An NoC Architecture with GA based Technique

Abdullah Mohd, Rafija Khan

Abstract— In olden days, the communication between different IP cores in System-on-Chip (SoC) was point-to-point communication. Due to this point-to-point communication the interconnection wires between the IP cores were more in number. So the complexity of SoC will be more and the power consumption is also high. To overcome these disadvantages a new SoC paradigm introduced. i.e. Network-on-Chip(NoC). Network-on-Chip (NoC) is an approach to design a communication subsystem between different IP cores in System-on-Chip (SoC). This paper presents a genetic algorithm based automated design that synthesizes application specific NoC topology and routes the communication traces on the interconnection network. The design technique solves multi-objective problem of minimizing power consumption and the router resource. Network-on-Chip (NoC) is regarded as a solution for future on-chip interconnects. However the performance advantages of conventional NoC architectures are limited by long latency and high power consumption due to multi-hop distance communication among process elements. To solve these limitations, we employ GA technique on-chip communication as express links for transferring data so that transfer latency can be reduced.

Index Terms Design Automation, Genetic Algorithms, Network-on-Chip (NoC), Pareto curve, Power consumption, Routing, Silicon-on-Chip (SoC).

1 INTRODUCTION

In future, System-on-Chip (SoC) architectures will be implemented in nano scale technologies which will be consist of hundreds of processing and memory elements communicating at several giga bytes per second. Network-on-Chip has emerged as the dominant solution for the interconnection architecture design problems of SoC design in nanoscale technologies by both academia and the industry [1] [2] [3]. This paper proposes a genetic algorithm based approach. Our GA-based technique solves multi objective problem that addresses NoC power consumption and interconnection resources. The GA-based technique has ability to escape local minima and generate excellent quality solutions in reasonable time. Further GA based technique can generate a set of Pareto points where each point represents a solution with a certain power and router resource consumption. The Pareto points provide the designer with an option to choose a solution corresponding to the desired tradeoff power and router resource consumption.

2 GENETIC ALGORITHMS FOR NOC TOPOLOGY DESIGN AND ROUTE GENERATION

As depicted in Fig.1 the overall design flow for application specific NoC synthesis, our technique takes communication trace graph (CTG), a system level floorplan and interconnection architecture as input. As a first step, for reducing the design space of the GA and thus its runtime, our technique allocates router at different locations are assumed to be at the corners of the computation cores as shown in Fig.1. As the possible physical locations of routers are known, we can determine shortest distance from any node to the routers and we can determine all inter-router distances. Therefore, we can estimate the link lengths (and resulting link power consumption) in the NoC very easily. In the next step, our GA-based design technique selects the routers to be utilized in the NoC and maps the nodes for router ports, constructs the topology of the network, and routes traffic traces on the interconnection architecture. As shown in fig.1 the final layout shows the node to router mapping by dotted lines, and the NoC topology by thick lines.

