

- pheromone sensitivity $\alpha = 1$;
- sensitivity to heuristic information $\beta = 2$ (in TSP, heuristic information is a distance between cities);
- initial pheromone level $\tau_{ij}(0) = 0.2$.

Number of ants N and pheromone evaporation coefficient ρ depend on number of cities and tend to increase as complexity of the problem increases.

During single iteration ants are sent through all cities and pheromone level is renewed according to the path length. The shortest found path, as a solution of TSP, is selected after all iterations are computed. Experiments are repeated 1 000 times for each total number of cities. Inevitably with the increase of the total number of cities, adjustments to the number of ants N and evaporation coefficient ρ are done. Fig. 2 present summarized experimental results, showing travel distance (part a), standard deviation of a travel distance (part b), number of used ants (part c) and computing time (part d) dependences on the total number of cities used in solving TSP.

Standard deviation of a travel distance is used as a measure of a solution quality and determinism of the algorithm.

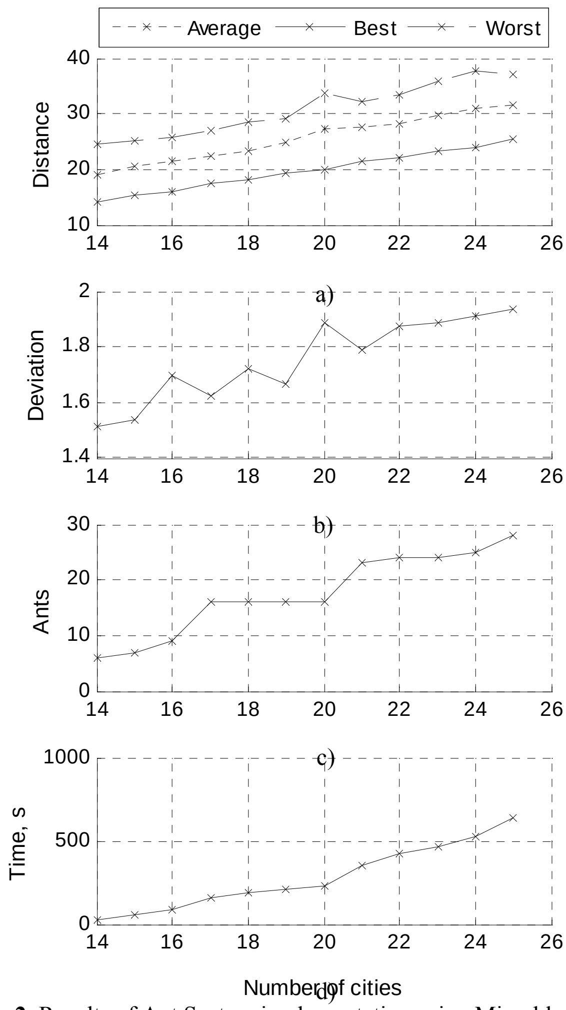

Fig. 2. Results of Ant System implementation using Microblaze

When standard deviation reaches zero, algorithm becomes deterministic. It may happen because of few reasons:

- Too many ants are used – solution average is close or equal to best solution;
- Bad parameters of Ant System causes solution to stuck in the same local minima.

Large standard deviation shows poor efficiency of the algorithm. Search becomes fully stochastic, heuristic and pheromone information has no or very small influence on solution, because of:

- Not enough ants are used – average is far from the best solution;
- Bad parameters cause Ant System to be unable to search around good solutions.

Increase of the total number of cities, leads to TSP search space increase thus more ants are needed (Fig. 2c) and more time it takes (Fig. 2d) to find the shortest path between cities. In Ant System case, time increase is linearly dependent on number of ants. During experimentation number of ants was increased only in the case of bad solutions. When the total number of cities was in 17–20 range, number of ants was kept constant as system delivered good solutions. Standard deviation (Fig. 2b) reflects system balance: increase when there is an excess or shortage of ants, and decrease when number of ants is sufficient for a right solution.

Conclusions

1. Pseudo-Random Number Generator plays an important role in Ant System implementation – the use of Multiply-With-Carry algorithm let us to get repeatable results on different platforms.
2. The use of Basic Floating Point Unit and 32 bits Integer Multiplier has the greatest influence on Ant System implementation performance – it 2.5 times speeds 14 cities Traveling Salesman Problem solution.
3. Rapid growth of standard deviation may be used as an indicator that system should be adjusted for current complexity of the problem.

Acknowledgement

This work was supported by Lithuanian State Science and Studies Foundation (projects T-08127 and T-09152, corresponding contracts T-112/08 and T-106/09).

