IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Digital Object Identifier

Energy-aware Successor Tree Consistent EDF Scheduling for PCTGs on MPSoCs

UMAIR ULLAH TARIQ[1], HAIDER ALI[2], MUHAMMAD SHAHROZ NADEEM [3], SYED ROOHULLAH JAN [3], FARIZA SABRINA [1], SRIMANNARAYANA GRANDHI[1], ZHENGLIN WANG[1], LU LIU[4]
[1]School of Computer Science of Engineering and Technology, The Central Queensland University, Australia. E-mail: {u.tariq, f.sabrina, s.grandhi, z.wang}@cqu.edu.au
[2]School of Computing, University of Derby, Derby, United Kingdom. E-mail: h.ali@derby.ac.uk
[3]School of Engineering and Arts, Science and Technology, Suffolk University, United Kingdom. E-mail: {s.nadeem3,syed.jan}@uos.ac.uk
[4]School of Computing and Mathematic Sciences, University of Leicester, Leicester, United Kingdom. E-mail: l.liu@leicester.ac.uk

Corresponding author: Haider Ali (e-mail: h.ali@derby.ac.uk)

ABSTRACT

Multiprocessor System-on-Chips (MPSoCs) computing architectures are gaining popularity due to their high-performance capabilities and exceptional Quality-of-Service (QoS), making them a particularly well-suited computing platform for computationally intensive workloads and applications. Nonetheless, The scheduling and allocation of a single task set with precedence restrictions on MPSoCs have presented a persistent research challenge in acquiring energy-efficient solutions. The complexity of this scheduling problem escalates when subject to conditional precedence constraints between the tasks, creating what is known as a Conditional Task Graph (CTG). Scheduling sets of Periodic Conditional Task Graphs (PCTGs) on MPSoC platforms poses even more challenges. This paper focuses on tackling the scheduling challenge for a group of PCTGs on MPSoCs equipped with shared memory. The primary goal is to minimize the overall anticipated energy usage, considering two distinct power models: dynamic and static power models. To address this challenge, this paper introduces an innovative scheduling method named Energy Efficient Successor Tree Consistent Earliest Deadline First (EESEDF). The EESEDF approach is primarily designed to maximize the worst-case processor utilization. Once the tasks are assigned to processors, it leverages the earliest successor tree consistent deadline-first strategy to arrange tasks on each processor. To minimize the overall expected energy consumption, EESEDF solves a convex Non-Linear Program (NLP) to determine the optimal speed for each task. Additionally, the paper presents a highly efficient online Dynamic Voltage Scaling (DVS) heuristic, which operates in O(1) time complexity and dynamically adjusts the task speeds in real-time. We achieved the average improvement, maximum improvement, and minimum improvement of EESEDF+Online-DVS 15%, 17%, and 12%, respectively compared to EESEDF alone. Furthermore, in the second set of experiments, we compared EESEDF against state-of-the-art techniques LESA and NCM. The results showed that EESEDF+Online-DVS outperformed these existing approaches, achieving notable energy efficiency improvements of 25% and 20% over LESA and NCM, respectively. Our proposed scheduler, EESEDF+Online-DVS, also achieves significant energy efficiency gains compared to existing methods. It outperforms IOETCS-Heuristic by approximately 13% while surpassing BESS and CAP-Online by impressive margins of 25% and 35%, respectively.

INDEX TERMS PCTGs, Scheduling, Shared Memory, MPSoCs, Conditional Precedence Constraints, DVS, Green Computing

I. INTRODUCTION

REVOLUTION has been witnessed in the recent past in integrated architecture design, shifting from single-processor systems to multicore architectures. This shift is driven by the limitations of traditional approaches, the impact of power consumption, and the continued advancement facilitated by Moore's law. Designers are leveraging multicore architectures to meet increasing computational requirements while managing power consumption and design complexity [1]. Multi-core architectures such as Multiprocessor System-on-Chips (MPSoCs) have received intensive interest in the embedded systems community due to their high performance, exceptional Quality-of-Service (QoS), and low power consumption [2], [3]. Prioritizing energy minimization in high-performance embedded systems yield a multitude of advantages, ranging from decreased heat dissipation and increased reliability to improved performance and greater environmental sustainability. Embracing energy-efficient design principles is key to unlocking the full potential of modern embedded systems while mitigating various challenges associated with power consumption [4]–[6]. Efficient task mapping and scheduling can be used to achieve green computing and reduce the carbon footprint. Task mapping and scheduling involve the systematic allocation of a set of tasks to the processors in MPSoCs in a way that satisfies specific requirements, such as optimizing power consumption or reducing overall execution time [2], [5]. Dynamic Voltage Scaling (DVS) is a highly effective and widely adopted technique in modern embedded systems for achieving energy efficiency. Its ability to dynamically adjust voltage and frequency levels based on workload requirements makes it a valuable technique in reducing energy consumption and enhancing the overall performance and reliability of embedded systems [7]–[9].

Shared-memory scheduling in MPSoCs offers manifold advantages and serves to enhance system performance while optimizing resource utilization and scalability. For instance, memory-shared scheduling makes it possible for several cores to share a single memory space, which optimizes memory access patterns. As a result, data can be cached and distributed more wisely among cores, reducing memory contention, and communication overhead, increasing memory access and energy efficiency [10], [11]. In task scheduling, tasks are categorized into dependent task graphs and independent task graphs. Dependent task graphs represent tasks with dependencies, where the execution of one task is dependent on the completion of another task. Dependent task graphs, such as Directed Acyclic Graphs (DAGs) or Periodic Conditional Task Graphs (PCTGs) are commonly used to model applications with complex inter-task dependencies, such as data processing pipelines or workflow-based applications. Independent task graphs in contrast represent tasks that can be executed independently of each other, without any inter-task dependencies. Tasks in independent task graphs can often be parallelized and executed concurrently, making them suitable for applications with parallelizable tasks, such as scientific simulations or parallel processing applications [5]. PCTGs play a crucial role in the realm of MPSoC architectures, finding applications in various domains. Some examples include real-time systems where tasks follow periodic patterns and may have dependencies based on specific conditions, such as in industrial automation, robotics, and automotive systems. PCTGs are also instrumental in scheduling periodic and condition-dependent tasks in Internet-of-Things (IoT) applications like smart homes, smart cities, and wearable devices, where tasks may dynamically adjust their execution based on events or sensor inputs. Additionally, PCTGs find effective use in multimedia streaming scenarios where tasks within the graph exhibit inter-dependencies. High-performance and high-efficiency scheduling for PCTGs ensures optimal resource utilization and timely task execution, enhancing system throughput, and responsiveness. However, it may introduce complexity in scheduling algorithms and require sophisticated optimization techniques, potentially leading to increased computational overhead and implementation challenges [8], [12], [13].

Applications involve a set of tasks where each task is subject to both timing and precedence constraints. Precedence constraints establish the relationships between tasks in terms of data and control dependencies. Traditionally, these constraints are unconditional, meaning that once a task is completed, it leads to the immediate execution of another specific task. However, in various applications, conditional precedence constraints are introduced, which adds complexity to the scheduling process. In the case of conditional task graphs, after the completion of a task v_i, it may lead to one of several possible tasks based on specific conditions. These conditions create alternative execution paths, introducing branches into the task graph. The incorporation of conditional precedence constraints significantly increases the number of potential scenarios that the scheduler must consider. This growth is exponential and directly proportional to the number of conditions present in the conditional task graph. As a result, the task scheduling problem becomes notably more challenging when dealing with conditional task graphs compared to non-conditional task graphs. Handling conditional task graphs efficiently and optimizing their schedules require specialized algorithms and methodologies to explore the multiple branching possibilities effectively. Scheduling such graphs demands careful consideration of the dependencies and conditions to achieve optimal performance and energy efficiency while respecting all timing constraints. Addressing conditional precedence constraints is crucial for accurately modeling real-world applications, as many modern systems involve tasks with complex dependencies and conditional execution paths. By developing innovative scheduling approaches capable of handling conditional task graphs, researchers can unlock the full potential of these advanced applications in energy-efficient and high-performance heterogeneous MPSoC systems.

This research investigates the scheduling problem associated with PCTGs comprising non-preemptible tasks, implemented on a MPSoC computing platform. The MPSoC is endowed with identical processors capable of functioning at discrete voltage levels and features shared memory. The main objective of this investigation is to minimize the overall expected energy consumption of tasks across diverse scenarios on processors. In pursuit of this aim, we consider two power models: the Total Power (TP) model and the Dynamic Power

(DP) model.