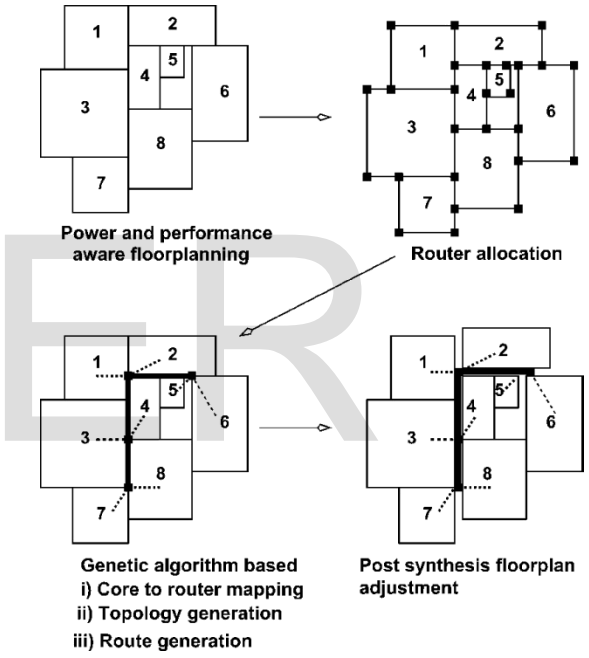

Fig.1 Design Flow for Synthesis of NoC

2.1 Overview of Genetic Algorithm

Genetic Algorithm is based on the biological phenomena of genetic evolution. It maintains population which is set of solutions. A GA-based technique is typically applies three operators namely reproduction, crossover, and mutation to produce new members. Reproduction duplicates a solution across generations, crossover combines two solutions to generate two new solutions, and mutation modifies an existing to generate one new solution. The algorithm continues to operate in an iterative manner. By applying genetic operators, GA evolves a new generation from current generation by operating in an iterative manner. By applying genetic operators a new generation is created by first increasing the population by generating new individual solutions, and then selecting a constant number of solutions based on their fitness criteria. Here the fitness criterion is a cost function that captures the optimization goal.

For supporting efficient application of genetic operators,

and GA-based optimization the population should be represented as data structure. In our GA-based technique the population is modeled hierarchically at three levels. The first represents the number of routers; the second level represents the mapping of the CTG nodes to the ports of intermediate routers. The population in our GA-based technique is represented as shown in Fig.2 in hierarchical manner.

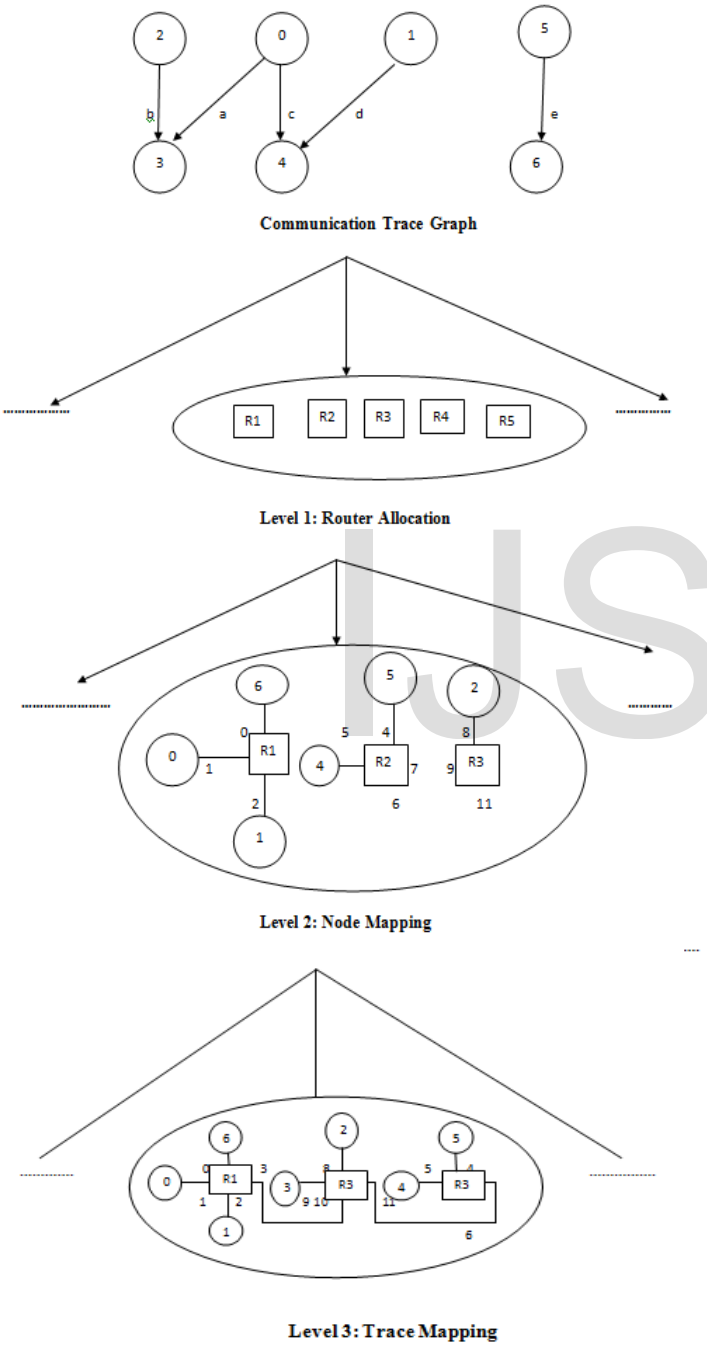

Fig.2 Hierarchical Representation

2.1.1 Router Level Data Structure

As discussed earlier, for application of genetic operators the population should be represented as data structure or strings of chromosomes. These strings of chromosomes are represented by sets of arrays. At the first level, the number of routers in a solution is represented by a binary array *arstr [MAXROUTER],* where MAXROUTER is the total number of routers in the architecture.

