Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture

W. Hung, C. Addo-Quaye, T. Theocharides, Y. Xie, N. Vijaykrishnan, and M. J. Irwin
Embedded & Mobile computing Design Center
The Pennsylvania State University
University Park, PA 16802, USA
{whung,addoquay,yuanxie,vijay,theochar,mji@cse.psu.edu}

Abstract

Networks-on-Chip (NoC), a new SoC paradigm, has been proposed as a solution to mitigate complex on-chip interconnect problems. NoC architecture consists of a collection of IP cores or processing elements (PEs) interconnected by on-chip switching fabrics or routers. Hardware virtualization, which maps logic processing units onto PEs, affects the power consumption of each PE and the communications among PEs. The communication among PEs affects the overall performance and router power consumption, and it depends on the placement of PEs. Therefore, the temperature distribution profile of the chip depends on the IP core virtualization and placement. In this paper, we present an IP virtualization and placement algorithm for generic regular Network on Chip (NoC) architecture. The algorithm attempts to achieve a thermal balanced design while minimizing the communication cost via placement. Our framework can also realize hardware virtualization which can further accomplish better performance. A case study on Low Density Parity Checks (LDPC) decoder is presented to evaluate our algorithm.

1. Introduction

Network-on-chip architecture has been proposed as a potential solution for the interconnect demands that arise with the nanometer era [2]. In the network-on-chip architecture, a general purpose on-chip interconnection network replaces the traditional design-specific global on-chip wiring, by the use of switching fabric or routers to connect *IP cores* or *processing elements* (*PEs*). The PEs communicate with each other by sending messages in packets, through the routers. This is usually called packet-based interconnect. An example implementation is shown in Figure 1. The diagram on the left illustrates a tiled single PE per node architecture arranged in a 2-D mesh network, where as the diagram on the right illustrates the underlying interconnect, where each router communicates to the PE via a local port, and to its neighboring routers via four global ports.

A particular property of Network-on-chip architecture is the concept of *hardware virtualization*, which maps one or more logical processing units onto a single PE, thus allowing the PE to virtually perform the computation of one or more (depending on the degree of virtualization) logical processing units. This method requires the presence of additional logic and memory as part of the PE hardware in order to identify the particular computation that is performed at any given time, and to read/write the required data per computation. It is possible that computations on virtualized PEs happen out of order; as a result synchronization mechanisms are implemented within the PEs in order to allow for out of order computation, as well as to synchronize the completed computations with the rest of the PEs. This attributes to a larger overall PE which consequently consumes more power; however it also increases the amount of logical units that can be mapped on the chip, which depending on the computation type, can result in computation performance gain [1].

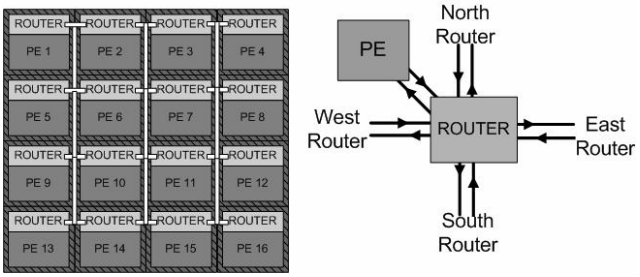

Figure 1. Left – an NoC Layout showing the physical placement of the PEs and routers. Right – an individual NoC router shown in detail with, the connections to the rest of the routers.

After the hardware virtualization (i.e., mapping of the computation onto a single PE), the energy consumption and the computation time for each PE is fixed. However, the communication delay between PEs depends on the PE locations. For example, message from PE1 has to go through at least seven routers to reach PE16 in Figure 1

(which means it needs six hops). On the other hand, the communication energy consumption depends on the location of the PE as well as the message volume sent through routers and the distance between PEs. Consequently, the performance and the overall energy consumption are determined by the IP placement. If two PEs exchange a lot of packets, it is better to place them closer to reduce both the communication energy and latency.

As the technology scales, the temperature in modern high-performance VLSI circuit increases dramatically due to smaller feature size, higher packing density and rising power consumption. The hotspot in a modern chip might have a temperature of more than 100^{o}C, while the intra-chip temperature differentials can be larger than 10~20^{o}C. Temperature can have dramatic impacts on circuit behavior. For example, interconnect (Elmore) delay increases approximately 5% for every 10^{o}C increase, and the leakage current increases exponentially with the temperature increase. Therefore, it is very important to reduce or eliminate hotspots and have a thermally balanced design. For NoC architecture, the thermal distribution profile of a design is largely determined by the PE locations, consequently, the IP placement algorithm is the key to achieve the thermal balance design goals.