The key contributions of our research and implementations are given as follows:

- We propose a two-phase scheduling framework integrating an efficient offline scheduler and a streamlined online scheduler. This framework is designed for the generalized task model of conditional task graphs (CTGs) and the specific case of task graphs, demonstrating its applicability and efficiency across both contexts. The approach ensures adaptability to both the broad constructs of CTGs and the nuanced specifics of task graphs, highlighting its universal relevance.
- Our offline scheduler comprises two essential components. The first is a pioneering task scheduling algorithm tailored for the periodic conditional task graph model, capable of creating a unified global schedule applicable across all possible scenarios of periodic CTGs. This aspect of our offline scheduler ensures its time complexity remains within polynomial bounds. The second component is our task-to-voltage assignment algorithm, based on convex Non-Linear Programming (NLP), specifically developed for the general task model of Periodic Conditional Task Graphs. Together, this two-part strategy significantly reduces energy consumption while preserving operational efficiency within a polynomial time, demonstrating its versatility across various task models.
- We have developed an efficient online Dynamic Voltage Scaling (DVS) heuristic with minimal time complexity $O(1)$. This heuristic is strategically designed to redistribute slack in a broad spectrum of scenarios, accommodating both the general framework of conditional task graphs and the specialized subset of task graphs. This development underscores our methodology's flexibility and wide applicability.
- Our Energy-efficient Successor Tree Consistent Earliest Deadline First (EESEDF) Algorithm achieved an average improvement of 15%, a maximum improvement of 17%, and a minimum improvement of 12% over the online DVS heuristic. These improvements highlight the effectiveness of our EESEDF Algorithm in optimizing task scheduling and reducing energy consumption. Our novel scheduling technique, EESEDF, also outperforms state-of-the-art LESA [14] and NCM [15] achieving energy efficiency of 25% and 20% respectively.
- Our two-phase approach not only optimizes energy consumption but also exhibits scalability. With its polynomial time complexity, it is exceptionally suited for scheduling Periodic Conditional Task Graphs (PCTGs). Compared to BESS [16] and CAP-Online [17], our approach achieves an average improvement of 25% over BESS and a 35% improvement over CAP-Online in terms of expected energy consumption. Additionally, we have demonstrated that both these approaches fall short in scalability, particularly for CTGs with a large number of conditions. This shortfall is due to both approaches being exponential in the number of conditions within CTGs.

The paper's organization is as follows: Section II provides an overview of the related work, highlighting the existing literature and approaches relevant to our research. Section III presents the power models, task models, and system models used in our study. This section outlines the fundamental models that form the basis of our offline scheduling approach and energy optimization techniques. In Section IV, we delve into the details of our novel offline scheduling algorithm Energy Efficient Successor Tree Consistent Earliest Deadline First (EESEDF). This section explains the workings of our proposed algorithm for task assignment and scheduling. Section V presents our NLP-based approach for determining an optimal speed assignment for each task in the scheduling phase. The experimental results are showcased in Section VI, illustrating the outcomes of our evaluations and providing insights into the performance and energy efficiency of our approach. Lastly, Section VII concludes the paper, summarizing the key findings and contributions of our research.

II. RELATED WORK

Aydin et al. introduced an energy-efficient scheduling algorithm, utilizing Dynamic Voltage and Frequency Scaling (DVFS) for independent real-time tasks with diverse power consumption characteristics on multiprocessor systems. The scheduling problem was formulated as a Nonlinear Programming (NLP) task, optimizing task speeds while ensuring optimality [18]. Other research studies have also explored DVFS techniques for energy optimization. For instance, Zhang et al. in [19] presented the Shuffled Frog Leaping Algorithm (SFLA), a meta-heuristic scheduling algorithm that combines Particle Swarm Optimization (PSO) and Memetic algorithms, and compared its energy efficiency with Genetic Algorithms (GA). Additionally, Kumar et al. in [20] integrated task mapping and voltage assignment using GA within a single optimization loop, leveraging DVFS to reduce dynamic energy consumption while maintaining an acceptable performance trade-off. Moreover, Wang et al. focused on preemptive periodic independent task scheduling through Discrete Event System (DES) supervisory control [21]. Furthermore, Liu et al. [22] deployed the Weighted Earliest Finish Time (WEFT) algorithm for task mapping and executing tasks with the minimum possible earliest completion time. These investigations in [18]–[22] aimed to reduce energy consumption for independent tasks operating on MPSoC architectures, without explicitly considering precedence constraints.

Several other studies have been conducted to optimize energy consumption and improve performance in heterogeneous MPSoC systems with various scheduling and voltage assignment techniques. Chen et al. in [23] formulated energy consumption-constrained scheduling as an optimization problem to reduce the schedule duration of workflows. First, they modeled the workflows and energy consumption of processors. They also devised a novel scheduling algorithm based on energy difference coefficients, coupled with an enhanced energy per-assignment strategy. This approach aims to generate an allocation of processors, frequencies, and start times for each task that approximates optimality while ensuring compliance with data dependency and energy limitation constraints. An integrated approach was developed by Ali et al. for task mapping, scheduling, and voltage assignment on Network-on-Chip (NoC) based heterogeneous MPSoCs. They introduced a heuristic named EIMSVS to reduce both the processing and communication energies [24]. Abd Ishak et al. in [25] investigated non-preemptive scheduling for tasks with precedence constraints and individual deadlines. They used NLP and Integer Linear Programming (ILP) techniques to assign optimal voltages to tasks and communications on NoC links, aiming to minimize energy consumption. Ali et al. in [7] developed an energy-efficient task scheduling approach using the CITM-VA meta-heuristic. This method integrated DVFS and Dynamic Power Management (DPM) techniques to achieve maximum energy savings by considering contention at NoC links. Ding et al. introduced the HGAAP heuristic for task mapping on heterogeneous multiprocessor architectures. The goal was to optimize energy consumption by finding efficient task mappings [26]. Tariq et al. in [27] performed energy-efficient and contention-aware static scheduling for tasks with precedence and deadline constraints on NoC-based MPSoCs with DVFS-enabled processors. They designed the ARSH-FATI metaheuristic that collectively performed task mapping, scheduling, and voltage scaling, resulting in superior performance. Additionally, they developed the EECDF scheduling algorithm, which considered communication contention awareness. However, it is important to note that these studies focused mainly on MPSoC systems with tasks having precedence constraints, and their approaches aimed to optimize energy consumption and performance in different ways. Each of these research works contributed valuable insights to the field of energy-efficient scheduling in heterogeneous MPSoCs, helping to address the challenges of modern embedded systems. Xie et al. in [28] designed a scheduling algorithm known as fairness on multiple HEFT (F-MHEFT). They optimized scheduling applications based on multiple DAGs, aiming to achieve both high performance and meet stringent timing constraints. The authors focused on fairness and prioritizing meeting timing constraints for applications.

More recently Chen et al. developed a List-based Energy-aware Scheduling Algorithm (LESA) that incorporates task prioritization and weight-based energy distribution strategies. This algorithm deploys DVFS to assign discrete speed levels to the tasks while seeking an approximate optimal schedule by considering the task dependencies and energy constraints of the system [14]. Maurya et al. in [15] introduced an enhanced version of the Not Changing Makespan (NCM) sub-algorithm within the Energy Aware Service Level Agreement (EASLA) task scheduling framework. The improved algorithm is tailored for DVFS-enabled heterogeneous cluster systems and incorporates the Predict Earliest Finish Time (PEFT) algorithm to efficiently compute the schedule length. This approach aimed to optimize energy consumption while maintaining the desired level of performance for task execution. Roy et al. [29] addressed the challenge of scheduling periodic real-time applications, each represented as DAG, on a distributed platform with heterogeneous processors connected by shared buses. It leverages DVFS to minimize energy consumption while ensuring that DAG instances meet their deadlines within a hyperperiod. Initially formulated as a constraint optimization problem and developed a three-phase list-based hierarchical scheduling algorithm named Slack Aware Frequency Level Allocator (SAFLA). In another study, Roy et al. [30] explored the challenge of scheduling tasks represented as DAG. An optimal solution utilizing ILP is initially introduced for execution on distributed heterogeneous processors connected by shared buses. However, the ILP approach proves computationally intensive and impractical for moderately large problems. Therefore, a low-overhead heuristic algorithm named the Contention Cognizant Task and Message Scheduler (CC-TMS) is developed. This algorithm offered efficient and fast solutions within a reasonable time frame. Devaraj and Sarkar presented an approach for generating fault-tolerant schedules for real-time tasks represented as precedence-constrained task graphs running on multicore systems. It also outlines strategies to optimize these schedules, maximizing fault tolerance and minimizing peak-power dissipation [31]. Sharma et al. introduced a heuristic method called ETA-HP designed to optimize energy and temperature efficiency when scheduling real-time periodic tasks on a heterogeneous multicore system with DVFS capability. This technique consists of four stages: Deadline Partitioning, Task-to-Core Allocation, Temperature-Aware Scheduling, and Energy-Aware Scheduling [32]. Moulik et al. developed a heuristic approach, called CEAT for scheduling real-time periodic tasks on a heterogeneous multicore platform with DVFS support, focusing on energy efficiency. The proposed strategy involves three main stages: Deadline Partitioning, Task-to-Core Allocation, and Energy-Aware Scheduling [33]. Several related studies have explored temperature-aware and energy-efficient schedulers for multicore processors. These include works such as [34]–[36] which have proposed various techniques to optimize task scheduling considering both temperature and energy consumption on multicore architectures. However, these scheduling techniques presented in [7], [14], [15], [23]–[36] do not consider PCTGs do not consider PCTGs.

