Journal of Computer and System Sciences 79 (2013) 457–474

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Reliable energy-aware application mapping and voltage–frequency island partitioning for GALS-based NoC

A. Mahabadi [a,b], S.M. Zahedi [a,c], A. Khonsari [a,c,*]

[a] Department of Electrical and Engineering, University of Tehran, Tehran, Iran
[b] Department of Science and Engineering, Shahed University, Tehran, Iran
[c] School of Computer Science, IPM, Tehran, Iran

ARTICLE INFO

Article history:
Received 30 December 2010
Received in revised form 30 April 2011
Accepted 26 September 2012
Available online 12 October 2012

Keywords:
Energy
MILP
Reliability
Voltage–frequency island partitioning
GALS-based NoC
Real-time

ABSTRACT

Reliable energy-aware application mapping, task scheduling, and voltage–frequency island partitioning so as to minimize the energy consumption while preserving the required bandwidth and latency is considered as a challenging problem in the designing of Multi-Processor System-on-Chip. To achieve modular design and low power consumption, Globally Asynchronous Locally Synchronous (GALS) design paradigm is a promising approach which fits very well with the voltage–frequency islands concept. In this paper, we formulate mapping problem of a real-time application with stochastic execution times onto multicore systems, scheduling tasks on processors, and assigning voltage–frequency levels to Processing Elements (PEs) as a Mixed Integer Linear Programming (MILP) in GALS-based Network-on-Chip. Furthermore, owing to the importance of reliability issue, we address the effects of transient faults in our proposed MILP formulation such that the reliability of the whole system incorporating several heterogeneous PEs is guaranteed to be better than a given threshold. Due to the *NP*-hardness of such a problem, a rounding by sampling-based heuristic algorithm is provided. Experimental results based on E3S benchmark suite and some real applications show the effectiveness of our proposed heuristic in achieving a near-optimal solution in a small fractional of time needed to find the optimal solution. Experimental results also show that, our formulation preserves the required reliability and increases the energy consumption by 70% in some cases.

1. Introduction

Due to the increasing demand for high computational capabilities, Network-on-Chip (NoC) has been emerged as a promising interconnect solution to achieve high performance systems [2]. Such chips consist of regular tiles and contain variety number of PEs. The PEs can be DSP cores, programmable general purpose cores, high-bandwidth Input/Output (IO) devices or task specific coprocessors which introduce heterogeneity within the chips. The NoC provides more scalability, flexibility, and performance over the bus-based communication mechanism [3]. Developing a design methodology for NoC-based multiprocessor System-on-Chip (MPSoC) while optimizing the total system energy consumption under the bandwidth, latency, and reliability constraints poses novel and exciting challenges to the research community [4].

* Corresponding author at: School of Computer Science, IPM, Tehran, Iran.
E-mail address: ak@ipm.ir (A. Khonsari).

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2012.09.006

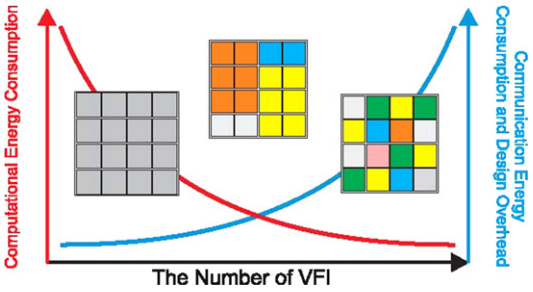

Fig. 1. Computational and communication energy consumption variation and design overhead of GALS-based NoC [1].

Energy-aware application mapping, task scheduling, and Voltage–Frequency Islands (VFIs) partitioning problem have been addressed in recent studies on the NoC-based MPSoC platform design [5–9]. Owing to the heterogeneity of PEs, assigning a specific application task to different PEs results in very different computational energy consumption. For each task assignment, diversity of inter-task communication volumes and existence of different paths between source and destination, can led to completely different communication energy consumption. Furthermore, PEs can operate at multiple voltage–frequency levels to achieve desirable performance and energy trade-offs; consequently, the solution to the energy-aware application mapping, task scheduling, and VFIs partitioning has a significant impact on the total energy consumption.

Assigning single supply/threshold voltage and operating clock speed to all PEs is not energy efficient and needs to distribute a single global clock signal throughout a chip. Asynchronous techniques can be used for finding a solution to overcome the synchronous clock problem challenges. However, a fully asynchronous system design needs a well adopted design technique for synchronous systems. The gap between synchronous and asynchronous design techniques is fitted by choosing a globally asynchronous and locally synchronous technique [9]. GALS design technique distributes and partitions the global clock tree as local trees. Hence, it can reduce clock tree size, spread, and clock buffering requirements [10–14]. It also inherits the advantages of both synchronous and asynchronous design and can be well adapted to VFI concept [15].

Multi-processor systems which are designed and implemented with GALS technique are partitioned into several VFIs. In spite of the fact that scaling down the voltage–frequency level of each PE minimizes the computational energy consumption, the extra communication energy consumption and the overhead of designing excessive number of voltage–frequency independent islands may diminish the energy saving within the chip as depicted in Fig. 1. Due to the fact that each VFI requires its own power grid, clock tree, mixed-clock/mixed-voltage FIFOs and voltage level converter in order to communicate with other VFIs [9], there must be limited number of VFIs to achieve fine-grain system-level power management in a single chip [1].

In order to solve the energy-aware application mapping, task scheduling, and voltage–frequency islands partitioning problem, several studies have been recently carried out [1,5–7,16,17]. Aiming at reducing energy consumption and confidence that tasks will not lose their deadlines, these studies benefit from applying wide range of techniques such as branch-and-bound algorithm, linear programming, randomized algorithm and greedy-based heuristics [6,8,16–19]. These studies considered the problem of optimizing energy under the assumption that all tasks execute their worst-case execution times. Nevertheless, as an important observation in many cases, tasks may finish before their Worst-Case Execution Time [20]. In this study, we consider tasks with worst-case execution times and probabilistic execution times. For simplicity, it is further assumed that a probability distribution of each task execution times is being provided.

Optimizing the energy consumption of running a target application on a NoC-based MPSoC architecture while satisfying bandwidth, latency, reliability and real-time constraints can be divided into five subproblems as depicted in Fig. 2: (a) assigning tasks to PEs, (b) mapping PEs onto the routers of the NoC architecture, (c) assigning voltage–frequency levels to PEs, (d) assigning task communications to routing paths among PEs, (e) scheduling tasks onto PEs. To achieve the optimal energy consumption, all five subproblems have to be solved simultaneously. Despite the fact that solving all subproblems simultaneously is a complicated task, it leads to optimal overall energy consumption.

In addition to the complexities imposed by the task constraints (e.g., task deadline, throughput and bandwidth requirements) on mapping and scheduling, reliability has been emerged as a significant application constraint in embedded systems [21,22]. As feature size continue to be scaled down in deep submicron technology, it is possible to integrate a large number of transistors into a single chip. This integration leads to considerable rate of transient faults caused by errors which arise from capacitive crosstalk, power supply, and neutron and alpha radiations [23–26]. It worth to mention that, the effect of voltage scaling on transient faults [27,28] make the reliability issues more complicated.

In this paper, we formulate the problem of energy-aware application mapping, task scheduling, and voltage–frequency islands partitioning as a Mixed Integer Linear Programming and solve all subproblems (a) to (e) mentioned above simultaneously in a unified manner. To the best of our knowledge, our proposed MILP formulation is the first that formulates reliability constraints and considers stochastic behavior of task execution times. We also address different constraints such as bandwidth, latency, task deadlines and the maximum number of VFIs in our proposed MILP formulation. The effect of transient faults on the reliability of the system is also formulated as an MILP which guarantees the minimum required reliability of the system.

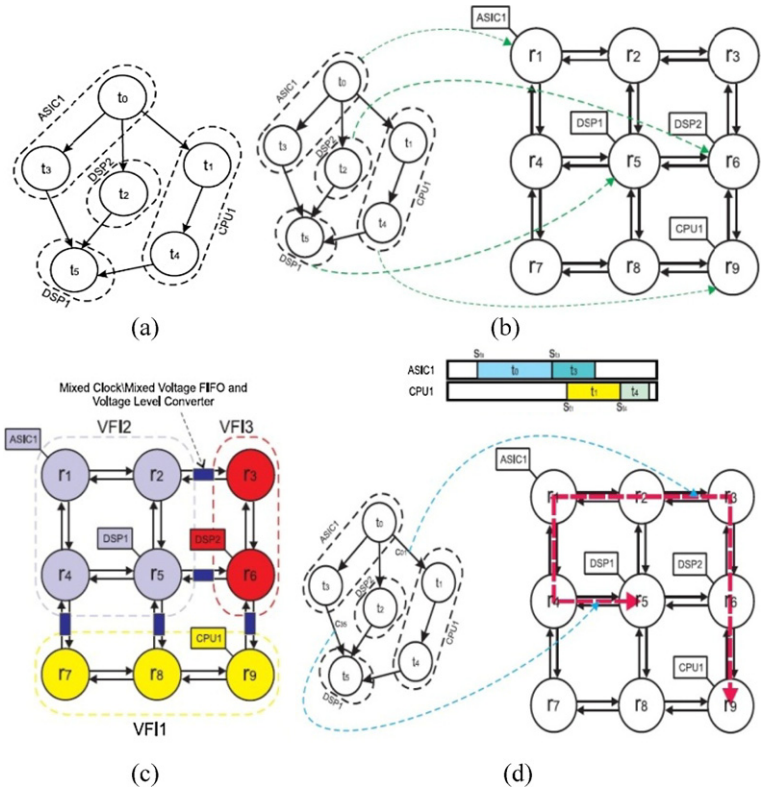

Fig. 2. Subproblems of optimizing the energy consumption of running a target application on a NoC-based MPSoC architecture: (a) assigning tasks to PEs; (b) mapping PEs onto the routers of the NoC architecture; (c) assigning voltage–frequency levels to PEs; (d) assigning task communications to routing paths among PE and scheduling tasks onto PEs.