In this paper, we present a thermal-aware IP virtualization and placement algorithm based on genetic algorithm. We demonstrate that a careful IP virtualization and placement can reduce the hotspots temperature and provide a thermally balanced design.

This paper is organized as follows: section 2 presents related work; section 3 discusses how to estimate temperature; section 4 gives a brief background on genetic algorithms; section 5 presents our mapping framework based on genetic algorithm; section 6 presents the case study on LDPC with the experimental result and finally we conclude the paper in section 7.

2. Related work

One of the major challenges for a successful adoption of the network-on-chip paradigm is in reducing the energy consumed during the interactions between the IPs (PEs) [2]. Hu and Marculescu [8, 9] proposed an energy-aware mapping algorithm which minimizes the total communication cost for a 2-D mesh NoC architecture under real-time performance constraints. Murali *et al.* [14] proposed an algorithm that maps IP cores onto a mesh NoC architecture under bandwidth constraints, minimizing the average communication delay. The potential bandwidth requirements were reduced by the partitioning of inter-core traffic across multiple paths. Al-Rawi *et al.* [1] also explored an optimal mapping on a set of LDPC nodes to physical computation units, with the purpose of minimizing the communication between the nodes.

Thermal placement for standard cell ASIC design has been investigated for several years. For example, Chu and Wong used matrix synthesis problem (MSP) to model the thermal placement problem and three algorithms were proposed to solve it [4]. Tsai and Kang also proposed a method [18] to calculate temperature based on power estimation for standard cell placement. Thermal placement can also be refined by partitioning; for example, Chen *et al.* proposed a partition-driven thermal placement model [3] for standard cells, making use of multigrid-like approach to simplify the thermal problem at each level of finer granularity, facilitating the inclusion of temperature constraints on the placement. For standard cell thermal placement, since the physical interconnect parasitic information is not available yet, the power consumption of interconnect is usually ignored. While in NoC architecture, the communication cost (latency and power consumption) can be affected by the IP placement.

The contribution of our work is that we map the IP virtualization and placement problem in NoC into genetic algorithm. By modifying the fitness function, we can achieve different design goals. We compare different optimization strategies (power-balanced placement, thermal-balanced placement as well as communication cost minimization placement), and the experimental result indicates that with the thermal-balanced placement, we can achieve the best thermal distribution profile for the NoC architecture.

3. Temperature estimation

The temperature of each IP block depends on the power consumption and the position of the IP blocks. Skadron *et al.* [17] proposed a thermal modeling tool called HotSpot, which is easy to use and computationally efficient for modeling thermal effects at the IP block level. HotSpot provides a simple compact model, where the heat dissipation within each functional block and the heat flow among blocks are accounted for. The basic idea is that, if we define the transfer thermal resistance R_{ij} of IP block PE_i with respect to PE_j as the temperature rise at PE_i due to one unit of power dissipated at PE_j:

$$R_{ij} = \Delta T_{ij}/\Delta P_j$$

such that we can get a transfer thermal resistance matrix as below:

$$R^t = \begin{bmatrix} R^t_{11} & R^t_{12} & \cdots & R^t_{1m} \\ R^t_{21} & R^t_{22} & \cdots & R^t_{2m} \\ \vdots & & \ddots & \vdots \\ R^t_{m1} & R^t_{m2} & \cdots & R^t_{mm} \end{bmatrix}$$

For any power distribution on the NoC architecture, we can calculate each block's temperature by applying the following equation:

$$\begin{bmatrix} T_1 \\ T_1 \\ \vdots \\ T_m \end{bmatrix} = \begin{bmatrix} R^t_{11} & R^t_{12} & \cdots & R^t_{1m} \\ R^t_{21} & R^t_{22} & \cdots & R^t_{2m} \\ \vdots & \ddots & & \vdots \\ R^t_{m1} & R^t_{m2} & \cdots & R^t_{mm} \end{bmatrix} \begin{bmatrix} P_1 \\ P_1 \\ \vdots \\ P_m \end{bmatrix}$$

where P_i is the power consumed by IP block PE_i and T_i is the temperature of the IP block PE_i. The transfer thermal resistance matrix can be obtained from Hotspot, given the IP block placement.

4. Genetic algorithm

Genetic algorithms (GA) [7] are a class of search and optimization methods that mimic the evolutionary principles in natural selection. Figure 2 shows a genetic algorithm optimization flow.

The solution is usually encoded into a binary string called *chromosome*. Instead of working with a single solution, the search begins with a random set of chromosomes called *initial population*. Each chromosome is assigned a *fitness score* that is directly related to the *objective function* of the optimization problem.

