Energy-efficient Static Task Scheduling on VFI based NoC-HMPSoCs for Intelligent Edge Devices in Cyber-Physical Systems

UMAIR ULLAH TARIQ* and HAIDER ALI*, University of New South Wales
LU LIU, (corresponding author) University of Leicester, United Kingdom
JOHN PANNEERSELVAM, University of Derby, United Kingdom
XIAOJUN ZHAI, (corresponding author) University of Essex, United Kingdom

The interlinked processing units in modern Cyber-Physical Systems (CPS) creates a large network of connected computing embedded systems. Network-on-Chip (NoC) based Multiprocessor System-on-Chip (MPSoC) architecture is becoming a de-facto computing platform for real-time applications due to its higher performance and Quality-of-Service (QoS). The number of processors has increased significantly on the multiprocessor systems in CPS therefore, Voltage Frequency Island (VFI) recently adopted for effective energy management mechanism in the large scale multiprocessor chip designs. In this paper, we investigated energy-efficient and contention-aware static scheduling for tasks with precedence and deadline constraints on intelligent edge devices deploying heterogeneous VFI based NoC-MPSoCs (VFI-NoC-HMPSoC) with DVFS-enabled processors. Unlike the existing population-based optimization algorithms, we proposed a novel population-based algorithm called ARSH-FATI that can dynamically switch between explorative and exploitative search modes at run-time. Our static scheduler ARHS-FATI collectively performs task mapping, scheduling, and voltage scaling. Consequently, its performance is superior to the existing state-of-the-art approach proposed for homogeneous VFI based NoC-MPSoCs. We also developed a communication contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling algorithm and gradient descent inspired voltage scaling algorithm called Energy Gradient Decent (EGD). We introduced a notion of Energy Gradient (EG) that guides EGD in its search for islands voltage settings and minimize the total energy consumption.

Conducted the experiments on 8 real benchmarks adopted from Embedded Systems Synthesis Benchmarks (E3S). Our static scheduling approach ARSH-FATI outperformed state-of-the-art technique and achieved an average energy-efficiency of ~ 24% and ~ 30% over CA-TMES-Search and CA-TMES-Quick respectively.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks →** Network reliability.

Additional Key Words and Phrases: CPS, SNS, Task, DAG, Mapping, Scheduling, Contention, Heterogeneous, VFI-NoC-HMPSoCs, Energy-efficiency

*Both authors contributed equally to this research.

Authors' addresses: Umair Ullah Tariq, tariqu@cse.unsw.edu.au; Haider Ali, h.ali@derby.ac.uk, University of New South Wales, Sydney, Australia, 2052; Lu Liu, (corresponding author) University of Leicester, Leicester, United Kingdom, l.liu@leicester.ac.uk; John Panneerselvam, University of Derby, Derby, United Kingdom; Xiaojun Zhai, (corresponding author) University of Essex, Colchester, United Kingdom.

XXXX-XXXX/2019/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Umair Ullah Tariq, Haider Ali, Lu Liu, John Panneerselvam, and Xiaojun Zhai. 2019. Energy-efficient Static Task Scheduling on VFI based NoC-HMPSoCs for Intelligent Edge Devices in Cyber-Physical Systems. 1, 1 (June 2019), 23 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Key aspect of the Smart Networked Systems (SNS) is interlinking physical and virtual worlds using sensors, digital devices, and actuators. These systems that tightly interlink physical and cyber worlds referred to as Cyber-Physical Systems (CPS). Specifically, a CPS is the "integration of computation with physical processes " [44, 51]. In the modern CPS real-world is monitored in real-time through intelligent edge devices and relevant data is transferred into the cyberspace for further cyber services, applications and, visualizations. The interlinked processing units in CPS produce a large network of connected computing embedded systems [40, 44]. Wireless Sensors Network (WSN) is one of the technologies that is core to the concept of SNS. WSN deploying intelligent edge devices is widely used for numerous smart applications e.g. manufacturing process automation, security, and surveillance [5, 8, 31].

The ever-growing demand for real-time applications has initiated to use Multiprocessor System-on-Chips (MPSoCs) in modern embedded systems for CPS/WSN applications [2, 16, 44]. MPSoCs integrate I/O units, memory, processors, and busses on a single silicon chip. Moreover, MPSoCs guarantee high performance, exceptional reliability, and remarkable Quality-of-Service (QoS). These overwhelming qualities make MPSoCs an ideal computing platform for numerous real-time applications [2, 52]. For example, MPSoC systems play a vital role in CPS for mobile systems such as automated robots autonomous cars, telecommunication, environmental/industrial monitoring, and automotive robots [15, 46]. The use of MPSoCs is not limited to CPS and Internet-of-Things (IoT) applications nowadays, they are being widely used in consumer electronic devices such as smart-phones, tablets, and personal computers [14, 15]. Xilinx Zynq® UltraScale+™ MPSoCs and Renesas R-Car H3 nona-core are the examples of MPSoCs commercially available for CPS applications. Similarly Samsung Exynos 9810, Apple A11 Bionic, Xilinx Zynq® UltraScale+™ RFSoCs, Intel® Stratix® 10, Tilera TILE-Gx72™, and EZchip TILE-Mx100™ are high performance multiprocessor architectures for consumer digital systems [2, 45].

In the near future multiprocessor systems will consist of hundreds of processors consequently, bus-based traditional MPSoCs will become a bottleneck because of their congestion and scalability issues. Network-on-Chip (NoC) based interconnects alternatively offer an improved scalability with higher flexibility [2, 7]. Recently Voltage Frequency Island (VFI), Globally Asynchronous Locally Synchronous (GALS) paradigm is introduced to NoC interconnect where the tiles are partitioned into islands. Each island in MPSoC system is optimized with its own threshold voltage, operating frequency, and supply voltage to reduce the overall energy consumption [21]. VFI based NoC-MPSoC is a suitable choice of selection for data-extensive applications due to its higher throughput, energy-efficiency, and lower complexity [26]. VFI based MPSoCs curtail the overall system's complexity by reducing the number of voltage level converters (VLCs) and multiple clock first-in-first-out (MCFIFO) circuitry requirement [33]. VFI based MPSoC with fine-grained power management capability can seamlessly combine with task scheduling algorithm in order to optimize over all system's energy. It is worth noticing that state-of-the-art commercially available multiprocessor systems e.g. Intel Itanium i7 and IBM Power 7 series use VFI based MPSoC architectures [22, 23].

Static task scheduling describes a mechanism to suitably allocate tasks on processors before the embedded systems run while fulfilling certain obligations such as energy consumption and/or performance [11, 24]. Proper scheduling approach can drastically increase the reliability of an embedded system [17]. Static task schedulers can be used in applications e.g. surveillance, human

recognition, person tracking, gait analysis, advanced healthcare, crowed and traffic monitoring [1, 42]. Meeting the constraints such as deadlines, performance, and QoS of the battery constrained multiprocessor systems in CPS applications play a critical role [49]. For example, missing a deadline for an application can reduce QoS and performance [15]. Moreover, energy-aware computing is a critical challenging facet in modern embedded systems because higher energy consumption not only causes an increased carbon dioxide (CO_2) emission but also limits their life [19, 41, 49]. Dynamic Voltage and Frequency Scaling (DVFS) is a technique applied to MPSoC architectures in order to reduce the power overhead while maintaining the desired performance. In DVFS, the energy-efficiency is achieved by reducing the supply voltage and processor's clock frequency [6].

In this paper, we investigated first time ever the problem of energy-efficient and contention-aware static scheduling on the edge computing devices using heterogeneous VFI based NoC-MPSoC (VFI-NoC-HMPSoC) system with DVFS-enabled processor for a set of tasks with precedence constraints and deadline. Our main contributions and innovations are given as follows:

(1) We performed task mapping, ordering, and voltage scaling in an integrated way using a novel search based meta-heuristic called ARSH-FATI. Our static scheduler also considers energy performance profiles of the processors, voltage levels within each processor, contention at the NoC links, and inter-VFI communications during task scheduling.
(2) Our meta-heuristic ARSH-FATI can dynamically switch between different search modes to achieve a satisfactory trade-off between explorative and exploitative search during run-time. Moreover, we presented a new a contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling algorithm and gradient descent inspired Energy Gradient Decent (EGD) voltage scaling technique.
(3) We compared the energy performance of our static scheduler ARSH-FATI with state-of-the-art CA-TMES-Search and CA-TMES-Quick [16] energy management approaches using 8 real benchmarks adopted from E3S benchmark suit. Our meta-heuristic based static task scheduler achieved an average energy-efficiency of $\sim$ 24% and $\sim$ 30% respectively.

We organize the rest of the paper as follows: Section 2 reviews the existing search based algorithms and state-of-the-art energy optimization approaches. Section 3 presents the application, computing platform, and power models. In Section 4 we discuss our static contention-aware energy optimization scheme. The simulation results on different benchmarks are discussed in Section 5, while in Section 6 we conclude this paper.