The remainder of this paper is organized as follows: The related work is addressed in Section 2. The power, energy, latency, reliability and application models are described in Section 3. Section 4 presents the energy-aware application mapping, task scheduling, and VFIs partitioning problem formulation while a novel rounding by sampling-based heuristic is provided in Section 5. Experimental results are provided in Section 6. Finally, Section 7 concludes the paper.

2. Related work

The authors in [1] developed a design methodology which partitions NoC architecture into multiple VFIs and assigns supply and threshold voltage levels to each VFI. Their methodology minimizes the overall system energy consumption, under performance constraints. In [29], Jang et al. construct the framework for energy-efficient VFI-aware partitioning, mapping and routing. The authors in [30] and [8] consider the mapping and routing problem in the presence of bandwidth as well as latency constraints as an application requirement.

In [6], Ghosh et al. take a unified approach to minimize the energy consumption while mapping the application tasks to the PEs, mapping the PEs to the routers, assigning operating voltage to the PEs, and routing of data paths subject to performance constraints. They propose MILP formulation to find the optimal solution and then provide randomized rounding-based heuristic to utilize the MILP relaxation. Considering regular mesh NoC topology, in [19], the authors formulate the mapping and voltage islanding problem as an optimization problem in a unified way and present both optimal solution, obtained by solving the MILP, and heuristic solution based on random greedy selection.

The authors in [7,16,31] minimize the total communication energy consumption in the mapping problem under different performance or bandwidth constraints. The authors in [5] propose a mapping technique which allocates available PEs to the incoming real-time application tasks in order to minimize communication energy, given some deadline constraints. In [18], the authors present an algorithm to map cores under bandwidth constraints, minimizing the average communication delay. The authors in [17] formulate the inter-tile network contention using ILP formulation and solve the contention-aware application mapping problem by a mapping heuristic.

3. Preliminaries and problem definition

In this section we review the necessary background required for the rest of the paper.

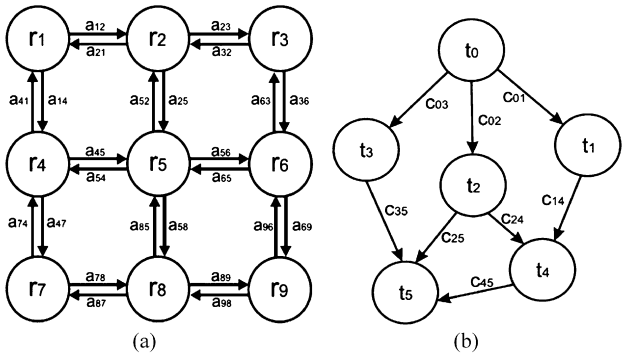

Fig. 3. The input graphs: (a) an ACG for a 3×3 mesh-based NoC architecture; (b) a sample CTG with 6 tasks and 8 edges.

3.1. Application characteristics

In order to formulate the energy-aware application mapping, task scheduling, and voltage–frequency islands partitioning in a formal way, the following definitions are provided which clarify the characteristics of the application:

Definition 1. An Architecture Characterization Graph (ACG) $G_R = (V_R, A_R)$, is a directed $N \times N$ mesh architecture graph, where each node $r_i \in V_R$ represents a router in the tile-base NoC architecture where each tile consists of a processing or storage element (referred to as PE) and a router, and each arc $a_{ij} \in A_R$ represents a link between two adjacent routers r_i and r_j [32]. Simply, the number of nodes is $n = N^2$ and the number of arcs is $m = 4N(N-1)$. Also, the bandwidth of each physical data channel a_{ij} is represented by $BW(a_{ij})$. Fig. 3(a) depicts an ACG for a 3×3 mesh-based NoC architecture.

Definition 2. Let $P = \{p_1, p_2, \ldots, p_n\}$ be the set of PEs, which can operate at any frequency in a set of available frequency–voltage levels $L = \{f_1, f_2, \ldots, f_{max}\}$ $(f_1 < f_2 < \cdots < f_{max})$. We use normalized voltages and frequencies which implies $f_{max} = 1.0$. The power consumption of each processing element j operating at frequency level $f_k \in L$ is represented by ρ_j^k. Also, β_{kl} is the energy overhead of connecting two adjacent tiles with their corresponding PEs operating at $f_k, f_l \in L$.

Definition 3. A Communication Task Graph (CTG) $G_T = (V_T, E_T)$, is a directed task graph, where V_T is the set of tasks and $c_{ij} \in E_T$ represents the $t_j's$ dependency on t_i [32]. Fig. 3(b) shows a sample CTG with 6 tasks and 8 edges. Each c_{ij} has the following properties:

- $V(c_{ij})$: The data communication volume (bit) between t_i and t_j.
- $bw(c_{ij})$: The minimum bandwidth (bits per second) required by c_{ij} that should be satisfied for each task communications.
- $\sigma(c_{ij})$: The maximum tolerable latency of c_{ij} which is given in the number of hop counts instead of an exact number of clock cycles [30] and also represents the performance constraint in the mapping problem.

Furthermore, each $t_i \in V_T$ has the following properties:

- The $deadline_i$ which is the deadline of t_i.
- Two vectors, $X_{ij} = \{x_{ij}^1, x_{ij}^2, \ldots, x_{ij}^{n_{ij}}\}$ and $Pr_{ij} = \{pr_{ij}^1, pr_{ij}^2, \ldots, pr_{ij}^{n_{ij}}\}$, which are considered to be the execution times of task t_i when it is executed on p_j operating at f_{max} and corresponding probabilities, respectively $(\forall t_i \in V_T \,\&\, \forall p_j \in P$, $\sum_{l=1}^{n_{ij}} pr_{ij}^l = 1)$. We further assume that $x_{ij}^1 < x_{ij}^2 < \cdots < x_{ij}^{n_{ij}}$, where x_{ij}^1 and $x_{ij}^{n_{ij}}$ are the *Best-Case Execution Time* (BCET) and *Worst-Case Execution Time* (WCET) of task t_i executing on p_j, respectively.

Definition 4. The minimum reliability R_0 indicates the minimum tolerable reliability of the real-time system which is required to be satisfied while mapping a real-time application and scheduling the real-time tasks [49].

3.2. Power, energy and delay models

In a general system-level power model, the power consumption of a computing system is given by [28] as:

$$P = P_s + h(P_{ind} + P_d), \qquad P_d = kC_{ef}V^2 f$$

where P_s is the *static power* which is used to keep the clock running, maintain basic circuits. This power can be only removed by turning down the whole system. P_{ind} is the *frequency independent active power* which indicates the constant

power consumed by off-chip and external devices. Putting the system components in sleep mode can efficiently remove the frequency independent active power. Finally, P_d represents the *frequency dependent active power* which consists of processor dynamic power and any power that depends on processing frequencies and system supply voltage. If the system operates in sleep mode h equals 0 otherwise $h = 1$. Here, C_{ef}, V and f represent the switch capacitance, supply voltage and frequency of the system, respectively.

Switching between on and off modes has a significant impact on time and energy overhead [33]. In the sequel, we assume an always "on" system. We know that p_s does exist, below we focus on frequency dependent active power and frequency independent active power.

For frequency scaling, when *frequency independent active power* is greater than zero energy increases if the frequency of the application decreases and at the same time, we keep the supply voltage constant [28]. However, *frequency dependent active power* term decreases as frequency of the application decreases [34]. In *voltage scaling* approach reduces the supply voltage when reduces the frequency [35]. Owing to the almost linear relation between circuit delay and V^{-1} [36] and [37], the authors in [28] indicate that for systems to operate properly, the operating frequency needs to scale down linearly when the supply voltage is decreased. They used normalized frequencies and voltages and assume $f_{max} = V_{max} = 1$ which implies that for frequency f, the corresponding supply voltage is $V = f$, $V_{max} = f$. We consider the similar assumptions. Thus, the dynamic energy consumption of executing a task at frequency f and supply voltage V, can be modeled as:

$$\begin{aligned} E_i^f &= \left(P_{ind} + C_{ef} V^2 f\right) \frac{x_i}{f} \\ &= P_{ind} \frac{x_i}{f} + C_{ef} V^2 x_i = P_{ind} \frac{x_i}{f} + C_{ef} f^2 x_i \end{aligned}$$

where x_i is the execution time of task t_i when it is executed at f_{max}.

Considering the probabilistic execution times for each task, the expected computational energy consumption $\bar{E}_i^f$ can be expressed as:

$$\bar{E}_i^f = \sum_{x_i^k \in X_i} Pr_i^k . \left(P_{ind} \frac{x_i^k}{f} + C_{ef} f^2 x_i^k \right)$$

where x_i^k represents the kth possible execution time of t_i when it is executed at frequency f and Pr_i^k denotes its corresponding probability. Here, X_i represents the set of all possible execution times for task t_i and intuitively, $\sum_{x_i^k \in X_i} Pr_i^k = 1$.

