Copyright © 2006 American Scientific Publishers
All rights reserved
Printed in the United States of America

Journal of
Low Power Electronics
Vol. 2, 1–12, 2006

Communication Power Optimization for Network-on-Chip Architectures

Dongkun Shin[1] and Jihong Kim[2,*]

[1] *Samsung Electronics Co., Korea*
[2] *School of Computer Science and Engineering, Seoul National University, Korea*

(Received: 15 February 2006; Accepted: 1 June 2006)

Network-on-Chip (NoC) architecture is emerging as a practical interconnection architecture for future systems-on-chip products. In this paper, an energy-efficient static algorithm which optimizes the energy consumption of task communications in NoCs with voltage scalable links is proposed. In order to find optimal link speeds, the proposed algorithm (based on a genetic formulation) globally explores the design space of NoC-based systems, including network topology, task assignment, tile mapping, routing path allocation, task scheduling, and link speed assignment. The experimental results demonstrate that the proposed design technique can reduce energy consumption by an average of 28% over existing techniques.

Keywords: Network-On-Chip, Dynamic Voltage Scaling, Real-Time Systems, Low-Power Design.

1. INTRODUCTION

Network-on-Chip (NoC) architecture has recently been proposed as a practical interconnection architecture for systems-on-chip (SoC) products.[1,2] NoCs are especially useful in overcoming complex on-chip communication problems, by providing a more structured and modular network interface. Network electrical parameters can be well controlled and optimized, since networks are structured and wired beforehand, making it possible to use aggressive signaling circuits, therefore significantly reducing power dissipation and propagation delay. A standard interface between modules facilitates reusability and interoperability.

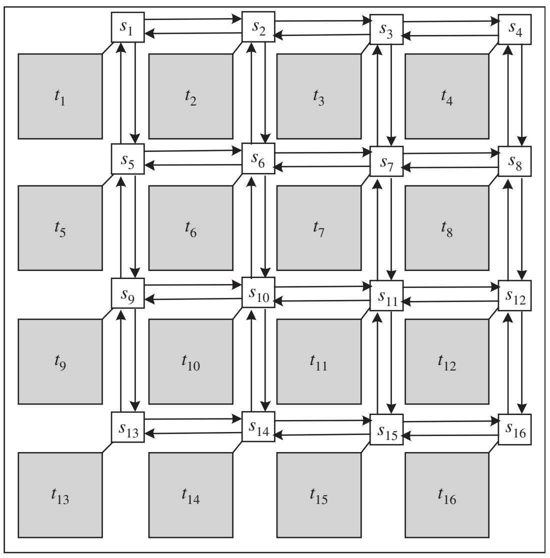
As presented in Figure 1(a), an NoC-based system is typically divided into regular tiles, where each tile might be a programmable microprocessor, an ASIC, or a FPGA. Instead of being connected by dedicated wires, each of these tiles is connected to an interconnection network that routes packets between tiles. As presented in Figure 1(b), the router in NoCs consists of input and output links, buffers and a crossbar switch.

In NoC-based systems, on-chip networks consume a substantial portion of system power budget. For example, the on-chip network of the MIT Raw microprocessor, which has 16 tiles, consumes 36% of the total chip power.[3] In the Alpha 21364 processor, 20% of the total chip power is consumed by the router and links.[4]

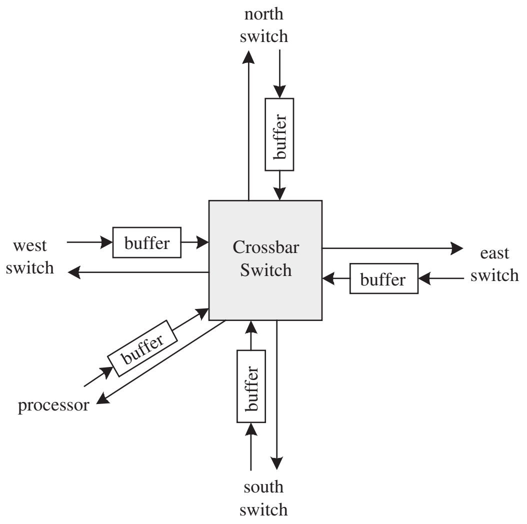
A promising low-power technique for energy-efficient NoCs is to scale the speeds of the communication links with the corresponding voltage level.[5] As with CPU dynamic voltage scaling, the link communication energy has a quadratic dependency on link speed. By adjusting the link speed based on identified idle communication intervals, a significant amount of communication energy can be saved without significant degradation in communication performance.

Two kinds of speed scaling techniques exist. The first is an *on-line* scheme, which adjusts the communication speed dynamically, based on variations in the run-time communication traffic. The other is an *off-line* scheme, which assigns an appropriate fixed communication speed to each link *statically*, based on the communication patterns of target applications. The *off-line* scheme is better suited to real-time applications, since system designers are able to predict communication delays at design time. In addition, the *off-line* scheme does not result in run-time overhead, which is indispensable to the *on-line* scheme. There are two kinds of run time overheads in the *on-line* scheme. The first is the communication traffic monitoring overhead required to determine an appropriate link speed. The second is the link voltage scaling overhead required for adjusting the operating voltage. For example, based on the multi-level DVS link model which supports ten discrete frequency levels and corresponding voltage levels

*Author to whom correspondence should be addressed.
Email: jihong@davinci.snu.ac.kr


s_1 s_2 s_3 s_4
t_1 t_2 t_3 t_4
s_5 s_6 s_7 s_8
t_5 t_6 t_7 t_8
s_9 s_{10} s_{11} s_{12}
t_9 t_{10} t_{11} t_{12}
s_{13} s_{14} s_{15} s_{16}
t_{13} t_{14} t_{15} t_{16}

(a) NoC-based system with 16 tiles



(b) Structure of router

Fig. 1. Architectural overview of NoC-based system.

in Ref. [5], the latency of voltage (frequency) transition between adjacent levels is set to 10 μs (100 link clock cycles).

In this paper, an *off-line* link speed assignment algorithm for energy-efficient NoCs with scalable voltage links is proposed. Given the task graph of a periodic real-time application, the proposed algorithm assigns an appropriate communication speed to each link, minimizing the energy consumption of NoC-based systems while guaranteeing the timing constraints of the real-time application. In addition, the proposed algorithm turns off links *statically*, when no communications are scheduled, so that the leakage power of an interconnection network can be saved. (The leakage power consumption can be a significant power consumer. For example, 21% of the total power consumption in 0.07 μm technology is caused by leakage current.[6])

As with other multiprocessor-based systems, the design flow of NoC-based systems involves several (interacting) steps, of which link speed assignment is the final step. In a typical multiprocessor system, the design flow includes two key steps, *task assignment* and *task scheduling*. Given a task graph with design constraints (e.g., the execution time and the power consumption) and processing elements (PEs), each task is first assigned to an appropriate PE (*task assignment*). Then, each task is scheduled for execution within the PE (*task scheduling*). However, in NoC-based systems, two additional steps are necessary, *tile mapping* and *routing path allocation*. The tile mapping step maps a PE to one of the tiles in an NoC-based system. The routing path allocation step determines communication paths between tiles. For example, if an NoC-based system has sixteen PEs, as presented in Figure 1(a), it is required to decide which tile each PE will be located. If data has to be transferred from tile t_1 to tile t_{16}, then the switches among $s_1, \ldots, s_{16}$ that forward the data, must be determined. (In this paper, the term *network assignment* is used to refer to both the tile mapping and routing path allocation steps.)

In an NoC-based system, design decisions made in the network assignment step, as well as the task assignment and the task scheduling steps, can significantly affect the communication speed of each link, because communication traffic patterns vary according to the result of the design steps. Therefore, in order to make an NoC-based system energy-efficient, each step should be taken with the awareness of its implication for energy consumption in links.