2 LITERATURE REVIEW

Task mapping and scheduling on multiprocessor architectures is an NP-hard problem and different heuristics have been proposed based on mathematical formulation such as Integer Linear Programming (ILP), Non Linear Programming (NLP), Linear Programming (LP) and Mixed Integer Linear Programming (MILP). Similarly, search based heuristic algorithms using selection, crossover, mutation, and elitism are also widely deployed. The popular examples of these search based algorithms are Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). Among these algorithms, GA is widely adopted for task mapping and scheduling [2, 34]. These evolutionary algorithms belong to stochastic generate and test algorithms which are based on (1) exploration of the search space and (2) exploitation of the promising information already found. Exploration primarily describes the ability of an algorithm to discover the unseen regions while exploitation demonstrates the capability to proceed in the desired direction for improvement. For example in GAs, mutation and crossover are hypothetically considered to perform exploration and exploitation respectively [10, 47]. However, there is strong criticism that crossover does not possess a competitive advantage over mutation [47]. Nevertheless, these search based algorithms

fail to efficiently exploit the available chunk of information i.e. schemata. Moreover, exploration and exploitation are the two opposing forces and a well-found balance between them determines the success of a search based algorithm.

Motivated by the fact that multiprocessor systems are reliable and high-performance computing platforms for edge devices in CPS, several researchers have investigated task mapping and scheduling. One of the earliest work in scheduling includes a scheme developed by Olafsson in order to efficiently distribute the tasks i.e. workload on heterogeneous multiprocessor system [38]. Aydin et al. [4] provided DVFS based energy-efficient scheduling algorithm with $O(n^2logn)$ complexity for independent real-time tasks with different power consumption characteristics on multiprocessors system. They formulated the scheduling problem as an NLP and assigned constant speed to the tasks while maintaining the optimality. Other energy management studies used DVFS technique for energy optimization. For example, Zhang et al. [50] presented a meta-heuristic scheduling algorithm called Shuffled Frog Leaping Algorithm (SFLA) by integrating the gains of PSO and Memetic algorithms while compared the energy-efficiency of SFLA with GA. Kumar and Vidyarth [25] integrated task mapping and voltage assignment in a single optimization loop of GA. They used DVFS technique to assign voltages to the tasks such that the dynamic energy consumption is reduced with an acceptable performance trade-off. Wang et al. performed preemptive periodic independent tasks scheduling using Discrete Event System (DES) supervisory control [48]. Liu and Qi mapped the tasks using Weighted Earliest Finish Time (WEFT) algorithm and executed the tasks with lowest possible earliest completion time [30]. These investigations reduced energy consumption of independent tasks running on MPSoC architectures without explicitly considering the precedence constraints.

Huang et al. [18] used an extended ILP formulation for energy optimization on heterogeneous NoC based MPSoC systems and developed a heuristic called Simulated Annealing with Timing Adjustment (SA-TA). Fundamentally SA-TA optimizes energy consumption by reaching near to the global optimum under timing constraints. Gammoudi et al. scheduled real-time periodic tasks on homogeneous NoC based MPSoC in order to meet deadline, energy and communication constraints using a heuristics manipulated by deterministic strategy [15]. Ali et al. performed integrated task mapping, scheduling, and voltage assignment on NoC based heterogeneous MPSoC (HMPSoC) systems using a heuristic called EIMSVS for reducing processing and communication energies [3]. Ishak et al. investigated a non-preemptive scheduling for tasks with precedence constraints and individual deadlines. They used NLP and ILP to assign optimal voltages to the tasks and communications on NoC links [20]. A similar ILP based approach is followed by Tariq et al. [46] using Iterative Offline Energy-aware Task and Communication Scheduling (IOETCS) algorithm for total energy consumption reduction. Ali et al. developed an energy-efficient task scheduling approach using Contention-aware Integrated Task Mapping and Voltage Assignment (CITM-VA) meta-heuristic algorithm. CITM-VA integrated DVFS and DPM to achieve maximum energy savings by reducing both static and dynamic power consumptions while considering the contention at NoC links [2]. Ding et al. presented a Hybrid Heuristic Genetic Algorithm with Adaptive Parameter (HGAAP) for energy-aware task mapping on heterogeneous multiprocessor architectures [13]. However, these studies consider MPSoC systems for tasks with precedence constraints but perform mapping and scheduling on single processor per VFI.

Ninomiya et al. [36] developed a task scheduling scheme for VFI based MPSoC architecture using SA based algorithm for energy consumption reduction and generated an optimal schedule for set of tasks under deadline constraints. Pagani et al. [39] presented a scheduling scheme called Single Frequency Approximation (SFA) to map the tasks and assign optimal voltage and frequency levels to each VFI. Liu and Guo [29] developed a Voltage Island Largest Capacity First (VILCF) algorithm for mapping the tasks on active VFI first in order to fully utilize it before activating

other inactive VFIs. Shin et al. [43] studied communication-aware VFI partitioning approach and developed a task mapping, voltage assignment algorithm for reducing inter-VFI communications. These investigations in [29, 36, 39, 43] deploy bus-based VFI-MPSoC systems for independent tasks mapping and scheduling. Some other researchers considered NoC based VFI-MPSoC systems for instance, Jang and Pan [21] performed energy-aware scheduling for dependent tasks by reducing VFI's power overheads. Digalwar et al. [12] presented a scheduling algorithm in order to optimize the total energy consumption for periodic tasks with hard deadline. Han et al. [16] developed a contention-aware static mapping and scheduling scheme for a set of tasks with precedence constraints in order to minimize the make-span and inter-VFI communications. They developed a contention and energy-aware task mapping and edge scheduling (CATMES) heuristics to assign tasks to processors while scheduling the edges on NoC. Two approaches CA-TMES-Quick and CA-TMES-Search were developed to select the processor for a task where it can start earliest among all processor. CA-TMES-Quick first performs task assignment and then determines routes for the communications that are sent to this task. CA-TMES-Search calculates start time for each task while considering communication contention. The processor offering earliest start time for a task is selected by CA-TMES-Search. Specifically, CA-TMES-Search relatively performs better than CA-TMES-Quick because it coordinates the task mapping in an exhaustive way therefore, make-span significantly reduces. We use these CA-TMES-Quick and CA-TMES-Search energy management schemes as baseline in order to determine the performance of our static task scheduling approach.

Though these state-of-the-art studies [12, 16, 21, 36, 43] addressed the energy-efficiency on VFI based NoC-MPSoC systems but non of them performed investigation on heterogeneous computing platform while considering processor energy performance profiles for achieving higher energy savings. Specifically, to the best of our knowledge non of the prior work focused on contention-aware and energy-efficient task scheduling on VFI-NoC-HMPSoC using DVFS technique for dependent tasks with precedence constraints and common deadline .

3 PRELIMINARIES

In this section first we present the relevant application model, second we discuss our VFI-NoC-HMPSoC architecture and finally we explain the energy model. Moreover, In this paper, we use the term tile and processor interchangeably.

3.1 Application Model

We characterize a real-time workload or application by Directed Acyclic Graph (DAG): $G(V, E, X)$ shown in Fig. 1. Where $V = \{v_1, v_2, v_3, \ldots, v_n\}$ represents a set of tasks and $E \subseteq V \times V$ shows directed edges set while each edge $(v_i, v_j) \in E$ denotes data dependency between two tasks. For example, if we have an edge from task v_i to task v_j then v_i is the predecessor of v_j and outputs the data to v_j, where v_j is the successor of v_i and it accepts input data from v_i. Moreover, X indicates set of directed edge weights while $\chi_{(i,j)}$ is the edge-weight of an edge (v_i, v_j) that shows the volume of data (in unit of bits) sent from v_i to v_j. We assume all tasks in the application have a common deadline, D.

3.2 System Architecture

We consider a VFI based NoC-MPSoC architecture with M processors $P = (pe_1, pe_2, pe_3, \ldots pe_M)$ demonstrated in Fig. 2. Each tile consists of a processor, local memory, and network interface card. Processors of the target architecture are partitioned into a set $C = \{c_1, c_2, c_3, \ldots c_m\}$ of m heterogeneous VFIs. Where each VFI, $c_i \in C$ consists a set of k homogeneous processors. We assume processors within an island (VFI) are of same type. Processors across different VFIs may be of different types i.e. inter-VFI processors may be heterogeneous. Each VFI can operate independently

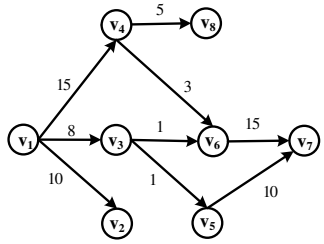

Fig. 1. Directed Acyclic Graph

at a set $\{(V_{dd_1}, f_1), (V_{dd_2}, f_2), (V_{dd_3}, f_3), \ldots, (V_{dd_n}, f_n)\}$ of n discrete voltage and frequency levels while a common supply voltage is shared by intra-VFI processors and routers.