The dynamic energy consumed by one bit of data sending through the router can be defined by *bit energy* E_{bit} metric as [38]:

$$E_{bit} = E_{L_{bit}} + E_{B_{bit}} + E_{S_{bit}} \tag{1}$$

where $E_{B_{bit}}$, $E_{L_{bit}}$ and $E_{S_{bit}}$ represent the communication energy consumption of the buffer, link and switch fabric, respectively. Assuming the *bit energy* values are measured at maximum supply voltage and frequency, the dynamic part of communication energy consumption of transmitting one bit from the source router to the destination router can be computed as [39]:

$$E_{bit}(src, dest) = \sum_{r_i \in P'}^{\infty} \left(E_{B_{bit}}(r_i) + E_{L_{bit}}(r_i) + E_{S_{bit}}(r_i) \right) f_{r_i}^2$$

where P' is the set of routers on the path from the source outer to the destination router and f_{r_i} represents normalized frequency which is assigned to router r_i.

Partitioning the chip into VFIs imposes extra energy overhead. This energy is required to connect nodes in one VFI with other nodes in other VFIs. The overhead of connecting two different voltage–frequency islands is modeled as [39]:

$$E_{Overhead} = E_{ClkGen}, E_{Vconv} + E_{MixClkFifo} \tag{2}$$

where E_{ClkGen}, E_{Vconv} and $E_{MixClkFifo}$ represent the energy consumed by generating additional clock signals [40], the voltage level converters [41] and mixed-clock/mixed-voltage FIFOs [42], respectively. The power delivery network is a complicated structure and considering implementation overheads prohibits to increase the number of VFIs without provision [6,9,39]. As discussed in [1], the energy consumption of level shifters which is used to connect voltage–frequency islands is not negligible and could be considerably high. This energy is derived based on the level converter circuit designed in [41], where energy overhead during the voltage transition between two levels is estimated to be proportional to the difference of the square of the voltage levels.

Finally, we assume that routers in each VFI are locally synchronous and routers in two VFIs communicate with each other through mixed-clock/mixed-voltage FIFOs. Therefore, the communication latency between source and destination routers, while sending a volume of data equal to $vol(src, dest)$ bits, in non-blocking traffics is expressed as [39]:

$$t_{comm}(src, dest) = \sum_{r_i \in p'} \left(\frac{D_S}{f_i} \right) + t_{fifo} \left\lceil \frac{vol(src, dest)}{W} \right\rceil \tag{3}$$

where P', W, and D_S represent the set of routers on the path from source router to destination router, the physical data channel width, and routing, switching, and propagation delay across wires between two routers at maximum frequency, respectively. Here, f_i is an operating frequency assigned to r_i and t_{fifo} equals to the delay of FIFO buffers.

3.3. Fault and reliability models

Focusing on *transient faults*, the authors in [43] assume that transient faults follow Poisson distribution with an average arrival rate λ. However, authors in [28] show the direct effect of dynamic voltage and frequency scaling on the fault rate variations. Therefore, the general model of average transient fault rate for systems running at frequency f and voltage V where $f_{min} \leqslant f \leqslant f_{max}$ and $V_{min} \leqslant V \leqslant V_{max}$, can be expressed as [28]:

$$\lambda(f) = \lambda(f, v) = \lambda_0 . g(f, v)$$

where λ_0 represents the average fault rate at f_{max} and voltage V_{max}. That is, $g(f_{max}, V_{max}) = 1$. Based on the results presented in [44] and [45], the fault rates increases exponentially when supply voltage decreases in Alpha processors family and also memory devices for different technologies. This is due to the fact that reducing the supply voltage leads to smaller critical charge which is responsible for exponentially increased fault rate [46] and [47]. Furthermore, lower energy particles with smaller critical charge could cause an error owing to the fact that there are lower energy particles than higher energy particles [48].

As discussed in Section 3.2, we assume that the supply voltage for lower operating frequencies is reduced [36]. Hence, the effects of voltage scaling on fault rates while the system runs at frequency f and corresponding voltage $V = f.V_{max} = f$, can be formulated as [28]:

$$\lambda(f) = \lambda_0 . g(f) = \lambda_0 . 10^{\frac{d(1-f)}{1-f_{min}}}$$

where $d > 0$ is a constant which represents the sensitivity of fault rate to voltage and frequency scaling.

The *reliability* of a task is defined in [49] as *the probability of completing the task successfully, i.e. without encountering errors triggered by transient faults*. Since we assumed that transient faults arrive according to a Poisson process in which its arrival rate depends on the operating frequency, the probability of having no transient fault during one running of a task t_i can be computed as:

$$R_i^f = e^{-\lambda(f) . \frac{x_i}{f}}$$

where x_i is execution time of task t_i when it is executed at f_{max} and simply its execution time at $f < f_{max}$ is given by x_i / f. We consider different possible execution times for each task to express more realistic behavior of a real-time system. The expected value of R_i^f can be expressed as:

$$\overline{R}_i^f = \sum_{x_i^k \in X_i} \left(Pr_i^k . e^{-\lambda(f) . \frac{x_i^k}{f}} \right)$$

We are more interested in the *Worst-Case Reliability* for each task which occurs at its WCET (e.g., x_i^{WCET}). The WCR is an underestimation of task reliability since $\forall k, x_i^k \leqslant x_i^{WCET}$. We also consider WCR as a constraint during mapping process which can guarantee the reliability of a restricted real-time system. Considering the definition of the reliability for each task and the fact that tasks are executed on PEs independently, if the reliability of task t_i at frequency f is R_i^f then the worst-case reliability of the whole system at frequency f is calculated as the product of the reliability of its tasks and is given by:

$$R_f = \prod_{t_i \in T} R_i^f = e^{-\sum_{t_i \in T} (\lambda(f) . \frac{x_i^{WCET}}{f})}$$

As discussed above reducing the supply voltage at lower operating frequencies leads to exponentially increased fault rates which results in less reliability. Hence, there is a trade-off between computational energy consumption and the reliability in terms of supply voltage and operating frequency. Increasing the chip operating frequency and supply voltage together increases reliability at the expense of more energy consumption. We are interested in minimizing the energy consumption as long as not deviating from minimum reliability requirement of the system.

Table 1
Output functions of the problem.

Function	Explanation
M_1	*Function*: Assigning tasks to PEs *Explanation*: $M_1(t_j) = p_j$, means $t_i \in V_T$ has been assigned to $p_j \in P$
M_2	*Function*: Assigning voltage–frequency to PEs *Explanation*: $M_2(p_j) = f_k$, means $p_j \in P$ has been assigned to $f_k \in L$
M_3	*Function*: Mapping PEs to routers *Explanation*: $M_3(p_j) = r_m$, means $p_j \in P$ has been mapped onto $r_m \in V_R$
M_4	*Function*: Mapping task communications to routing paths *Explanation*: $M_4(c_{ij}) = \Gamma$, means $C_{ij} \in E_T$ has been mapped onto $\Gamma \to e_{r_1 r_2} \to e_{r_2 r_3} \to \cdots \to e_{r_{n-1} r_n}$, where $\forall i \in [1, n-1]$, $e_{r_i, r_{i+1}} \in E_R$ and $M_3(M_1(t_i)) = r_1$ and $M_3(M_1(t_i)) = r_n$

3.4. Problem definition

We have the following formal definition for the energy-aware application mapping, task scheduling, and voltage–frequency islands partitioning as:

$$Minimize(E_{comp} + E_{comm} + E_{VFIs}) \tag{4}$$

where E_{comp}, E_{comm}, and E_{VFIs} represent computational energy consumption, communication energy consumption, and VFIs overhead energy consumption, respectively.

Constraints:

- Each task is assigned to exactly one PE.
- Each processor operates at exactly one frequency and voltage level.
- Each PE is mapped onto exactly one router and exactly one PE is assigned to each router.
- Execution deadlines are met for each task and all task dependencies are considered.
- The reliability of the whole system need to be above a given threshold.
- Number of VFIs are less than or equal to a given upper bound.
- The link bandwidth constraint and communication latency constraint are met for router links and task communications, respectively.

4. Proposed MILP formulation

In this section, we develop a formal definition of energy-aware application mapping, task scheduling, and voltage–frequency islands partitioning, subject to bandwidth, latency, deadline constraints and the maximum number of permissible VFIs. There are four possible functions M_1: for task to processing element assignment function, M_2: for PEs to voltage–frequency level assignment function, M_3: for processing element to router mapping function, and M_4: represent the task communication to path mapping function as depicted in Table 1. We formulate these assignment and mapping functions step by step and, we model the scheduling problem and all constraints in an MILP formulation.

We first formulate E_{comp}, E_{comm}, and E_{VFIs} as an MILP formulation according to the models presented in Section 3.2 and formulate the first three constraints simultaneously in Sections 4.1, 4.2, and 4.3. Then we formulate the rest of the constraints in Section 4.4.