For example, consider the task graph G presented in Figure 2(a) with a period set to 400 time units. In this example, the tasks τ_1, τ_2, τ_3, and τ_4 are assumed to be

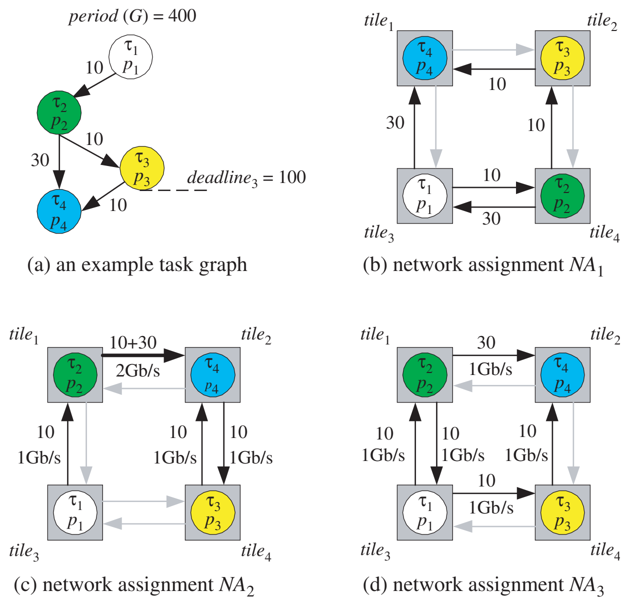


Fig. 2. Motivational example.

assigned to the PEs p_1, p_2, p_3, and p_4, respectively. It is assumed that a target SoC has four tiles with voltage-scalable links. Each link can be switched to one of two data transfer rates, 1 Gb/s or 2 Gb/s. A tile mapping algorithm may generate the network assignment presented in Figure 2(b), where each number on a link indicates the corresponding communication cost, i.e., the total amount of data transfers for the link.

Assuming that the routing paths are allocated by the XY-routing algorithm,[7] packets are first routed along the X-axis. When a packet reaches the column under which the destination tile is located, it is then routed along the Y-axis. Therefore, the communication cost of the network assignment NA_1 is 90. However, if tile mapping is changed to the network assignment NA_2 (as presented in Fig. 2(c)), the communication cost is reduced to 70. Therefore, it can be said that the network assignment NA_2 has greater energy-efficiency than NA_1 because NA_2 requires less energy to satisfy the application's communication requirements.

Now let us consider how to assign the communication speed of each link in the network assignment NA_2. If the task τ_3 has a hard deadline, then the link from $tile_1$ to $tile_2$ should have a high speed (2 Gb/s) connection, this is because the link is on the critical path; therefore, it must transfer the data between τ_2 and τ_4 as well as between τ_2 and τ_3. This suggests that the network assignment NA_2 could be improved so that the critical path does not share links with non-critical paths. Therefore, routing path allocation should be performed with awareness of its effect on link speed scaling.

If the routing path of the edge (τ_2, τ_3) in NA_2 is changed from $tile_1 \rightarrow tile_2 \rightarrow tile_4$ to $tile_1 \rightarrow tile_3 \rightarrow tile_4$, the network assignment NA_3 is obtained, as presented in Figure 2(d). Although the network schedule NA_3 has the same amount of communication traffic as the network assignment NA_2, a lower speed (1 Gb/s) can now be assigned to the link from $tile_1$ to $tile_2$. If it is assumed that the communication link dissipates 10 mW at 1 Gb/s, but 40 mW at 2 Gb/s, by assigning the lower speed to the link from $tile_1$ to $tile_2$ (even though the link from $tile_3$ to $tile_4$ is now activated), the overall communication energy is reduced by 36% over NA_2.

$$\begin{aligned}(\text{Energy } (\text{NA}_2) &= 40 \text{ mW} \times \frac{40}{2 \text{ Gb/s}} + 10 \text{ mW} \times \frac{30}{1 \text{ Gb/s}} \\ &= 1100 \text{ mW} \\ \text{Energy } (\text{NA}_3) &= 10 \text{ mW} \times \frac{70}{1 \text{ Gb/s}} = 700 \text{ mW})\end{aligned}$$

In this paper, it is demonstrated that the existing design algorithm for NoC-based systems is inappropriate for systems with voltage scalable links, and a novel optimization algorithm (based on a genetic formulation) is proposed, which explores the overall design space efficiently. The experimental results demonstrate that the proposed design algorithm can reduce energy consumption by an average of 28% compared with the existing algorithm.

The subsequent sections of this paper are organized as follows. In Section 2, the related work on NoC-based design techniques is briefly reviewed. The overall design flow and problem formulation is presented in Section 3. The detailed design techniques are described in Section 4. Experimental results are presented in Section 5. Section 6 concludes with a summary and directions for future work.

2. RELATED WORKS

Several research groups have investigated design techniques for minimizing the energy consumption in NoC-based systems. For example, Simunic and Boyd[8] proposed a power management technique for NoC-based systems. Based on a network-centric power management scheme, the proposed technique makes better predictions of future workload than techniques based on a node-centric power management approach. While this work focused on PEs, other techniques,[4, 5, 9–13] have been developed, with the aim of reducing the energy consumption of communication links in NoC-based systems, because the communication links consume significant energy.[4] Kim and Horowitz[5] proposed variable-frequency links, which can track and adjust the voltage level to the minimum supply voltage as the link frequency changes, thus reducing the power dissipation. The variable-frequency link dissipates power varying from 21 mW at 1 Gb/s to 197 mW at 3.5 Gb/s, providing a potential 10X improvement in power.

Based on the variable-frequency links proposed by Kim et al.,[5] Sang et al.[4] developed a history-based dynamic voltage scaling (DVS) policy which adjusts the operating voltage and clock frequency of a link according to the utilization of the link and the input buffer. Soterious et al.[9] proposed a simple dynamic power management technique for communication links based on the communication traffic variance, in order to reduce leakage power consumption. Worm et al.[10] proposed an adaptive low-power transmission scheme for on-chip networks. They minimized the energy required for reliable communications, while satisfying QoS constraints by dynamically, varying the voltage on the links.

Unlike these existing techniques, that are all *on-line* schemes, the proposed technique is an *off-line* technique which assigns the appropriate constant speed to each link. The information on communication patterns from a task graph is exploited. Several researchers tackled a similar problem. There have been off-line scheduling techniques for combined voltage scaling of processors and communication links.[14, 15] However, the target system is not an NoC-based system, thus these techniques did not address the network assignment problem.

Hu et al.[11] proposed a network assignment algorithm which is designed to minimize the dynamic power consumption by reducing the communication traffic. Marcon et al.[12] improved Hu's algorithm by considering the communication dependency in addition. Ogras et al.[13] proposed the optimization technique for communication architecture.

However, they did not address the issue of link speed scaling, but only the network assignment problem, assuming task assignment and task scheduling had been completed. Lei and Kumar[16] have also used the communication patterns of a task graph in tile mapping. However, the objective of the algorithm was to find tile mapping that minimizes the overall execution time of the task graph.

3. OVERALL DESIGN FLOW FOR NoC-BASED SYSTEMS

3.1. Specification and Architectural Model

A periodic real-time application is represented as a task graph (TG) $G = \langle V, E\rangle$, which is a directed acyclic graph, where V is the set of tasks and E is the set of directed edges between tasks. In G, each directed edge $e(\tau_i, \tau_j)$ represents a precedence relationship between τ_i and τ_j. That is, $e(\tau_i, \tau_j)$ means that the task τ_i must complete its execution before task τ_j starts its execution. (For descriptive purposes, $e(\tau_i, \tau_j)$ will be denoted by $e_{i,j}$.) The period of the task graph G is denoted by *period*(G). A task τ_i in G may have a deadline d_i, which must be met to ensure correct functionality of the application. Each edge $e_{i,j}$ is associated with a value $w(e_{i,j})$, which indicates the amount of communication data required between τ_i and τ_j, in the case that τ_i and τ_j are allocated to different PEs. Figure 2(a) presents an example of a task graph. Each edge e has a value of $w(e)$, and the task τ_3 has a deadline.

Although the proposed design technique can support NoC topology selection, it is assumed that the regular tile-based mesh NoC architecture such as Figure 1(a) is given as target architecture for simple modeling. The network topology selection algorithm is presented in subsection 4.6. In the tile-based mesh NoC architecture, an NoC-based system N with $m \times m$ tiles is denoted as a tuple $\langle T, L\rangle$, where $T = \{t_1, \ldots, t_m, \ldots, t_{m^2}\}$ is the set of tiles and $L = \{\ell_1, \ldots, \ell_{4m(m-1)}\}$ is the set of links between tiles. Tiles are assumed to have the same area $\mathbb{A}$. The link between t_i and t_j is denoted by $\ell_{i\to j}$. The notation src$(\ell_{i\to j})$ and dst$(\ell_{i\to j})$ is also used to represent the source and destination of $\ell_{i\to j}$, respectively. For a link ℓ_i, $W(\ell_i)$ indicates the total amount of data transferred across the link. The set of PEs is denoted as $R = \{r_1, \ldots, r_n\}$, where r_i indicates the i-th PE. It is assumed that the number of PEs and the number of tiles are the same, i.e., $|R| = |T|$. Each tile has associated coordinate values, $t_{i.x}$ and $t_{i.y}$ which specify row and column. In this paper, $t_{i.x}$ and $t_{i.y}$ are set to be the quotient and the remainder of $(i-1)/m$, respectively.

