01 Jan 2004

Swarm Intelligence for Digital Circuits Implementation on Field Programmable Gate Arrays Platforms

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Venu Gopal Gudise

Swarm Intelligence for Digital Circuits Implementation on Field Programmable Gate Arrays Platforms

Ganesh K. Venayagamoorthy and Venu G. Gudise
Dept. of Electrical and Computer Engineering
University of Missouri – Rolla, MO, 65409, USA
gkumar@ieee.org and venug@ieee.org

Abstract

Field programmable gate arrays (FPGAs) are becoming increasingly important implementation platforms for digital circuits. One of the necessary requirements to effectively utilize the FPGA's resources is an efficient placement and routing mechanism. This paper presents an optimization technique based on swarm intelligence for FPGA placement and routing. Mentor graphics technology mapping netlist file is used to generate initial FPGA placements and routings which are then optimized by particle swarm optimization (PSO). Results for the implementation of a binary coded decimal bidirectional counter and an arithmetic logic unit on a Xilinx FPGA show that PSO is a potential technique for solving the placement and routing problem.

1. Introduction

Field programmable gate arrays (FPGA's) have been attracting alot attention for digital platform implementations because of their programmability and relatively high density. In particular, SRAM-based FPGA's make use of lookup tables (LUT's) or similar circuits, as their basic blocks, called logic blocks.

Since logic blocks and routing resources are predefined in an FPGA chip, it is difficult to fit a large, dense design on any given FPGA while meeting aggressive system-level delay constraints. Optimizing for 100% wirability is often at odds with optimization for speed. Critical paths must be given priority during placement. Simulated annealing has been applied to the FPGA placement problem in a manner similar to the placement of standard cells [1]. While standard cell techniques are sufficient for those FPGAs that invest a large portion of their chip area in routing resources [2], special care must be taken in FPGA architectures that seek to limit the cost of routing. Min-cut placement combined with hierarchical global routing that introduces signal congestion into placement process is used in [3]. A penalty-driven improvement algorithm is used in [4].

A new technique called the PSO that emerges and allies itself to evolutionary algorithms based on simulation of the behavior of a flock of birds or school of fish. Swarm algorithms differ from evolutionary algorithms most importantly in both metaphorical explanation and how they work. What is new with the swarm algorithm is that the individuals (particles) persist over time, influencing one another's search of the problem space. The particles in PSO are known to have fast convergence to local/global optimum position(s) over a small number of iterations [5].

In this paper the concept of PSO is applied to solve FPGA placement and routing. Mentor graphics technology mapped netlist file is used to generate the initial FPGA placements and routings which are then optimized by PSO. This is demonstrated on the implementation of a 4-bit BCD counter and an ALU on a Xilinx FPGA.

The organization of this paper is as follows: Section 2 gives a brief introduction of a FPGA, the placement and routing problem; Section 3 explains the PSO algorithm. Section 4 describes the PSO based placement and routing and Section 5 presents some results.

2. FPGA placement and routing

FPGAs are programmable devices with relatively high density. Symmetrical array (Fig. 1), row-based and hierarchical-PLD are most commonly used architectures with either multiplexer or look-up table logic. In this paper, Xilinx FPGAs are considered.

Xilinx LCA (logic cell array) basic logic cells, called as configurable logic blocks (CLBs) contain both combinational logic and flip-flops. CLBs are based on the use of SRAM as a look-up table. The truth table for a K-input logic function is stored in a 2^Kx1 SRAM. The address lines of the SRAM function as inputs and output line of the SRAM provides the value of the logic function.

Xilinx FPGA has three major configurable elements: configurable logic blocks (CLBs), input/output blocks (IOB), and interconnects. The CLBs provide the

Proceedings of the 2004 NASA/DoD Conference on Evolution Hardware (EH'04)
0-7695-2145-2/04 $ 20.00 © 2004 IEEE

functional elements for constructing logic. The IOBs provide the interface between the package pins and internal signal lines. The programmable interconnects provide routing paths to connect the inputs and outputs of the CLBs and IOBs to appropriate networks.

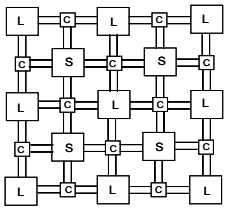

Fig. 1 Symmetrical array FPGA model

3. Particle swarm optimization

PSO is a form of evolutionary computation technique developed by Kennedy and Eberhart [6-7]. PSO like a genetic algorithm (GA) is a population (swarm) based optimization tool. One major difference between PSO and traditional evolutionary computation methods is that particles' velocities are adjusted, while evolutionary individuals' positions are acted upon; it is as if the "fate" is altered rather than "state" of the PSO individuals [7].