4.1. Computational energy consumption

The computational energy consumption plays a significant role in total energy consumption and below we formulate this problem as an MILP formulation. Before doing so, we need to model all related parameters. Computational energy consumption of each PE while running a set of tasks depends on two factors: the tasks execution times and PE's frequency level. At first, we consider $M_1(t_i) = p_j$ which states the assignment of t_i to p_j and can be modeled by an indicator variable as:

$$x_{ij} = \begin{cases} 1 & \text{if } M_1(t_i) = p_j \text{ where } p_j \in P \\ 0 & \text{otherwise} \end{cases}$$

Each task must be assigned to exactly one PE, so the following constraint is provided as:

$$\forall t_i \in V_T, \quad \sum_{p_j \in P} x_{ij} = 1$$

Secondly, frequency level assignment (e.g., $M_2(p_j) = f_k$) can be modeled by another indicator variable as:

$$y_{jk} = \begin{cases} 1 & \text{if } M_2(p_j) = f_j \text{ where } f_k \in L \\ 0 & \text{otherwise} \end{cases}$$

The following constraint is used to ensure that exactly one frequency level has to be assigned to each PE as:

$$\forall p_j \in P, \quad \sum_{f_k \in L} y_{ik} = 1$$

Considering these two assignment function simultaneously, we can provide a new decision variable as:

$$\gamma_{ij}^k = \begin{cases} 1 & \text{if } M_1(t_i) = p_j \,\&\, M_2(p_j) = f_k \\ 0 & \text{otherwise} \end{cases}$$

This variable can be expressed in a quadratic form as $\forall t_i \in V_T,\ f_k \in L$: $\gamma_{ij}^k = x_{ij}.y_{jk}$. It is possible to reduce the quadratic expression to a corresponding linear expression by eliminating quadratic terms and adding new constraints. Let us consider a quadratic expression which comprises quadratic terms, such as $a.b$, where a and b are binary variables. Let us substitute c for the multiplication of a and b. This substitution could be ensured by adding the following set of new linear constraints as:

$$c \geqslant a + b - 1, \qquad 2c \leqslant a + b \tag{5}$$

Accordingly, the following constraints provide conditions in the definition of γ as:

$$\forall t_i \in V_T,\ p_j \in P,\ f_k \in L: \quad x_{ij} + y_{jk} \geqslant 2\gamma_{ij}^k$$
$$x_{ij} + y_{jk} - 1 \leqslant \gamma_{ij}^k$$

The expected *computational energy consumption* now can be formulated as:

$$E_{comp} = \sum_{t_i \in V_T} \sum_{p_j \in P} \sum_{f_k \in L} \sum_{l \in N_{ij}} \left(\gamma_{ij}^k p_{ij}^l pr_{ij}^l \frac{x_{ij}^l}{f_k} \right) \tag{6}$$

where the elements in the set $N_{ij} = \{1, \ldots, n_{ij}\}$ correspond to elements in X_{ij}.

4.2. VFIs overhead energy consumption

Mapping PEs onto routers can be modeled as:

$$z_{jm} = \begin{cases} 1 & \text{if } M_3(p_j) = r_m \text{ where } r_m \in V_R \\ 0 & \text{otherwise} \end{cases}$$

Similarly, the following constraints are used in order to map each PE to exactly one router and assign exactly one processor to each router, respectively.

$$\forall p_j \in P, \quad \sum_{r_m \in V_R} z_{jm} = 1$$
$$\forall r_m \in V_R, \quad \sum_{p_j \in P} z_{jm} = 1$$

Mapping p_j with assigned frequency f_k onto a router r_m is modeled by the following indicator variable as:

$$\zeta_{mk}^j = \begin{cases} 1 & \text{if } M_2(p_j) = f_k \,\&\, M_3(p_j) = r_m \\ 0 & \text{otherwise} \end{cases}$$

These conditions can be modeled by the following constraints:

$$\forall r_m \in V_R,\ p_j \in P,\ f_k \in L: \quad z_{jm} + y_{jk} \geqslant 2\zeta_{mk}^j$$
$$z_{jm} + y_{jk} - 1 \leqslant \zeta_{mk}^j$$

The following indicator variable indicates assigning a frequency level to a router:

$$\theta_{mk} = \begin{cases} 1 & \text{if } \exists p_j \in P,\ M_2(p_j) = f_k \,\&\, M_3(p_j) = r_m \\ 0 & \text{otherwise} \end{cases}$$

The definition of θ_{mk} can be formulated as:

$$\forall r_m \in V_R,\ f_k \in L, \quad \theta_{mk} = \sum_{p_j \in P} \zeta_{mk}^{j}$$

We define another indicator variable to model two adjacent routers with their assigned frequencies as:

$$\alpha_{ij}^{kl} = \begin{cases} 1 & \text{if } \theta_{ik} = 1 \,\&\, \theta_{jl} = 1 \\ 0 & \text{otherwise} \end{cases}$$

This conditional definition can be written as the following constraints:

$$\forall a_{ij} \in A_R,\ f_k, f_l \in L: \quad \theta_{ik} + \theta_{jl} \geqslant 2\alpha_{ij}^{kl}$$
$$\theta_{ik} + \theta_{jl} - 1 \leqslant \alpha_{ij}^{kl}$$

Now the overhead energy consumption of connecting two VFIs can be modeled as:

$$E_{VFIs} = \sum_{e_{ij} \in A_R} \sum_{f_k \in L} \sum_{f_l \in L} \alpha_{ij}^{kl}.\beta_{kl} \tag{7}$$

4.3. Communication energy consumption

There may be multiple paths between a source router and a destination router in a given network. Each path results in different throughput, latency and energy consumption. Let us define $M_4(c_{ij})$ as a task communicate to path function which means:

$$\forall c_{ij} \in E_T, \quad M_4(c_{ij}) = \Gamma \in A_R$$

where Γ is the set of links which are involved in delivering data communication volume between t_i and t_j, $t_j \in V_T$. In order to specify the routing path for a given task communication, say c_{ij}, it is necessary to know routers which t_i and t_j are mapped onto. We define, φ_{im} indicator variable as:

$$\varphi_{im} = \begin{cases} 1 & \text{if } \exists p_j \in P,\ M_1(t_i) = p_j \,\&\, M_3(p_j) = r_m \\ 0 & \text{otherwise} \end{cases}$$

Modeling this variable needs a new indicator variable which is defined as follow:

$$\vartheta_{im}^{j} = \begin{cases} 1 & \text{if } M_1(t_i) = p_j \,\&\, M_3(p_j) = r_m \\ 0 & \text{otherwise} \end{cases}$$

Considering the definitions of φ and ϑ, now we can write the following equation:

$$\forall t_i \in V_T,\ r_m \in V_R, \quad \varphi_{im} = \sum_{p_j \in P} \vartheta_{im}^{j}$$

It is clear that each task needs to be mapped onto exactly one PE and each PE must be assigned to exactly one router, hence the following constraint is given:

$$\forall t_i \in V_T, \quad \sum_{r_m \in V_D} \varphi_{im} = 1$$

If $M_4(c_{ij}) = \Gamma$ and $|\Gamma| \geqslant 1$ (e.g., t_i and t_j are mapped to deferent routers) then Γ needs to satisfy three main constraints.

- *Loop avoidance constraint*: There must be no loop in the path from $M_3(M_1(t_i))$ to $M_3(M_1(t_j))$ for each $c_{ij} \in E_T$ which means that $\forall l_1, l_2 \in \Gamma$, if $l_1 = a_{x_1 y_1}$ and $l_2 = a_{x_2 y_2}$, then $y_1 \neq y_2$.
- *Source and destination constraints*: The routing path must be from $M_3(M_1(t_i))$ to $M_3(M_1(t_j))$. These routers, therefore, must be visited once through the routing path which implies that $\exists l_1 = a_{x_1 y_1} \in \Gamma$ such that $\varphi_{ix_1} = 1$ and $\exists l_2 = a_{x_2 y_2} \in \Gamma$ such that $\varphi_{jy_2} = 1$. It could be also inferred that $\nexists l_3 = a_{x_3 y_3} \in \Gamma$ such that $\varphi_{iy_3} = 1$ and $\nexists l_4 = a_{x_4 y_4} \in \Gamma$ such that $\varphi_{jx_4} = 1$.
- *Path existence constraint*: Links selected for the routing path must construct a single connected path. Hence, $\forall l_1 \in \Gamma$ if $l_1 = a_{x_1 y_1}$ and $\varphi_{jy_1} = 0$ then $\exists l_2 = a_{x_2 y_2} \in \Gamma$, such that $x_2 = y_1$.

To model these routing path constraints we use the variable δ_{xy}^{ij}. If $\delta_{xy}^{ij} = 1$ then a_{xy} is supposed to be used in the routing path for c_{ij}. Thus, δ_{xy}^{ij} can be written as:

$$\forall c_{ij} \in E_T, \quad \delta_{xy}^{ij} = \begin{cases} 1 & \text{if } a_{xy} \in M_4(c_{ij}) \\ 0 & \text{otherwise} \end{cases}$$

In order to express the above constraints, another indicator variable sep_{ij} is given by:

$$sep_{ij} = \begin{cases} 1 & \text{if } M_1(t_i) \neq M_1(t_j) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Moreover, another indicator variable $contain_{ijk}$ is considered as following formulations to model sep_{ij}. The $contain_{ij}^{k}$ equals 1 if both t_i and t_j are mapped to p_k and equals 0 otherwise.

$$\forall t_i, t_j \in V_T, \ \forall p_k \in p:$$
$$x_{ik} + x_{jk} \geqslant 2contain_{ij}^{k}, \qquad x_{ik} + x_{jk} - 1 \leqslant contain_{ij}^{k}$$
$$\forall t_i, t_j \in V_T, \quad sep_{ij} = 1 - \sum_{p_k \in p} contain_{ij}^{k}$$

Clearly, Γ is an empty set if t_i and t_j are mapped onto the same router. This could be written as:

$$\forall c_{ij} \in E_T, \ \forall a_{xy} \in A_R, \quad \delta_{xy}^{ij} \leqslant sep_{ij}$$

Putting together all the elements, now the routing path constraints could be expressed formally in our MILP formulation. It could be inferred from the *loop avoidance constraint* that if $M_4(c_{ij}) = \Gamma$, then for each router $r_x \in V_R$ there is at most one link in Γ that r_x appears as its head and also there is one link in Γ that r_y appears as its tail. The source and destination of the path, additionally, must be visited once. All these constraints could be modeled as:

$$\forall c_{ij} \in E_T, \ \forall r_y \in A_R:$$
$$\sum_{a_{xy} \in A_R} \delta_{xy}^{ij} \leqslant (1 - \varphi_{iy}), \qquad \sum_{a_{yx} \in A_R} \delta_{yx}^{ij} \leqslant (1 - \varphi_{jy})$$