3.2. Problem Formulation

For a given task graph $G = \langle V, E\rangle$, an initial step assigns each task in G into one of the available PEs. The function $\Phi\colon V \to R$ is used to represent this task assignment step. The task assignment affects the total communication load because only the tasks assigned into different PEs generate communication loads. Each PE is then assigned to one of the tiles in an NoC-based system. The function $\Psi\colon R \to T$ is used to represent this tile mapping step. The mapping also affects the total communication load because the distances between tiles have changed. The routing path between tiles is then allocated, and the function $\Omega\colon E \to P$ is used to denote this routing path allocation step, where P is the set of link sequences. After the routing path allocation, $W(\ell_i)$ is set for all ℓ_i in L to be $\sum_{\forall e_j, \ell_i \in \Omega(e_j)} w(e_j)$.

In this paper, only the static minimal-path routing algorithm is considered, because it is more suitable for an on-chip network and generates less communication traffic than non-minimal routing paths. The routing path allocation step does not affect the total communication load because only the minimal routing path is considered. However, since the allocation affects the communication load of each link $W(\ell)$, it also affects the speed of the links. The task scheduling step determines the execution order of tasks assigned to the same PE. Since the task scheduling step converts the task graph G to G' by inserting additional edges, it is described with the function $\mathrm{O}\colon G \to G'$. Because the communication delay between tasks must be known for task scheduling, the tile mapping and the routing path allocation steps are executed first. The link speed assignment step decides the clock speed of each link in reducing the energy consumption, by utilizing what would otherwise be slack time. The function $S\colon L \to C$ is used to denote the link speed assignment step, where C is the set of possible clock speeds for the links.

The link energy optimization problem for NoC-based systems can be defined as follows:

Link Energy Optimization Problem
Given $G = \langle V, E\rangle$, R and $N = \langle T, L\rangle$,
Find the functions $\Phi, \Psi, \Omega, \mathrm{O}$, and S such that

$$E = \sum_{\ell_i \in L} (C_L \cdot W(\ell_i) \cdot f(\ell_i)^2 + period(G) \cdot P_{\text{leakage}}(\ell_i))$$

is minimized
subject to $\forall \tau_i \in V, \quad \theta(\tau_i) \le d_i$

where C_L is the average switching capacitance of the links. $f(\ell_i)$ and $P_{\text{leakage}}(\ell_i)$ are the clock speed and the leakage power of a link ℓ_i. For a link ℓ_i with $W(\ell_i) = 0$, $P_{\text{leakage}}(\ell_i)$ is 0, because an inactive link can be turned off. $\theta(\tau_i)$ is the end time of τ_i. The energy consumption can be estimated more accurately with the high-level power model of the on-chip network[17] or the cycle-accurate on-chip network simulator.[18]

Since the task assignment may change the total computation energy consumption on PEs, because of the heterogeneous architecture, both the computation energy of PEs and the communication energy of links should be considered. However, in this paper, only the communication energy is considered to concentrate on the network assignment problem, assuming the homogeneous PEs.

4. ENERGY-EFFICIENT NoC-BASED SYSTEM DESIGN

Figure 3 presents the design flow for NoC-based systems, which consists of five optimization steps. Given a task graph and NoC architecture, the design flow generates an optimized NoC system design. If the NoC architecture (network topology) is not provided, it should be pre-established additionally by the topology selection step. When a predefined routing algorithm is used, the routing path allocation step is not applied. If the target NoC-based system does not provide a variable speed communication link, the link speed assignment step is also excluded. Three genetic algorithms (GAs) are used to explore the design space efficiently. These are a GA-based task assignment algorithm (GA-TA), GA-based tile mapping algorithm (GA-TM), and GA-based routing path allocation algorithm (GA-RPA). These algorithms organize *nested* iteration loops. For example, for each solution at GA-TM, routing path allocation, task scheduling and link speed assignment steps are performed to evaluate the fitness value. The GA-based topology selection algorithm (GA-TS) is used if there is no given network topology architecture.

The genetic algorithm is chosen due to following two reasons. Unfortunately, the communication optimization problem has a very large solution space, as the search space increases factorially with the number of tiles in an NoC-based system.[11] Even for 16 tiles, there is 16 numbers of tile mappings. Moreover, since all the design steps are closely related to the link speed assignment step, it is unlikely that good speed assignment will be found by optimizing each step independently. In other words, the total complexity of the design problem is the product of the complexity of each design step instead of the sum of each step.

Genetic algorithms imitate the principles of natural evolution to solve search and optimization problems, and are a promising technique for system-level design with a large solution space. GA is particularly suitable for multiple-objective optimization.[19] Figure 4 presents the typical structure of a genetic algorithm. Starting with an initial population, a genetic algorithm evolves a population using the crossover and mutation operations. In the right part of Figure 4, various design factors which affect the performance of GA, are presented. These factors need to be selected carefully.

Table I presents the summary of GA operations used for the link energy optimization problem.

4.1. GA-Based Task Assignment

The more tasks that are assigned to a PE, the larger its area becomes, because it requires more memory or gates. Since each tile has the same size, the area constraint may be expressed as $A_\Phi(r_i) \le \mathbb{A}$, where $A_\Phi(r_i)$ means the area of a PE r_i under the task assignment function Φ. If there is an edge e between two tasks assigned to the same PE, its value $w(e)$ is changed to 0. This task assignment step

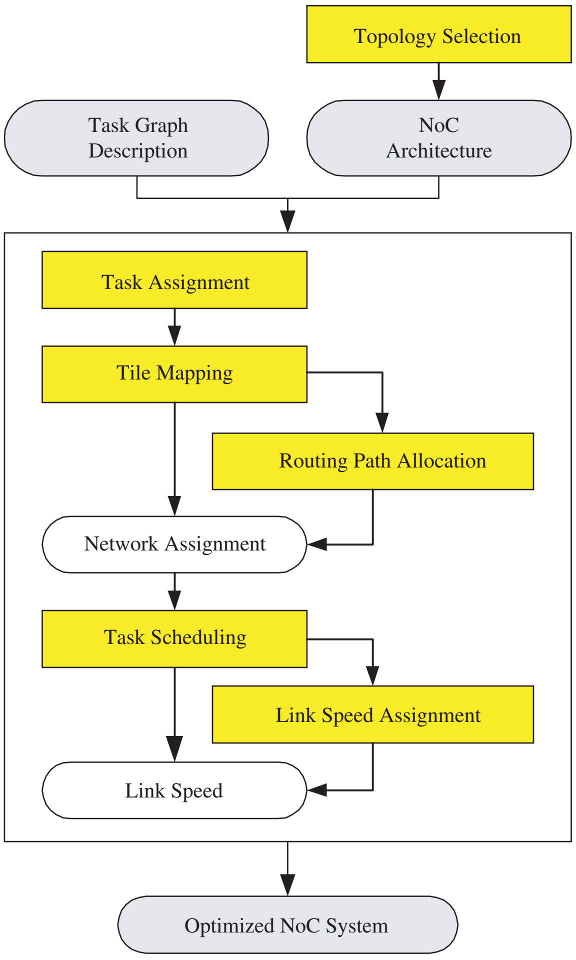


Fig. 3. Overall design flow for energy-efficient NoC-based systems.