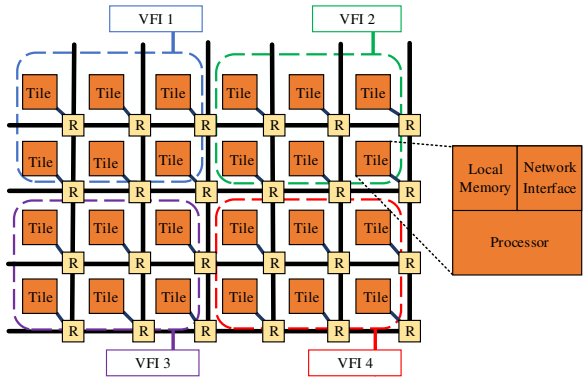

Fig. 2. VFI based Heterogeneous NoC-MPSoC Architecture

3.3 Communication Model

We assume a 2D-mesh topology NoC for communication architecture of the VFI based heterogeneous NoC-MPSoC (VFI-NoC-HMPSoC) shown in Fig. 3. Each tile of the computing system associated with a router to communicate with other processors. In NoC buffers are used in routers to host the incoming flits when immediate transfer to next processor and/or Intellectual property (IP) is not possible because of the congestion. NoC mesh consists of N_R rows and N_C columns therefore, number of processors in VFI-NoC-HMPSoC is equal to $N_R \times N_C$. Each router has five ports, four ports are used to communicate with the neighbor routers and one is dedicated for the purpose of communicating with the processor. A link is used to connect two routers and/or a router with a processor. We consider that all links are identical, full duplex, and have the same bandwidth, b_w.

3.3.1 Switching Technique. Virtual cut-through (VCT) and wormhole (WH) are the two most popular packet switching techniques for NoC interconnects. In WH each packet is split into small pieces known as FLITS. When in the network a packet traverses the WH immediately determines its next hop, forwards it and then the subsequent FLITS worm their way through the network. In VCT routing the buffer size is large and the entire packet is sent to the next node Thus, VCT has lower latency, higher link utilization, and lesser packet blocking probability. Though WH switching is simple and possesses higher efficiency of flow control over VCT in case of congestion occurrence,

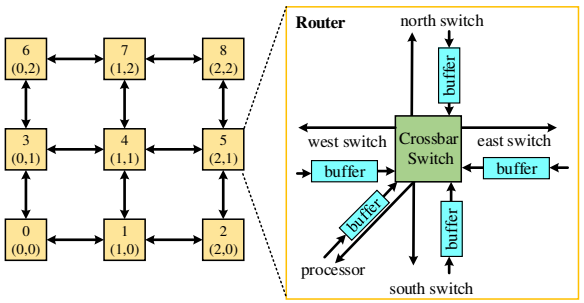

Fig. 3. 2D-mesh topology NoC interconnect

the stalling packet can block all the links and produces a low link utilization. Therefore, we consider VCT packet switching technique in this paper.

3.3.2 Routing Technique. Routing technique in a network decides the path of a packet from source to the destination router. We use a well-known XY deterministic routing on NoC which is most suitable option for 2D-mesh topology networks. Moreover, XY routing is a simple but yet effective approach additionally one of the major advantages includes that a deadlock does not occur in it. In XY routing the packets at the routers are routed in X-direction first and later on in the Y-direction.

3.4 Energy Model

We adopt the energy model described in [37]. The total energy consumption of an application is the sum of processing and communication energy consumption E_p and Ec respectively. The parameter E_p is the energy consumed in the execution of tasks on the processor, whereas communication energy is consumed in transmission of communications on the network that includes switch fabric, links, and buffers. E_p and E_c are discussed in detail in [37].

The total energy E consumed by an application is given given as follows:

$$E = E_p + E_c \tag{1}$$

Concisely, we consider DAG applications and heterogeneous VFI -NoC-HMPSoC architecture with VCT switching, XY routing, and energy consumptions that occur due to processors and communications.

4 STATIC CONTENTION-AWARE ENERGY OPTIMIZATION APPROACH

VFI-NoC-HMPSoCs consist of processors with different power-performance profiles. These processors operate at distinctive frequency and voltage i.e. speed levels. Furthermore, precedence constraints and the deadline of the tasks essentially must be observed. Subsequently, the execution order of the tasks and communications could significantly affect the total energy consumption. A substantial amount of energy could be saved by assigning priorities to the tasks with shorter deadline because DVFS can efficiently utilize the available idle slack by assigning a lower speed to the tasks. Therefore, the obtained quality of the solution is influenced by three factors: (1) task mapping, (2) ordering, and (2) voltage assignment. The state-of-the-art approach by [27] performs task orderings and voltage scaling in an integrated manner and performs task mapping separately. However, we think task and communication ordering and voltage scaling can be helpful in steering the task mapping optimization process towards more energy-efficient solution. This is one of the major factors that we consider in the design of our energy-aware integrated mapping, scheduling

and voltage scaling (ARSH-FATI) algorithm. ARSH-FATI algorithm considers mapping, scheduling, and voltage scaling in an integrated manner.

Algorithm 1: ARSH-FATI

input : A DAG G, tasks Deadlines, an MPSoC, total number of iterations Ω and population size μ
output : Task to processor mapping *map* and islands voltage levels *vol*
Construct two matrices Π and Ψ of zeros having dimensions $\mu \times |V|$ and a vector f of zeros having dimension $\mu \times 1$;
for $\eta \leftarrow 1$ *to* μ **do**
  for ***each*** $v_i \in G$ **do**
    $\Pi[\eta][i] \leftarrow \lceil rand()(|P| - 1) + 1 \rceil$;
  end
  for ***each*** $c_j \in C$ **do**
    $\Psi[\eta][j] \leftarrow$ maximum island voltage;
  end
end
for $\eta \leftarrow 1$ *to* μ **do**
  Generate the extended graph G_e;
  $[m, e] \leftarrow IVS(G, G_e, \Pi, \Psi, \eta)$;
  $f[\eta] \leftarrow fitness(m, e)$;
end
while *stopping criteria is not satisfied* **do**
  Find the best solution π_b and the worst solution π_w;
  for $\eta \leftarrow 1$ *to* μ **do**
    $f_b' \leftarrow -\infty$;
    for ***each*** $v_i \in G$ **do**
      $r \leftarrow rand()$;
      $\theta \leftarrow \Pi[\eta][i]$;

$$\Pi[\eta][i] \leftarrow \begin{cases} \Upsilon(\theta, \pi_b[i], \pi_w[i]) & \text{if } r \leq DR \\ \Pi[\eta][i] & \text{otherwise} \end{cases}$$

    end
    Construct an extended graph G_e given a mapping ;
    $[m, e] \leftarrow IVS(G, G_e, \Pi, \Psi, \eta)$;
    $f[\eta] \leftarrow fitness(m, e)$;
    if $f_b' < f[\eta]$ **then**
      $f_b' \leftarrow f[\eta]$;
    end
  end

$$DR \leftarrow \begin{cases} \frac{DR}{\lambda} & \text{if } f_b' > f_b \\ \lambda DR & \text{otherwise} \end{cases}$$

end
Set *map* and *vol* to mapping and islands voltage settings respectively with the highest fitness in the population;

The details of ARSH-FATI are given in Algorithm 1. Before we start explaining our algorithms we first define an extended graph G_e illustrated in Fig. 4. In an application, there are two kinds of

events communications and tasks. In order to schedule communications using traditional DAG based scheduling approaches we transform a DAG i.e. G into an extended graph G_e. Given a task to processor mapping an extended graph G_e is constructed by inserting an additional node v_s to graph G for each edge (v_i, v_j) whose tail node v_i and head node v_j are mapped on different processors. The edge (v_i, v_j) in extended graph is replaced by two edges (v_i, v_s) and (v_s, v_i). The additional inserted nodes are called communication nodes. The extended graph is represented by $G(V + V^*, E)$, where V is a set of the task nodes, V^* is a set of the communication nodes, and E is a set of the edges.

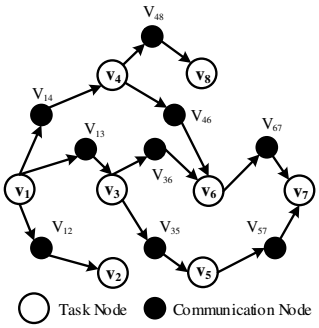

Fig. 4. Extended Graph

ARSH-FATI is a population-based algorithm in which only the best and worst solutions of the previous population are used to generate μ number of candidate solutions for the current population. Such kind of selection algorithms in the literature are commonly referred to as $(1 + \mu)$ selection algorithms.

Robustness of ARSH-FATI algorithm lies in the notion of updating the parameter dimensional rate (DR) at run-time during the searching process. Our algorithm attains a satisfactory trade-off between the exploitation and exploration attributes of the search process. We define the parameter DR as the percentage of tasks that are re-mapped probabilistically to generate a new solution (mapping) form current (best and worst) solutions. The need for only re-mapping a percentage of tasks and not all the tasks stems from the sensitivity of energy consumption to task mapping in this (energy optimization) problem. In other words re-mapping, even a small subset of tasks may generate a schedule with energy consumption significantly different than the schedule generated by original mapping. Hence, the role of DR is to adjust at run-time the exploitation and exploration features of ARSH-FATI algorithm that we explain in the following.

Step1. Initial population generation: First we generate a matrix Π of dimensions $\mu \times |V|$, where each row in this matrix represents a task to processor mapping. Each row is in matrix Π is generated by randomly mapping tasks to processors.

Step2. Evaluation: We define the following fitness function to gauge the quality of each member of the population:

$$fitness(m, e) = \begin{cases} \frac{1}{e} & \text{if } m \leq D \\ -\infty & \text{otherwise} \end{cases}$$

We define the following two terms:

(1) **Best solution:** is a member of the population that has the highest fitness value.
(2) **Worst solution:** is a member of the population that has the lowest fitness value.

