Microprocessors and Microsystems 45 (2016) 187–197

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Implementation of harmony search on embedded platform

Mohammed El-Shafei, Imtiaz Ahmad, Mohammad Gh. Alfailakawi*

Department of Computer Engineering, Kuwait University, P.O. Box 5969, 13060 Safat, Kuwait

ARTICLE INFO

Article history:
Received 23 October 2015
Revised 23 February 2016
Accepted 5 May 2016
Available online 6 May 2016

Keywords:
Harmony search
Embedded system
Field programmable gate array (FPGA)
Very high speed integrated circuit
Hardware description language (VHDL)

ABSTRACT

Harmony Search (HS) is relatively a new population-based meta-heuristic optimization algorithm that imitates the music improvisation process of musicians to search for a perfect state of harmony. HS has attracted a lot of attention by showing excellent results for a wide range of optimization problems in diverse fields. HS is typically implemented on a software platform, which restrict its applications to real-time applications. In order to accelerate the algorithm, one can proceed with the parallelization of the algorithm and/or map it directly onto hardware to achieve faster execution time. This paper presents an efficient architecture for parallel HS algorithm in FPGA platform in order to improve HS performance in terms of execution time, resource utilization and power consumption while searching several solution candidates for a problem. The implementation is tested using a suite of well-known benchmark functions. Analysis of the experimental results show that the proposed concurrent implementation has a promising performance up to 175x and no less than 16x as compared with software implementation.

1. Introduction

The last decade has witnessed the advent and prevalent use of population based meta-heuristic optimization algorithms to solve a wide range of real world problems in diverse fields. Meta-heuristics are intelligent self-learning algorithms inspired and developed by mimicking the intelligent processes and behaviors arising in nature, sociology, and other disciplines [1]. Harmony Search (HS) is a population-based meta-heuristic optimization algorithm developed by Geem et al. in 2001 [2] which imitates the music improvisation process where musicians improvise their instruments' pitch searching for a perfect state of harmony [3]. The improvisation process which happens when a musician searches for a better state of harmony, such as jazz improvisation seeks to find musically pleasing harmony as determined by an aesthetic standard, just as the optimization process seeks to find a global solution determined by an objective function. The pitch of each musical instrument determines the aesthetic quality, just as the objective function value is determined by the set of values assigned to each design variable [4]. To enhance the overall search performance, HS needs to balance the tradeoffs between two important aspects: diversification and intensification. Diversification stands for covering the search space randomly by existing partial solutions in order to expand and explore search area diversely. When diversification is strong, the search will cover large search area of the problem to be explored. However, if the diversification is kept low, there is a possibility of failing to explore areas that could provide better solution for the problem. The other aspect, i.e. intensification, refers to improving the results of particular solutions in specific region by finding the best solution in a given local area; this aspect will develop harmonies distributed in different problem areas to provide best local solutions for each region such harmonies cover. Having suitable intensification will allow the harmony to learn from previous results and tune the parameters that controls the algorithm's performance in order to improve convergence [5].

HS mimics musicians in the improvisation process to enrich their experiences by practice in searching for better harmonies based on randomness (diversification) or their experiences (intensification). The HS is one of the major modern meta-heuristic optimization techniques which is simple in concept, few in parameters, easy in implementation, and can be used efficiently and effectively to find excellent solutions to extremely hard numeric maximization/minimization problems. Although, HS is a relatively new meta-heuristic algorithm, its effectiveness and robustness have been demonstrated by successfully applying it to various optimization problems in diverse fields such as science, engineering, and medicine [1,3].

In the last decade, a new computing paradigm called Reconfigurable Computing (RC) has emerged. RC systems have soft-definable features that can overcome the limitations of the two well-known computing paradigms, the General Purpose Processor (GPPs) in the form of software, and Application Specific Integrated Circuits (ASICs) in the form of hardware. RC systems

* Corresponding author. Fax: +96524839461.
E-mail address: alfailakawi.m@ku.edu.kw (M.Gh. Alfailakawi).

http://dx.doi.org/10.1016/j.micpro.2016.05.003
0141-9331/

combine the flexibility offered by software and the performance offered by hardware [6,7] by making use of reconfigurable Field Programmable Gate Arrays (FPGAs). In addition, re-configurability provided by RC is mandatory for design adaptability to survive unforeseen physical effects such as aging and temperature variation and/or new application requirements [8]. Furthermore, RC system efficiency and flexibility have been demonstrated by successfully accelerating a wide variety of applications such as string set matching for bioinformatics research [9], artificial intelligence (AI) applications [10], and cryptography [11] due to the possibility of massive parallel processing. Nowadays, FPGAs are common computational elements used in exploring many embedded applications, such as in small mobile robots and industrial control units. Their memory and communication capacity is also often limited. Therefore, designs targeted on such systems need to take into account the constraints in available computational resources, cost, size, and energy consumption. Thus, one of the main goals is to find the right balance between these system constraints and execution performance desired.