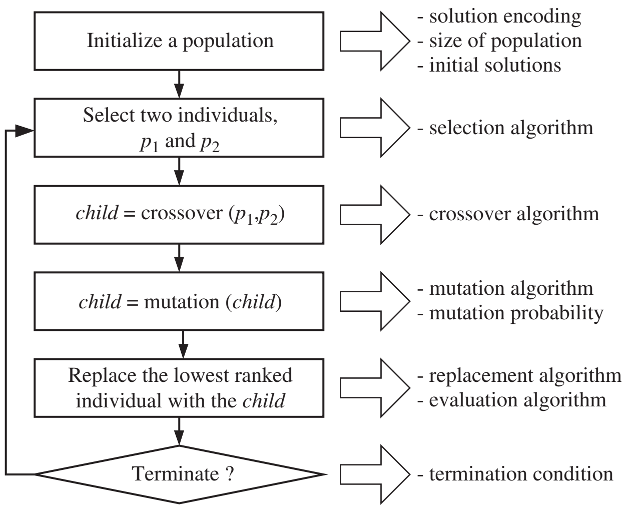


Fig. 4. Typical structure of genetic algorithm and design considerations.

Table I. Summary of GA operations for link energy optimization problem.

Design flow	Gene encoding	Crossover	Mutation
GA-TA	1-D array of integer	2-points crossover	change a randomly selected gene
GA-TM	1-D array of integer	Cycle crossover[20]	exchange two randomly selected genes
GA-RPA	1-D array of integer	Coordinate crossover	change a routing path between two randomly selected genes
GA-TS	2-D array of binary	2-D geographic crossover	change a randomly selected gene

affects total communication load. A task assignment solution is represented as an array of integers. For example, in Figure 5(a), the task τ_1 is mapped to r_8 in the individual p_1. The crossover operation is the two-point crossover, which is widely used in GAs. The parent individuals p_1 and p_2 are divided at the same two points and the child individual c_1 is generated from the first part of p_1, the second part of p_2, and the third part of p_1. The mutation operation changes the values of randomly selected genes into new values, which makes up a new individual.

4.2. GA-Based Tile Mapping

A tile mapping solution is encoded as an array of integers. For example, in Figure 5(b), the PE r_1 is mapped to t_8 in the individual p_1. In order to achieve GA-based tile mapping, care must be taken in designing the crossover operation. If a two-point crossover is used for tile mapping, illegal solutions would be obtained, because different PEs may be allocated into the same tile. Therefore, the cycle crossover is used,[20] which is appropriate when the encoding represents a sequence. Figures 5(b)–(d) demonstrate the method of making child individuals using the cycle crossover. In the mutation operation, two randomly selected genes are exchanged to make a new individual.

For each individual, in order to evaluate fitness value, routing path allocation, task scheduling, and link speed assignment steps are performed. In order to reduce the computation time, whether one individual is topologically identical to another individual is verified, and the evaluation step is omitted if an identical individual has already been evaluated. In order to achieve this operation, each individual is first transformed into an ordered form, and the ordered forms of individuals are compared. The ordered form has the following two properties:

(1) the PE $\Psi^{-1}(t_1)$ has a smaller index than the indices of $\Psi^{-1}(t_m)$, $\Psi^{-1}(t_{m^2-m+1})$, and $\Psi^{-1}(t_{m^2})$, where t_1, t_m, t_{m^2-m+1} and t_{m^2} are four corner tiles;

(2) the PE $\Psi^{-1}(t_m)$ has a smaller index than the index of $\Psi^{-1}(t_{m^2-m+1})$. A tile mapping can be transformed into an ordered form by rotating or mirroring the tile mapping structure.

After tile mapping, the set of communication loads, CL, are composed. For each edge $e_{i,j}$ for which $w(e_{i,j}) > 0$, a communication load v is made, consisting of three properties: $v_{\text{src}} = \Psi(\Phi(\tau_i))$, $v_{\text{dst}} = \Psi(\Phi(\tau_j))$, and $v_{\text{data}} = w(e_{i,j})$.

4.3. GA-Based Routing Path Allocation

In order to transfer the information between tiles, NoC-based systems require a method of routing data packets through the network. There have been several routing algorithms proposed in the past. In general, these algorithms can be divided into static routing and adaptive routing, based on when the routing path is decided. While routing paths are determined at design time in static routing, they change depending on the network status at run time in adaptive routing. In this paper, only the static and minimal path routing algorithm is considered, because it is more suitable for on-chip networks, and generates less communication traffic than the non-minimal routing paths. The most popular static and minimal path routing algorithm is the XY routing algorithm.

In routing path allocation, it is assumed that the source tile transmits the data packet with routing information represented by the sequence of tile identifiers. The intermediate routers between the source tile and destination tile determine the forward direction from the first integer of the routing information and forward the remaining routing information. In the regular 2-D Mesh topology such as Figure 1(a), the routing path allocation step determines n directions, where the Manhattan distance between two tiles is n.

The individuals in the GA-based routing path are represented by one-dimensional arrays of integers. Each array represents the routing path for a communication load

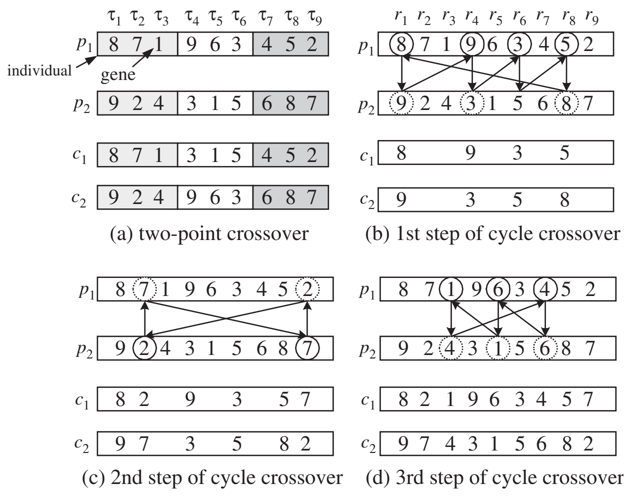


Fig. 5. Crossover operations in GA-TA and GA-TM.

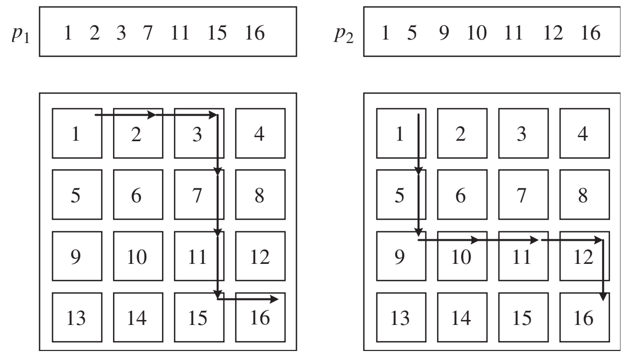


Fig. 6. Routing path encoding.

$v \in CL$. For example, Figure 6 presents two different routing paths for communication load v where $v_{\text{src}} = t_1$ and $v_{\text{dst}} = t_{16}$. The routing paths p_1 and p_2 can be represented as follows:

$$p_1 = \{\ell_{1\to2}, \ell_{2\to3}, \ell_{3\to7}, \ell_{7\to11}, \ell_{11\to15}, \ell_{15\to16}\}$$

$$p_2 = \{\ell_{1\to5}, \ell_{5\to9}, \ell_{9\to10}, \ell_{10\to11}, \ell_{11\to12}, \ell_{12\to16}\}$$

Care should be taken in generating a routing path solution, because it is illegal if no communication link exists between two tiles, which are adjacent in the routing path solution for the given network topology. Therefore, routing path allocation requires a special crossover operation. As presented in Figure 7(a), if a one-point crossover is used for path routing, illegal solutions may be obtained, because the communication links $\ell_{2\to9}$ and $\ell_{5\to3}$ are not provided in the network topology. The crossover operation should guarantee that it generates only legal solutions, while passing on properties from parent individuals to child individuals. In order to satisfy these requirements, a special crossover is invented, called the *coordinate crossover*, for path routing. Figure 7(b) presents the *coordinate crossover* operation. First, the same tile identifiers are located from two parent individuals (*meeting points*). Two different routing paths represented by the parents meet at the meeting point. For example, in Figure 6, two routing paths represented by p_1 and p_2 meet at tile t_{11}. (The start tile t_1 and the end tile t_{16} are excluded from the meeting point.) Second, both parent individuals are divided just before the meeting points and child individuals are created, by mixing parts from two parents, similar to the multi-point crossover operation.