The system initially has a population of random solutions. Each potential solution, called particle, is flown through the problem space. The particles have memory and each particle keeps track of previous best position and corresponding fitness. The previous best value is called as 'p_{best}'. It also has another value called 'g_{best}', which is the best value of all the particles p_{best} in the swarm. The basic concept of PSO technique lies in accelerating each particle towards its p_{best} and the g_{best} locations at each time step.

The main steps in the PSO are described as follows:

(i) Initialize a population (array) of particles with random positions and velocities of d dimensions in the problem space.
(ii) For each particle, evaluate the desired optimization fitness function in d variables.
(iii) Compare particle's fitness evaluation with particle's p_{best}. If current value is better than p_{best}, then set p_{best} value equal to the current value and the p_{best} location equal to the current location in d-dimensional space.
(iv) Compare fitness evaluation with the population's overall previous best. It the current value is better than g_{best}, then reset g_{best} to the current particle's array index and value.
(v) Change the velocity and position of the particle according to equations (1) and (2) respectively. V_{id} and X_{id} represent the velocity and position of i^{th} particle with d dimensions respectively and, $rand_1$ and $rand_2$ are two uniform random functions.

$$V_{id} = W \times V_{id} + c_1 \times rand_1 \times (P_{bestid} - X_{id}) + c_2 \times rand_2 \times (G_{bestid} - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

(vi) Repeat step (ii) until a convergence criterion is met.

The parameters of PSO are described as follows: W called the inertia weight controls the exploration and exploitation of the search space because it dynamically adjusts velocity. V_{max} is the maximum allowable velocity for the particles. If V_{max} is too high, then particles will move beyond good solution and if V_{max} is too low, then particles will be trapped in local minima. c_1, c_2 termed as cognition and social components respectively are the acceleration constants which change the velocity of a particle towards p_{best} and g_{best}. A swarm of particles can be used locally or globally in a search space.

4. PSO placement and routing

For the preliminary PSO based placement and routing work presented in this paper, the following assumptions are made:

(i) The distances between the CLBs and IOBs are taken in terms of the normalized units.
(ii) Congestion of the channels is not considered for routing.
(iii) All channels are of equal capacity.

The PSO based placement and routing is demonstrated on the implementation of a 4-bit BCD counter and a 4-bit ALU on a Xilinx XC4000 FPGA platform.

4.1 X74_168 counter

X74_168 [8] is a 4-stage, 4-bit, synchronous, loadable, cascadable, bidirectional binary-coded-decimal counter. The data on the D - A inputs is loaded into the counter when the load enable (LOAD) is Low. The LOAD input, when Low, has priority over parallel clock enable (ENP), trickle clock enable (ENT), and the bidirectional (U_D) control. The outputs (QD - QA) increment when U_D and LOAD are High and ENP and ENT are Low during the Low-to-High clock transition. The outputs decrement when LOAD is High and ENP, ENT, and U_D are Low during the Low-to-High clock transition. The counter ignores clock transitions when LOAD and either ENP or ENT are High.

4.2 Arithmetic logic unit (ALU)

A four bit arithmetic logic unit (ALU) performing 32 functions [9] is considered for FPGA implementation. It has four select signals and two modes of operation. There

are 16 logical functions and 16 arithmetic functions which are performed when mode is set to high and low respectively. Different functions are chosen based on the select signals.

4.3 Xilinx XC4000 FPGA model

The Xilinx XC4000 FPGA contains 196 CLBs in a 14×14 matrix. The four bit BCD counter (X74_168) and the arithmetic logic unit (SN74181 ALU) are implemented on Xilinx XC4000 FPGA and its placement and routing are carried out using Mentor Graphics (Figs. 2 and 3 respectively). The output of the netlist file uses 7 CLBs and 14 IOBs to implement the BCD counter (Fig. 2) and, 13 CLBs and 22 IOBs to implement ALU (Fig. 3). The netlist files are used in generating random placement and routing for use by PSO particles in the next subsection.