Note that the location and number of routers are determined before the GA is invoked. MAXROUTER is only the upper-limit on the routers that can be utilized. For the example, if MAXROUTER=24. We denote each router by r_i, where i is an integer such that 0≤ i≤ MAXROUTER. Each router that can be possibly utilized in the architecture is assigned a random location in the array given by *loc* (r_i).

The GA maintains I instances of array *arstr* is a binary array where a "1" in the location "i" denotes the router assigned to *arstr*[i] is possibly utilized in the architecture, and a "0" denotes that a router is not utilized. We say that the router is possibly utilized because even though the router is in the architecture, a communication trace may not be routed through it.

2.1.2 Node Level Data Structure

In the second level, The GA maintains J instances of node to port mapping. The node to port mapping at the second level is stored in an integer array given by *nprstr* [|v|]. The location contains the port number to which node is v_i€ V assigned. Fig.2, the string nprstr represents one such node to port mapping where node 0 is mapped to port 8, and so on. Note that since each port can map only one node, a port is not repeated in the string.

2.1.3 Trace Level Data Structure

Our GA maintains instances K of communication trace mappings for every instance of nprstr array. The communication trace mapping is represented by a linked list trlist of integer arrays refers to the communication mapping of trace i.

2.2 Overview of Optimization Technique

Consider the top level flowchart of our GA-based technique as shown in Fig.3. The input to the technique is the set of router architectures, the communication trace graph, and the system-level floorplan. An initial population of solutions is generated using the algorithms described, and the fitness of each trace level string is calculated. Initially, the exit criterion is set to false. Our technique applies genetic operators at the three levels of solution hierarchy with different probabilities. For each genetic operation at a higher level of hierarchy, the GA explores several operations at the lower levels. This approach aids in a structured design space exploration for the problem.

At each level the number of solutions produced by the crossover is much larger than those produced by mutation. At each level new individual solutions are produced by the application of genetic operators. A new generation is produced by selection of M fittest members among the current generation and the new individual solutions, where for level 1, for level2, and for level 3. Selection of members of current generation for the next generation models the reproduction operation.

At all the three levels of hierarchy, the fitness is given by the strongest complete solution at the trace level. Since each router allocation has J instances of node mapping and every node mapping has K instances of trace mapping, the fitness of a particular router allocation is given by the strongest trace among the possible trace level mapping. Termination condition is reached if the N successive generations don't result any change in the Pareto curve. We set N to summation of the number of nodes and edges in the CTG, that is N.