If there are two meeting points mp_α and mp_β, two parent routing paths divided by the meeting points can be represented as $p_1 = \{sp_i, sp_j, sp_k\}$ and $p_2 = \{sp_a, sp_b, sp_c\}$, where the first tile of the subpaths sp_j and sp_b is mp_α and the first tile of the subpaths sp_k and sp_c, is mp_β. The coordinate crossover operation generates two child solutions, $c_1 = \{sp_i, sp_b, sp_k\}$ and $c_2 = \{sp_a, sp_j, sp_c\}$. If two parent solutions are legal, the child solution c_1 is also legal because there are communication links between the last tile of sp_i and the first tile of sp_b, and between the last tile of sp_b and the first tile of sp_k. Therefore, it can be said that the *coordinate crossover* only generates legal child individuals. The child individuals are also guaranteed to inherit the links both parents share.

For the mutation operation, the routing path between two randomly selected tiles is changed. Using these specially designed crossover and mutation operations, only the legal solution space is explored. For regular tile-based mesh topology, the simpler GA encoding, crossover and mutation operations can be used, as presented in Ref. [22].

4.4. Task Scheduling

For task scheduling, a list scheduling algorithm is adopted, which uses the mobility of each task to determine its priority. The mobility of a task is defined as the difference between the ASAP start time and the ALAP end time. In order to obtain these times, the communication delay of an edge must be known.

Marcon et al.[12] used the communication dependence and computation graph (CDCG), in order to calculate the communication delay of an edge. Using CDCG, the packet delay, the routing delay, and the contention delay of each communication load was estimated. Contention delay may occur when more than one communication load shares the same communication link.

In the proposed scheme, the communication dependency can be identified from the task graph. For two edges $e_{i,j}$ and $e_{n,m}$, if a path exists from τ_j to τ_n in the task graph, the edge $e_{n,m}$ has a dependency on the edge $e_{i,j}$. When two independent communication edges $e_{i,j}$ and $e_{n,m}$ share the same communication link ℓ_k, the contention delay can be generated, thus it can be said that $e_{n,m}$ is a candidate conflicting edge of $e_{i,j}$ for $\ell_k(e_{i,j} \overset{\ell_k}{\leftrightarrow} e_{n,m})$. However, the contention delay is generated only when two communication edges $e_{i,j}$ and $e_{n,m}$ use link ℓ_k *simultaneously*. In this case, it can be said that $e_{n,m}$ is a conflicting edge of $e_{i,j}(e_{i,j} \overset{\ell_k}{\Leftrightarrow} e_{n,m})$.

It is assumed that the target NoC architecture uses the wormhole switching algorithm, where a message between

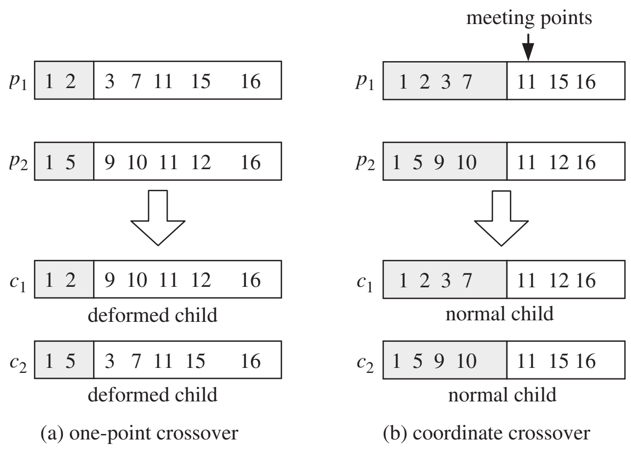


Fig. 7. Crossover operation for routing.

tiles is transferred in the unit of flit. Prior to estimating the communication delay of an edge $e_{i,j}$, two primitive factors, σ_r and $\sigma_l(e_{i,j})$ should be estimated. σ_r is the time required for taking a routing decision inside a router. $\sigma_l(e_{i,j})$ is the maximum time required to transmit a flit through a link. $\sigma_l(e_{i,j})$ has a different value, depending on the operating speeds of communication links used for the communication edge $e_{i,j}$. $\sigma_l(e_{i,j})$ is estimated as follows:

$$\sigma_l(e_{i,j}) = \left\lceil \frac{F}{B} \right\rceil \frac{1}{\min_{\ell_k \in \Omega(e_{i,j})}(f(\ell_k))}$$

where F and B are the flit size and the bit-width of communication link, respectively. $\min_{\ell_k \in \Omega(e_{i,j})}(f(\ell_k))$ represents the clock speed of the communication link with the lowest speed among the links in $\Omega(e_{i,j})$.

If the message header size is 1 flit, the communication latency without contention delay can be computed as follows:[7]

$$\sigma'(e_{i,j}) = \eta(e_{i,j})(\sigma_r + \sigma_l(e_{i,j})) + \sigma_l(e_{i,j}) \left\lceil \frac{w(e_{i,j})}{F} \right\rceil$$

where $\eta(e_{i,j})$ is the number of routers through which a packet of $e_{i,j}$ goes from the source tile to the destination tile. Depending on the clock speed of a communication link $f(\ell_k)$, the communication delay $\sigma'(e_{i,j})$ of the edge $e_{i,j}$ is changed.

If there is an edge conflicting with $e_{i,j}$, the contention delay should be considered. However, it is very complex to calculate the exact value of contention delay, because it depends on various factors such as switching algorithm, virtual channel, and communication start time. If the exact communication start time for the wormhole switching without virtual channel is known, the contention delay can be estimated, as presented in Figure 8. If message C arrives at the shared link ℓ_k, it should wait until messages A and B release the shared resource ℓ_k because the messages arrived at ℓ_k before C. Therefore, the contention delay of $e_{i,j}$ can be represented as follows:

$$\begin{aligned}\sigma_c(e_{i,j}) &= \sum_{\ell_k \in \Omega(e_{i,j})} \sigma_c(e_{i,j}, \ell_k) \\ &= \sum_{\ell_k \in \Omega(e_{i,j})} \left\{ \max_{e_c \overset{\ell_k}{\Leftrightarrow} e_{i,j}} (\delta_r(e_c, \ell_k) - \delta_a(e_{i,j}, \ell_k)) \right\}\end{aligned}$$

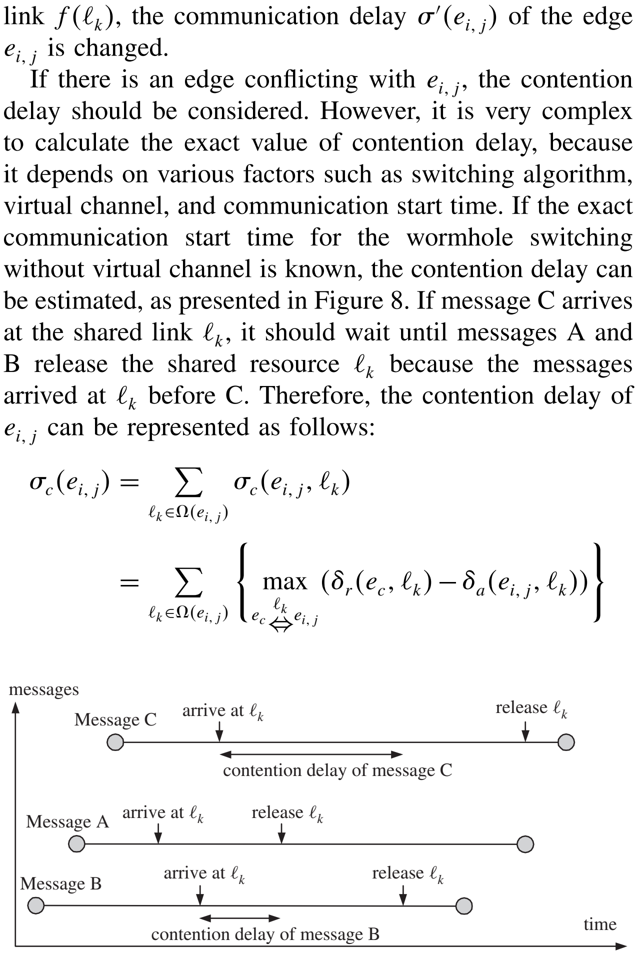


Fig. 8. Contention delay of conflicting communications.

where $\delta_r(e_c, \ell_k)$ is the time when the message e_c releases the link ℓ_k and $\delta_a(e_{i,j}, \ell_k)$ is the time when the message $e_{i,j}$ arrives at link ℓ_k.