Step3. Setting parameter *DR***:** We set the value of DR to 0.3 for the initial population. This value is determined empirically after extensive experiments. The *DR* value for the other populations generated during the optimization process is determined as follows:

$$DR = \begin{cases} \frac{DR}{\lambda} & \text{if the best solution is improved} \\ \lambda DR & \text{otherwise} \end{cases} \tag{2}$$

According to equation (2) if the best solution found so far is improved in the previous iteration then the value of *DR* is increased by dividing it by $0 < \lambda < 1$ otherwise we decrease *DR* by multiplying it with λ. We refer to λ as the dimensional rate adaption parameter as it determines the new value of *DR* during the optimization process. The larger the value of *DR* the more explorative the search is as this enables the moves in the search space by re-mapping many tasks at the same time, thereby leading to large and unconstrained step sizes. Compared to this a small value of *DR* motivates a more exploitative search by allowing small and conservative steps in the search space. The motivation behind equation (2) is to encourage the re-mapping of more tasks and thus, support more explorative search if the energy consumed by the schedule generated by the mapping in the previous iteration reduces. On the other hand if the energy does not reduce then the explorative search is rather restricted and ARSH-FATI takes small steps near the current mapping.

It is worth noticing that ARSH-FATI also reduces the communication contention. The energy function has two components, the communication energy, and the processing energy. Notice that the most effective mechanism of minimizing communication energy is to reduce the traffic over the network. As the traffic over the network reduces so does the communication contention. In scenarios where the communication energy dominates the total energy, ARSH-FATI will choose the solution that minimizes the traffic over the network and consequently, reduced contention among the communications. The prime objective of ARSH-FATI is to minimize the total energy. Therefore, in scenarios where the total energy is dominated by processing energy, it may choose a solution that generates high contentions between communications but minimizes the total energy consumption. Steps involved in the working principle of ARSH-FATI are given as follows:

New population: In every iteration for each candidate solution, we only remap a subset of tasks. These tasks are selected based on the value of *DR*. We remap the selected task v_i as follows:

$$\Pi[\eta][i] = \begin{cases} \Upsilon(\theta, \pi_b[i], \pi_w[i]) & \text{if } r \leq DR \\ \Pi[\eta][i] & \text{otherwise} \end{cases}$$

where θ is the processor where v_i is currently mapped, r is a random numbers and $\pi_b[i]$ and $\pi_w[i]$ are the processor where v_i is mapped in the best and worst solutions respectively. The function $\Upsilon(\theta, \pi_b[i], \pi_w[i])$ is defined as follows:

$$\Upsilon(\theta, \pi_b[i], \pi_w[i]) = \begin{cases} |\upsilon(\theta, \pi_b[i], \pi_w[i])| & \text{if } \upsilon \leq |P| \\ |P| & \text{otherwise} \end{cases}$$

$\upsilon(\theta, \pi_b[i], \pi_w[i])$ is defined as follows:

$$\upsilon(\theta, \pi_b[i], \pi_w[i]) = \lceil \theta + r_1(\pi_b[i] - \theta) - r_2(\pi_w[i] - \theta) \rceil \tag{3}$$

where r_1 and r_2 are random numbers. The term $r_1(\pi_b[i] - \theta)$ reflects the likelihood of the solution to move closer to the best solution in the population and the term $r_2(\pi_w[i] - \theta)$ reflects the likelihood of the solution to avoid the worst solution.

4.0.1 Earliest edge consistent deadline first (EECDF) algorithm. Before we describe *EECDF* given in Algorithm 2 we define some notations. The worst case execution time of a task node v_i mapped on processor pe_k operating at frequency f_j is $et(v_i) = \frac{NCC(v_i,k)}{f_j}$, where $NCC(v_i, k)$ is the worst case clock cycles of v_i on processor pe_k. The start and finish times of a task node v_i are respectively denoted by $\rho(v_i)$ and $\zeta(v_i)$. Similarly for a communication node v_j (corresponding to edge (v_a, v_b)) the transmission time on a link L between processors pe_s and pe_d is $et(v_j, L) = \frac{\chi_{a,b}}{b_w \min\{f_s, f_d\}}$, where b_w is the link width, f_s and f_d are the frequencies of pe_s and pe_d respectively. The start and finish time of v_j on link L are respectively denoted by $\rho(v_j, L)$ and $\zeta(v_j, L)$, where $\zeta(v_j, L) = \rho(v_j, L) + et(v_j, L)$.

Algorithm 2: *EECDF*

input : A DAG G, an extended DAG G_e, matrix Π, matrix Ψ and current chromosome index η
output: Energy e and make-span m of schedule
Calculate the *ECD*, d'_i of each task in $v_i \in G$;
Insert all source node in a ready queue R;
while *there are ready nodes in R* **do**
  Find a node v_i in R with smallest d'_i while ties are broken in favour of smallest index;
  if v_i *is a task node* **then**
    Schedule v_i subject to rules **R1**, **R2** and **R3**;
  end
  else
    Schedule v_i subject to rules **R4**, **R5**, **R6** and **R7**;
  end
  Delete v_i from R;
  Insert all ready nodes in R;
end
Calculate the energy e and make-span m of the schedule;

Scheduling, in general, is an NP-hard problem. Hence, in this work, we propose an earliest edge consistent deadline first (*EECDF*) heuristic algorithm. *EECDF* is a static list scheduler that prioritizes nodes with shorter edge consistent deadline (*ECD*) over nodes with longer *ECD*. The motivation behind this is to allow the DVFS algorithm to efficiently utilize the available slack.

Given task to processor mapping, operating frequencies of processors and a DAG G we calculate the *ECD* by the following dynamic programming algorithm.

Traverse the DAG G in the reverse topological order of G. If the task v_i is a sink node then its ECD, d'_i is equal to its pre-assigned deadline d_i otherwise:

$$d'_i = \min\{d_i, \min\{d'_j - et_j : \forall v_j \in ISucc_i\}\} \tag{4}$$

where $ISucc_i$ is a set of immediate successors of v_i. The *ECD*, d'_j of a communication node is same as its parent (task) node.

The *EECDF* algorithm is described in Algorithm 2. We performs four major steps.

(1) Calculate the *ECD* of each task $v_i \in G$ (Line 1).
(2) Create a ready queue R and insert all the source nodes in G_e to R (Line 2).
(3) Find a node v_i that has minimum *ECD* in R and schedule it. Then delete v_i from R and insert all the ready nodes in G_e to R. Repeat this until R is empty (Line 3-10).
(4) Calculate the energy E and make-span m of the schedule.

We define seven rules to schedule the highest priority node $v_i \in R$. The first three rules deal with the schedule of a task node and the remaining four deal with the schedule of a communication node.

Task scheduling rules: The schedule of a task node v_i is obtained by applying the following rules collectively in order:

- **R1:** The start time of v_i is equal to release time of v_i, $\rho(v_i) = r_(v_i)$.
- **R2:** The release time of each node $v_j \in ISucc(v_i)$ is equal to $r(v_j) = \max\{\zeta(v_i), r(v_j) : \forall v_j \in ISucc(v_i)\}$.
- **R3:** The release time of each unscheduled task node v_j mapped on same processor of v_i is $r(v_j) = \max\{\zeta(v_i), r(v_j)\}$.

R3 enforces *EECDF* rule on the schedule of task nodes. Under this rules task nodes with shorter *ECD* have higher priority than task nodes with longer *ECD*. High priority tasks are scheduled earlier in time than low priority tasks.

Communication scheduling rules: In communication scheduling, network resources such as links are treated as processors in a way that each communication can only use one resource at a time. Hence, communication nodes are scheduled on the links for the time they occupy them.

Consider a communication node v_j whose source is mapped on pe_{src} and destination is mapped on pe_{dest}, the routing algorithm used by the network generates the route R_j from pe_{src} to pe_{dst}. The route $R_j =< L_1, L_2, \ldots, L_l >$ is an ordered list of links, where L_1 is the first link and L_l is the last link on the route.

Note that the route depends only on the source and destination of the communication because in our network model we assume deterministic (XY) routing. Furthermore, the entire communication must be transmitted on the established route because in the network model we suppose circuit switching. A communication node utilizing this route must be scheduled on all the links (of this route). The data traverses these links in the order they appear in the route vector.

Link causality constraints: The schedule of a communication node v_j on links of route R_j must abide by the link causality constraints defined as follows:

$$\rho(v_j, L_1) \leq \rho(v_j, L_k) \tag{5}$$

$$\zeta(v_j, L_{k-1}) \leq \zeta(v_j, L_k) \tag{6}$$

$$for\ 1 < k \leq l$$

The causality constraints impose bounds on the schedule of v_j on the links of R_j. The finish time of v_j must not be sooner on link L_k than its predecessor link L_{k-1}.

Given a communication node v_j whose parent node is v_a and child node is v_b, the schedule of a v_j on $R_j =< L_1, L_2, \ldots, L_l >$ is obtained by applying the following rules collectively in order:

- **R4:** The start time of v_j is equal to finish time of v_a, $\rho(v_j) = \zeta(v_a)$.
- **R5:** The start time of v_j on each link $L_k \in R_j$ is:

$$\rho(v_j, L_k) = \begin{cases} \max\{\beta, \rho(v_j)\} & \text{if } k = 1 \\ \max\{\beta, \alpha, \rho(v_j, L_1)\} & \text{if } k > 1 \end{cases}$$

  where $\alpha = \zeta(v_j, L_{k-1}) - et(v_j, L_k)$ and β is the finish time of latest communication node scheduled on link L_k. On the link L_1 the start time of v_j is constrained only by the finish time of its parent node v_a and on all subsequent links the schedule of v_j follows the causality constraints. Note that **R5** enforces the *EECDF* rule on the schedule of communication nodes. Under rule **R5** communication nodes with shorter *ECD* have priority over nodes with longer *ECD*.
- **R6:** The finish time of v_j is equal to the finish time of v_j on the last link L_l, $\zeta(v_j) = \zeta(v_j, L_l)$.