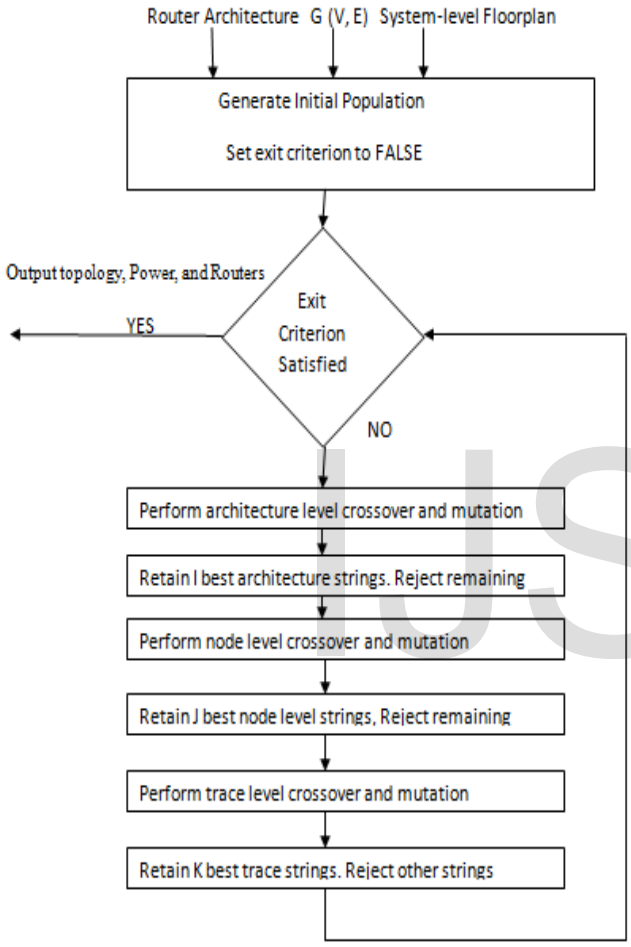

Fig. 3 GA flow for NoC Design

3. PROPOSED ARCHITECTURE

The proposed NoC architecture in this paper is as shown in the Fig.6. By applying Genetic Algorithm at three different levels by using GA hardware architectures as discussed above, the optimal solutions for optimizing the NoC is obtained. There is one router, four processors and one external memory units are used in this architecture. The tasks for the processor are given from the external memory unit. The router in this architecture is efficiently used such that no processor is idle at any time. On the other hand the interconnection power consumption is reduced by using the router which avoids point-to-point communication between the different processors. Thus the design technique solves the multi-objective problem of minimizing the power consumption and increase router resource efficiency.

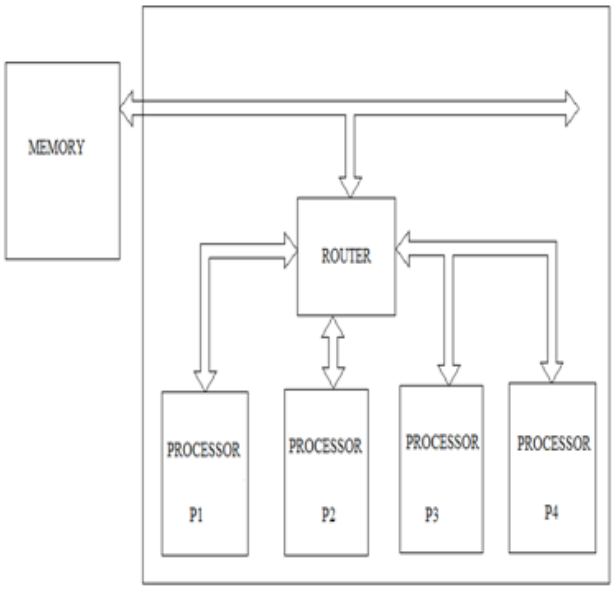

Fig. 4 Proposed Architecture

4 IMPLEMENTATION RESULTS

The proposed architecture has been realized in Xilinx ISE, when realizing in Xilinx, the resulting system takes 112 4-input LUTs and number of slice flipflops 83. The proposed architecture is realized by VHDL code, synthesized by using Xilinx and simulated by using Modelsim. By applying Genetic Algorithm at three levels, the optimal solutions are found for each level. Eventually, the system complexity and power consumption is reduced [4] [5].

5 CONCLUSION

Our overall design flow consists of two phases namely, system level floorplanning, and interconnection architecture generation. In the first phase, we invoke an existing flooplanner with an objective of minimizing power-performance cost function. For the second phase, we presented GA-based technique for application specific on-chip interconnection network generation.

Further, in comparison to an approach that synthesizes NoC architecture without the knowledge of a system-level floorplan our methodology can accept link length constraints, and our designs on an average consume over 36.83% lower power. Our future work will address integration of NoC design techniques with the computation architecture design stage.

REFERENCES

[1] Design of Network-on-chip Architectures with a Genetic

Algorithm-Based Technique Glenn Leary, Krishnan Srinivasan, Krishna Mehta, and Karam S. Chatha, *Member, IEEE.*

[2] W.J. Dally and B. Towles, "Route packet, not wires: On chip interconnection networks," in *Proc.* DAC, Jun. 2002, pp.684-689.
[3] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm, "*IEEE Computer,* vol.35, no. 1, pp.70-78, Jan. 2002.
[4] ST Microelectronics, Geneva, Switzerland, "ST network on-chip, "2005[online}.
[5] K. Srinivasan, K.S. Chatha, and G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures, "*IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 14, no. 4, pp. 407-420, Apr. 2006