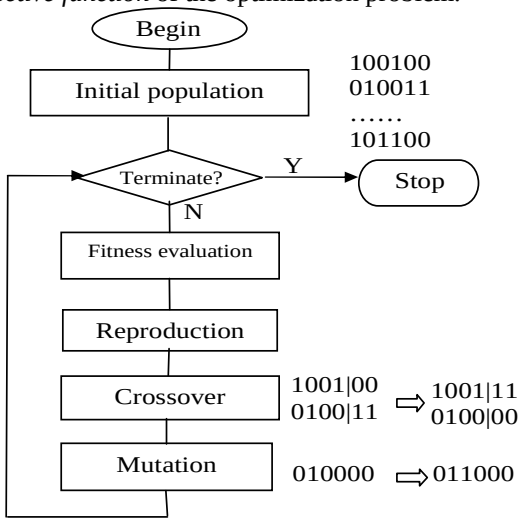

Figure 2. Genetic algorithm flow.

The population of chromosomes is modified to a new *generation* by applying three *operators* similar to natural selection operators – *reproduction*, *crossover* and *mutation*. Reproduction selects good chromosomes based on the fitness function and duplicates them. Crossover picks two chromosomes randomly and some portions of the chromosomes are exchanged with a probability P_c. Finally, mutation operator changes a 1 to a 0 and vice versa with a small mutation probability P_m. A genetic algorithm successively applies these three operators in each generation until a termination criterion is met. It can very effectively search a large solution space while ignoring regions of the space that are not useful. This methodology leads to very time-efficient searches. In general, a genetic algorithm has the following steps:

1. Generation of initial population.
2. Fitness function evaluation.
3. Selection of chromosome.
4. Reproduction, Crossover, Mutation operations.

5. IP virtualization and placement framework

The proposed NoC mapping optimization flow uses a genetic algorithm as shown in Figure 2. The IP virtualization and placement information is encoded into integer strings called chromosomes. The optimization flow begins with a randomly generated *initial population*, which consists of many randomly generated IP placements. The optimization flow is an iterative procedure. The chromosomes with better fitness will survive at each generation and are operated on with three different operations (reproduction, crossover and mutation) to form a new set of chromosomes – or new IP virtualization and placement. The iteration continues until the termination criterion is met.

5.1 Chromosome encoding

One example of the chromosome encoding is shown in Figure 3. It contains 16 unique integers, which represents the 16 IP cores (physical PEs). The position of each integer indicates its placement location. For example, chromosome A in Figure 3 represents the placement as shown in Figure 1, while chromosome B represents the placement where the PE 1 and PE 6 in Figure 1 are swapped. With virtualization, logical processing units 22, 73, 19 and 31 are clustered into physical PE 4 while logical processing unit 44, 37, 56 and 85 are grouped together in physical PE 11 in chromosome A. With this representation, the optimizations of mapping and virtualization can be done simultaneously.

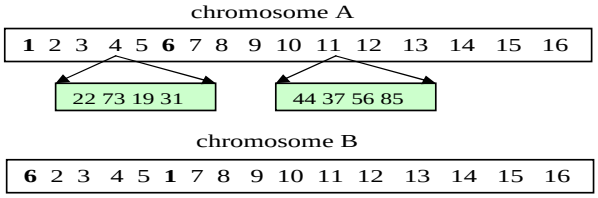

Figure 3. Chromosome encoding for IP placement.

5.2 Fitness function

The fitness function, which decides the survival chance for a specific chromosome, is related to the mapping goals. Depending on the optimization goal, the fitness function of the genetic algorithm is different.

5.2.1 Thermal balanced design

The goal of thermal placement is to distribute temperature across a chip evenly and minimize the hotspot temperature. We use an approach similar to that proposed by Chu, *et al.* [4]. Their work tried to solve the problem of thermal placement for gate arrays. Here, we model our NoC architecture as an ***m*** *x* ***n*** matrix with the given temperatures such that the maximum sum among all ***t*** *x* ***t*** submatrices is minimized. The number assigned to each cell in the matrix is a nonnegative temperature value. The parameter ***t*** is used to account for the heat transfer ability. Increasing ***t*** means that the heat transfer is good, so the number of affected cells near the heat source cell will be larger. For any matrix ***M***, let $S_t(M)$ be the set of all ***t*** x ***t*** submatrices, the fitness of a solution can be defined as follow:

$$Fitness = \frac{1}{Max(S_t)}$$

5.2.2 Power balanced design

This optimization strategy is to achieve a balanced power distribution on the chip. Basically, we use the same approach to calculate the fitness of a chromosome here as in thermal balanced design. The only difference is the numbers in the matrix. For power-balanced design, we provide the power consumption of each IP and its corresponding router. The power consumption for the IP is fixed when the hardware virtualization is done. However, the router power consumption depends on the communication links. The temperature in one IP location will not be equal to that of another IP location, since the heat can flow to the adjacent IPs. But power consumption of each IP will remain the same even when placed in different physical locations. Thus, we can observe the different temperature results from Power and Thermal balanced designs.