References

1. **Mateika D., Martavičius R.** Large Image Formation using Harris-Plessey Corner Detection Algorithm // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No, 5(85). – P. 21–24.
2. **Laptik R., Navakauskas D.** MAX-MIN Ant System in Image Preprocessing // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. – No. 1(89). – P. 21–24.
3. **Matuzevičius D., Navakauskas D.** Investigation of Segmentation Methods for Proteomics // Electronics and Electrical Engineering. – Kaunas: Technologija, 2005. – No. 7(63). – P. 66–70.
4. **Serackis A., Matuzevičius D., Navakauskas D.** Reconstruction of Protein Spots Using DSP Modules // Proceedings of 29th International Conference on Fundamentals of Electrotechnics and Circuit Theory. – 2006. –Vol. 2. – P. 573–576.
5. **MicroBlaze Processor.** Xilinx Inc. Accessed at: http://www.xilinx.com/products/design_resources/proc_central/microblaze.htm.
6. **Huerta P., Castillo J, Martinez I. J.,** Multi MicroBlaze System for Parallel Computing // Proceedings of 9th International Conference on Circuits. – 2005. – P. 1–6.
7. **Dorigo M., Maniezzo V., Colorni A.** The Ant System: Optimization by a colony of cooperating agents // IEEE transactions On Systems, Man, and Cybernetics. Part B. – 1996. – Vol. 26, No. 1. – P. 1–13.
8. **Laptik R., Navakauskas D.** Application of Ant Colony Optimization for Image Segmentation // Electronics and Electrical Engineering. – 2007. – No. 8(80). – P. 13–18.
9. **Stutzle T., Hoos H. H.** The MAX-MIN Ant System and local search for the traveling salesman problem // Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), Piscataway, NJ. – 1997 – P. 309–314.
10. **Marsaglia G.** Yet another RNG // Electronic billboard sci.stat.math. Accessed at: http://groups.google.com/group/sci.stat.math/browse_thread/thread/a7b68b5b74ec272b/dd90bad892f56640.
11. **Couture R., L'Ecuyer P.** Linear Recurrences with Carry a Uniform Random Number Generators // Proceedings of the 1995 Winter Simulation Conference. – P. 263–267.
12. **Metropolis N., Ulam S.** The Monte Carlo Method // Journal of the American Statistical Association. – 1949. – 44(247). – P. 335–341.
13. **DN8000K10PSX board**. The Dini Group Accessed at: http://www.dinigroup.com/DN8000k10psx.php.

Received 2009 04 16

R. Laptik, V. Arminas, D. Navakauskas. Ant System Implementation using Microblaze: Some Preliminary Results on Efficiency Study // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. – No. 6(94). – P. 27–30.

The paper presents some preliminary results on efficiency study of Ant System implementation using software processor Microblaze. By the use of Monte-Carlo tests of number π calculation the best use of Pseudo-Random Number Generator – implementation of Multiply-With-Carry algorithm in a single precision floating point numbers – is grounded. By experimentation the usefulness of eight supplemental Microblaze core units is assessed and the advantage of the use of Basic Floating Point Unit together with 32 bits Integer Multiplier is proven. Experimental investigation of Traveling Salesman Problem solution by implemented Ant System is presented and confirms that rapid growth of standard deviation may be used as an indicator that system should be adjusted for current complexity of the problem. Ill. 2, bibl. 13. (in English; summaries in English, Russian and Lithuanian).

Р. Лаптик, В. Арминас, Д. Навакаускас. Воплощение муравьиной системы, используя Microblaze: предварительная оценка производительности // Электроника и электротехника. – Каунас: Технология, 2009. – № 6(94). – С. 27–30.

Представлены первичные результаты исследования производительности муравьиной системы, используя программный процессор Microblaze. Для обоснования применения генератора псевдо-случайных чисел, основанного на алгоритме умножения с переносом, используется метод Монте-Карло для расчета числа π. Экспериментально исследуются восемь внутренних счетных устройств Microblaze, доказывается преимущество использования основного устройства вычисления с плавающей запятой и 32 битного умножителя. Представлено экспериментальное исследование задачи путешествующего коммивояжера с подтверждением, что резкий рост среднего квадратичного отклонения может быть использован как индикатор требования изменения параметров системы. Ил. 2, библ. 13 (на английском языке; рефераты на английском, русском и литовском яз.).

R. Laptik, V. Arminas, D. Navakauskas. Skruzdžių sistemos įdiegimas naudojant Microblaze procesorių: pirminis našumo įvertinimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2009. – Nr. 6(94). – P. 27–30.

Straipsnyje pateikiami pirmieji programiniame procesoriuje Microblaze įdiegtos skruzdžių sistemos našumo tyrimų rezultatai. Skruzdžių sistemos pseudo atsitiktinių skaičių generatoriaus naudojimas pagrindžiamas perkeliamosios daugybos algoritmu – Monte Karlo metodu nustatant skaičiaus vertę. Iš aštuonių procesoriaus Microblaze branduolio modulių eksperimentiškai atrenkami daugiausiai įtakos turintys moduliai: pagrindinis slankiojo kablelio aritmetikos modulis ir 32 bitų sveikųjų skaičių daugintuvas. Skruzdžių sistema sprendžiant *keliaujančio prekeivio* uždavinį parodoma, jog iš staigaus vidutinio kvadratinio nuokrypio augimo galima spręsti apie poreikį keisti sistemos parametrus. Il. 2, bibl. 13. (anglų kalba; santraukos anglų, rusų ir lietuvių k.).