SOPC-Based Parallel Genetic Algorithm

M. Salmani Jelodar, M. Kamal, S. M. Fakhraie, M. Nili Ahmadabadi

Abstract—The ever-growing complexity of the modern chips is forcing fundamental changes in the way systems are designed. System-on-a-Programmable-Chip (SOPC) concept is bringing a major revolution in the design of integrated circuits, due to the fact that it makes unprecedented levels of in-field integration possible. Genetic Algorithm (GA) is a powerful function optimizer that is used successfully to solve problems in many different disciplines. A major drawback of GA is that it needs huge computation time for sequential execution on PCs. Therefore, the hardware implementation of GA has been the focus of some recent studies. Parallel GA (PGA) is particularly important for efficient hardware implementation and promise substantial gains in performance and results. In this paper, a SOPC-based PGA framework is proposed. Our proposed framework can be used in real-time applications. We have implemented our proposed system on an Altera® Stratix Development Kit and we compare its performance with the corresponding software simulation. The results obtained indicate a speedup of up to 50 times in the elapsed computation time.

I. INTRODUCTION

SOPC platforms are becoming more prevalent as a solution for the implementation of embedded computing systems. This is due to their ease of implementation and highly customizable nature. SOPC systems provide the ability to have multiple processors in a single chip, and consequently, we can implement a parallel processing system on a SOPC.

Genetic algorithm (GA) is a general-purpose and powerful search algorithm used to solve hard optimization problems by simulating natural evolution over populations of candidate solutions [1],[2]. However, GA process is time-consuming. For many real-world applications, GA can run for days, even when it is executed on a high performance workstation. To reduce the execution time of GA, several methods have been offered, including parallel and/or distributed processing of GA along with its hardware implementation[3]-[5]. A myriad of hardware-based GA solutions have been put forward in recent years [6]-[15]. Evolutionary algorithms (EAs) have been hardware implemented in three main areas [16]:

1- A means of implementing the fitness functions of the GA or genetic programming (GP)[17].

2- A platform for implementing the EAs (especially for GA) like the GA processor (GAP) [7],[18],[19] for general optimization problems.

3- An evolutionary engine in intrinsic evolvable hardware [20].

In the following paragraphs, these three areas are surveyed in brief.

For the hardware implementation of GA, Scott et al. proposed a hardware-based genetic algorithm (HGA), which was implemented on a set of field-programmable gate arrays (FPGAs) [7]. The HGA is based on the standard Genetic Algorithm (SGA)[23].

Yoshida et al. [7],[18] proposed a hardware-based GA, called the GA Processor (GAP). The GAP is based on a steady-state GA.

Shakleford et al.[24] proposed their original GA procedure. This procedure was called Survival based GA, and it was somewhat similar to the steady-state GA. They implemented a complete GA system using a Xilinx XC3200E chip. Their implementation uses extensive pipelines and parallel fitness evaluation to get a performance increase of 320 times when compared to the same algorithm running on a 366MHz Pentium CPU.

Kajitani et al. [6] proposed a GA hardware for evolvable hardware on a single LSI chip. Wakabayashi et al. [25] also proposed a VLSI implementation of an adaptive GA, called the GA Accelerator (GAA) chip as a general purpose GA hardware.

Koza et al. [17] used an FPGA to speed up the fitness evaluation of a sorting network, in which the FPGA was used solely to perform the fitness evaluation. The initial population was created by a host computer, and then individuals were downloaded to the pre-programmed FPGA and the FPGA was instructed to evaluate the fitness of the individual. Subsequent selection and breeding stages were again performed by the host computer.

Yamaguchi et al. [26] used an FPGA to implement a coprocessor for evolutionary computation to solve the iterated prisoners dilemma (IPD) problem. They reported a 200 times performance speedup in processing the IPD functions on a FPGA when compared to a 750MHz Pentium processor.

Graham and Nelson [27] implemented a complete GA system using four FPGAs. Each FPGA was programmed to carry out a different function; Selection, Crossover, Fitness and Mutation. Each FPGA passed its results to the next, forming a pipeline. The performance of their system was compared to a software implementation running on a