However, the arrival time of a message can be changed depending on the corresponding task's execution time. That is, contention delay is determined by the task's execution time. Unfortunately, since the task's execution time cannot be known at design time, all candidate conflicting edges are considered for calculating communication delay:

$$\sigma_c(e_{i,j}) = \sum_{e_c \overset{\ell_k}{\Leftrightarrow} e_{i,j},\, \ell_k \in \Omega(e_{i,j})} \sigma'(e_c)$$

Then, the communication delay is $\sigma(e_{i,j}) = \sigma'(e_{i,j}) + \sigma_c(e_{i,j})$.

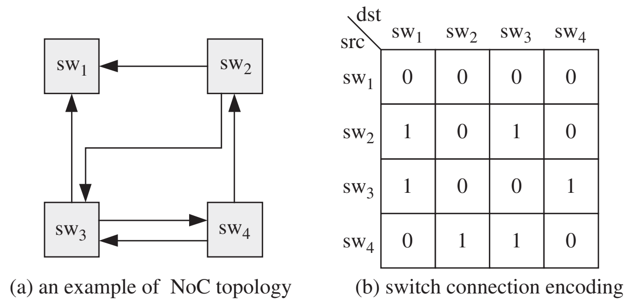
4.5. Link Speed Assignment

For link speed assignment, a similar idea as the voltage and clock speed selection algorithm proposed by Schmitz and Al-Hashimi[23] can be used. The algorithm first estimates the slack time of each task considering the deadline and precedence constraint. It then calculates $\Delta E(\tau_i)$ for a task τ_i which has slack time. $\Delta E(\tau_i)$ is the energy gain when the time slot for τ_i is increased by Δt (with a lower clock speed). After increasing the time slot for task τ_i with the largest $\Delta E(\tau_i)$, by a time increment Δt, the same sequence of steps are repeated until there is no task containing slack time.

While this algorithm determines the operating speed of each task assigned on the DVS-enabled PE, the proposed link speed assignment algorithm determines an operating speed for each link, which does not change dynamically at run time. In order to estimate the slack time for each link, the edges whose communication loads share that link need to be considered. The slack time of a link is the minimum value among the slack times of the edges, i.e., $\xi(\ell_i) = \min_{\ell_i \in \Omega(e_j)}(\xi(e_j))$, where $\xi(\ell_i)$ and $\xi(e_j)$ are the slack times of ℓ_i and e_j, respectively.

4.6. GA-Based Topology Selection

For a more energy-efficient NoC-based system, it is better to use application-specific NoC topology,[13, 24] where the communication links and switches can be optimized based on the communication pattern of the target application. The GA-based topology selection (GA-TS) algorithm is used to explore all possible NoC topologies while existing work attempts to find an efficient algorithm among predefined network topologies in an NoC library. Figure 9(a) presents an application-specific NoC topology. The 1×2 switch (one input link and two output links) is used for switch sw_2, instead of the 2×2 switch. For the topology presented in Figure 9(a), the connection between switches can be encoded with a 2-D array of binary numbers, as presented in Figure 9(b). The value at the coordinate (i, j) means whether there is a link from sw_i to sw_j.


sw1
sw2
sw3
sw4
dst
src
(a) an example of NoC topology
(b) switch connection encoding

src \ dst	sw1	sw2	sw3	sw4
sw1	0	0	0	0
sw2	1	0	1	0
sw3	1	0	0	1
sw4	0	1	1	0

Fig. 9. GA encoding for NoC topology.

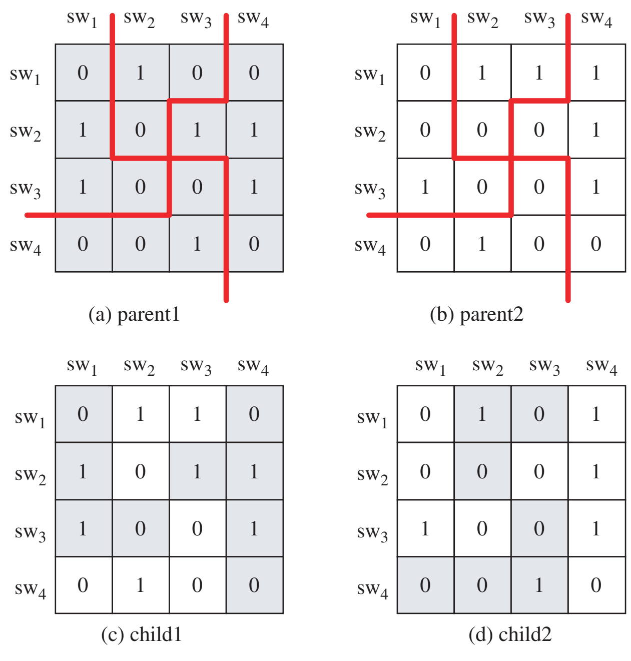
For example, since there is a link from sw_2 to sw_3, the entry in (2, 3) is 1.

Different NoC topologies require different hardware costs. For example, the NoC topology presented in Figure 9 requires one 2×0 switch (sw_1), one 2×2 switch (sw_3), and two 1×2 switches (sw_2 and sw_4). Comparing with the mesh NoC architecture, which requires four 2×2 switches, the NoC architecture presented in Figure 9 is more efficient in terms of energy and area cost.

When the NoC topology is determined by taking advantage of application-specific communication patterns, the GA operations discussed in Table I should be used with some modification. Unlike the mesh NoC architecture, there can be no possible routing path for a communication load in some tile mapping solutions. For example, if the tile mapping of Figure 2(c) is used for the network topology of Figure 9, there is no routing path for the communication from task τ_2 to tasks τ_3 and τ_4.

In this case, tile mapping should be changed using the mutation operation, such that it may have one or more possible paths for all communication loads. For example, if a link is inserted from sw_1 to sw_2 into the network topology of Figure 9, a routing path for the communication from the task τ_2 to tasks τ_3 and τ_4 can be found.

For the crossover operation of GA-TS, the 2-D geographic crossover algorithm[21] is used. In this crossover operation, two parent solutions are separated into several pieces in the same manner. Then, child solutions are generated by mixing the pieces, as presented in Figure 10.



(a) parent1

	sw1	sw2	sw3	sw4
sw1	0	1	0	0
sw2	1	0	1	1
sw3	1	0	0	1
sw4	0	0	1	0

(b) parent2

	sw1	sw2	sw3	sw4
sw1	0	1	1	1
sw2	0	0	0	1
sw3	1	0	0	1
sw4	0	1	0	0

(c) child1

	sw1	sw2	sw3	sw4
sw1	0	1	1	0
sw2	1	0	1	1
sw3	1	0	0	1
sw4	0	1	0	0

(d) child2

	sw1	sw2	sw3	sw4
sw1	0	1	0	1
sw2	0	0	0	1
sw3	1	0	0	1
sw4	0	0	1	0

Fig. 10. 2-D geographic crossover operation for NoC topology.

5. EXPERIMENTAL RESULTS

The efficiency of each optimization technique at the task assignment, tile mapping, routing path allocation, and link speed assignment steps, were estimated. For the experiments, random task graphs g_1 to g_{16} were generated. Figure 11 presents the energy consumptions of the communication links under various optimization configurations.[†] The results were normalized against the energy consumption obtained by a design technique using random task assignment, random tile mapping, XY-routing, and no link speed scaling.

The first bar for each task graph represents the result when the link speed scaling technique is applied (opt_{LS}). The second bar represents the result when the GA-RPA algorithm is used for routing path allocation, as well as link speed scaling ($\text{opt}_{\text{LS+R}}$). The third bar presents the result when the GA-TM algorithm for tile mapping is also applied ($\text{opt}_{\text{LS+R+TM}}$). The fourth bar presents the energy efficiency when all optimization algorithms are used ($\text{opt}_{\text{LS+R+TM+TA}}$). The energy consumption is reduced by 8%, 17%, 27%, and 43% on average by the opt_{LS}, $\text{opt}_{\text{LS+R}}$, $\text{opt}_{\text{LS+R+TM}}$, and $\text{opt}_{\text{LS+R+TM+TA}}$ techniques, respectively. These reduction ratios are dependent on the characteristics of the task graph (e.g., slack time and communication load) and the performance of random configurations. For example, link speed scaling (opt_{LS}) presented small energy reductions because the random task assignment, the random tile mapping and the XY-routing generated little slack time. From these results, it can be known that all design steps have considerable effect on communication energy, and it is necessary to optimize the energy consumption at all design steps. (The task scheduling step is not compared against other techniques because list scheduling is universally popular.)