Furthermore, other constraints in the *source* and *destination constraints* need to be satisfied. There must be a link where t_i is mapped onto its head and also there must be a link where t_j is mapped onto its tail. Thus, the *path existence constraint* can be stated as:

$$\forall c_{ij} \in E_T, \ \forall r_y \in A_R:$$
$$\sum_{a_{xy} \in A_R} \delta_{xy}^{ij} + \varphi_{iy} = \sum_{a_{yx} \in A_R} \delta_{yx}^{ij} + \varphi_{jy}$$

Considering routing paths, it is now possible to model the *Communication Energy Consumption* as:

$$E_{comm} = \sum_{c_{ij} \in E_T} \sum_{a_{xy} \in A_R} \sum_{f_k \in L} \delta_{xy}^{ij} . v_{ij} . \theta_{xk} \in_{xk} \tag{9}$$

where v_{ij} and ε_{xk} represent $v(c_{ij})$ and $(E_{B_{bit}}(x) + E_{L_{bit}}(x) + E_{S_{bit}}(x)) f_k^2$, respectively. Note that (9) is in the Quadratic Programming (QP) form and they can be reduced to an Integer Linear Programming (ILP) by inequalities presented in (5).

4.4. Constraints

(1) *Real-time application deadline*: A valid schedule satisfies two kinds of constraints: *edge constraints* and *inprocessor constraints* [50]. Edge constraint induced by a_{ij} keeps the order between each t_i and its successor t_j. It is necessary that any two tasks that are assigned to the same PE should not overlap in their execution time. This can be achieved by imposing in-processor constraint to the tasks assigned to a single PE. In order to express these constraints, sep_{ij} which was introduced in (8) is used. Moreover, the worst-case execution time of each task can be calculated as:

$$\forall t_i \in V_T, \quad ex_i = \sum_{p_j \in P} \sum_{f_i \in L} \gamma_{ij}^{l} . \frac{n_{ij}^{n_{ij}}}{f_l}$$

Edge constraint, therefore, can be formulated as:

$$\forall c_{ij} \in E_T:$$
$$s_i + ex_i + Comm_{ij} \leqslant s_i + (1 - sep_{ij}).MAX\_VAL,$$
$$s_i + ex_i \leqslant sj + sep_{ij}.MAX\_VAL$$

where $Comm_{ij}$ and MAX_VAL denote communication delay and the largest integer, respectively. According to communication delay model expressed in (3) we can rewrite $Comm_{ij}$ as follow:

$$\forall c_{ij} \in E_T, \quad Comm_{ij} = \sum_{e_{xy} \in A_R} \sum_{f_k \in L} \delta_{xy}^{ij}.\theta_{xk}.\tau_{xk} + t_{fifo}.\frac{v_{ij}}{W} \tag{10}$$

where τ_k represents routing, switching and propagation delay across wires between two routers at frequency f_k. We can linearize (10) by technique introduced in (5). Note that, due to the tight scheduling constraints and the need for predictability in real-time systems, the task *WCET* is applied to the scheduling constraints in order to check that tasks are able to meet their deadlines.

In-processor constraints can be expressed by the following inequalities:

$$\forall t_i, t_j \in V_T:$$
$$s_i + ex_i \leqslant s_j + (1 - b_{ij} + sep_{ij}).MAX\_VAL,$$
$$s_j + ex_j \leqslant s_i + (b_{ij} + sep_{ij}).MAX\_VAL$$

where b_{ij} equals 1 if t_i is executed before t_j; otherwise, it equals 0 as stated in (11). Lastly, the following constraints represent start time constraint and deadline constraint for each task.

$$\forall t_i \in V_T:$$
$$s_i \geqslant 0,$$
$$s_i + ex_i \leqslant deadline_i \tag{11}$$

(2) *Worst-case reliability*: As mentioned in Section 3.3, *WCR* of each task t_i when it is executing on p_j at frequency f can be written as:

$$R_{ij}^f = e^{-\lambda(f).\frac{x_{ij}^{n_{ij}}}{f}}$$

Now we can write the WCR of whole system as a product of all task reliabilities:

$$R = \prod_{t_i \in V_T} R_i = e^{-(\sum_{t_i \in V_T} \sum_{p_j \in P} \sum_{f_k \in L} \gamma_{ij}^k.\sigma_{ij}^k)}$$

where σ_{ij}^k represents $\lambda(f_k).\frac{x_{ij}^{n_{ij}}}{f_k}$. The *WCR* constraint can be expressed as $R > R_0$, that is:

$$\sum_{t_i \in V_T} \sum_{p_j \in P} \sum_{f_k \in L} \gamma_{ij}^k.\sigma_{ij}^k \leqslant \ln\left(\frac{1}{R_0}\right)$$

(3) *Number of VFIs*: Voltage–frequency level assignment can be viewed as coloring the vertexes of the grid graph. Let the routers be represented by vertexes, and the voltage–frequency levels be represented by colors. Construct new graph by defining new edges between each two adjacent vertexes with the same color. The problem of finding the number of voltage–frequency islands is the same as finding the number of connected components in this new constructed graph as (12). The variable κ is used to represent the new constructed graph.

$$\forall a_{xy} \in A_R, \quad k_{xy} = \begin{cases} 1 & \exists f_k \in L,\ \theta_{xk} = \theta_{yk} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

which means:

$$\forall a_{xy}, \quad k_{xy} = \sum_{f_k \in L} a_{xy}^{kk}$$

The number of connected components in a graph equals to the number of vertexes minus the number of edges in the spanning forest. The spanning forest is a set of spanning trees, one for each connected component. Since there are two

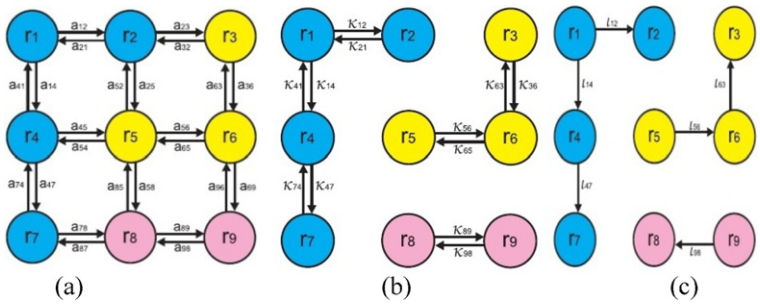

Fig. 4. Constructing a spanning forest: (a) colored graph G; (b) new constructed graph G; (c) spanning forest.

edges between each two adjacent routers r_x and r_y, in our architecture characterization graph (e.g., a_{xy} and a_{yx}), we use rooted spanning trees instead of spanning trees as in Fig. 4. A rooted tree is a directed acyclic graph in which all edges point away from the root. The root vertex has no parent (e.g., a vertex with no entering edges) and all other vertexes have exactly one parent. In order to represent the spanning directed forest we introduce new variable ι_{xy} which is 1 if κ_{xy} is used in a spanning directed forest and 0 otherwise.

As discussed before, ι_{xy} must be 0 when κ_{xy} equals 0. To enforce this constraint, following inequality needs to be satisfied:

$$\forall a_{xy} \in A_R, \quad l_{xy} \leqslant k_{xy}$$

Additionally, each vertex must be a tail of at most one edge. This constraint can be ensured by the following condition:

$$\forall r_y \in V_R, \quad \sum_{e_{xy} \in A_R} l_{xk} \leqslant 1$$

Furthermore, there must be a constraint to prevent loop formation. The number of edges in any given set of vertexes chosen from a spanning forest is at most one less than the number of vertexes in the given set. So, loops can be prevented by:

$$\forall S \subseteq V_R,\ S \neq \emptyset, \quad \sum_{\{a_{xy} | x,y \in S\}} l_{xy} \leqslant |S| - 1$$

where S represents a non-empty subset of V_R and $|S|$ is the size of S. Finally, for a large constant C, the following constraint assures that l is spanning and covers all vertexes in the connected components of new constructed graph:

$$\forall r_y \in V_R:$$
$$c \times \sum_{a_{xy} \in A_R} (l_{xy} + l_{yx}) \geqslant \sum_{a_{zy} \in A_R} (k_{zy} + k_{yz})$$

Note that, C can be set here to be equal to $2 \times |V_R|$ due to the fact that each node has at most $|V_R| - 1$ neighbors which implies that $\sum_{a_{zy} \in A_R} (k_{zy} + k_{yz})$ cannot exceed $2 \times |V_R|$. The constraint on the number of VFIs can now be formulated as:

$$|V_R| - \sum_{a_{xy} \in A_R} l_{xy} \leqslant \pi$$

where $|V_R|$ and π represent the number of routers and the bound on maximum number of VFIs, respectively.

(4) *Link bandwidth and communication latency*: The link bandwidth and communication latency are considered as the key performance metrics in NoC communication architecture analysis and optimization [4]. The link bandwidth constraint states that the total traffic passing through a network link does not exceed the link capacity. The link bandwidth requirements must be satisfied for each router link which can be expressed as the following inequality:

$$\forall a_{xy} \in A_R, \quad \sum_{c_{ij} \in E_T} bw_{ij}.\delta_{xy}^{ij} \leqslant BW_{xy}$$

where bw_{ij} and BW_{xy} represent $bw(c_{ij})$ and $BW(a_{xy})$, respectively.