The energy minimization problem for tasks with conditional precedence constraints has seen only a few proposed approaches. One such approach, presented by Shin and Kim in [37] focuses on scheduling conditional task graphs while considering energy minimization through DVS. However, their solution does not address task mapping and assumes it is fixed. They use an insertion-based approach for task ordering, considering mutual exclusion relations between tasks. Walsh in [38] proposed a stochastic non-linear model for task speed assignment to minimize energy consumption, resulting in a schedule represented in a table format with tasks assigned at different speeds and start times based on conditions. Unfortunately, the size of the schedule table grows exponentially with the number of conditions, leading to the problem of finding an optimal schedule table becoming P-Space complete. Another technique proposed by Wu et al. [39] utilizing DVS for energy consumption minimization. They use the schedule table from [40] to identify the available slack time in worst-case scenarios. Furthermore, they demonstrate that combining their DVS-based technique with a genetic algorithm for task mapping can yield additional energy savings. However, their approach assumes that all branches are equally taken, which may not be realistic in practice. Additionally, the size of the schedule table still grows exponentially with the number of conditions, and the genetic algorithm-based mapping can result in very high complexity due to the task speed assignment and ordering being handled by the inner loop of the algorithm for the entire conditional task graph [16]. In summary, the limited existing approaches for minimizing energy consumption in tasks with conditional precedence constraints suffer from challenges related to the exponential growth in schedule table size with the number of conditions and high complexity when using genetic algorithms for task mapping. Developing more efficient algorithms to address these issues remains an ongoing research area. Malani et al. in [17] introduced an online scheduling algorithm named CAP-Online, designed for conditional task graphs (CTGs) with a shared deadline, operating under the dynamic power model. In this algorithm, when a task is scheduled, it calculates the critical path, identifies the available slack time for that task, and then stretches it to make use of the available slack. However, since CAP-Online needs to enumerate all the paths of the conditional graph, its time complexity grows exponentially as the number of tasks increases. Tariq and Wu and Tariq et al. in [3], [41] proposed an energy-efficient priority-based list scheduler for scheduling CTGs on MPSoCs. They use successor-tree-consistent-deadline as a priority for each task and employ NLP-based dynamic voltage scaling algorithms to assign voltage to tasks. Additionally, techniques presented in [3] assume that processors operate at discrete frequency levels whereas the approach in [41] considers processors operate at continuous frequency levels. The methods proposed by [3], [16], [17], [41] are tailored for the scheduling of CTGs on MPSoCs. However, a pressing requirement exists for an energy-efficient approach to schedule PCTGs on MPSoCs. TABLE 1 comprehensively summarizes the literature on task scheduling techniques on multiprocessor systems.

In summary, our novel scheduling algorithms focus on the energy-aware scheduling problem for dependent tasks, particularly PCGTs in multiprocessor systems to increase energy efficiency and overall performance using novel heuristics.

III. SYSTEM MODELS

The target MPSoC is composed of a set $P = pe_1, pe_2, ..., pe_m$ consisting of m identical processors as depicted in FIGURE 1. Each processor pe_i is equipped with DVFS capabilities. This feature enables each processor to operate at a discrete set of k finite frequency levels, denoted as $[f_1, f_2, \ldots, f_k]$. The total power consumption of a processor, pe_i, is determined by considering both dynamic power due to switching activity and static power resulting from leakage. This total power can be computed using the following equation (1), as presented in [42]:

$$P_{tot} = C_{eff}.V_{dd}^2.f + L_g.(V_{dd}.K_3.e^{K_4.V_{dd}}.e^{K_5.V_{bs}} + |V_{bs}|.I_j) \quad (1)$$

Here, $C_{eff}.V_{dd}^2.f$ represents the dynamic power, where C_{eff} is the effective capacitance, V_{dd} denotes the supply voltage, and f represents the frequency at which the processor operates. The term L_g denotes the number of logic gates in the circuit. The equation also includes parameters K_3, K_4, and K_5, which are specific to the processor technology being used. The values of these parameters depend on the manufacturing process and the characteristics of the processors. Additionally, V_{bs} represents the body-bias voltage, and I_j is the body junction leakage current. These terms are associated with the static power component due to leakage. For a comprehensive understanding of the power model and its intricacies, further details can be referred to in the original work presented in [42]. In essence, this power model serves as a crucial tool in assessing the total power consumption of each processor within the MPSoC considering both dynamic and static power contributions. By leveraging DVFS capabilities the system can dynamically adjust the frequency and voltage of each processor to achieve energy-efficient operation while meeting the performance requirements.

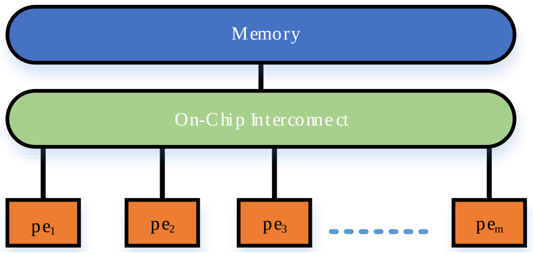


FIGURE 1: Shared-Memory MPSoC Platform

In this study, we consider a set of applications denoted by $\Gamma = \tau_1, \tau_2, \ldots, \tau_n$. These are n independent periodic applications. Each application, denoted as $\tau_i \in \Gamma$, is described by a 3-tuple (G_i, D_i, T_i). The components of this tuple are as follows:

$G_i = (V_i, E_i, A_i)$ represents a CTG. A CTG is a DAG with the following characteristics:

- $V_i = v_{i,1}, v_{i,2}, \ldots, v_{i,k}$ is a set of vertices, where each vertex $v_{i,j} \in V_i$ represents a task. A task $v_{i,j}$

TABLE 1: Summary of Literature on Task Scheduling Techniques

Reference	Task Model	Approach	Limitations
[18]–[22]	Independent	Offline	Minimize energy usage for independent tasks running on multicore architectures without explicitly considering task dependencies or order of execution.
[7], [14], [15], [23]–[36]	TG	Offline	While these methods effectively lower energy consumption in multicore architectures however they overlook conditional precedence constraints.
[37]–[40]	CTG	Offline	The exponential increase in schedule table size with the growth of conditions, coupled with the complexity involved in utilizing genetic algorithms for task mapping, renders these approaches unsuitable for PCTGs. Additionally, the time complexity of these approaches is exponential in the number of conditions in CTGs, making them ill-suited for PCTGs due to the large number of conditions.
[3]	CTG	Offline	A promising offline scheduler, designed for CTGs for deployment on heterogeneous systems, operates within polynomial time. However, this offline scheduler is missing an efficient online counterpart required for effectively distributing slack that becomes available during runtime.
[41]	CTG	Hybrid	Tailored for the continuous speed processor model, this approach introduces both an offline and an online scheduler for scheduling a single CTG on an MPSoC platform.
[16], [17]	CTG	Hybrid	The offline and online schedulers are specifically developed for single CTGs, featuring a time complexity that is exponential in the number of conditions within CTGs. This design limitation restricts their straightforward extension for scheduling PCTGs, rendering them unsuitable for PCTG scheduling.

corresponds to a sequential unit of execution and has a worst-case execution time denoted by $w_{i,j}$ at maximum processor frequency.
- $E_i \subset V_i \times V_i$ is a set of directed edges that represent the dependencies among tasks. For example, an edge $(v_{i,j}, v_{i,k})$ indicates that task v_{ij} must be completed before task $v_{i,k}$.
- A_i is a set of triplets $(e_{i,j}, c_{i,j}, p(c_{i,j}))$, where $e_{i,j} \in E_i$. The components $c_{i,j}$ and $p(c_{i,j})$ represent the condition associated with $e_{i,j}$ and its probability, respectively.