Algorithm 3: Energy Gradient Descent (*EGD*)

input : A DAG G, an extended DAG G_e, matrix Π, matrix Ψ and index η.
output: Schedule make-span m and energy e.

```
[e, m] ← EECDF(G, G_e, Π, Ψ, η);
if m ≤ D then
    while there are extensible islands do
        Γ ← −1;
        for each extensible island c_j ∈ C do
            EG^best ← −∞;
            Find the voltage V_dd^L of island c_j exactly one level lower than current voltage level Ψ[η][j];
            for each V_dd ∈ {V_dd^min, ..., V_dd^L} do
                temp ← Ψ[η][j]; Ψ[η][j] ← V_dd^L;
                [e', m'] ← EECDF(G_e, Π, Ψ, η);
                Ψ[η][j] ← temp;
            end
            if m ≤ D then
                Calculate EG(c_j);
                if EG^best < EG(c_j) then
                    EG^best ← EG(c_j);
                    Γ ← V_dd; τ ← j; e'' ← e'; m'' ← m';
                end
            end
        end
        if Γ ≠ −1 then
            Ψ[η][τ] ← Γ; e ← e'' ; m ← m'';
        end
    end
end
```

- **R7:** The release time of v_b is equal to finish time of v_j, $r(v_b) = \zeta(v_j)$

4.0.2 Energy gradient descent (EGD). *EGD* in Algorithm 3 is inspired by gradient descent. Given task mapping and the initial islands operating voltages, *EGD* explores the solution space to find voltage settings for islands such that total energy consumption is minimized and the resulting schedule under these settings is feasible.

Before we describe EGD we define the two terms, an extensible island and an island energy gradient.

An island $c_j \in C$ is extensible, if by reducing its operating voltage the resulting schedule under the new voltage settings is feasible.

EGD is guided by energy gradient in its search for the island voltage setting that minimize energy consumption. Given the operating voltage V_{dd} of an island c_j, the energy consumption E and make-span m of the schedule, the energy gradient of c_j is defined as:

$$EG(c_j, E, m, E', m') = \begin{cases} \gamma(E - E') & \text{if } E > E', m \geq m' \\ \frac{E-E'}{m'-m} & \text{otherwise} \end{cases}$$

where γ is a large number, E' and m' is the energy consumption and make-span of the schedule respectively, when c_j operates at V'_{dd}, where V'_{dd} is a voltage level lower than V_{dd}.

EGD repeats the following two steps until there are no extensible islands:
Step 1: First find a set of extensible islands. Then for each extensible island c_j do the following:

- Find a set $\{V_{dd}^{min}, \ldots, V_{dd}^{L}\}$ of operating voltages, where V_{dd}^{L} is the maximum operating voltage of c_j under which the energy consumption of the schedule reduces.
- Tentatively adjust the operating voltage of c_j to each voltage level in set $\{V_{dd}^{min}, \ldots, V_{dd}^{L}\}$, call *EECD* to calculate the make-span, the energy consumption of the schedule under new voltage settings and calculate the *EG*.

Step 2: Find the island c_j and its operating voltage V_{dd} that maximizes the energy gradient and adjust the operating voltage of c_j to V_{dd}.

EGD may repeat the above mentioned two steps several times before it converges. In each iteration *EGD* can find many extensible islands and can adjust their operating voltages to many different levels. Each of these island voltage pairs may lead to some reduction in energy consumption. *EGD* chooses the pair that maximizes the *EG*. This is because for each island voltage pair the energy consumption of the schedule under the new voltage settings reduces without or with an increase in the make-span of the schedule. Both of these cases are reflected in the *EG* function. The first case is an ideal one because energy is reduced without any reduction in the available slack. Hence, the *EG* gives more weight to island voltage pairs that lie in case 1 by multiplying the energy difference with a large integer λ. In the second case energy reduces but with an increase in schedule make-span. In this case *EG* is the ratio between energy difference and the make-span difference. Higher the ratio the better the island voltage pair. A large value of this ratio is an indication of a large numerator and a small denominator. A large numerator reflects a big energy difference. This is desirable because it indicates that by changing the voltage level the schedule under new voltage settings reduces energy significantly. A small numerator reflects a small make-span difference. This is also desirable as this indicates more slack will be available for the nodes in the subsequent iterations.

5 EXPERIMENTAL SETUP AND RESULTS

In this section, we explain the experimental set up used for simulation. We also generate energy consumption values for different real benchmarks and discuss the results.

5.1 Experimental Setup

We use 8 real benchmarks in our experimental analysis on VFI-NoC-MPSoC computing architecture while generate results for different scenarios. The real benchmarks are adopted from Embedded System Synthesis Benchmarks Suite (E3S) which is a widely used benchmark suit in the task mapping and scheduling research [9]. Automatic Target Recognition (ATR) benchmark is a real-time streaming application used for pattern recognition. Benchmark MP3-decoder performs Huffman decoding and Inverse Discrete Transform (IDCT). JPEG-encoder contains tasks for Huffman encoding and Discrete Cosine Transform (DCT). Office benchmark contains tasks for text processing, image rotation, and gray-scale to binary conversion. Auto-industry represents an embedded application that includes tasks such as Fast Fourier Transform (FFT), finite/infinite impulse response filter, IDCT, Inverse Fast Fourier Transform (IFFT), matrix arithmetic, table lookup, road speed calculation, and interpolation. Consumer-1 and Consumer-2 benchmarks perform JPEG compression and/or decompression, conversions such as from RGB to CMYK and RGB to YIQ.

We use Samsung Exynos 5422 chip power and energy model for our simulations adopted from [28] and use two types of processors i.e. type 1: high-performance Cortex A15 (big) and type 2: low-power Cortex A7 (little). The Cortex A7 consumes $\sim 6-12$ times less power compared to Cortex A15 [32]. The operating frequencies and relative power consumption of both types are

Table 1. Operating Frequency and Power Consumption of Type 1 and Type 2 Processors

Type 1 (Cortex A15)							
Frequency (GHz)	2.0	1.8	1.6	1.4	1.2	1.0	0.8
Power (mW)	2500	1750	1350	1000	850	650	400
Type 2 (Cortex A7)							
Frequency (GHz)	1.4	1.2	1	0.8	0.6	0.4	0.2
Power (mW)	82.0	76.0	74.0	72.0	68.0	66.0	64.0

Table 2. The 70 nm Processor Technology Parameter Values

Parameter	Value	Parameter	Value
K_1	0.063	K_2	0.153
K_3	5.38×10^{-38}	K_4	1.83
K_5	4.19	K_6	5.26×10^{-12}
C_{eff}	4.30×10^{-10}	α	2.00
I_j	4.80×10^{-10}	L_g	4.00×10^6
V_{bs}	- 0.70	V_{th}	0.244

given in Table 1. Moreover, we adopt 70 nanometers (nm) processor technology parameters from Ali et al. [2] listed in Table 2.

We built the simulation environment in Matlab version R2016a moreover, we conducted the experiments using hardware platform of Intel (R) Xeon (R), i5-3570 CPU with the clock frequency of 3.50 GHz and 16.00 GB memory, 10 MB cache. We also use intlinprog solver for programming ILP problems. We first select real-world then synthetic benchmarks and report on the energy-efficiency evaluation of our $ARSH - FATI$ meta-heuristic.

Fig. 5 shows the impact of DR on $ARSH - FATI$ performance. We set $DR = 0.3$ initially though it can acquire values $0.1 \leq DR \leq 0.5$ with small impact on the total energy performance for static task scheduling. The results indicate that the energy performance of the $ARSH - FATI$ slightly decreases when $DR = 0.1$ and $DR = 0.5$. However, our algorithm automatically sets the DR value to produce maximum energy-efficiency but initially setting $DR = 0.10$ means $ARSH - FATI$ performs an insufficient exploration while $DR = 0.5$ leads to an excessive exploration. Thus, $DR = 0.3$ is the nominal initial value for our meta-heuristic. $ARSH - FATI$ converges i.e. DR value relatively stabilizes at 200 number of iterations (NI) and a minute variation occurs till 500 while no variations occur when $NI > 500$. Therefore, we consider $NI = 500$, $\mu = 5$, and $\lambda = 0.9$ for our experiments.

5.2 Results

We generate results for four scenarios considering different metrics such as homogeneous MPSoC platform, heterogeneous multiprocessing computing system, PPI, deadline, and CCR. In this section, we refer to different parameters listed in Table 3.

5.2.1 Scenario 1. We set the default parameters $NVFI = 4$, $PPI = 2 \times 2$, $M = 16$, $DR = 0.30$, and perform experiments on 8 real benchmarks deploying both homogeneous and heterogeneous VFI-NoC-MPSoC computing architectures.