Communication latency is a crucial real-time constraint that needs to be met as well. As mentioned above, $\sigma(c_{ij})$ represent the maximum tolerable latency for the task communication c_{ij} which is given in the number of hop count. For each

task communication c_{ij}, $\delta_{xy}^{ij} = 1$ indicates that a_{xy} is involved in delivering data communication volume between t_i and t_j. Hence the inequality given in (13) ensures that the latency constraint is satisfied after the mapping:

$$\forall c_{ij} \in E_T, \quad \sum_{a_{xy} \in A_R} \sigma_{xy}^{ij} \leqslant \sigma_{ij} \tag{13}$$

5. Heuristic solution

Since the MILP formulation presented above is *NP*-hard we now provide a randomized sampling-based heuristic. The proposed heuristic provides near optimal solution in a significantly low computation time compared to MILP-based approach.

Algorithm 1 Rounding by sampling-based heuristic

Mapping, scheduling, and VFI partitioning GALS-based approach

Input: The parameters and corresponding MILP formulation specified in Section 4.
Output: Functions M_1, M_2, M_3, and M_4, and the starting times for tasks $t_i \in V_T$ specified in Table 1.

Relax all the integer constraints in the MILP formulation.
Solve the LP using CPLEX.
cost(*bestSolution*) $\leftarrow \infty$
for *iter* $= 0$ **to** *numofiter* **do**
  for all $t_i \in V_T$ **do**
    repeat
      Independently round x_{ij} to 1 with probability as its value.
    until $x_{ij} = 1$ **for** some $p_j \in P$
  end for
  for all $p_j \in P$ **do**
    repeat
      Independently round y_{jk} to 1 with probability as its value.
    until $y_{jk} = 1$ **for** some $f_k \in L$
  end for
  Round z variables using rounding by sampling technique presented in [51] for rounding fractional matchings.
  Round other variables following correlation constraints.
  if (any constraint Violations) **then**
    Eliminate constraint violations by modifying the rounding of x_{ij}, y_{jk}, and z_{jm} variables for the corresponding $p_j \in P$.
  end if
  if (*cost*(*bestSolution*) $>$ *cost*(*currentSolution*)) **then**
    best $\leftarrow$ *currentSolution*
  end if
end for
return *bestSolution*

Our heuristic takes as input the parameters specified in Section 4 and the corresponding MILP formulation and produces functions M_1, M_2, M_3, and M_4 specified in Table 1 and the starting times for tasks $t_i \in V_T$ as the output. In the first step we relax all the integer constraints in the MILP formulation and solve the relaxed *Linear Problem* (*LP*) using CPLEX [54]. The algorithm then runs in an iterative manner. At each iteration, the best solution is compared with the current solution found in this iteration and updated accordingly. Increasing the number of iterations increases the probability of finding a better solution, but it increases the execution time of the heuristic algorithm.

The feasible solution of the LP, as opposed to the MILP, could not be guaranteed to be an integral feasible solution. Hence a rounding scheme is needed to transform the linear program solution to an integral one. This solution must satisfy all constrains specified in Section 3.4. Therefore, a sufficient approximate solution for the original problem is desired.

The presented above algorithm uses both independent randomized rounding and rounding by sampling techniques introduced in [51]. The rounding by sampling technique is based on sampling from a maximum entropy distribution over the combinatorial structures hidden in the feasible solutions. The main implication of this technique is that it keeps the combinatorial structures intact, while aiming at preserving the solutions quantitatively. The technique transforms a feasible solution of LP to an integral one while preserving marginal probabilities imposed by the linear values obtained from solving the LP. In [51], the author provides an intuitive combinatorial algorithm that samples a fractional matching from the maximum entropy distribution while using the least biased Bayesian update rule. The interested readers are referred to [51] for more detail implementation of the method. Algorithm 1 applies these techniques in order to achieve a near optimal solution of the main problem.

6. Experimental results

In this section we present the experimental results to demonstrate the effectiveness of our proposed approach in minimizing the energy consumption. The experimental results are gathered from benchmark applications (*office-automation*,

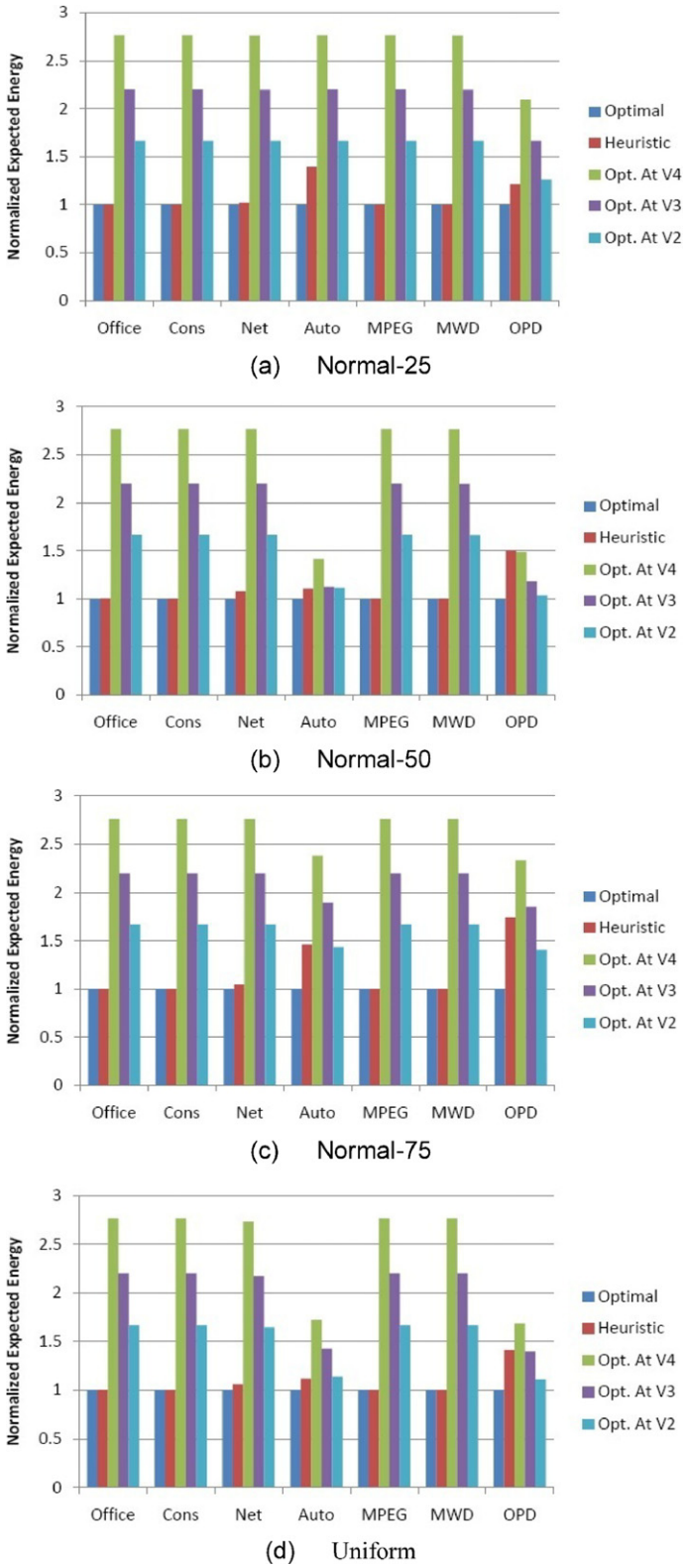

Fig. 5. Comparison of the normalized expected energy consumption values using the MILP approach, our heuristic and the fixed voltage–frequency level approach.

Table 2
Benchmark applications and mesh dimension.

Benchmark	Task nodes	Task edges	Network size
Office-automation	5	5	2×2
Consumer	12	12	3×3
Networking	13	9	3×3
Auto-industry	24	21	4×4
MPEG	12	13	3×3
MWD	12	11	3×3
OPD	16	17	4×4

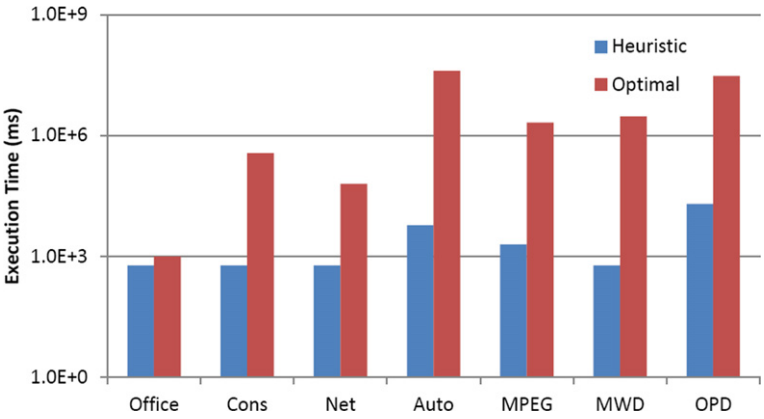

Fig. 6. Comparison of the execution time values using the MILP approach and our heuristic (log scale).

consumer, *networking*, and *auto-industry*) collected from the Embedded System Synthesis Benchmarks Suite (E3S) [52] and three real applications MPEG4 [53], Multi-Window Display (MWD) [53] and Object Place Decoder (OPD) [18].

E3S benchmark suite was designed for use in embedded systems synthesis research. In particular, it was designed for use in automated system-level allocation, assignment, and scheduling studies. It contains 17 processors which are characterized based on the measured execution times of 47 tasks, power quantities derived from processor datasheets, and additional information. In addition, E3S contains communication resources that model a number of different buses. There is one task set for each of the four application suite: *office-automation, consumer, networking,* and *auto-industry*. The number of tasks, communication edges between them and the mesh-based network sizes are depicted in Table 2.