The parameters $D_{i,j}$ and T_i are positive integers that denote the relative deadline of j^{th} task of τ_i and period of τ_i, respectively. It holds that $D_{i,j} \leq T_i$. In the context of a CTG $\tau_i \in \Gamma$, a node is considered a source node if its in-degree is zero, while a node is classified as a sink node if its out-degree is zero. We assume that each τ_i possesses a single source node v_{source} and a single sink node v_{sink}. If this condition is not met, we add dummy nodes v_{source} and v_{sink} with worst-case execution times set to zero. Additionally, we connect v_{source} to all source nodes and establish connections from all sink nodes to v_{sink}.

In a CTG, the edges are categorized as either conditional or unconditional. A conditional edge is linked to a condition that determines whether the subsequent task will be executed. In contrast, an unconditional edge has no such associated condition. Every non-sink node in a CTG is considered a FORK node. A FORK node with multiple outgoing edges can be classified as either an AND-FORK node or an OR-FORK node. In an OR-FORK node, all outgoing edges represent conditional edges with mutually exclusive conditions, ensuring that only one of the immediate successors will be executed. The probabilities associated with all the conditional edges of an OR-FORK node add up to one. An AND-FORK node, on the other hand, has all its outgoing edges as unconditional edges, meaning that all immediate successors will be executed without any conditional restrictions.

Additionally, any non-source node in a CTG is also a JOIN node. A JOIN node with multiple incoming edges can be either an AND-JOIN node or an OR-JOIN node. In the case of an OR-JOIN node, all the parent nodes are mutually exclusive, meaning that only one among them will be executed. An AND-JOIN node is defined by having all its parent tasks executed.

OR-FORK and OR-JOIN nodes are always present in pairs and are collectively referred to as a conditional construct, enabling the modeling of conditional constructs such as **if-then-else**.

The properties of OR-FORK and OR-JOIN Nodes in Conditional Task Graphs are.
- **Correspondence and Traversal:** An OR-FORK node with k children (where $k > 1$) corresponds to an OR-JOIN node with exactly k parents. Notably, the OR-JOIN node does not directly succeed the OR-FORK node. Nevertheless, regardless of the path taken, any traversal starting from the OR-FORK node will inevitably lead to the OR-JOIN node.
- **Disjoint Branch-Spanning Sub-graphs:** For each OR-FORK node and its corresponding OR-JOIN node, there exists a property where any two paths originating from the OR-FORK node and leading to different immediate predecessors of the OR-JOIN node remain separate and do not overlap. A path refers to a sequence of vertices and edges traversed from an immediate successor of the OR-FORK node to an immediate predecessor of the OR-JOIN node. This definition ensures that each path from the OR-FORK to the OR-JOIN node remains distinct and does not intersect with other paths.

- **Absence of External Edges:** There are no external edges connected to the path encompassing all the vertices and edges traversed from the OR-FORK node to the OR-JOIN node. An external edge is defined as an edge connecting a vertex outside of this path. This property ensures that the path of an OR-FORK and its corresponding OR-JOIN node remains self-contained, with no external edges interfering with the paths connecting these nodes.

In the conditional task graph (CTG) depicted in FIGURE 2(a), v_2 and v_4 are OR-FORK nodes, v_1 is an AND-FORK node, v_9 and v_{10} are OR-JOIN nodes, and v_4 is an AND-JOIN node.

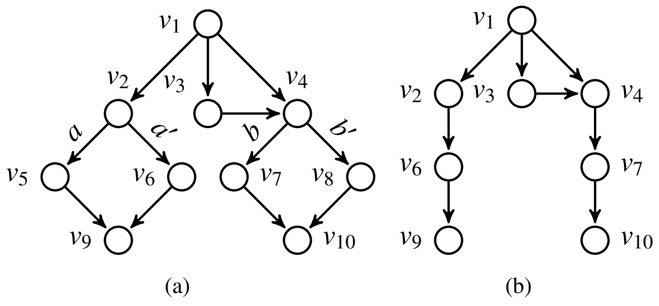


FIGURE 2: (a) CTG G_A of application τ_A (b) A scenario of the CTG G_A shown in Fig. 2(a)

A scenario in a CTG G_i represents a graph generated by a single complete execution trace of the CTG. FIGURE 2(b) illustrates an example scenario of the CTG G_A from FIGURE 2(a), where $a =$ false and $b =$ true.

In a CTG G_i, the activation space AS_i comprises all possible conditions, each corresponding to a unique scenario. For instance, the activation space of CTG G_A in FIGURE 2 is $AS_A = ab, ab', a'b, a'b'$. The probability of a scenario $s \in AS_A$, denoted by $p(s)$, is calculated using the following equation:

$$p(s) = \prod_{c \in s} p(c) \tag{2}$$

Here, c represents a condition belonging to the scenario s, and $p(c)$ is the probability when c is true.

Each task in the conditional task graph is associated with its "activation probability," which represents the likelihood of the task being executed. Let's denote the set of scenarios in which a specific task v_j belongs to S_j. The activation probability of task v_j is determined using the following calculation:

$$p(v_j) = \sum_{s \in S_j} p(s) \tag{3}$$

where $p(s)$ is defined in Equation (2).

We use two subscripts to refer to a task of a CTG for example $v_{i,j}$ refers to j^{th} task of CTG G_i.

In the embedded applications, tasks represent segments of code, often with dependencies dictating their order of execution. Classic DAG task models capture these relationships, where precedence constraints link tasks. However, some tasks have conditional dependencies, relying on specific conditions for execution. CTGs can model applications with such constraints. An example is the MPEG decoder [43] as demonstrated in FIGURE 3, where the decoding process varies based on the video frame, with different procedures for I, P, and B macro-block classifications. Traditional DAG-based models cannot effectively represent these variable behaviors found in applications.

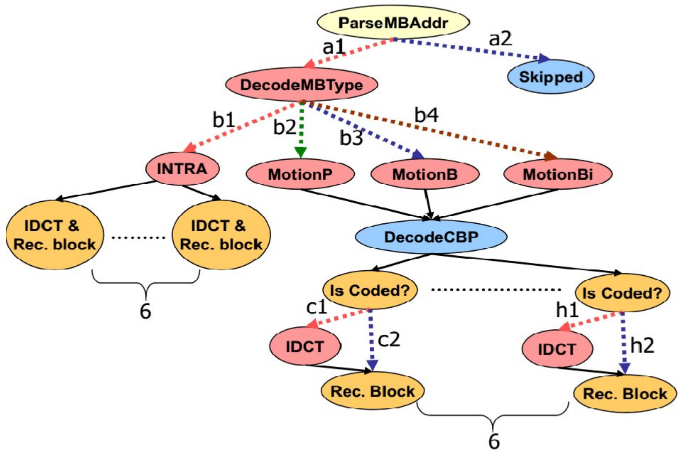


FIGURE 3: MPEG Decoder: A Real-world Example of the CTG

IV. ENERGY EFFICIENT SUCCESSOR TREE CONSISTENT EARLIEST DEADLINE FIRST SCHEDULER

In this section, we explain our scheduler, EESEDF. The symbols used in explaining our novel scheduling and mapping approach are listed in TABLE 2.

A. CTG PRIORITY

EESEDF schedules CTGs in the Γ one by one. For this purpose each CTG G_i in set Γ is assigned a priority ω_i as follows:

$$\omega_i = \frac{\Omega_i}{T_i} \tag{4}$$

where Ω_i is the worst-case workload of the CTG G_i and it calculated as follows:

$$\Omega_i = \max\{\sum_{v_{i,j} \in s} w_{i,j} : s \in AS_i\} \tag{5}$$

The parameter ω_i signifies the priority of individual applications, where a greater ω_i value implies a higher priority. ω characterizes the computational weight of an application. Scheduling heavier applications at an earlier instance ensures ensures effective utilization of MPSoC resources.

