FPGA placement and routing using particle swarm optimization

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology, ganeshv@mst.edu

Venu Gopal Gudise

Follow this and additional works at: http://scholarsmine.mst.edu/faculty_work
Part of the Electrical and Computer Engineering Commons

Recommended Citation
http://scholarsmine.mst.edu/faculty_work/1699

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. For more information, please contact weaverjr@mst.edu.
FPGA Placement and Routing Using Particle Swarm Optimization

Venu G. Gudise and Ganesh K. Venayagamoorthy
Dept. of Electrical and Computer Engineering, University of Missouri – Rolla, MO 65409, USA.
venug@ieee.org and gkumar@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 fixed resources is an efficient placement and routing mechanism. This paper presents particle swarm optimization (PSO) for FPGA placement and routing. Preliminary results for the implementation of an arithmetic logic unit on a Xilinx FPGA show that PSO is a potential technique for solving the placement and routing problem.

1. Introduction

Placement is an NP-complete problem, thus new challenges arise from the continuous growth in the number of logic elements contained into commercial FPGAs. Since the solution space grows exponentially with the size of the circuit, it is necessary to resort to efficient combinatorial optimization algorithms. Simulated annealing (SA) is a combinatorial optimization method known for the quality of solutions it provides in solving placement problems. SA algorithm consists of heating the system at high temperature, then lowering the temperature slowly until no further changes occur. Problem with SA is that the system must be close to equilibrium throughout the process, which demands a careful adjustment of the annealing schedule parameters.

The limitation of routing resources is crucial. Optimizing for 100% wirability is often at odds with optimization for speed. Given the extremely granular, rigid nature of the routing resources here, optimizations for delay minimization are very difficult to estimate during placement, yet an overly conservative estimate may compromise overall routability [1].

In this paper, the concept of PSO [2, 3] to solve FPGA placement and routing obtained from the Mentor Graphics technology mapping netlist file is presented. This is demonstrated on the implementation of an arithmetic logic unit (ALU) on a Xilinx FPGA. Preliminary results are presented to show the potential of the PSO. To the knowledge of the authors, this is first application of PSO algorithm for FPGA placement and routing.

2. Particle Swarm Optimization

Particle swarm optimization is a form of evolutionary computation technique [2]. The PSO starts with random initial population (particles) with random velocities which are 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 ‘pbest’. It also has another value called ‘gbest’, which is the best value of all the particles pbest in the swarm. The basic concept of PSO technique lies in accelerating each particle towards its pbest and the gbest locations at each time step. The velocity and position of the particles are calculated 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.

$$ V_{id} = W \times V_{id} + c_1 \times \text{rand}_1 \times (P_{bestid} - X_{id}) + c_2 \times \text{rand}_2 \times (G_{bestid} - X_{id}) $$

$$ X_{id} = X_{id} + V_{id} $$

where $W$ is the inertia weight that controls the exploration and exploitation of the search space. $c_1$ and $c_2$, the cognition and social components respectively are the acceleration constants which changes the velocity of a particle towards the pbest and gbest.

3. PSO Placement and Routing

For the preliminary PSO based placement and routing work presented in this paper, the following assumptions are made: the distances between the CLBs are taken in terms of the normalized units; the input and output connections from the IOB ports are neglected and their distances are not considered; Congestion of the channels is not considered for routing; and all channels are of equal capacity.

The PSO based placement and routing is demonstrated on the implementation of a four bit ALU on a Xilinx XC4000 FPGA. A four bit ALU performing 32 functions [4]. There are 16 logical functions and 16 arithmetic functions. The Xilinx XC4000 FPGA contains 196 CLBs in a 14 × 14 matrix. A four bit arithmetic and logic unit (SN74181 ALU [4]) is implemented on Xilinx XC4000 FPGA and its placement and routing are carried out using...
Mentor Graphics. The output of the netlist file uses 13 CLBs to implement ALU. This netlist file is used in generating random placement and routing for use by PSO particles.

Each PSO particle represents a Xilinx XC4000 FPGA with $14 \times 14$ CLBs. For the ALU implementation the 13 CLBs are randomly placed on the FPGA and allowed to move within the $14 \times 14$ space. For this problem, the coordinates (row, column) of the 13 CLBs on the FPGA are taken as the “position vector” of each swarm particle. This means each particle’s position is matrix of $13 \times 2$. The fitness or performance function $f$ of the particles is evaluated as the sum of the normalized distances of the respective connections between the CLBs where applicable. For example, for an input to a CLB at location (row1, column1) from the output of a CLB at location (row2, column2), the fitness is given by $f = (\text{absolute}(\text{row1} - \text{row2}) + \text{absolute}(\text{column1} - \text{column2}))$.

The $p_{\text{best}}$ of each particle stores the position vector (locations of all the 13 CLBs on the FPGA) where the fitness function is the lowest. The $g_{\text{best}}$ stores the position vector (locations of all the 13 CLBs) with the lowest fitness function of the particle in the whole swarm. The $p_{\text{best}}$ and the $g_{\text{best}}$ are continuously updated whenever a position vector with a lower fitness is found for each particle and the swarm respectively. The $g_{\text{best}}$ is the global optimal position vector for the FPGA placement and routing. Every $p_{\text{best}}$ is a near optimal position vector after a number of PSO exploration for the FPGA placement and routing.

4. Results

A swarm of 25 particles is used to carry out the FPGA placement and routing for the ALU implementation described above. The PSO particles start with randomly initialized position vectors for the CLBs placement and routing on the FPGA. Figure 1 shows the position vector of the CLBs corresponding to the initial $g_{\text{best}}$ of the swarm with a fitness of 302.

A number of trials yielded a fitness of 91 on average over 5000 PSO explorations. Figure 2 shows the position vector of the $g_{\text{best}}$ obtained after 1500 explorations on a given trial. This figure shows that when PSO is applied to choose optimal positions for the CLBs placement and routing, the CLBs have been found to be placed almost adjacent 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 $p_{\text{best}}$ and $g_{\text{best}}$ fitness’s of the particles and the swarm respectively zero. This proves that the PSO algorithm finds the global minimum for all the particles for the FPGA placement and routing problem. Therefore, it can be concluded that PSO is a global optimization technique.

5. Conclusions

Preliminary results on the Xilinx FPGA have been presented to minimize the interconnection lengths between the CLBs using PSO. Future work is to include the minimization of the interconnection distances between the CLBs and the IOBs taking channel congestion into account.

6. References