The following discrete voltage levels are used for voltage level assignment: $V_0 = 1.9V$, $V_1 = 2.3V$, $V_2 = 2.5V$, $V_3 = 3.3V$, and $V_4 = 3.6V$. The power consumption estimation of the tasks on the PEs is provided in the benchmark for E3S benchmark applications. We use the same approach as [19] for estimating the power consumption of the real applications tasks. The estimated power consumption of each task is mentioned in the E3S benchmark suite. Tasks in E3S benchmark suite can be classified as computing intensive, I/O read-write or memory read-write intensive tasks. Applying similar classifications to the real application tasks, we used similar values for power consumption given in the E3S benchmark suit as an estimation of the power consumption values for the tasks in the real applications.

The BCET for tasks is assumed to be 50% of their WCETs. We suppose that there are 10 different execution times within BCET and WCET of a task. We consider the same two different probability distributions as [49] regarding the execution times of tasks. For uniform distribution, the probability of a task to take any one of its execution times is the same and equal to 0.1. The second one is the modified discrete normal distribution. We consider three cases with the average of $\mathrm{BCET} + (\mathrm{WCET} - \mathrm{BCET})/4$, $(\mathrm{WCET} - \mathrm{BCET})/2$, and $\mathrm{BCET} + 3(\mathrm{WCET} - \mathrm{BCET})/4$ which are represented as Norm25, Norm50, and Norm75, respectively (see Fig. 5).

All the experiments were performed on an Intel® Core™ i7 CPU 860 2.80 GHz PC with 4.00 GB RAM. The rounding by sampling-based heuristic is implemented in C++. The MILP optimal solution and the corresponding relaxed LP solution as a part of the heuristic are achieved using ILOG CPLEX 11.1 Concert technology. As depicted in Fig. 6, the provided heuristic is able to find near optimal solution within a few seconds. The extra time is negligible compared to that required to solve the MILP for finding the optimal solution.

The quality of our proposed heuristic is depicted in Fig. 5. In all cases our heuristic can find a near optimal solution. On average for all benchmark applications, our heuristic solution can save 56%, 45%, and 28% energy consumption over the optimal solution for fixed voltage at V_4, V_3, and V_2, respectively. Note that, operating at lower voltage–frequency levels would result in missing the task deadlines due to the slow execution times.

We have assumed that transient faults obeys the Poisson distribution [43] and the average fault rate at the maximum voltage–frequency level, λ_0, is equal to 10^6. This rate corresponds to 100,000 FITs (failure in time, in terms of errors per billion hours of use) per megabit which is a reasonable fault rate as reported in [46]. Taking the effects of voltage frequency

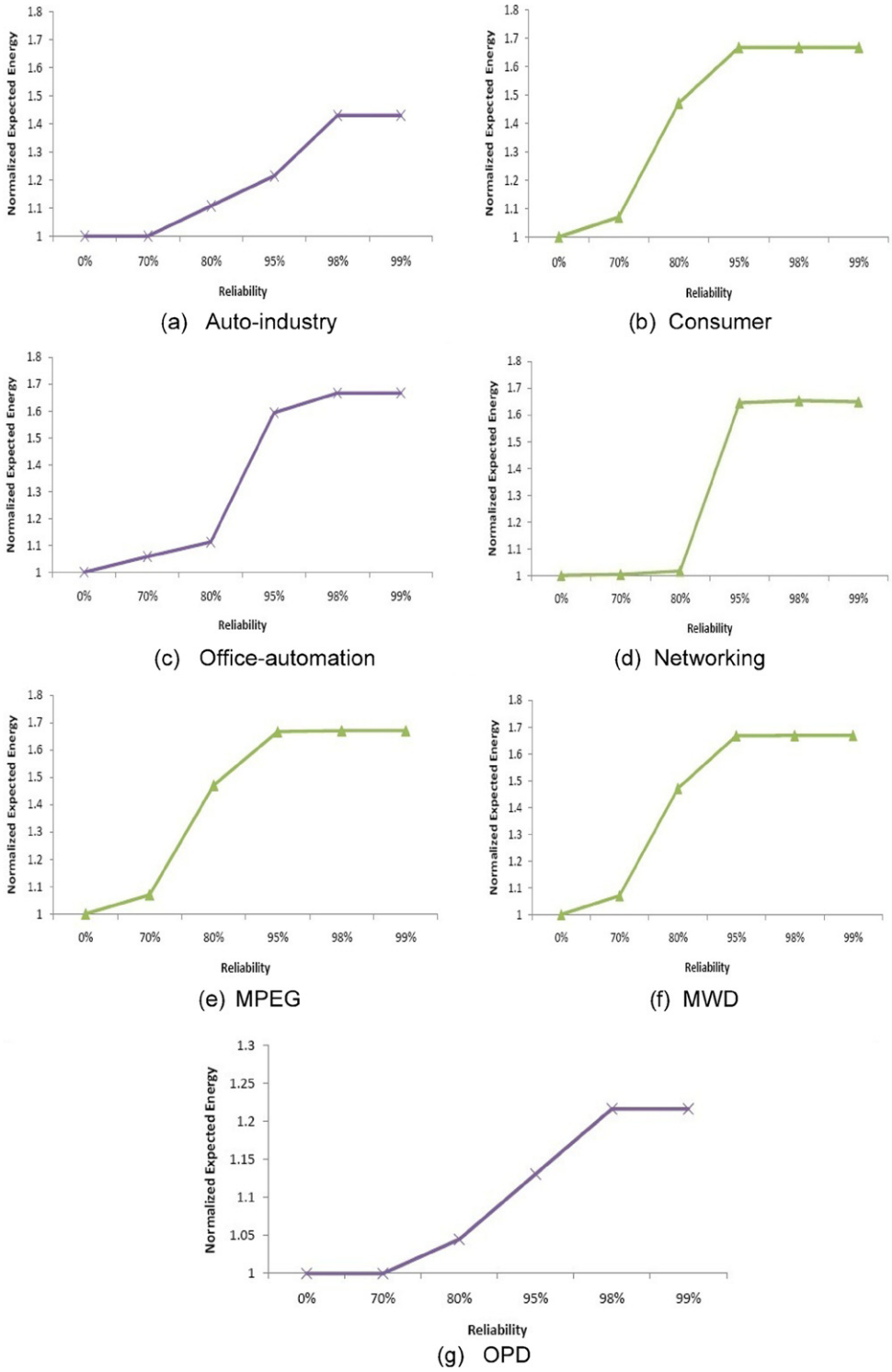

Fig. 7. Normalized expected energy consumptions vs. the constrained Worst-Case Reliability (WCR) for E3S benchmark and three real applications (MPEG, MWD, OPD).

scaling on transient fault rates into account, the exponent in the exponential fault model (see Section 3.3) is assumed to be $d = 6$ (or 7). These assumptions have been previously presented in [49].

Fig. 7 depicts normalized expected energy consumption which has been computed in Eq. (4) vs. the constrained worst-case reliability for different benchmark applications. The graphs have shown in the Fig. 7 verifies that the applications

consume more energy as they require higher worst-case reliability. The normalized energy consumption grows gradually when we have no constraint on worst-case reliability until reaching to 70% required worst-case reliability for most benchmark applications. Then, it increases dramatically from 70% required worst-case reliability to 95% required worst-case reliability. When the applications request 99% worst-case reliability they consume 1.7 times energy compared to the case that we have no constraint on reliability.

7. Conclusion

In this paper we formulated energy-aware mapping problem of a real-time application with stochastic execution times onto multi-core systems, scheduling tasks on processors, and assigning voltage–frequency levels to Processing Elements (PEs) as a Mixed Integer Linear Programming (MILP) in GALS-based Network-on-Chip (NoC). Due to the *NP*-hardness of the energy-aware application mapping, task scheduling, and voltage–frequency islands partitioning problem we presented a novel rounding by sampling-based heuristic algorithm to achieve a near optimal solution to main problem. Experimental results based on E3S benchmark suites and some real applications reveal our proposed heuristic demonstrate that using multiple voltage–frequency levels is more efficient than using fixed voltage–frequency level. Experimental results also show that, preserving the required worst-case reliability of the system could increase the energy consumption by 70% in some scenarios.