TABLE 2: Symbols and Descriptions

Symbol	Description
E	Set of directed edges
AS	Set of all possible scenarios
A_i	Set of triplets
μ^k	Worst-case utilization
C_i^k	Maximum sum of worst-case execution time
CTG	Conditional task graph
TG	Task graph
G_s	Super graph
WCS_{ij}	partial worst-case sets
ω_i	Priority of each PCTG
$\mathrm{w}_{i,j}$	Worst case execution time of j^{th} task of τ_i.
v_{ij}	Task
$v_{1,sink}$	Sink node
N	Number of jobs
$est(v_{i,j,u}, pe_k)$	Earliest start time of job
G_s	Precedence constraints
E_R	New set of edges
G_R	Reduced super graph
$CurrTime$	The time when online heuristic is invoked
ECD_i	Edge consistent deadline
et_i	The worst-case execution time
NCC_i	Clock cycles
CP_i	Critical path
V_{dd}	Supply voltage
pe_k	Finish time of a job
$r_{i,j,u}$	Release time of a job
$D_{i,j}$	Relative deadline of j^{th} task of τ_i
$d_{i,j,u}$	Deadline of a job
$\rho_{i,j,u}$	Start time of a job

B. TASK SCHEDULING

The task assignment algorithm allocates each task to a processor such that the total worst-case utilization of all the processors is minimized. This way, the workload across all processors is balanced. The worst-case utilization of pe_k is computed as follows:

$$\mu^k = \sum_{\tau_i \in \Gamma} \frac{C_i^k}{T_i} \tag{6}$$

where C_i^k is the maximum sum of the worst-case execution time of all the tasks of CTG τ_i assigned to pe_k in all possible scenarios of CTG τ_i:

$$C_i^k = \max\{ \sum_{v_{i,j} \in s, pe_k} w_{i,j} : s \in AS_i\} \tag{7}$$

Thus, equation (6) can be written as:

$$\mu^k = \sum_{\tau_i \in \Gamma} \frac{\max\{\sum_{v_{i,j} \in s, pe_k} w_{i,j} : s \in AS_i\}}{T_i} \tag{8}$$

Example 1: Consider the applications shown in FIGURE 4. Assume that the two applications are assigned to two processors and the task assignment is $pe_1 = [v_{1,1}, v_{1,4}, v_{1,5}, v_{1,6}, v_{2,1}, v_{2,2}, v_{2,3}, v_{2,4}]$ $pe_2 = [v_{1,2}, v_{1,3}]$. The worst case execution times are $w_{1,1} = 0.5, w_{1,2} = 1, w_{1,3} = 5, w_{1,4} = 2.5, w_{1,5} = 1, w_{1,6} = 0.5, w_{2,1} = 1, w_{2,2} = 2, w_{2,3} = 1, w_{2,4} = 1$. The periods of the applications are $T_1 = 9, T_2 = 18$. For simplicity, we assume that deadlines are equal to the periods. For the given mapping the worst case utilization for pe_1 using Equation (8) is:
$\mu^1 = \frac{\max\{\sum_{v_{1,j} \in s, pe_k} w_{1,j} : s \in AS_1\}}{T_1} + \frac{\max\{\sum_{v_{2,j} \in s, pe_k} w_{2,j} : s \in AS_2\}}{T_2}$
The activation space for the application τ_1 is $AS_1 = \{a, a'\}$ and for application τ_2 is $AS_2 = \{b, b'\}$. FIGURES 3(c), 3(d), 3(e) and 3(f) show the scenarios corresponding to conditions a a' b and b'respectively. Given the activation spaces and corresponding scenarios the utilization of processor pe_k is:
$\mu^1 = \frac{\max\{ \{w_{1,1}+w_{1,4}+w_{1,6}\}_a, \{w_{1,1}+w_{1,5}+w_{1,6}\}_a{}'\}}{T_1} + \frac{\max\{ \{w_{2,1}+w_{2,2}+w_{2,4}\}_b, \{w_{2,1}+w_{2,3}+w_{2,4}\}_b{}'\}}{T_2} = \frac{\max\{3.5,2\}}{9} + \frac{\max\{4,3\}}{18}$
$=0.6111$.

Equation (8) can be used directly to determine the worst-case utilization of pe_k. However, it requires enumerating all possible scenarios of CTGs mapped to pe_k. Since the number of scenarios of a CTG, in general, is exponential, therefore, calculating the worst-case utilization using the equation (8) has exponential time complexity. The worst-case utilization of a processor can be computed in polynomial time if we can determine $C_i^k = \max\{\sum_{v_{i,j} \in s, pe_k} w_{i,j} : s \in AS_i\}$ in polynomial time. The value C_i^k can be determined if we can find a set of tasks amongst all the tasks of τ_i assigned to pe_k such that the sum of worst-case execution times of all the tasks in the set is the maximum among all possible scenarios of τ_i. Let such a set be represented by S_i^k and Γ^k be a set of all the CTGs, each of which has at least one task assigned to pe_k, we have:

$$S^k = \bigcup_{\tau_i \in \Gamma^k} \{S_i^k\} \tag{9}$$

Using S^k, Equation (8) can be written as:

$$\mu^k = \sum_{S_i^k \in S^k} \frac{\sum_{v_{i,j} \in S_i^k} w_{i,j}}{T_i} \tag{10}$$

We develop a polynomial time Algorithm 2, for calculating the S_i^k for pe_k. Given the set L_i^k of all the tasks of G_i assigned to pe_k, Algorithm 2 computes the S_i^k by computing the partial worst-case sets WCS_{ij} of each task $v_{ij} \in V_i$ in reverse topological order. The definition of $WCS_{i,j}$ is as follows:

The worst-case set of a task $v_{i,j} \in V_i$ is computed based on the following three cases.

- $v_{i,j}$ is a sink node. If $v_{i,j} \notin L_i^k$ holds, we have $WCS_{i,j} = \emptyset$. Otherwise we have $WCS_{i,j} = \{v_{i,j}\}$.
- $v_{i,j}$ is an OR-FORK node and $v_{i,o} \in ISucc_{i,j}$ satisfies $\sum_{v_{i,l} \in WCS_{i,j}} W_{i,l} = \max\{\sum_{v_{i,l} \in WCS_{i,q}} W_{i,l} : v_{i,q} \in ISucc_{i,j}\}$. If $v_{i,j} \notin L_i^k$ holds, we have $WCS_{i,j} = WCS_{i,o}$. Otherwise, we have $WCS_{i,j} = WCS_{i,o} \cup \{v_{i,j}\}$.
- $v_{i,j}$ is an AND-FORK node. We have

$$WCS_{i,j}^k = \begin{cases} \bigcup_{v_{i,l} \in ISuc_{i,j}} WCS_{i,l} & if\ v_{i,j} \notin L_i^k \\ \bigcup_{v_{i,l} \in ISuc_{i,j}} WCS_{i,l} \cup \{v_{i,j}\} & otherwise \end{cases}$$

Example 2: Let's demonstrate how ALGORITHM 2 computes each set S_i^k in this example. We consider the applications τ_1 and τ_2, and their task assignments as given in

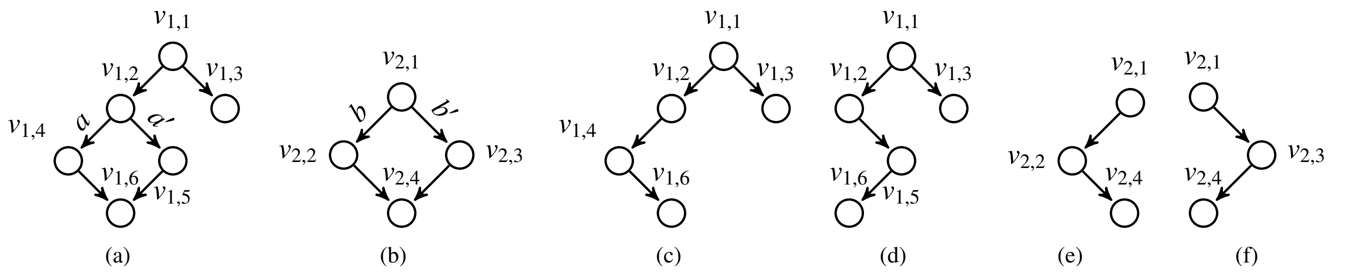


FIGURE 4: (a) CTG G_1 of application τ_1 (b) CTG G_2 of application τ_2 (c) Scenario corresponding to a (d) Scenario corresponding to a' (e) Scenario corresponding to b (f) Scenario corresponding to b'.