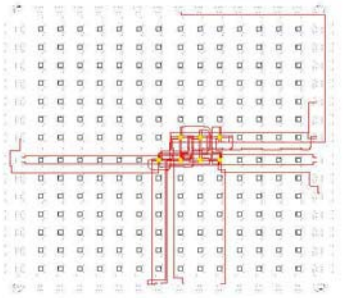

Fig. 2 X74_168 Counter implementation by mentor graphics using the xilinx XC4000 family

Fig. 3 ALU implementation by mentor graphics using the xilinx XC4000 family of logic gates

4.4 PSO placement and routing

The position vectors for both the IOB and CLB locations are randomly initialized. This is a two fold process. First, the IOB positions selected randomly are fixed and the CLBs are moved keeping their connections same and changing the CLBs positions on the FPGA for finding their optimal locations. After the CLBs move for some iterations and get a relatively better position, measured by the fitness function, the CLBs are fixed and the IOBs are moved keeping the connections same and changing the IOBs positions on the FPGA. This process is repeated until no change in the fitness function is found.

Each PSO particle represents a Xilinx XC4000 FPGA with 14×14 CLBs. For the BCD counter, the 7 CLBs and 14 IOBS are randomly placed on the FPGA and allowed to move within the 14×14 space. First, the coordinates (row, column) of the 7 CLBs on the FPGA are taken as the "position vector" of each swarm particle. This means each swarm particle position is matrix of 7×2. The fitness function or the performance function of the particles is evaluated as the sum of the distances of the respective connections between the CLBs wherever applicable. For example, if the output of a CLB at location [row2, column2] is an input to a CLB at location [row1, column1], the fitness is calculated as *[absolute (row1-row2) + absolute (column1-column2)]*.

The p_{best} of each particle stores the position vector (locations of all the 7 CLBs on the FPGA) where the fitness function is the lowest. The g_{best} stores the position vector (locations of all the 7 CLBs) with the lowest fitness function of the particle in the whole swarm. The p_{best} and the g_{best} are continuously updated whenever a position vector with a lower fitness is found for each particle and the swarm respectively. The g_{best} is the global optimal position vector for the FPGA placement. The same process is then repeated but this time the CLBs positions are fixed and IOBs are moved, and optimized.

The procedure is similar to the ALU circuit with the only difference that the number of IOBs is now 22 and CLBs is 13.

5. Results

A swarm of 25 particles randomly initialized is used for FPGA placement and routing for the BCD counter and ALU described above.

Figure 4 shows the position vector of the CLBs and IOBs corresponding to initial g_{best} of the swarm for the counter circuit with an initial fitness value of 533. A number of trials yielded a fitness of 386 on average over 2000 PSO iterations. Figure 5 shows the position vector of the g_{best} obtained after 2000 explorations on a given trial.

Figure 6 shows the position vector of the CLBs and IOBs corresponding to the initial g_{best} of the swarm for the ALU circuit with an initial fitness value of 892. A number of trials yielded a fitness of 672 on average over 2000 PSO iterations. Figure 7 shows the position vector of the g_{best} for the ALU circuit obtained after 2000 iterations on a given trial.

The results show that when PSO is applied to choose optimal positions for the CLBs placement and routing, the CLBs have been found to be placed close to each other. In this experiment, the CLBs' positions are restricted from overlapping. If this restriction is removed, all the CLBs are found to overlap with the p_{best} and g_{best} fitness's of the particles and the swarm respectively zero. The results obtained above for the counter and the ALU can be further improved over a large number of PSO iterations.

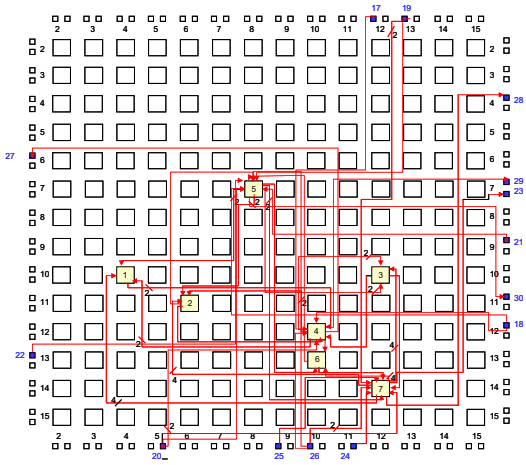

Fig. 4 initial gbest vector for counter (cost 533)