5.2.3 Communication cost minimization design

The communication cost is given by:

$$Commcst = \sum_{i=1}^{|E|} vol(d^i) * dist(source(d^i), dest(d^i))$$

where the d^i is any communication between two IPs (where the $source(d^i)$ is the source IP core and the $dest(d^i)$ is the destination IP core), $vol(d^i)$ is the message volume that has to flow between these two IP cores. The *dist* is the number of hops that the messages have to go through. To minimize the communication cost, the fitness function of our algorithm is given by:

$$Fitness = \frac{1}{Commcst}$$

5.3 Crossover operator and mutation operator

Due to the nature of Genetic Algorithms, the number in a chromosome will be generated randomly. For our placement problem, we map each physical IP as a nonnegative unique number as the encoding method. When doing crossover operation, the offspring's chromosome is generated from mating of parents' chromosomes. At this stage, there are possibilities that some numbers are redundant. It's important to guarantee that each number should only exist once in a chromosome to make the evolution proceeds. As for mutation operation, we have two different operators to explore more solution spaces. One is the mutation by swapping and another is mutation by shifting. Either operator can be used with a random probability.

5.4 Control parameters

While generating the initial population, we have to set an appropriate population size, and the crossover probability P_c, as well as the mutation probability P_m. If the population size is too small, the genetic diversity within the population may not increase for many generations. On the other hand, a large population size increases the computation time for each generation but it may take fewer generations to find the best solution. Schaffer *et al.* [16] have conducted extensive simulation on a wide range of functions and concluded that a small population of size 20 to 30, a crossover probability in the range of 0.75 to 0.95, and a mutation probability in the range of 0.005 to 0.01 perform very well. In our implementation, we set the population size to be 30~35, crossover probability P_c to be 0.9 and the mutation probability P_m to be 0.01.

The termination of the iterative evolution can be user-defined. We set a maximum generation to be 5000 and specify that if the fitness improvement is less than 0.001% during the last 100 generations, the evolution stops without going through all generations.

6. Case study on LDPC and experimental results

In this section, we present a case study of implementing Low Density Parity Check (LDPC) decoder on networks-on-chip architecture, and evaluate our algorithm by using this real application.

6.1 A brief introduction on LDPC

Low Density Parity Check (LDPC) codes are a form of iterative error correction codes similar to Turbo codes, that can achieve near Shannon-limit communication channel capacity [6,12]. They offer excellent decoding performance and good block error performance. The most notable advantage of LDPC codes is their suitability for parallel hardware implementation. An LDPC code is a linear message encoding technique, defined by a set of two very sparse parity check matrices, G and H. The message to be sent is encoded using the G matrix. When it reaches its destination, it is decoded using the H-matrix. The LDPC decoding algorithm consists of a series of intensive computations derived from a message-passing iterative bipartite graph, as shown in Figure 4. The bipartite graph consists of two types of nodes, the bit node, and the check node [6]. Connections between the two nodes in the bipartite graph depend on the row and column weight of the H-Matrix, where the weight is the number of 1-entries in the row/column. Columns represent the number of bit nodes and rows represent the number of check nodes. A 1 in the ij^{th} entry of the H-Matrix represents an edge between the i^{th} check and the j^{th} bit nodes as shown in Figure 4.

Message passing iterations are performed by the two computation units - the bit node and the check node [12]. Each type of node interacts with a number of other nodes, all of the opposite type, to decode a word. The number of nodes involved in the computation depends on the desired block size.

The H-matrix is usually sparse, and needs to be large, in order to decode large blocks of data; that consequently implies a relatively large amount of edges in the bipartite graph [12]. Hence, the two major challenges identified when designing LDPC decoders are the interconnect structure between the nodes, and the amount of memory required for computation as well as configuration purposes per node [19].

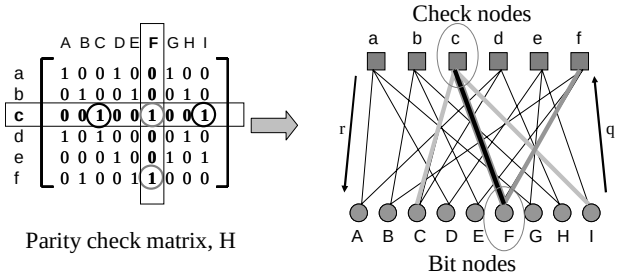

Figure 4: Derivation of the Bipartite Graph from the H-Matrix.