The proposed GA-based NoC-based system design technique is also compared with previous techniques.[‡]

[†]The real energy consumptions of communication links were not estimated. Instead, it is assumed that the energy consumption is proportional to the communication traffic and the square of communication speed.

[‡]The proposed algorithm is not compared with the on-line techniques which adjust the link speed at run time since the energy performance is dependent on the run-time traffic pattern. If the communication traffic varies significantly at run time, the on-line scheme will show better results. But, the on-line scheme does not guarantee the real time constraint.

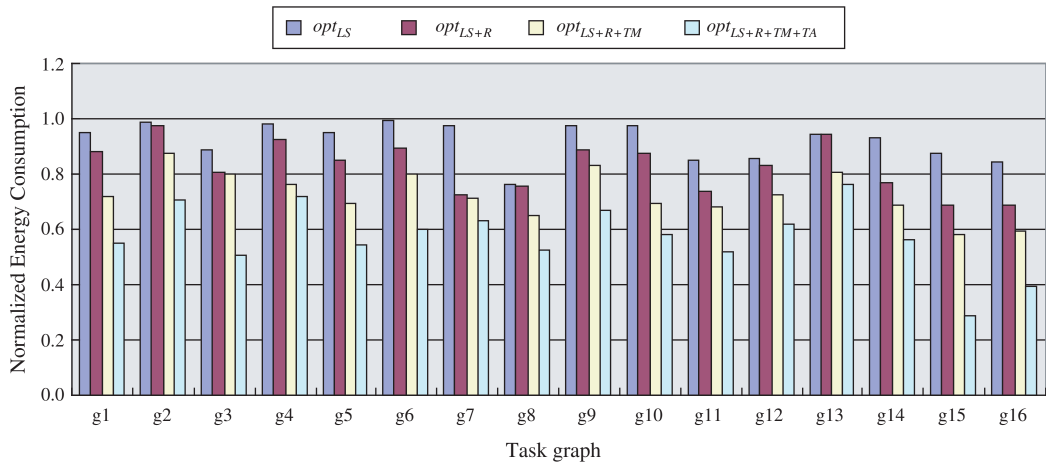


Fig. 11. Effects of nested optimization techniques.

Hu and Marculescu proposed a tile mapping technique that used a branch-and-bound (B&B) algorithm and a routing path allocation algorithm to balance the communication workload across the links.[11] Their algorithm is expanded by integrating it with the proposed task assignment, task scheduling, and link speed assignment algorithms. The integrated algorithm is denoted as **BB** in this paper.

The B&B algorithm finds optimal solution by walking through the search tree. The algorithm consists of two steps, *branch* and *bound*. Starting from the totally unmapped solution (root node), the *branch* step generates a new partially mapped solution (internal node) by mapping one unmapped PE of the previous partially mapped solution until all PEs are mapped (leaf node). In order to reduce the search space, the *bound* step determines whether to branch further from an internal node. The *bound* step compares UBC, the best cost of leaf nodes generated until the current time, and LBC, the lower bound cost of the leaf nodes to be generated from the current internal node. If UBC < LBC, the *branch* step stops to branch the internal node and generates another internal node. The technique in Ref. [11] solves the routing path allocation problem by using a heuristic algorithm. The heuristic determines the routing path between tiles mapped at each *branch* step during the B&B search algorithm. The heuristic attempts to balance the communication workload across the links.

In **BB**, the cost of a solution is estimated from the amount of traffic, which is proportional to the dynamic power consumption of the communication links. The **BB** algorithm is improved by assuming that a link with no communication traffic can be turned off. This technique is called **BB**$^+$ to distinguish it from **BB**. The **BB**$^+$ technique takes into account the leakage power in estimating the cost of a solution.

The heuristic for routing path allocation is also improved in **BB**$^+$, which provides priority to the links with non-zero traffic. Using this algorithm, the number of links without traffic can be increased.[¶] Figure 12 presents the improved heuristic for routing path allocation. First, the list of communication load (*CL*) is sorted by the number of possible routing paths of v. Then, a routing path of each v is allocated beginning from the v with the smallest number of possible routing paths. The function *choose_link* selects one link between the x-directional output-link and the y-directional output-link considers the traffic of links. The functions *get_xlink*(cur_tile, tar_tile) and *get_ylink*(cur_tile, tar_tile) return the x-directional link and the y-directional link among the output-links of cur_tile, respectively. The routing path algorithm of **BB** is identical to Figure 12 except for lines 16 and 17.

```
1: void path_allocation( ) {
2:   Sort CL by the number of possible paths of each $v$;
3:   for each $v$ in CL {
4:     cur_tile := $v_{src}$ ; tar_tile := $v_{dst}$;
5:     while (cur_tile $\neq$ tar_tile) {
6:       $\ell$ := choose_link(cur_tile, tar_tile);
7:       cur_tile := dst($\ell$) ;
8:       W ($\ell$) += $v_{data}$;
9: }}}
10: struct link choose_link(cur_tile, tar_tile) {
11:   $\ell_X$ := get_xlink(cur_tile, tar_tile);
12:   $\ell_Y$ := get_ylink(cur_tile, tar_tile);
13:   if (cur_tile.$x$ = tar_tile.$x$) return $\ell_Y$;
14:   else if (cur_tile.$y$ = tar_tile.$y$) return $\ell_X$;
15:   else {
16:     if (W($\ell_X$) = 0) return $\ell_Y$;
17:     else if (W($\ell_Y$) = 0) return $\ell_X$;
18:     else if (W($\ell_X$) > W($\ell_Y$)) return $\ell_Y$;
19:     else return $\ell_X$ ;
20: }}
```

Fig. 12. Routing path allocation in BB$^+$.

In Ref. [11], it is demonstrated that the proposed algorithm can find an optimal solution within a short period of time. However, if task scheduling and link speed

[¶]In routing path allocation, the total bandwidth of a link should be considered. This is related to the buffer size of a link. However, in this paper, we assume that there is sufficient buffer capacity and there is no constraint on bandwidth.

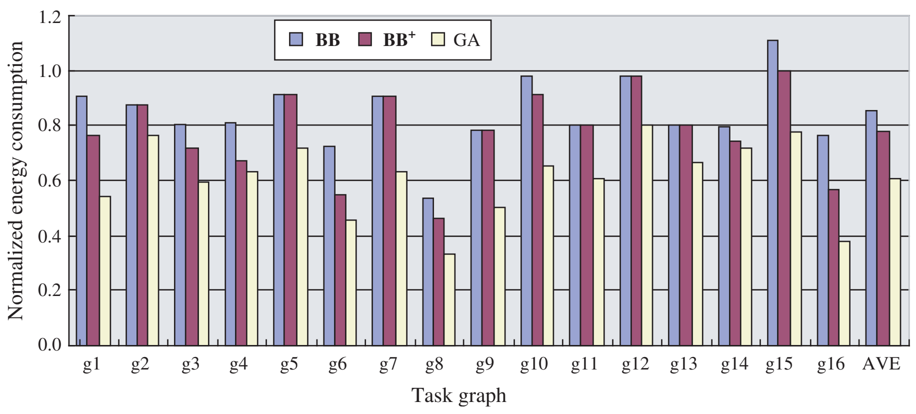


Fig. 13. Performance comparison between B&B and GA techniques.

assignment are considered, **BB** and **BB**$^+$ cannot find an optimal solution because the exact cost of an internal node cannot be estimated before task scheduling and link speed assignment.

Figure 13 presents experimental results that compare the GA-based algorithm with **BB** and **BB**$^+$. In order to only compare the network assignment step, all algorithms were executed with the same predetermined task assignment. The results were normalized against the energy consumption of the random tile mapping and XY-routing technique. As seen from the results, **BB** reduces the energy consumption by 16% on average, but sometimes generates worse results than random mapping (e.g., the task graph g_{15}). This is because **BB** does not consider leakage power or link speed assignment. The **BB**$^+$ technique and the GA-based technique reduced energy consumption by 22% and 39% on average, compared with the random mapping and XY-routing technique. The GA-based technique reduced energy consumption by 28% on average, compared with the **BB** technique.