Example 1. The first step is to transform CTG G_1 into a single sink node graph by adding a sink node $v_{1,sink}$ and the corresponding edges: $V_1 = V_1 \cup v_{1,sink}$ and $E_1 = E_1 \cup (v_{1,6}, v_{1,sink}), (v_{1,3}, v_{1,sink})$. The weight of the sink node is set to $w_{1,sink} = 0$. Let's focus on determining the set S_1^1. In ALGORITHM 2, we have a list Θ that contains all the tasks of CTG G_1 sorted in topological order. For L_1^1 (with $i = 1$ and $k = 1$), it contains all the tasks of CTG G_1 mapped to pe_1, i.e., $L_1^1 = [v_{1,1}, v_{1,4}, v_{1,5}, v_{1,6}]$. The list Θ is as follows: $\Theta = [v_{1,1}, v_{1,2}, v_{1,4}, v_{1,5}, v_{1,3}, v_{1,6}, v_{1,sink}]$. ALGORITHM 2 traverses the list Θ in reverse topological order, starting from the sink node and moving towards the source node. It computes the partial worst-case sets for each task, initially set to empty. Let's go through the algorithm step by step:

1) First, the algorithm selects the sink node $v_{1,sink}$. Since $v_{1,sink}$ is a sink node and $v_{1,sink} \notin L_1^1$, we have $WCS_{1,sink} = \emptyset$.
2) Next, it selects $v_{1,6}$. As $v_{1,6}$ is an AND-FORK node with only one immediate successor, which is $v_{1,sink}$, and $v_{1,6} \in L_1^1$, we have $WCS_{1,6} = WCS_{1,6} \cup WCS_{1,sink} \cup v_{1,6} = v_{1,6}$.
3) Then, $v_{1,3}$ is selected. Since $v_{1,3} \notin L_1^1$, we have $WCS_{1,3} = WCS_{1,3} \cup WCS_{1,6} = \emptyset$.
4) Next, $v_{1,5}$ is chosen. As $v_{1,5} \in L_1^1$, we have $WCS_{1,5} = WCS_{1,5} \cup WCS_{1,6} \cup v_{1,5} = v_{1,6}, v_{1,5}$. Additionally, $WCS_{1,4} = WCS_{1,4} \cup WCS_{1,6} \cup v_{1,4} = v_{1,4}, v_{1,6}$.
5) $v_{1,2}$ is an OR-Fork node with two immediate successors, $v_{1,4}$ and $v_{1,5}$. Since $\sum_{v_{1,j} \in WCS_{1,4}} w_{1,j} > \sum_{v_{1,l} \in WCS_{1,5}} w_{1,l}$ (i.e., $(2 + 0.5) > (1 + 0.5)$) and $v_{1,2} \notin L_1^1$, we have $WCS_{1,2} = WCS_{1,2} \cup WCS_{1,4} = v_{1,4}, v_{1,6}$.
6) Finally, we have $WCS_{1,1} = WCS_{1,1} \cup WCS_{1,2} \cup WCS_{1,3} \cup v_{1,1} = v_{1,1}, v_{1,4}, v_{1,6}$ as $v_{1,1} \in L_1^1$.

After traversing the list Θ, we obtain $S_1^1 = WCS_{1,1}$ since $v_{1,1}$ is the source node in G_1. Similarly, we can compute S_2^1 as $S_2^1 = v_{2,1}, v_{2,2}, v_{2,4}$. Given the sets S_1^1, S_2^1, and $S^1 = S_1^1, S_2^1$, the utilization of pe_1 can be calculated using Equation (10): $\mu^1 = \sum_{S_i^1 \in S_1^1, S_2^1} \frac{\sum_{v_{1,j} \in S_i^1} w_{1,j}}{T_i} = \frac{\sum_{v \in S_1^1} w}{T_1} + \frac{\sum_{v \in S_2^1} w}{T_2} = \frac{(w_{1,1}+w_{1,4}+w_{1,6})}{T_1} + \frac{(w_{2,1}+w_{2,2}+w_{2,4})}{T_2} = \frac{3.5}{9} + \frac{4}{18} = 0.6111$. It's worth noting that the worst-case utilization for pe_1 computed using Equation (10) is the same as the utilization obtained using Equation (8). The key difference is that using Equation (10), the worst-case utilization of any processor can be obtained in polynomial time.

The details of our task assignment algorithm are given in ALGORITHM 1. It first selects a CTG in $\tau_i \in \Gamma$ that has the maximum value of ω (line 8) and computes the successor tree deadline of each $v_{i,j} \in V_i$ (line 9). Then it constructs a list Θ that contains all the tasks of G_i sorted in non-increasing order of successor tree deadlines (line 10). Sorting Θ in non-increasing order of successor tree deadlines implies that Θ is sorted in topological order. Once a CTG G_i is selected, all its tasks are assigned to processors and scheduled (lines 11-35). ALGORITHM 1 tentatively assigns each task $v_{ij} \in V_i$ to each processor $pe_k \in P$ and computes the total worst-case utilization of pe_k (lines 11-20). A list of L_i^k is constructed that contains all the tasks of G_i assigned to pe_k (line 11). The tasks that are mutually exclusive to $v_{i,j}$ are removed from L_i^k (line 12). Notice that each time only the set S_i^k of G_i is re-computed because only the set S_i^k is affected if v_{ij} is assigned to pe_k (lines 15-18). The task v_{ij} is assigned to a processor that has minimum worst-case utilization (line 21). Task $v_{i,j}$ can submit an infinite number of jobs separated by the period T_i, where each of its jobs has a worst-case execution time that equals to $w_{i,j}$. We use three subscripts to refer to a job of a task. $v_{i,j,u}$ refers to u^{th} job of the task $v_{i,j}$. For periodic CTGs, it is sufficient to construct a schedule for one hyper-period, which is the least common multiple of the periods of all the applications in the task set Γ. ALGORITHM 1 determines the number of jobs N of task $v_{i,j}$ (line 21) as follows:

$$N = \frac{H}{T_i} \tag{11}$$

For each job $v_{i,j,u}$ release time is computed as follows:

$$r_{i,j,u} = \max\{(u-1)T_i, \max\{\zeta(v_{i,l,u}) : v_{i,l,u} \in IPred(v_{i,j,u})\}\} \tag{12}$$

and the deadline is computed as follows as follows:

$$d_{i,,j,u} = (u-1)T_i + D_{i,j} \tag{13}$$

The start time (line 24) of a job $v_{i,j,u}$ is calculated as follows:

$$\rho_{i,j,u} = \max\{r_{i,j,u}, est(v_{i,j,u}, pe_k)\} \tag{14}$$

ALGORITHM 1: Energy Efficient Earliest Successor Tree Consistent Earliest Deadline First Algorithm (EESEDF)

```
input : Set $\Gamma$ of periodic applications, MPSoC with $m$
        processors, Set $\Omega$ of Priorities of all $\tau_i \in \Gamma$.
output: Super Graph $G_s(V, E)$.
1  $S^k \leftarrow \emptyset, \forall pe_k \in P$ ;
2  $\Gamma_1 \leftarrow \Gamma$;
3  $Failure \leftarrow False$;
4  Create conditional graph $G_s$ with empty sets E, V, and A;
5  while $\Gamma_1$ is not Empty do
6      Select $\tau_i \in \Gamma_1$ that has maximum value of $\omega$;
7      Compute the successor tree consistent deadline of each
         $v_{i,j} \in V_i$;
8      Construct a list $\Theta$ of all tasks $v_{ij} \in V_i$ sorted in
         non-increasing order of successor tree consistent
         deadline;
9      for each $v_{i,j} \in \Theta$ from source to sink do
10         for each $pe_k \in P$ do
11             $temp^k \leftarrow \emptyset$ ;
12             Construct set $L_i^k$ of all the tasks of CTG $\tau_i$
                 assigned to processor $pe_k$;
13             $L_i^k \leftarrow L_i^k \setminus Mutex_{i,j}$ ;
14             $L_i^k \leftarrow L_i^k \cup \{v_{i,j}\}$;
15             Find the previous $S_i^k \in S^k$ and assign to
                 $temp^k$;
16             $S_i^k \leftarrow worst\_case\_set(L_i^k, \Theta)$;
17             $S^k \leftarrow (S^k \setminus \{temp^k\}) \cup \{S_i^k\}$;
18             $\mu^k = \sum_{S_i^k \in S^k} \frac{\sum_{v_{i,l} \in S_i^k} w_{i,l}}{T_i}$;
19             $S^k \leftarrow (S^k \setminus \{S_i^k\}) \cup \{temp^k\}$ ;
20         Find $pe_k$ that has minimum worst case utilization
             $\mu^k$ and assign $v_{i,j}$ to $pe_k$;
21         $S^k \leftarrow (S^k \setminus \{temp^k\}) \cup \{S_i^k\}$;
22         Compute the number of instances $N$ of $v_{i,j}$ in $H$
             using equation (11);
23         for $u = 1$ to $N_i$ do
24             Compute the release time and deadline of $v_{i,j,u}$
                 using equations (12) and (13);
25             Compute $\rho_{i,j,u}$ the start time of $v_{i,j,u}$;
26             $\zeta_{i,j,u} \leftarrow \rho_{i,j,u} + w_{i,j,u}$;
27             $V \leftarrow V \cup \{v_{i,j,u}\}$;
28             $E \leftarrow E \cup \{(v_{i,l,u}, v_{i,j,u}) : \forall v_{i,l} \in IPred_{i,j}\}$;
29             $A \leftarrow A \cup \{(v_{i,l,u}, v_{i,j,u}) : \forall v_{i,l} \in$
                 $IPred_{i,j} \wedge v_{i,j}$ is OR-FORK node $\}$;
30     $\Gamma_1 \leftarrow \Gamma_1 \setminus \tau_i$;
31 Insert additional edges in $E$ subject to constraints C1, C2
     and C3;
32 Solve the NLP to assign each job an optimal speed;
```