For the reasons explained, we therefore use the LDPC decoder as an example to evaluate our mapping algorithms for NoC architecture. The underlying interconnections between the LDPC nodes are implemented as a NoC architecture as described in Section 1. We explore the mapping of the check and bit nodes on the NoC architecture, in such a way to obtain an even temperature distribution and reduce the overall communication on-chip. It has been shown [11] that while architectural modifications in the PE can reduce the overall power consumption; most of the chip power is consumed in the communication links and the routers which are constantly active. In addition, the LDPC nodes consume different amounts of power because of the variations in the number of connections per node [12], hence the thermal distribution problem. As a result, an emphasis is also placed on reducing both the number of hops (defined as a message transmission between two on-chip routers) as well as the number of messages transmitted overall. The mapping and placement problems are magnified by the hardware constraints in terms of area however. Initially we have the H-matrix, which provides the total number of nodes, as well as the connections between them and their degrees (# of inputs). Based on the area constraints, we then have a limited number of processing units that we can place on the chip. The problem therefore becomes mapping the pool of computation nodes given by the H-Matrix, into the limited area we have on the chip. The overall computation can be enhanced by hardware virtualization; this is again a technique that thrives from the NoC architecture, as described in Section 1. Using hardware virtualization, we can have a physical PE function as two or more logical PEs (depending on the virtualization factor). The tradeoff comes in the PE area and power consumption, which increase by a small factor to incorporate the extra memory and logic required for node identification.

An optimal mapping of the LDPC nodes into the physical PEs of the NoC architecture provides potential reduction both in the communication (# of hops) between the PEs as well as the number of messages transmitted. The proposed NoC architecture provides a fast and reliable underlying structure, allowing the bit and check nodes to effectively communicate with each other.

6.2 Design methodology

In order to obtain the power models for each PE, we used the following tool flow. Firstly, the LDPC Software [12] was used in order to generate both the H-Matrix and the encoded messages. We used three types of LDPC codes – a (7, 4) Hamming code, a (2000, 1000) LDPC code with three checks per bit and six bits per check, and a (10000, 5000) LDPC code with three checks per bit and six bits per check. For the (7, 4) Hamming code, we transmitted the message using a Binary Symmetric Channel (with error probability 5%) and for all three codes, we transmitted the messages using Additive White Gaussian Noise channels, with noise standard deviation of 5%. Once the bipartite graph was obtain, we then used NOCSim [12] to set up either a *4x4* or

5x5 2-D mesh network, with different physical bit nodes and check nodes and an I/O communication-oriented node.

NOCSim sets up a predefined network of PE's and routers, and in addition provides mechanisms to handle virtualized nodes. NOCSim takes as input the Network topology defined by the H-Matrix and the encoded message to be decoded. It then simulates the Network, generating real network traffic by packetizing data and headers and simulating cycle-accurate data transmission between routers and PE's. NOCSim generates the entire network traffic between physical nodes, taking into consideration potential mapping of more than one virtual node on a physical PE (virtualization). NOCSim outputs include the number of messages from PE to PE and the routing path they follow, in the form of the real binary data that travels across the network. In parallel with NOCSim simulations, we created and synthesized in commercial 160nm technology using Synopsys Design Compiler, Verilog models of each PE as well as the entire underlying NoC architecture.These NOCSim outputs were then used as our test vectors for the synthesized models of the physical PE's, and Synopsys Power Compiler was used to give the power models of the individual PE's. The operating clock frequency was at 500MHz, with Vdd of 1.8V. Figure 5 shows the overall modeling methodology.

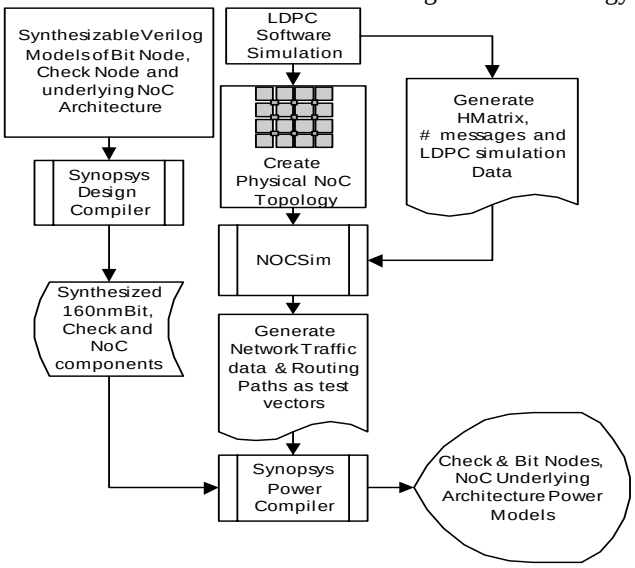

Figure 5. The overall design methodology for NoC LDPC Power Estimation.