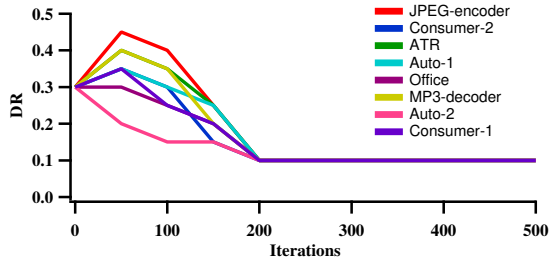

Fig. 5. Dimensional rate parameter variations

Table 3. List of Parameters Used In Results

Parameters	Description
NVFI	Number of Voltage Frequcny Island
PPI	Processors per Voltage Frequency Island
M	Total number of processors
CCR	Communication to Computation Ratio
MULT	Multiplier
ΔE	Change in Energy
Task Set-1	ATR, MP3-decoder, JPEG-encoder
Task Set-2	Office, Auto, Consumer

Table 4. Real Benchmarks Energy Consumption Comparison in Joule (J) at $NVFI = 4$ and $PPI = 2 \times 2$

Benchmarks	$ARSH - FATI$ (Homogeneous)	$ARSH - FATI$ (Heterogeneous)	$ARSH - FATI$ +EGD (Heterogeneous)	$CA - TMES$ (Search)	$CA - TMES$ (Quick)
ATR	1.1201	1.0324	0.9586	1.2115	1.2501
MP3-decoder	1.3208	1.2828	1.1627	1.4950	1.5793
JPEG-encoder	1.3808	1.3593	1.2396	1.5439	1.6415
Office	0.4431	0.4107	0.3981	0.4906	0.4910
Auto-1	0.5136	0.4931	0.4682	0.5748	0.5791
Auto-2	0.2850	0.2546	0.2381	0.2915	0.3064
Consumer-1	0.4321	0.4121	0.3917	0.4701	0.4920
Consumer-2	0.3802	0.3602	0.3334	0.4043	0.4201

We compare the energy performance of $ARSH - FATI$ with state-of-the-art CA-TMES-Search and CA-TMES-Quick [16]. First, we consider a homogeneous VFI-NoC-MPSoC system where all the processors are of type 1. We set the operating frequencies of the processors to their maximum (f_{max} = 2.0 GHz). Second, we use a VFI-NoC-HMPSoC deploying both type 1 and type 2 processors without voltage scaling technique. We randomly select the type of processor for each VFI to generate a heterogeneous computing platform in order to ensure unbiased experimentation. Third, we consider a VFI-NoC-HMPSoC computing architecture and use EGD in order to efficiently avail

the slack in the processors. Table 4 summarizes the energy consumption values for these three cases on 8 real benchmarks.

Fig. 6 demonstrates the energy performance of our static task scheduler $ARSH - FATI$ compared to CA-TMES-Search and CA-TMES-Quick. X-axis denotes real benchmarks while y-axis represents energy consumption in joule (J). Not surprisingly when all the processors are of type 1, $(ARSH - FATI)_{homogeneous}$ consumes lower energy because our population-based meta-heuristic performs better solution space exploration during task mapping and subsequently, reduces communication energy. In other words $(ARSH - FATI)_{homogeneous}$ schedules dependent tasks closer to each other in order to avoid energy dissipation occurring due to the utilization of links, switches, and buffers for communications. Specifically, $(ARSH - FATI)_{homogeneous}$ achieves an average energy-efficiency of $\sim$ 15%, $\sim$ 8% over CA-TMES-Qucik and CA-TMES-Search respectively.

The energy savings further increases when both type 1 and type 2 processors are deployed to form VFI-NoC-HMPSoC system. Task scheduler, $(ARSH - FATI)_{heterogeneous}$ attains an average-efficiency of $\sim$ 13%, $\sim$ 20% compared to CA-TMES-Search and CA-TMES-Qucik respectively. Unlike, CA-TMES-Quick and CA-TMES-Search energy management approaches our static scheduler $(ARSH - FATI)_{heterogeneous}$ is aware of the energy performance profiles and generates a task schedule such that higher energy consuming tasks are mapped on low performance and high energy-efficient processor.

Our static scheduler $ARSH - FATI$ when integrated with voltage scaling algorithm EGD i.e. $(ARSH - FATI)_{heterogeneous+EGD}$ achieves the highest energy-efficiency. It produces an average energy savings of $\sim$ 24%, $\sim$ 30% over CA-TMES-Search and CA-TMES-Quick respectively. EGD tends to find the voltage settings for islands such that energy consumption is minimized and the deadline constraints are satisfied. In other words, EGD reduces the computation energy consumption by intelligently exploiting the available slack in the processors.

Summarizing the observations in these experiments in scenario 1, $ARSH - FATI$ reduces both the communication and computation energy consumptions while not sacrificing the constraints. Our approach $ARSH - FATI$ for static task scheduling on VFI-NoC-MPSoC architecture outperforms both CA-TMES-Search and CA-TMES-Qucik.

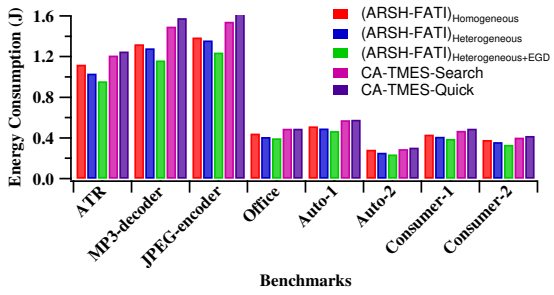

Fig. 6. Energy Consumption at $NVFI = 4$ and $PPI = 2 \times 2$

5.2.2 Scenario 2. Next, we examine the impact of PPIs on energy consumption while determining the ability of $ARSH - FATI$ to utilize the resources experimenting on 9 real benchmarks. We set $NVFI = 4$, heterogeneous computing system, and systematically upgrade $PPI = 2 \times 2, 4 \times 2, 4 \times 3$ i.e. M = 16, 32, 64.

Fig. 7 illustrates that except MP3-decoder, JPEG-encoder, and Robot other benchmarks do not show a significant decrease in the energy consumption when PPI is gradually increased. These

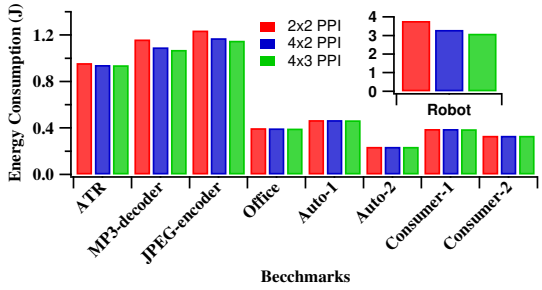

Fig. 7. Energy Consumption using $NVFI = 4$ and Different PPI

benchmarks contain relatively higher number of task nodes and degree of parallelism. MP3-decoder consumes 1.1627 J energy at $PPI = 2 \times 2$ while it decreases to 1.0936 J and 1.0728 J for $PPI = 4 \times 2$ ($\Delta E = 0.0.0691$ J), and $PPI = 4 \times 3$ ($\Delta E = 0.0899$) respectively. Similarly, JPEG-encoder depletes 1.2396 J energy at $PPI = 2 \times 2$ and this energy consumption reduces to 1.1722 J ($\Delta E = 0.0674$ J), 1.1517 J ($\Delta E = 0.0879$ J) at $PPI = 4 \times 2, 4 \times 3$ respectively. We also evaluate the performance of our static scheduler $ARSH - FATI$ on more complex real benchmark, Robot containing 88 tasks. Compared to $PPI = 2 \times 2$ (M = 16), $ARSH - FATI$ achieves energy savings of $\sim$ 13% and $\sim$ 18% for robot when $PPI = 4 \times 2$ (M = 32) and $PPI = 4 \times 3$ (M = 64) respectively. These results demonstrate that our meta-heuristic $ARSH - FATI$ can efficiently utilize the resources and degree of parallelism in the benchmarks to reduce the total energy consumption.

5.2.3 Scenario 3. We now conduct experiments to analyze the robustness of ARSH-FATI under deadline variations and compare its performance with CA-TMES-Search. We consider voltage scalable heterogeneous computing architecture with $NVFI = 4$, $PPI = 1 \times 2$ and $M = 8$. We set the base-line deadline for each benchmark in set 1 and set 2 (described in TABLE II) to the make-span of the schedule generated by CA-TMES-Search under the condition of all VFIs operating at maximum frequencies.

Fig. 8 and Fig. 9 show the energy consumption of $ARSH - FATI$ and CA-TMES-Search for set-1 and set-2 respectively. The MULT represents the factor multiplied to the baseline-deadline. For example, $MULT = 1.00$ at horizontal axis in Fig. 8, 9 indicates the deadline of each benchmark is set to 1.00 $\times$ baseline-deadline. The dotted lines represent our $ARSH - FATI$ while the straight lines show CA-TMES-Quick. The condition $MULT < 1$ indicates a strict deadline while $MULT > 1$ shows a relaxed deadline.