References

[1] U.Y. Ogras, R. Marculescu, P. Choudhary, D. Marculescu, Voltage–frequency island partitioning for GALS-based networks-on-chip, in: 44th Annual Design Automation Conference DAC, San Diego, CA, USA, Jun. 2007, pp. 249–262.
[2] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, IEEE Computer 35 (1) (2002) 70–78.
[3] G.D. Micheli, L. Benini, Networks on Chips, Morgan Kaufman, 2006.
[4] R. Marculescu, U.Y. Ogras, L.S. Peh, N.E. Jerger, Y. Hoskote, Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 28 (1) (2009) 3–21.
[5] C.L. Chou, U.Y. Ogras, R. Marculescu, Energy-and performance-aware incremental mapping for networks on chip with multiple voltage levels, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 27 (10) (2008) 1866–1879.
[6] P. Ghosh, A. Sen, A. Hall, Energy efficient application mapping to NoC processing elements operating at multiple voltage levels, in: 3rd ACM/IEEE International Symposium on Networks-on-Chip, San Diego, CA, USA, May 2009, pp. 80–85.
[7] X. Wang, M. Yang, Y. Jiang, P. Liu, A power-aware mapping approach to map IP cores onto NoCs under bandwidth and latency constraints, ACM Trans. Archit. Code Optim. TACO 7 (1) (2010).
[8] A. Hansson, K. Goossens, A. Radulescu, A unified approach to mapping and routing on a network-on-chip for both best-effort and guaranteed service traffic, Hindawi VLSI Design 2007 (2007) 1–17.
[9] W. Jang, D. Ding, D.Z. Pan, Voltage and frequency island optimizations for many-core/networks-on-chip designs, in: International Conference on Green Circuits and Systems ICGCS, Jun. 2010, pp. 217–220.
[10] A. Iyer, D. Marculescu, Power and performance evaluation of globally asynchronous locally synchronous processors, in: 29th Annual International Symposium on Computer Architecture ISCA, May 2002, pp. 158–168.
[11] G. Semeraro, G. Magklis, R. Balasubramonian, D.H. Albonesi, S. Dwarkadas, M.L. Scott, Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling, in: 8th International Symposium on High-Performance Computer Architecture ISHPC, May 2002, pp. 29–40.
[12] G. Semeraro, D.H. Albonesi, G. Magklis, Hiding synchronization delays in GALS processor micro architecture, in: 10th International Symposium on Asynchronous Circuits and Systems, May 2004, pp. 159–169.
[13] K. Niyogi, D. Marculescu, Speed and voltage selection for GALS systems based on voltage/frequency islands, in: Asia and South Pacific Design Automation Conference ASP-DAC, May 2005, pp. 292–297.
[14] D. Marculescu, E. Talpes, Variability and energy awareness: A microarchitecture-level perspective, in: Asia and South Pacific Design Automation Conference ASP-DAC, Jun. 2005, pp. 11–16.
[15] D.M. Chapiro, Globally-asynchronous locally synchronous systems, PhD dissertation, Stanford University, Oct. 1984.
[16] J. Hu, R. Marculescu, Energy- and performance-aware mapping for regular NoC architectures, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 24 (4) (2005) 551–562.
[17] C.L. Chou, R. Marculescu, Contention-aware application mapping for network-on-chip communication architectures, in: International Conference on Computer Aided Design, Lake Tahoe, CA, USA, Oct. 2008, pp. 164–169.
[18] S. Murali, G.D. Micheli, Bandwidth constrained mapping of cores onto NoC architectures, in: Design, Automation and Test in Europe Conference and Exhibition DATE, vol. 2, Feb. 2004, pp. 20896–20902.
[19] P. Ghosh, A. Sen, Energy efficient mapping and voltage islanding for regular NoC under design constraints, Internat. J. High Perform. Systems Archit. 2 (3/4) (2010) 132–144.
[20] R. Ernst, W. Ye, Embedded program timing analysis based on path clustering and architecture classification, in: The Intl Conference on Computer-Aided Design, 1997, pp. 598–604.
[21] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, C.R. Das, Design and analysis of an NoC architecture from performance, reliability and energy perspective, in: ACM Symposium on Architecture for Networking and Communications Systems ANCS, Oct. 2005, pp. 173–182.
[22] V. Narayanan, Y. Xie, Reliability concerns in embedded system designs, IEEE Computer 39 (1) (2006) 118–120.
[23] S. Manolache, P. Eles, Z. Peng, Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC, in: 42nd Annual Design Automation Conference DAC, Jul. 2005, pp. 266–269.
[24] S. Murali, T. Theocharides, N. Vijaykrishnan, M. Irwin, L. Benini, G.D. Micheli, Analysis of error recovery schemes for networks-on-chips, IEEE Des. Test Comput. 22 (5) (2005) 433–442.
[25] S.S. Mukherjee, J. Emer, S.K. Reinhardt, The soft error problem: An architectural perspective, in: 11th International Symposium on High-Performance Computer Architecture HPCA-11, Feb. 2005, pp. 243–247.
[26] K.L. Shepard, V. Narayanan, Noise in deep submicron digital design, in: IEEE/ACM International Conference on Computer Aided Design ICCAD, Nov. 1996, pp. 524–531.

[27] V. Degalahal, L. Lin, V. Narayanan, M. Kandemir, M.J. Irwin, Soft errors issues in low-power caches, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 13 (10) (2005) 1157–1166.
[28] D. Zhu, R. Melhem, D. Mosse, The effects of energy management on reliability in real-time embedded systems, in: IEEE/ACM International Conference on Computer Aided Design ICCAD, San Jose, CA, USA, Nov. 2004, pp. 35–40.
[29] W. Jang, D. Ding, D.Z. Pan, A voltage–frequency island aware energy optimization framework for networks-on-chip, in: IEEE/ACM International Conference on Computer Aided Design ICCAD, San Diego, CA, USA, Nov. 2008, pp. 264–269.
[30] K. Srinivasan, K.S. Chatha, A technique for low energy mapping and routing in network-on-chip architectures, in: International Symposium on Low Power Electronics and Design ISLPED, New York, USA, May 2005, pp. 387–392.
[31] J. Hu, R. Marculescu, Energy-aware mapping for tile-based NoC architectures under performance constraints, in: Proc. Asia and South Pacific, Design Automation Conference, 2003, pp. 233–239.
[32] J. Hu, et al., Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints, in: Design, Automation and Test in Europe Conference and Exhibition DATE, Los Alamitos, CA, USA, 2004, pp. 234–239.
[33] E.N. Elnozahy, M. Kistler, R. Rajamony, Energy-efficient server clusters, in: Power-Aware Computer Systems-PACS, 2002, pp. 179–196.
[34] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, R. Rajkumar, Critical power slope: understanding the runtime effects of frequency scaling, in: 16th Annual ACM International Conference on Supercomputing ICS, New York, NY, USA, Jun. 2002, pp. 35–44.
[35] T. Pering, T.D. Burd, R.W. Brodersen, The simulation and evaluation of dynamic voltage scaling algorithms, in: International Symposium on Low Power Electronics and Design ISLPED, Monterey, CA, USA, Aug. 1998, pp. 76–81.
[36] T.D. Burd, R.W. Brodersen, Energy efficient CMOS microprocessor design, in: 28th Hawaii International Conference on System Sciences HICSS, Hawaii, USA, Jan. 1995, pp. 288–297.
[37] S.M. Martin, K. Flautner, T. Mudge, D. Blaauw, Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads, in: IEEE/ACM International Conference on Computer Aided Design ICCAD, San Jose, CA, USA, Nov. 2002, pp. 721–725.
[38] T.T. Ye, L. Benini, G.D. Micheli, Analysis of power consumption on switch fabrics in network routers, in: 39th Annual Design Automation Conference DAC, Jun. 2002, pp. 524–529.
[39] U.Y. Ogras, R. Marculescu, D. Marculescu, E.G. Jung, Design and management of voltage–frequency island partitioned networks-on-chip, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 17 (3) (2009) 330–341.
[40] D.E. Duarte, N. Vijaykrishnan, M.J. Irwin, A clock power model to evaluate impact of architectural and technology optimizations, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 10 (6) (2002) 844–855.
[41] T.D. Burd, R.W. Brodersen, Design issues for dynamic voltage scaling, in: International Symposium on Low Power Electronics and Design ISLPED, New York, NY, USA, Jul. 2000, pp. 9–14.
[42] T. Chelcea, S.M. Nowick, A low-latency fifo for mixed-clock systems, in: IEEE Computer Society Workshop on VLSI, Apr. 2000, pp. 119–126.
[43] Y. Zhang, K. Chakrabarty, V. Swaminathan, Energy-aware fault tolerance in fixed-priority real-time embedded systems, in: International Conference on Computer Aided Design CAD, San Jose, CA, USA, Nov. 2003, pp. 209–214.
[44] N. Seifert, D. Moyer, N. Leland, R. Hokinson, Historical trend in alpha-particle induced soft error rates of the AlphaTM microprocessor, in: 39th Annual IEEE International Reliability Physics Symposium, 2001, pp. 259–265.
[45] J.F. Ziegler, Trends in electronic reliability: Effects of terrestrial cosmic rays, http://www.srim.org/SER/SERTrends.htm.
[46] P. Hazucha, C. Svensson, Impact of CMOS technology scaling on the atmospheric neutron soft error rate, IEEE Trans. Nucl. Sci. 47 (6) (2000) 2586–2594.
[47] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, L. Alvisi, Modeling the effect of technology trends on the soft error rate of combinational logic, in: International Conference on Dependable Systems and Networks DSN, 2002, pp. 389–398.
[48] J.F. Ziegler, Terrestrial cosmic ray intensities, IBM J. Res. Dev. 42 (1) (1998) 117–139.
[49] D. Zhu, H. Aydin, J.J. Chen, Optimistic reliability aware energy management for real-time tasks with probabilistic execution times, in: 29th IEEE Real-Time Systems Symposium RTSS, Barcelona, Spain, Dec. 2008, pp. 313–322.
[50] H. Yang, S. Ha, ILP based data parallel multitask mapping/scheduling technique for MPSoC, in: International SoC Design Conference ISOCC, Busan, Nov. 2008, pp. 134–137.
[51] A. Asadpour, Rounding by sampling, PhD dissertation, Stanford University, Jun. 2010.
[52] R. Dick, Embedded system synthesis benchmarks suite (E3S), http://ziyang.eecs.umich.edu/~dickrp/e3s/.
[53] K. Srinivasan, K.S. Chatha, A low complexity heuristic for design of custom network-on-chip architectures, in: Design, Automation and Test in Europe Conference and Exhibition DATE, 2006, pp. 130–135.
[54] CPLEX 11.1ILOG: http://www.ilog.com/product/cplex/, 2011.