In the genetic algorithm, we use an intelligent crossover mechanism which guarantees that bit nodes are always swapped with bit nodes only and similarly for check nodes. Also in the case of virtualization logical PE node is decoupled from its physical PE mapping in the previous generation during crossover and mutation i.e. the physical to logical mapping is not static.

6.3 Experimental results

First we implement a *10x10* NoC architecture with 100 PEs on the chip to choose the window parameter value ***t***. Each PE has a size of 1 mm *x* 0.8 mm in a commercial 160nm standard cell library. Based on the physical layout, we use Hotspot to obtain the transfer thermal resistance matrix and estimate the temperature for each IP block, as described in section 3.

Figures 6 and Figure 7 show the peak and average temperature respectively by using two different windows (*1x1* window and *2x2* window, as defined in section 5.2). From the figures we can see that *1x1* window performs better in both, reducing peak and average temperatures. For the following experiments, we use ***t*** =1 to obtain better results. The IP virtualization and mapping algorithm is implemented in C and the experiment is done on an Intel Pentium 4 processor machine (2.8 GHz 512M RAM) running Linux, the runtime is about 15~18 minutes for 3000 generations in *10x10* NoC example.

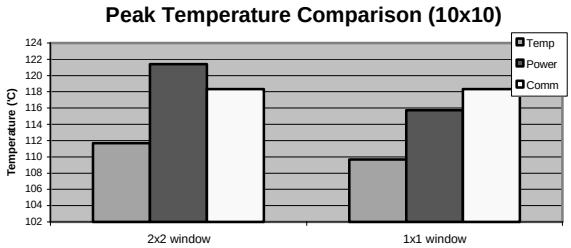

Figure 6. Peak temperature comparison of 1x1 and 2x2 window schemes.

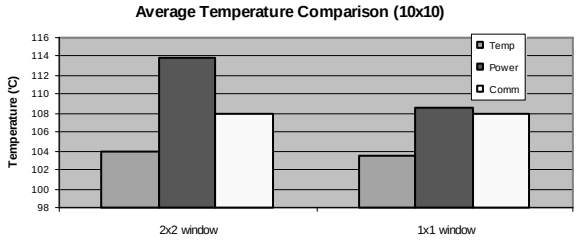

Figure 7. Average temperature comparison of 1x1 and 2x2 window schemes.

6.3.1 Thermal-aware IP placement

The first experiment we conduct is to do the hardware virtualization manually using a custom algorithm that sequentially places virtual nodes to same physical PE and use the proposed algorithm to do thermal-aware IP placement. Table 1 shows the set up data for the LDPC nodes used in our experiment. The degree of a node is the number of outgoing connection edges to complementary nodes. Each node in the setup has two different degree values. The virtualization is done through manual

assignments before the mapping process.

Figures 8, 9 and 10 show the comparisons for 5 different sets of LDPC codes implemented on a *4x4* NoC.

Table 1. LDPC Nodes Configuration Profile

Data Set	Bit Node Degrees	Check Node Degrees	Volume
set 1	3,4	6,8	16087000
set 2	3,4	8,10	13315840
set 3	3,4	8,12	14128000
set 4	5,6	6,12	20591360
set 5	3,9	6,9	16473088

Figure 8 shows the average temperature for the NoC chip. We can see that the temperature optimization approach performs better than the others. The running times for these experiments are 12~15 seconds for 1000 generations. Beyond 1000 generation, there is little improvement.

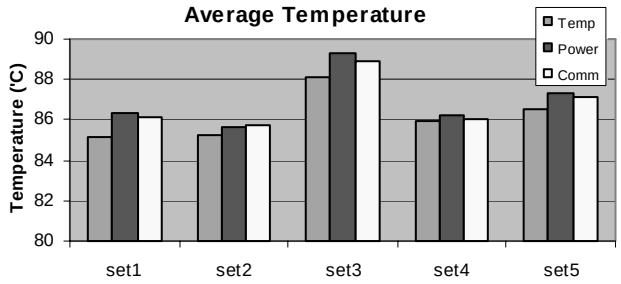

Figure 8. Average temperature for a 4x4 LDPC decoder with 5 sets of hardware virtualization under different optimization strategies.