ALGORITHM 2: The Worst-case scenario

```
1  $worst\_case\_set(L_i^k, \Theta)$
2  for each task $v_{i,j}$ in $\Theta$ from sink to source do
3      if $v_{ij}$ is a sink node then
4          $WCS_{i,j} \leftarrow \emptyset$;
5      else if $v_{i,j}$ is an OR-FORK node then
6          Find a child $v_{i,l}$ of $v_{i,j}$ among all the children of
             $v_{i,j}$ in the $\tau_i$ such that $\sum_{v_{i,q} \in WCS_{i,l}} w_{i,q}$ is
             maximized;
7          $WCS_{i,j} \leftarrow WCS_{i,l}$;
8      else
9          $WCS_{i,j} \leftarrow \emptyset$;
10         for each $v_{i,l} \in ISuc_{i,j}$ do
11             $WCS_{i,j} \leftarrow WCS_{i,j} \cup WCS_{i,l}$;
12     if $v_{i,j} \in L_i^k$ then
13         $WCS_{i,j} \leftarrow WCS_{i,j} \cup \{v_{i,j}\}$
14 $S_i^k \leftarrow WCS_{i,source}$;
15 return $S_i^k$;
```

where $est(v_{i,j,u}, pe_k)$ is the earliest start time of job $v_{i,j,u}$ on processor pe_k and it is the finish time of the latest scheduled job on pe_k that is concurrent to $v_{i,j,u}$. Tasks that are concurrent with $v_{i,j}$ are part of the set $cSet_{i,j}$. Two tasks, $v_{i,j}$ and $v_{i,k} \in G_i$, are considered concurrent if they cannot be reached from each other in G_i and are not mutually exclusive. Moreover, $cSet_{i,j}$ includes the following tasks:

$$cSet_{i,j} = cSet_{i,j} \cup \bigcup_{\tau_q \in \Gamma \setminus \tau_i} V_q \tag{15}$$

Additionally, all jobs of the same tasks are also concurrent.

The algorithm constructs the super graph $G_s(V, E, A)$ containing all jobs of the tasks of every CTG in the set Γ step by step (lines 29-31). Initially, set V is empty, at each step it adds the job $v_{i,j,u}$ to set V (line 29). Each job $v_{i,j,u}$ inherits the precedence constraints of task $v_{i,j}$ (line 30). Edges $(v_{i,l,u}, v_{i,j,u})$ for all $v_{i,l} \in IPred_{i,j}$ are added to the set E which is initially empty. Furthermore, if $(v_{i,l,u}, v_{i,j,u})$ is a conditional edge, it is inserted into the set A. Algorithm 1 repeats the process until all the CTGs have been successfully scheduled.

To account for the resource constraints introduced by the schedule, the algorithm inserts additional edges into the set E (Line 38). This insertion of edges helps manage and optimize the use of resources within the system. Specifically, an edge $(v_{i,j,u}, v_{w,o,l})$ is inserted between jobs $v_{i,j,u}$ and $v_{w,o,l}$ if the following three constraints are simultaneously satisfied:

- **C1:** Both $v_{i,j,u}$ and $v_{w,o,l}$ are scheduled on the same processor.
- **C2:** The start time of task $v_{w,o,l}$, denoted as $\rho_{w,o,l}$, is greater than the start time of $v_{i,j,u}$, denoted as $\rho_{i,j,u}$.
- **C3:** Task $v_{w,o,l}$ belongs to the concurrent set $cSet_{i,j,u}$, indicating that $v_{i,j,u}$ and $v_{w,o,l}$ are concurrent tasks.

It takes $O(ne)$ time to calculate successor-tree-consistent deadline, where n represents the count of vertices and e

the count of edges in G_S. The worst-case time complexity for computing the activation probabilities of all the tasks is $O(nClog(C)\eta)$ [13] where C is the number of conditions in the CTG and η is the maximum number of outgoing edges of all the OR-FORK nodes. To determine whether two tasks are mutually exclusive, we use the algorithm presented in [13]. It takes $O(n^2C^3)$ time to check if each pair of tasks are mutually exclusive. Demonstrating the worst-case time complexity for ALGORITHM 2 as $O(n+e)$ is straightforward. Given that ALGORITHM 2 executes m times per task, the resulting worst-case time complexity becomes $O(mn(n+e))$. Consequently, the worst-case time complexity for ALGORITHM 1 can be expressed as $O(ne+mn^2+mne+nClog(C)\eta+n^2C^3)$ excluding the NLP.

V. ASSIGNING SPEED TO TASKS

Our offline speed assignment approach deploys NLP for assigning each job the optimal speed.

A. NLP APPROACH

To address the task speed assignment problem, we formulate it as a convex Non-Linear Programming (NLP) problem. The main goal of this NLP problem is to minimize the expected energy consumption of a single global schedule for the hyperperiod H, while also ensuring that all timing and precedence constraints are met. In essence, the NLP problem seeks to find the optimal speeds for each task in the global schedule, such that the overall energy consumption is minimized, and the tasks can be executed within their specified time frames while respecting any dependencies between them. By formulating the problem as a convex NLP, we can effectively find a solution that simultaneously achieves energy efficiency and satisfies the various constraints imposed by the task dependencies and timing requirements. By solving this NLP problem, we obtain the optimal speed assignments for each task in the schedule, leading to a well-optimized global schedule that minimizes energy consumption and meets all the necessary timing and precedence constraints.

The energy consumed by a task, v_i running at a frequency f_i, is calculated as follows:

$$E(v_i, V_{dd_i}) = P(v_i)\ .\ w_i \tag{16}$$

In the context of the task speed assignment problem, we encounter two scenarios based on whether we aim to minimize the total processor energy or the total dynamic processor energy. This distinction is determined by the function $P(v_i)$, which can represent either the dynamic power function or the total power function. Under a specific power model, the expected total energy consumption of CTG G_s is expressed as follows:

$$\sum_{s\in AS} E(s)p(s) \tag{17}$$

In the context of the super graph G_s, the activation space AS represents the set of all possible scenarios. In each scenario, $s \in AS$, $p(s)$ denotes the probability, and $E(s)$ indicates the energy consumption.

However, calculating the expected energy consumption of a conditional task graph using Equation (17) incurs exponential time complexity. To alleviate this, for a global schedule where each task in $G_s(E, V, A)$ has a single speed for each scenario, the expected energy consumption is computed as follows:

$$\sum_{v_j\in V} E(v_j, V_{dd_i})p(v_j) \tag{18}$$

The super graph G_s effectively represents the precedence constraints imposed by both the initial global schedule and the original task graphs. However, it may contain redundant edges, which in turn introduce unnecessary constraints in the NLP formulation. To address this issue, we construct a reduced super graph denoted as G_R by applying transitive reduction to G_s. The transitive reduction process eliminates all redundant edges from G_s, resulting in a streamlined representation.

An edge (v_i, v_j) is considered redundant if there exists a path from task v_i to task v_j via an intermediate task v_z (where $z \neq i, j$). By removing such redundant edges, we obtain a new set of edges: E_R.