Recently, there has been a widespread use of meta-heuristic algorithms to solve challenging problems in diverse fields of science and engineering. Meta-heuristic algorithms are generally compute intensive and are slow when implemented in software. FPGAs provide a suitable platform for hardware implementation of software algorithms. The low execution time of FPGA in comparison with its software counterparts is the main reason of using FPGA as a platform to implement meta-heuristics algorithms for real-time applications. Because of the aforementioned reasons, successful hardware implementation for meta-heuristics algorithms such as genetic algorithms [12], particle swarm optimization [13], and differential evolution [14] in FPGAs have been reported recently. The salient features of these hardware implementations are that they exploit the intrinsic parallelism of the algorithm, and the flexibility and parallel processing capabilities of FPGAs to devise architectures in order to improve the performance of these algorithms for real-time applications. Although, there exist hardware implementation of many meta-heuristic algorithms, to the author's best knowledge, the reported work for hardware/software co-design implementation of HS is to accelerate the protein folding simulations [15].

The main focus of the work in [15] is to develop approaches for accelerating the intractable 3D-HP-SC (three dimensional Hydrophobic-Polar Side-Chain) model based Protein Folding Problem (PFP) using Harmony Search algorithm. The solution to PFP is important for medical advancements and development of new drugs. The authors proposed three different FPGA-based implementations in VHDL, two of them used an embedded processor (Altera's NIOS II) as part of its hardware design. In one of the NIOS II processor based implementation, the HS was implemented fully in software by using ANSI-C programming language. In the second NIOS II processor based implementation, a dedicated hardware block for computing the fitness function was attached to the embedded processor as a slave component. In this version, all HS algorithm related computation is performed by the processor except the fitness computation, which is offloaded to fitness calculation module. The third hardware-based approach is fully implemented in hardware and does not use an embedded processor. This approach has dedicated hardware blocks, specifically developed to perform the HS algorithm. The main blocks of the architecture includes: harmony memory, harmony search core, random number generator and fitness calculation unit. First, the harmony memory is initialized producing a new harmony for each position on the harmony memory. Each variable of each new harmony is independent of the others. Therefore, each new harmony is generated in one clock pulse using a set of N random number generators, where N is the number of variables in the harmony. Once the harmony memory is loaded with the initial harmonies, the iterative process starts, where at each iteration, four individuals (harmonies) are evaluated simultaneously (in parallel). In the improvisation step, the selection of each variable of the new harmony is performed independently. Only block level architecture was given in [15] and was designed specifically for that application.

The hardware implementation of HS for general optimization is not reported in literature. Therefore, unlike the work in [15] which is application-specific, the purpose of this work is to present an efficient architecture for parallel HS algorithm for general optimization problems in an FPGA platform that improves HS performance in terms of execution time, resource utilization and power consumption while searching several solution candidates for a problem. In addition, a different parallelism strategy is adapted to attain maximum speedup for a variant of HS algorithm. A key feature which enhances the performance of FPGA is its parallel processing capability. This feature enables the algorithm to enhance its performance by running different parts of the algorithm simultaneously which is very important for real-time applications [15,16].

The remainder of the paper is organized as follows. Section 2 presents the basic HS algorithm. The system architecture and implementation on embedded platform is presented in Section 3. Synthesis results and analysis on benchmark functions are reported in Section 4. Finally, Section 5 concludes the paper and present directions for future work.

2. Basic harmony search algorithm

Harmony Search is a population-based meta-heuristic algorithm inspired by the improvisation process of musicians. Musicians in their process of improvising a pitch use their skills of composition (i.e. playing a new pitch selected from the possible sound range), memorization (i.e. playing a pitch from their own memory) or playing an adjacent pitch from the pitch in their memory in pursuit of a composition with a perfect harmony. HS algorithm uses a similar search strategies when deciding to select new values for a variable to find an optimum solution to an optimization problem. HS chooses a value from harmony memory (memory consideration), a close-by value to a one from memory (pitch adjustment), or a totally random value from problem-specific range (randomization). HS imitates musicians seeking attractive harmony by finding best pitches between different music instruments. HS algorithm works to find the best possible and suitable solution based on objective function unlike other approaches that depend on gradient or newton's method. The pseudo-code of HS algorithm adopted from [2,15] is presented in Algorithm 1.

Harmony Search (HS) algorithm starts with a Harmony Memory of size HMS, where each memory position is occupied by a harmony of size N (musicians). Each improvisation step of a new harmony is generated from harmonies already present in the memory. If the newly generated harmony is better than the worst harmony in memory, the worst harmony is replaced with the new one. Harmony improvising/update process is repeated until the maximum number of improvisations (NI) is reached. The HS algorithm can be described by five main steps described below [2,15]:

Initialization and Parameters Settings: This is the first step of the algorithm where the optimization problem is defined (line 3) and key parameters that directly affect the performance of the algorithm are identified, which are:

- Harmony Memory size (HMS): represents the total number of harmonies (solutions) stored in memory to guide the search;
- Harmony Memory Consideration Rate (HMCR): a value in the range (0.7, 0.95) that represents the percent of values