Figure 9 shows the peak temperature, or hotspot temperature. It shows that among all three optimization strategies, the temperature optimization strategy results in the lowest peak temperature, with the average difference of 4°C for all the sets of optimization approaches.

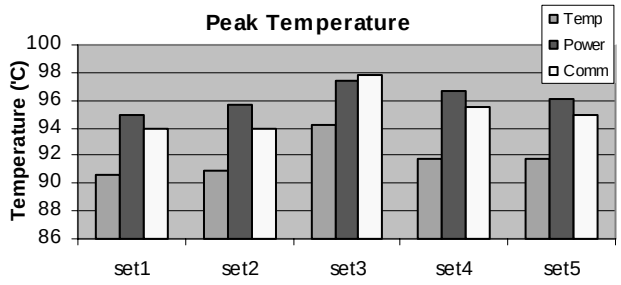

Figure 9. Peak temperature comparison for a 4x4 LDPC decoder with 5 sets of hardware virtualization under different optimization strategies.

Figure 10 shows the communication cost for different optimization strategies. It is obvious that the communication cost minimization strategy is the best choice in a communication critical environment.

The conclusion we draw from our experimental result for thermal-aware mapping with manual hardware virtualization is that, to reduce the temperature of the hotspot, we should use the thermal balanced optimization strategy.

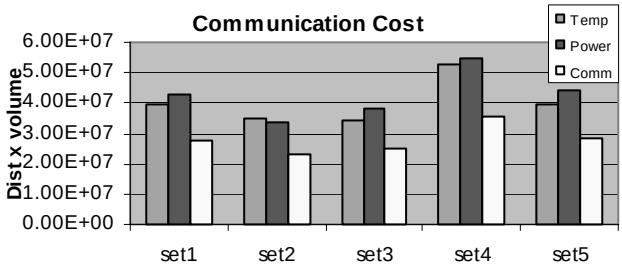

Figure 10. Communication cost comparison for a 4x4 LDPC decoder under different optimization strategies.

6.3.2 Thermal-aware IP virtualization and placement

Using five different LDPC codes which had different bit/check node connectivities, we performed a comparison of simulataneously performing virtualization and mapping (denoted Virtualized) is shown in Figure 11. As shown in the figure the Temp. approach still outperforms the other two approaches in both average and peak temperature. This confirms the results shown in Figures 8 and 9.

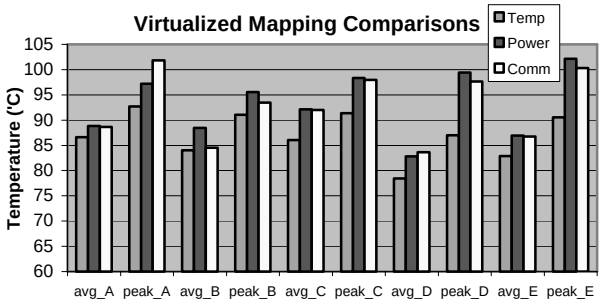

Figure 11. The experimental result for virtualized mapping of 960 virtual nodes of a LDPC decoder.

Figure 12 shows the comparisons of our custom virtualization followed by our genetic algorithm based placement approach (denoted non-virtualized) and virtualized approach for temperature optimization. The number beside the name of a data set is the size of the IP array required for this mapping configuration. The effectiveness of performing simultaneous virtualization and placement is established here as it reduces both the peak and average temperature by 2~3°C as compared to the non-virtualized approach. The reason of virtualized mapping being better than non-virtualized mapping is that we consider the connection of virtual nodes at a finer granularity that can place closely related nodes onto the same physical PE to further reduce the number of communication links; as a consequence, achieve further temperature reduction.

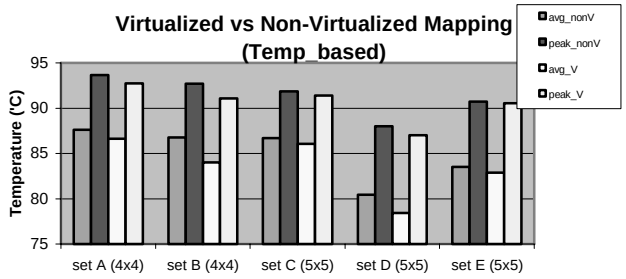

Figure 12. The comparisons of non-virtualized and virtualized mapping under temp-based approach.

The communication-based comparison of virtualized mapping and non-virtualized mapping is shown in Figure 13. We can see that the virtualized approach has about 10% of the average communication cost reduction.

The runtime for the experiments of doing virtualization and placement concurrently is about 9~13 minutes for 5000 generations.