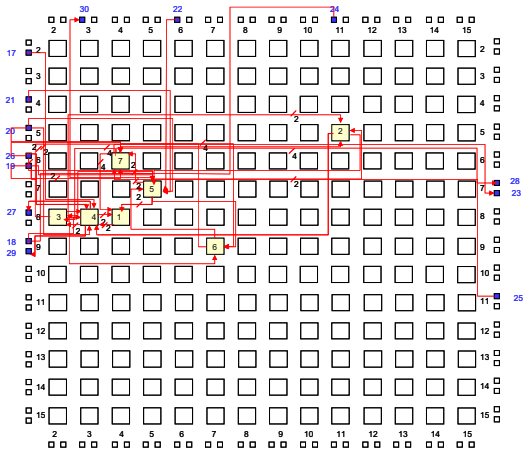

Fig. 5 Final gbest vector for counter (cost 337)

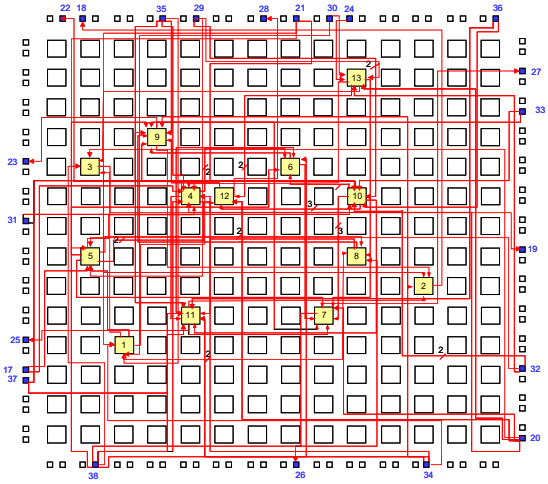

Fig. 6 Initial gbest vector for ALU (cost 892)

Conclusions

The preliminary work presented in this paper shows that PSO has the potential to be used for solving the FPGA placement and routing problem. The digital circuit implementation of FPGA platforms can be carried out more efficiently by optimizing the placement and routing of the logic blocks. Preliminary results on the Xilinx FPGA have been presented to minimize the interconnection lengths between the CLBs and IOBs for a counter and an ALU. Future work is to include the minimization of the interconnection distances between the CLBs and the IOBs subject to the channel congestion of the FPGA and the compare with existing placement algorithms. Different fitness functions for the PSO search will be explored such as the bounding box function.

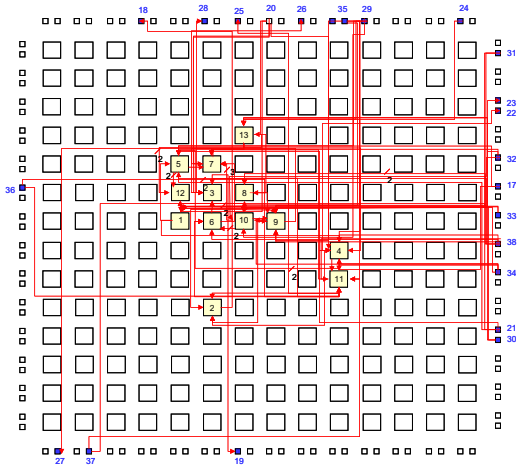

Fig. 7 final gbest vector for ALU (cost 669)

References

[1] C. Sechen, K. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Pplacement," *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 478 – 481, Nov 1987.

[2] Xilinx, Inc., *XACT Development System Reference Guide*, Jan 1993.

[3] N. Togawa, M. Sato, T. Ohtsuki, "A Simultaneous Placement and Global Routing Algorithm for Field-programmable Gate Arrays," *presented at FPGA94*, Berkeley, CA, 1994.

[4] J. Beetem, "Simultaneous Placement and Routing of the LABYRINTH Reconfigurable Logic Array," *Int. Workshop on Field-Programmable Logic and Applications*, pp.232 – 243, Oxford, England, 1991.

[5] V. G. Gudise, G. K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks", *IEEE Swarm Intelligence Symposium,* April, 2003, pp. 110 - 117.

[6] J. Kennedy, R. Eberhart, "Particle swarm optimization". *Proceedings, IEEE International Conf. on Neural Networks*, Perth, Australia. Vol. IV, pp. 1942–1948, 1995.

[7] J. Kennedy, Russell C. Eberhart, Yuhui Shi, *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.

[8] http://toolbox.xilinx.com/docsan/2_1i/data/common/lib/lib11_20.htm

[9] K. Hwang, *Computer Arithmetic: Principles, Architecture and Design*, John Wiley, 1979, ISBN 0-471-03496-7.