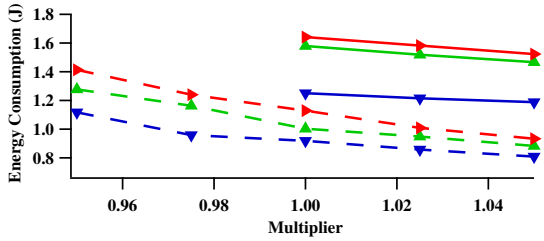

Fig. 8. Set-1 Energy Consumption using $NVFI = 4$ and $PPI = 2 \times 2$

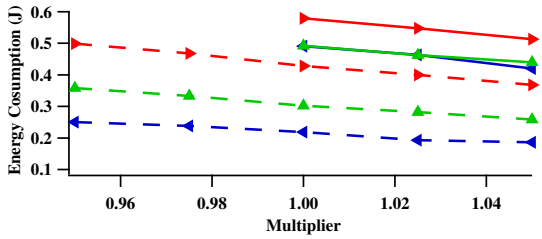

Fig. 9. Set-2 Energy Consumption using $NVFI = 4$ and $PPI = 2 \times 2$

The energy-efficiency of ARSH-FATI gradually reduces starting from $MULT < 1$ to $MULT = 0.95$. This increase in energy consumption occurs due to the reduction in slack. Though, energy consumption slightly increases under the strict deadline conditions (of $MUL < 1$), $ARSH - FATI$ can still successfully generate a feasible schedule. Moreover, as deadline decreases ARSH-FATI tends to schedule more tasks on high-performance processors. These processors reduce task execution time at a cost of higher energy consumption. This is another reason in the increase of energy consumption along with the reduction in slack. The same is not true for CA-TMES-Search because it neglects to consider the energy performance profiles of the processors during the task mapping phase. The *EECDF* prioritizes nodes with shorter *ECD* thereby, increasing the chance of generating a feasible schedule. This is because *ECD* of a node depends on the pre-assigned deadline. As the deadline varies so does *ECD* consequently, the relative urgency of nodes may change. This additional information reflected by *ECD* can be exploited by *EECDF*. On the contrary, CA-TMES-Search uses b-level to reflect the relative urgency of tasks. The metric b-level is independent of the application deadlines hence, the CA-TMES-Search is unaware of the deadline variations and is unable to respond accordingly.

Under the condition $MULT > 1$ the energy-efficiency of $ARSH - FATI$ rapidly increases. $ARSH - FATI$ being aware of processor energy performance profiles tends to map more tasks on lower performance but energy-efficient processors. Contrarily CA-TMES-Search is inadequate to avail energy performance profiles consequently, it maps more tasks on high performance, lower energy-efficient processors. ARSH-FATI schedules nodes in *EECDF* manner hence *EGD* can efficiently utilize the slack because nodes with longer *ECD* are not blocked by nodes with shorter *ECD*. The same is not true for $CA - TMES - Search$. Furthermore, uniform voltage scaling used by $CA - TMES - Search$ is an inefficient technique for a heterogeneous system.

Thus, $ARSH - FATI$ maintains its remarkable energy performance, robustness, and QoS for real benchmarks at $0.95 \leq MULT \leq 1.05$.

5.2.4 Scenario 4. Now, we evaluate the energy performance of $ARSH - FATI$ at $NVFI = 4$, PPI = 2×2, $M = 16$, and $CCR = 0.2 - 3.0$.

Fig. 10 illustrates the impact of CCR on $ARSH - FATI$ energy performance while CA-TMES-Quick (represented by blue line) being used as a baseline. Evidently, $ARSH - FATI$ static scheduler consumes lesser energy compared to CA-TMES-Search due to performing task mapping, scheduling, and voltage scaling in an integrated manner. With the increase in communication volume, the energy consumption of $ARSH - FATI$ reduces and reaches to its minimum value at $CCR = 1.0$. $ARSH - FATI$ maps the dependent tasks (parent and child nodes) on the same processor when $0.2 \leq CCR \leq 1.0$ in order to decrease the communication energy. $ARSH - FATI$ at $CCR > 1$ tends to map all the dependent tasks on the closest possible processors which leads to a slight increase in energy consumption. Our static scheduler $ARSH - FATI$ performs relatively better

Table 5. ARSH-FATI Energy Performance Summary

Our Static Scheduler	CA-TMES-Search	CA-TMES- Quick
$(ARSH-FATI)_{Homogeneous}$	08%	15%
$(ARSH-FATI)_{Heterogeneous}$	13%	20%
$(ARSH-FATI)_{Heterogeneous+EGD}$	24%	30%

when $0.5 \leq CCR \leq 2.0$ i.e. when network contention is medium. Our static scheduler $ARSH-FATI$ outperforms CA-TMES-Search in terms of energy-efficiency for $0.2 \leq CCR \leq 3.0$.

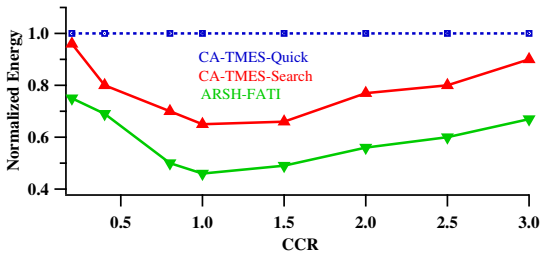

Fig. 10. CCR impact on ARSH-FATI at $VFI = 4$ and $PPI = 2 \times 2$

Table 5 summarizes the energy performance of $ARSH-FATI$ compared to the base-line state-of-the-art $CA-TMES-Search$ and $CA-TMES-Quick$ energy management approaches when $NVFI = 4$ and $PPI = 2 \times 2$ in the multiprocessor computing system. Energy consumption of the dependent DAG tasks decreases when computing platform is changed from homogeneous to heterogeneous. The energy-efficiency further improves when voltage scaling technique EGD is deployed. Concisely, $ARSH-FATI$ outperforms CA-TMES-Search and CA-TMES-Quick in terms of energy savings while maintains higher robustness.

6 CONCLUSION

Cyber-Physical Systems (CPS) integrate computation with physical processes using battery constrained intelligent edge devices. The computational complexity of real-time applications in CPS is rapidly increasing. Consequently, Network-on-Chip (NoC) based Voltage Frequency Islands (VFIs), Globally Asynchronous Locally Synchronous (GALS) are widely adopted in large scale multiprocessor chip designs due to their higher performance, simple architecture, and energy-efficiency. Unlike, other scheduling techniques [16, 35, 38, 48], we investigated a harder scheduling problem i.e. contention-aware and energy-efficient DAG tasks scheduling on heterogeneous VFI based NoC-MPSoC (VFI-NoC-HMPSoC) computing architecture with DVFS-enabled processors. We proposed a novel static task scheduler ARSH-FATI that performs task mapping, scheduling, and voltage scaling in an integrated manner while considering the energy performance profiles of the processors and contention at the NoC links. Our meta-heuristic ARSH-FATI can intelligently switch at run-time between explorative and exploitative search modes for performance trade-off. We also integrated communication contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling approach and Energy Gradient Decent (EGD) algorithm for voltage scaling in ARSH-FATI to reduce the computation energy consumption. We performed experiments on eight real benchmarks considering different scenarios. Our static scheduler outperformed state-of-the-art

CA-TMES-Search and CA-TMES-Quick [16] energy management approaches and achieved ~ 24% and ~ 30% an average energy-efficiency respectively.

REFERENCES

[1] Imran Ahmed, Awais Ahmad, Francesco Piccialli, Arun Kumar Sangaiah, and Gwanggil Jeon. 2018. A Robust Features-Based Person Tracker for Overhead Views in Industrial Environment. *IEEE Internet of Things Journal* 5, 3 (2018), 1598–1605.

[2] Haider Ali, Umair Ullah Tariq, Yongjun Zheng, Xiaojun Zhai, and Lu Liu. 2018. Contention & Energy-Aware Real-Time Task Mapping on NoC Based Heterogeneous MPSoCs. *IEEE Access* 6 (2018), 75110–75123.

[3] Haider Ali, Xiaojun Zhai, Umair Ullah Tariq, and Lu Liu. 2018. Energy Efficient Heuristic Algorithm for Task Mapping on Shared-Memory Heterogeneous MPSoCs. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 1099–1104.

[4] Hakan Aydin, Rami Melhem, Daniel Mossé, and Pedro Mejía-Alvarez. 2001. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Real-Time Systems, 13th Euromicro Conference on, 2001.* IEEE, 225–232.

[5] Mossaad Ben Ayed and Mohamed Abid. 2017. An Automated Surveillance System based on Multi-Processor System-on-Chip and Hardware Accelerator. *International Journal of Advanced Computer Science and Applications* 8, 9 (2017), 59–66.

[6] Mario Bambagini, Mauro Marinoni, Hakan Aydin, and Giorgio Buttazzo. 2016. Energy-aware scheduling for real-time systems: A survey. *ACM Transactions on Embedded Computing Systems (TECS)* 15, 1 (2016), 7.

[7] Guilherme Castilhos, Marcelo Mandelli, Guilherme Madalozzo, and Fernando Moraes. 2013. Distributed resource management in NoC-based MPSoCs with dynamic cluster sizes. In *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 153–158.

[8] Feng-Cheng Chang and Hsiang-Cheh Huang. 2016. A survey on intelligent sensor network and its applications. *J. Netw. Intell* 1, 1 (2016), 1–15.

[9] Yuanqing Cheng, Lei Zhang, Yinhe Han, and Xiaowei Li. 2013. Thermal-constrained task allocation for interconnect energy reduction in 3-D homogeneous MPSoCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 2 (2013), 239–249.

[10] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)* 45, 3 (2013), 35.