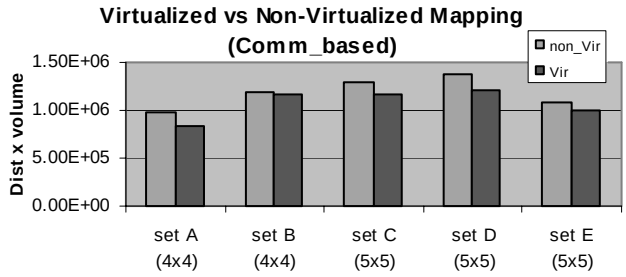

Figure 13. The comparison of communication cost for virtualized mapping and non- virtualized mapping.

7. Conclusion and future work

In this paper, we present a thermal-aware IP placement algorithm based on a Genetic Algorithm. Our experimental results show that our optimization strategy can reduce the hotspot temperature and achieve a thermally balanced design. By changing the fitness function, the IP placement algorithm can also be applied to minimize the communication cost of the NoC, such that the communication latency and energy is reduced. Incorporating virtualization into the scheme, our results showed that a simultaneous virtualization and placement optimization is better than sequential virtualization and placement. Currently, our IP placement framework works for tile-based network-on-chip architecture; we will extend our work in the future for irregular Networks-on-chip architecture.

Acknowledgments

This work was supported in part by a MARCO/DARPA GSRC Award, NSF Awards CAREER 0093085 and 0130143.

References

[1] G. Al-Rawi, J. Cioffi and M. Horowitz., "*Optimizing the mapping of LDPC Codes on parallel decoding architectures*", Proceedings of the IEEE ITCC, 2001. pp. 578-586.

[2] L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm, *IEEE Computer*, Volume 35, pp. 70--78, January 2002.

[3] Guoqiang Chen and Sachin Sapatnekar, "*Partition-Driven Standard Cell Thermal Placement*", ISPD 2003.

[4] C.N. Chu and D.F. Wong, "*Matrix Synthesis Approach to Thermal Placement*", Proc. Int. Sym. On Physical Design, pp.163-168, 1997.

[5] W. J. Dally, B. Towles, "*Route packets, not wires: on-chip interconnection networks*," Proc. DAC, pp. 684–689, June 2001.

[6] R. G. Gallager, "*Low-Density Parity-Check Codes*", IEEE Transactions on Information Theory, Jan. 1962, pp. 21-28.

[7] Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*, New York: Addison-Wesley. 1989.

[8] J. Hu and R. Marculescu, "*Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures*", Proceedings of DATE 2003, February 2003.

[9] J.Hu, R.Marculescu., "*Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints*", ASP-DAC 2003.

[10] B. Levine et al., "*Implementation of Near Shannon Limit Error-Correcting Codes using Reconfigurable Hardware*", Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines,2000. Page(s):217-226.

[11] T. Theocharides, G. Link et.al., "*Evaluating Alternative Implementations for the LDPC Check Node Function*", Proceedings of the IEEE International Symposium on VLSI, February 2004.

[12] D. Mackay R. Neal, "*Near Shannon limit performance of low density parity check codes*", IEE Electronics Letters, Vol.33, no. 6, March 1997, pp 457-458.

[13] M. M. Mansour and N. R. Shanbag, "*Low-Power VLSI Decoder Architectures for LDPC Codes*", Proc. of the 2002 International Symposium on Low Power Electronics and Design, Page(s): 284-289.

[14] S. Murali and G. De Micheli, "*Bandwidth-Constrained Mapping of Cores onto NoC Architectures*", Proceedings of DATE'04, February 2004.

[15] K. Skadron, T. Abdelzaher, and M. Stan, "*Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management*", In Proceedings of the Eighth International Symposium on High-Performance Computer Architecture, pp. 17-28, Feb. 2002.

[16] J.Schaffer, J. Caruana, L. Eshelman and R.Das, "*A study of control parameters affecting online performance of genetic algorithms for function optimization*," Proc. of the Third international Conference on Genetic Algorithms, pp.51-60, 1989

[17] Kevin Skadron, Mircea R. Stan, Wei Huang, et al., "*Temperature-Aware Microarchitecture*", ISCA 2003.

[18] Ching-Han Tsai and Sung-Mo Kang, "*Stand Cell Placement for Even On-Chip Thermal Distribution*", ISPD 1999.

[19] E. Yeo, B. Nikolic and V. Anantharam, "*Architectures and Implementations of Low-Density Parity-Check Decoding Algorithms*", invited paper at IEEE International Midwest Symposium on Circuits and Systems, Aug 4-7, 2002.

[20] T. Zhang et al, "*On Finite Precision Implementation of Low Density Parity Check Codes Decoder*", Proc. Of the 2001 IEEE International Symposium on Circuits and Systems.