The task graph of the real application (a multimedia system with an H.263 encoder/decoder and an MP3 encoder/decoder) introduced previously is also used in the experiment.[11] Since each task is assigned to a processing element in the task graph, only the tile mapping and routing path allocation steps are evaluated. The GA-based algorithm reduced energy consumption by 35% compared with the random tile mapping and XY-routing technique.

Table II presents the features of the task graphs and the execution times of the opt$_{\text{LS+R+TM}}$ algorithm running

Table II. Execution times of opt$_{\text{LS+R+TM}}$ algorithm.

TG	$N_{\text{node}}/N_{\text{edge}}$	3×3	4×4	TG	$N_{\text{node}}/N_{\text{edge}}$	3×3	4×4
g1	26/43	5.2	8.4	g9	30/29	6.5	43.9
g2	40/77	9.3	35.5	g10	36/50	12.5	57.9
g3	20/33	9.6	38.8	g11	37/36	9.0	26.3
g4	40/77	11.6	38.6	g12	24/33	5.0	9.6
g5	20/26	5.7	20.1	g13	31/56	9.9	58.6
g6	20/27	3.6	13.8	g14	29/56	6.2	19.3
g7	18/26	5.1	15.8	g15	12/15	3.2	9.4
g8	16/15	3.2	12.2	g16	14/19	2.8	4.2

on a Pentium-III 500 MHz Linux machine. The execution time of the genetic algorithm depends on various design factors, such as the population size, mutation probability, and termination condition. The population sizes are initialized as $|T|^2/2$ and $|T|$ in GA-TM and GA-RPA, respectively. In the experiments, the number of tiles, $|T|$, was 9 (3×3) or 16 (4×4). From Table II, it can be seen that the GA-based algorithm takes a long period of time when a task graph has a large number of edges, i.e., there are many communication loads. Exceptions such as the task graphs g_3 and g_{14} are due to the nature of the link speed assignment algorithm. The link speed assignment algorithm iterates, increasing the delay time of a link by Δt until there is no slack time. Therefore, it takes longer as a task graph has long slack times. The speed of the algorithm can be improved, with a fast link speed assignment module. The **BB** and **BB**$^+$ performed well when $|T| = 9$, but become very slow when $|T| = 16$. Without special speedup techniques, **BB** takes over 1 hour for the graph g_2 when $|T| = 16$. However, the GA-based algorithm takes under a minute, even for reasonably complex task graphs, this makes it acceptable for a design methodology.

6. CONCLUSION

In this paper, an energy-efficient algorithm to optimize the communication energy consumption in NoC-based systems with voltage scalable links is proposed. The proposed algorithm optimizes communication energy at the various design steps, i.e., topology selection, task assignment, tile mapping, routing path allocation, and link speed assignment. Genetic algorithms are used to explore the large design space of energy-efficient NoC-based systems effectively. The algorithm reduces the energy consumption of on-chip network by an average of 39%, compared with an algorithm which uses random tile mapping and XY-routing.

This paper can be extended in several directions. Due to the asynchronous communication protocol, the worst-case communication delay was estimated pessimistically. However, in identifying the exact conflicting relationship between communication edges, a tighter bound on the communication delay could be obtained. Another issue is the buffer size of each router. Since the required buffer size changes depending on the tile mapping and routing path allocation, it is required to design NoC-based systems under a buffer size constraint.

Acknowledgments: This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2005-C1090-0502-0031).

References

1. L. Benini and G. De Micheli, Networks on chip: A new SoC paradigm. *IEEE Computer* (**2002**), Vol. 35, pp. 70–78.
2. W. J. Dally and B. Towles, Route packets, not wires: On-chip interconnection networks. *Proceedings of Design Automation Conference* (**2001**), pp. 684–689.
3. H.-S. Wang, L.-S. Peh, and S. Malik, Power-driven design of router microarchitectures in on-chip networks. *Proceedings of International Symposium on Microarchitecture* (**2003**), pp. 105–116.
4. L. Shang, L.-S. Peh, and N. K. Jha, Dynamic voltage scaling with links for power optimization of interconnection networks. *Proceedings of International Symposium on High-Performance Computer Architecture* (**2003**), pp. 79–90.
5. J. Kim and M. Horowitz, Adaptive supply serial links with sub-1V operation and per-pin clock recovery. *Proceedings of International Solid-State Circuits Conference* (**2002**), pp. 1403–1413.
6. X. Chen and L.-S. Peh, Leakage power modeling and optimization in interconnection networks. *Proceedings of International Symposium on Low Power Electronics and Design* (**2003**), pp. 90–95.
7. J. Duato, S. Yalamanchili, and L. Ni, Interconnection networks, Morgan Kaufmann (**1997**).
8. T. Simunic and S. Boyd, Managing power consumption in networks on chips. *Proceedings of Design, Automation, and Test in Europe* (**2002**), pp. 110–116.
9. V. Soteriou and L.-S. Peh, Dynamic power management for power optimization of interconnection networks using on/off links. *Proceedings of Symposium on High Performance Interconnects* (**2003**), pp. 15–20.
10. F. Worm, P. Ienne, P. Thiran, and G. De Micheli, An adaptive low power transmission scheme for on-chip networks. *Proceedings of International System Synthesis Symposium* (**2002**), pp. 92–100.
11. J. Hu and R. Marculescu, Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures. *Proceedings of Design, Automation, and Test in Europe Conference* (**2003**), pp. 10688–10693.
12. C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, and F. Hessel, Exploring NoC mapping strategies: An energy and timing aware technique. *Proceedings of Design, Automation, and Test in Europe Conference* (**2005**), pp. 502–507.
13. Umit Y. Ogras and R. Marculescu, Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach. *Proceedings of Design, Automation, and Test in Europe Conference* (**2005**), pp. 352–357.
14. J. Liu, P. H. Chou, and N. Bagherzadeh, Communication speed selection for embedded systems with networked voltage-scalable processors. *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis* (**2002**), pp. 169–174.
15. A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. Al Hashimi, Simultaneous communication and processor voltage scaling for dynamic and leakage energy reduction in time-constrained systems. *Proceedings of International Conference on Computer Aided Design* (**2004**), pp. 362–369.
16. T. Lei and S. Kumar, A two-step genetic algorithm for mapping task graphs to a network on chip architecture. *Proceedings of Euromicro Symposium on Digital Systems Design* (**2003**), pp. 180–187.
17. N. Eisley and L.-S. Peh, High-level power analysis for on-chip networks. *Proceedings of Design, Automation, and Test in Europe Conference* (**2003**), pp. 10688–10693.
18. H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, Orion: A power-performance simulator for interconnection networks. *Proceedings of International Symposium on Microarchitecture* (**2002**), pp. 294–305.
19. R. P. Dick and N. K. Jha, MOGAC: A multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (**1998**), Vol. 17, pp. 920–935.
20. I. Oliver, D. Smith, and J. Holland, A study of permutation crossover operations on the traveling salesman problem. *Proceedings of International Conference on Genetic Algorithm* (**1987**), pp. 224–230.
21. A. B. Kahng and B. R. Moon, Toward more powerful recombinations. *Proceedings of International Conference on Genetic Algorithms* (**1995**), pp. 96–103.
22. D. Shin and J. Kim, Power-aware communication optimization for networks-on-chips with voltage scalable links. *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis* (**2004**), pp. 170–175.
23. M. T. Schmitz and B. M. Al-Hashimi, Considering power variations of DVS processing elements for energy minimisation in distributed systems. *Proceedings of International Symposium on System Synthesis* (**2001**), pp. 250–255.
24. A. Jalabert, S. Murali, L. Benini, and G. De Micheli, xPipesComiler: A tool for instantiating application-specific NoCs. *Proceedings of Design, Automation, and Test in Europe* (**2004**), pp. 20884–20889.

Dongkun Shin

Dongkun Shin *is a senior engineer at Samsung Electronics Co., Korea. His research interests include low-power systems, computer architecture, embedded systems, and real-time systems. Shin has a B.S. in computer science and statistics, an M.S. in computer science, and a Ph.D. in computer science and engineering from Seoul National University, Korea.*

Jihong Kim

Jihong Kim *is an associate professor at the School of Computer Science and Engineering, Seoul National University, Korea. His research interests include embedded systems, computer architecture, Java computing, multimedia, and real-time systems. Kim has a B.S. in computer science and statistics from Seoul National University, and a M.S. and Ph.D. in computer science and engineering, respectively, from the University of Washington. He is a member of the IEEE and ACM.*