[11] Ewerson Luiz de Souza Carvalho, Ney Laert Vilar Calazans, and Fernando Gehm Moraes. 2010. Dynamic task mapping for MPSoCs. *IEEE Design & Test of Computers* 27, 5 (2010), 26–35.

[12] Mayuri Digalwar, Praveen Gahukar, and Sudeept Mohan. 2018. Energy Efficient Real Time Scheduling on Multi-core Processor with Voltage Islands. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 1245–1251.

[13] Shan Ding, Jinhui Wu, Guoqi Xie, and Gang Zeng. 2017. A hybrid heuristic-genetic algorithm with adaptive parameters for static task scheduling in heterogeneous computing system. In *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 761–766.

[14] Christian El Salloum, Martin Elshuber, Oliver Höftberger, Haris Isakovic, and Armin Wasicek. 2013. The ACROSS MPSoC–A new generation of multi-core processors designed for safety–critical embedded systems. *Microprocessors and Microsystems* 37, 8 (2013), 1020–1032.

[15] Aymen Gammoudi, Adel Benzina, Mohamed Khalgui, and Daniel Chillet. 2018. Energy-Efficient Scheduling of Real-Time Tasks in Reconfigurable Homogeneous Multicore Platforms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2018).

[16] Jian-Jun Han, Man Lin, Dakai Zhu, and Laurence T Yang. 2015. Contention-aware energy management scheme for NoC-based multicore real-time systems. *IEEE Transactions on Parallel and Distributed Systems* 26, 3 (2015), 691–701.

[17] Menglan Hu, Jun Luo, Yang Wang, and Bharadwaj Veeravalli. 2017. Adaptive scheduling of task graphs with dynamic resilience. *IEEE Trans. Comput.* 66, 1 (2017), 17–23.

[18] Jia Huang, Christian Buckl, Andreas Raabe, and Alois Knoll. 2011. Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*. IEEE, 447–454.

[19] Jun Huang, Yu Meng, Xuehong Gong, Yanbing Liu, and Qiang Duan. 2014. A novel deployment scheme for green internet of things. *IEEE Internet of Things Journal* 1, 2 (2014), 196–205.

[20] Suhaimi Abd Ishak, Hui Wu, and Umair Ullah Tariq. 2017. Energy-aware task scheduling on heterogeneous noc-based mpsocs. In *Computer Design (ICCD), 2017 IEEE International Conference on*. IEEE, 165–168.

[21] Wooyoung Jang and David Z Pan. 2011. A voltage-frequency island aware energy optimization framework for networks-on-chip. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 1, 3 (2011), 420–432.
[22] Song Jin, Yinhe Han, and Songwei Pei. 2014. Variation-aware statistical energy optimization on voltage-frequency island based MPSoCs under performance yield constraints. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*. IEEE, 720–725.
[23] Arvind Kandhalu, Junsung Kim, Karthik Lakshmanan, and Ragunathan Rajkumar. 2011. Energy-aware partitioned fixed-priority scheduling for chip multi-processors. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on*, Vol. 1. IEEE, 93–102.
[24] Shin-haeng Kang, Hoeseok Yang, Sungchan Kim, Iuliana Bacivarov, Soonhoi Ha, and Lothar Thiele. 2014. Static mapping of mixed-critical applications for fault-tolerant MPSoCs. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*. IEEE, 1–6.
[25] Neetesh Kumar and Deo Prakash Vidyarthi. 2017. A GA based energy aware scheduler for DVFS enabled multicore systems. *Computing* 99, 10 (2017), 955–977.
[26] David E Lackey, Paul S Zuchowski, Thomas R Bednar, Douglas W Stout, Scott W Gould, and John M Cohn. 2002. Managing power and performance for system-on-chip designs using voltage islands. In *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*. IEEE, 195–202.
[27] Dawei Li and Jie Wu. 2016. Energy-efficient contention-aware application mapping and scheduling on NoC-based MPSoCs. *J. Parallel and Distrib. Comput.* 96 (2016), 1–11.
[28] Di Liu, Jelena Spasic, Gang Chen, and Todor Stefanov. 2015. Energy-efficient mapping of real-time streaming applications on cluster heterogeneous mpsocs. In *Embedded Systems For Real-time Multimedia (ESTIMedia), 2015 13th IEEE Symposium on*. IEEE, 1–10.
[29] Jun Liu and Jinhua Guo. 2016. Energy efficient scheduling of real-time tasks on multi-core processors with voltage islands. *Future Generation Computer Systems* 56 (2016), 202–210.
[30] Lindong Liu and Deyu Qi. 2018. An Independent Task Scheduling Algorithm in Heterogeneous Multi-core Processor Environment. In *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, 142–146.
[31] Yang Liu, Jonathan E Fieldsend, and Geyong Min. 2017. A framework of fog computing: Architecture, challenges, and optimization. *IEEE Access* 5 (2017), 25445–25454.
[32] Andrew Lukefahr, Shruti Padmanabha, Reetuparna Das, Ronald Dreslinski Jr, Thomas F Wenisch, and Scott Mahlke. 2014. Heterogeneous microarchitectures trump voltage scaling for low-power cores. In *Proceedings of the 23rd international conference on Parallel architectures and compilation*. ACM, 237–250.
[33] Aminollah Mahabadi, SM Zahedi, and Ahmad Khonsari. 2013. Reliable energy-aware application mapping and voltage–frequency island partitioning for GALS-based NoC. *J. Comput. System Sci.* 79, 4 (2013), 457–474.
[34] Seyedali Mirjalili and Andrew Lewis. 2016. The whale optimization algorithm. *Advances in engineering software* 95 (2016), 51–67.
[35] James D Monte and Krishna R Pattipati. 2002. Scheduling parallelizable tasks to minimize make-span and weighted response time. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 32, 3 (2002), 335–345.
[36] Sho Ninomiya, Keishi Sakanushi, Yoshinori Takeuchi, and Masaharu Imai. 2012. Task allocation and scheduling for voltage-frequency islands applied NOC-based MPSOC considering network congestion. In *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium on*. IEEE, 107–112.
[37] Umit Y Ogras, Radu Marculescu, Diana Marculescu, and Eun Gu Jung. 2009. Design and management of voltage-frequency island partitioned networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17, 3 (2009), 330–341.
[38] Sverrir Olafsson. 1995. A general model for task distribution on an open heterogenous processor system. *IEEE transactions on systems, man, and cybernetics* 25, 1 (1995), 43–58.
[39] Santiago Pagani, Jian-Jia Chen, and Minming Li. 2015. Energy efficiency on multi-core architectures with multiple voltage islands. *IEEE Transactions on Parallel and Distributed Systems* 26, 6 (2015), 1608–1621.
[40] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. 2010. Cyber-physical systems: the next computing revolution. In *Design Automation Conference*. IEEE, 731–736.
[41] Tifenn Rault, Abdelmadjid Bouabdallah, and Yacine Challal. 2014. Energy efficiency in wireless sensor networks: A top-down survey. *Computer Networks* 67 (2014), 104–122.
[42] Amin Safaei, QM Jonathan Wu, and Yimin Yang. 2018. System-on-a-chip (soc)-based hardware acceleration for foreground and background identification. *Journal of the Franklin Institute* 355, 4 (2018), 1888–1912.
[43] Dongkun Shin, Woojoong Kim, Soontae Kwon, and Tae Hee Han. 2011. Communication-aware VFI partitioning for GALS-based networks-on-chip. *Design Automation for Embedded Systems* 15, 2 (2011), 89–109.
[44] Eric Simmon, Kyoung-Sook Kim, Eswaran Subrahmanian, Ryong Lee, Frederic De Vaulx, Yohei Murakami, Koji Zettsu, and Ram D Sriram. 2013. *A vision of cyber-physical cloud computing for smart networked systems.* US Department of

Commerce, National Institute of Standards and Technology.

[45] Umair Ullah Tariq and Hui Wu. 2017. Energy-Aware Scheduling of Periodic Conditional Task Graphs on MPSoCs. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*. ACM, 13.

[46] Umair Ullah Tariq, Hui Wu, and Suhaimi Abd Ishak. 2018. Energy-Aware Scheduling of Conditional Task Graphs on NoC-Based MPSoCs. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.

[47] Fatemeh Vafaee and Peter C Nelson. 2010. An explorative and exploitative mutation scheme. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8.

[48] Xi Wang, Zhiwu Li, and Walter Murray Wonham. 2017. Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on DES supervisory control. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47, 7 (2017), 1082–1098.

[49] Jiao Zhang, Xiping Hu, Zhaolong Ning, Edith C-H Ngai, Li Zhou, Jibo Wei, Jun Cheng, and Bin Hu. 2018. Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. *IEEE Internet of Things Journal* 5, 4 (2018), 2633–2645.

[50] Weizhe Zhang, Enci Bai, Hui He, and Albert MK Cheng. 2015. Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms. *Sensors* 15, 6 (2015), 13778–13804.

[51] Zhiwei Zhao, Geyong Min, Weifeng Gao, Yulei Wu, Hancong Duan, and Qiang Ni. 2018. Deploying edge computing nodes for large-scale IoT: A diversity aware approach. *IEEE Internet of Things Journal* 5, 5 (2018), 3606–3614.

[52] Junlong Zhou, Tongquan Wei, Mingsong Chen, Jianming Yan, Xiaobo Sharon Hu, and Yue Ma. 2016. Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 8 (2016), 1269–1282.