Having obtained the reduced super graph $G_R(V, E_R, A)$, the offline speed assignment problem minimizes the total expected energy given as follows:

$$\min \sum_{v_j\in V} E(v_j, V_{dd_j}).p(v_j) \tag{19}$$

subject to the following constraints:

- **Execution time constraints**:
$$\forall v_j,\ et_j = NC_j \frac{K_6 L_d V_{dd_j}}{((1+K_1)V_{dd_j} + K_2 V_{bs} - V_{th_1})^{\alpha}} \tag{20}$$
- **Precedence constraints**:
$$\forall (v_j, v_l) \in E_R, \quad \rho_j + et_j \leq \rho_l \tag{21}$$
- **Deadline constraints**:
$$\forall v_j \in V, \quad \rho_j + et_j \leq d_j \tag{22}$$
- **Release time constraints**:
$$\forall v_j \in V, \quad \rho_j \geq r_j \tag{23}$$
- **Upper and lower bound on supply voltage**:
$$\forall v_j \in V, \quad V_{dd_1} \leq V_{dd_j} \leq V_{dd_k} \tag{24}$$
- **Execution time non-negativity constraints**:
$$\forall v_j \in V, \quad \rho_j \geq 0 \tag{25}$$

Any invalid values are rounded up to the nearest higher discrete voltage level to guarantee the accuracy of the discrete voltage levels assigned to tasks using NLP. It is important to

ALGORITHM 3: Computing ECD, ECR and CP

```
input : Super Graph $G_s$, Execution times of tasks in super
        graph $G_s$ determined by NLP, Probabilities of all
        tasks in $G_s$
output: $ECD_i, ECR_i$ and $CP_i$ of each $v_i \in G_s$
1  Topological sort graph $G_s$;
2  for each $v_i \in G_s$ starting from Highest topological order
    do
3      $ECR_i \leftarrow \max\{r_i, \max\{ECR_j + et_j : \forall v_j \in IPred_i\}\}$;
4  for each $v_i \in G_s$ starting from lowest topological order do
5      $ECD_i \leftarrow \min\{d_i, \min\{ECD_j - et_j : \forall v_j \in ISuc_i\}\}$;
6      $CP_i \leftarrow 0$;
7      $\mathcal{P} \leftarrow 0$;
8      for each $v_j \in ISuc_i$ do
9          if $\mathcal{P} < p(v_j)$ || $(\mathcal{P} == p(v_j)$ && $CP_i < CP_j)$
            then
10             if $ECR_j < ECD_i$ then
11                 $CP_i \leftarrow CP_j$;
12                 $\mathcal{P} \leftarrow p(v_j)$;
13     $CP_i \leftarrow CP_i + et_i$;
```

ALGORITHM 4: Online Dynamic Voltage Scaling

```
input : $v_i, CurrTime, ECD_i, w_i, NCC_i$ and $CP_i$
output: $f_i$
1  Calculate the slack available for $v_i$:
     $Slack = ECD_i - CurrTime - et_i$;
2  Calculate the MultRatio for $v_i$: $MulRatio \leftarrow \frac{slack+CP_i}{CP_i}$;
3  Calculate the frequency $f_i$ for $v_i$: $f_i \leftarrow \frac{NCC_i}{et_i \times MulRatio}$;
4  Round $f_i$ to the nearest higher discrete frequency if $f_i$ is an
     invalid frequency level and execute it at $\min\{f_i, f_i'\}$,
     where $f_i'$ is the frequency assigned by NLP;
```

note that during this conversion process, where voltages assigned to tasks are discretized, the task-to-voltage assignment may no longer remain optimal.

Please note that adding voltage selection overhead is trivial and can be seamlessly integrated into our NLP model in a manner similar to the approach discussed in [44].

It is noteworthy that the presented optimization problem features a convex objective function as well as linear constraints. This places the problem within the realm of general convex nonlinear optimization problems. Thanks to the nature of these problems, they can typically be addressed within a polynomial time [44]. This is due to the availability of efficient algorithms with polynomial complexity for both Linear Programming (LP) and convex Nonlinear Programming (NLP) [45].

B. ONLINE DVS HEURISTIC

The NLP-based approach has a high time complexity. Therefore, it is not suitable for distributing slack online which arises from early job completion or the possibility of certain jobs not being executed in some scenarios. To address this limitation, we propose an efficient online Dynamic Voltage Scaling (DVS) heuristic that assigns an appropriate speed (frequency/voltage) to each job. The algorithm, outlined in ALGORITHM 4, operates in constant time complexity ($O(1)$).

ALGORITHM 3 operates on the super graph G_s, which captures both the precedence constraints and the constraints introduced by the offline schedule. The et_i is the execution time of i^{th} task in G_s determined by the NLP. The algorithm begins by topologically sorting the graph G_s (line 1). Then, it calculates the edge consistent release time (ECR_i) for each job $v_i \in G_s$ (lines 2-3). The edge consistent release time determines the earliest possible start time for a job. Subsequently, the algorithm computes the edge consistent deadline (ECD_i) (line 5). The ECD_i for job v_j signifies the latest time by which the job must be completed. Finally, the critical path of each job v_i is determined according to the following procedure:

The probabilistic critical path CP_i is used to determine the amount of slack to be allocated to the job v_i (lines 6–13). The probabilistic critical path is the path that has the highest probability of execution. Initially CP_i and $\mathcal{P}$ are both set to zero. The CP_i and $\mathcal{P}$ are updated as follows $\forall v_j \in ISuc_i$:

$$CP_i = \begin{cases} CP_j, & \text{if } \mathcal{P} < p(v_j) \text{ or} \\ & (\mathcal{P} = p(v_j) \wedge CP_i < CP_j) \\ & \text{and } ECR_i < ECD_i, \\ CP_i, & \text{otherwise.} \end{cases} \tag{26}$$

$$\mathcal{P} = \begin{cases} p(v_j), & \text{if } \mathcal{P} < p(v_j) \text{ or} \\ & (\mathcal{P} = p(v_j) \wedge CP_i < CP_j) \\ & \text{and } ECR_i < ECD_i, \\ \mathcal{P}, & \text{otherwise.} \end{cases} \tag{27}$$

Finally, update CP_i as:

$$CP_i = CP_i + et_i \tag{28}$$

Condition $ECR_j < ECD_i$ ensures that the jobs on the critical path can share the slack available for job v_i. Notice that CP_i and ECD_i are computed in reverse topological order.

Please note that although ALGORITHM 3 is delineated within this section, its execution occurs in the offline phase. This takes place once the super graph G_s is fully constructed and the resource constraints are integrated within G_s. ALGORITHM 3 is used to ascertain the necessary inputs (ECD, ECR, and CP) for ALGORITHM 4. It is straightforward to show that the worst-case time complexity of ALGORITHM 3 is $O(n + e)$, where n represents the count of vertices and e denotes the count of edges in G_s.

The online DVS heuristic takes several input parameters, including v_i, $CurrTime$, ECD_i, w_i, NCC_i, and CP_i. Here, $CurrTime$ represents the time at which the online heuristic is invoked, ECD_i denotes the edge consistent deadline, et_i refers to the execution time determined by NLP and

TABLE 3: CTGs Sets.

Set	Conditional Task Graphs
Set-1	CTG1, CTG2, CTG3, CTG4
Set-2	CTG5, CTG6, CTG7, CTG8
Set-3	CTG9, CTG10, CTG11, CTG12
Set-4	CTG13, CTG14, CTG15, CTG16, CTG17
Set-5	CTG18, CTG19, CTG20, CTG21, CTG22
Set-6	CTG23, CTG24, CTG25, CTG26, CTG27
Set-7	CTG28, CTG29, CTG30, CTG31

NCC_i is clock cycles of job v_i at maximum frequency, and CP_i represents the critical path of job v_i. The offline computation of ECD_i and CP_i is performed using ALGORITHM 3, which is described above.

Before executing each job, the online-DVS ALGORITHM 4 is invoked to determine the appropriate frequency for that particular job. Our online heuristic assumes that the jobs are executed in the order specified by the offline constructed schedule. In ALGORITHM 4, the following steps are performed:

1) Firstly, the algorithm calculates the available slack for each job v_i (line 1).
2) Based on the critical path CP_i, it determines the amount of slack to allocate to job v_i (line 2). Any remaining slack is utilized by the jobs on the critical path.
3) It computes the frequency for job v_i (line 3).
4) Finally, discretize the frequency assigned to job v_i.

VI. EXPERIMENTAL RESULTS

In our study, we conducted a comprehensive evaluation of both our offline scheduling approach, EESEDF, and the online DVS heuristic. To provide a thorough analysis, we constructed seven sets of CTGs, as illustrated in TABLE 3. Each set was carefully formed using the CTGs listed in TABLE 4. The evaluation was performed on a diverse set of benchmarks presented in TABLE 3. To assess the performance of our scheduling approaches, EESEDF, and EESEDF+Online-DVS, in a realistic scenario, we evaluated our schedulers using real-world benchmarks while using LESA [14], NCM [15], IOETCS-Heuristic [3], BESS [16] and CAP-Online [17] for comparison. Chen et al. [14] developed LESA scheduler, integrating task prioritization and weight-based energy distribution strategies. This method employs DVFS to assign discrete speed levels to tasks, aiming to approximate an optimal schedule by considering task dependencies and energy constraints. Maurya et al. [15] presented an enhanced version of the NCM sub-algorithm within the EASLA task scheduling framework. This improved algorithm, tailored for DVFS-enabled heterogeneous cluster systems, incorporates the PEFT algorithm to efficiently compute schedule length.

The offline scheduling approach presented in [3] is tailored for tasks with conditional precedence constraints on heterogeneous NoC-based MPSoCs featuring heterogeneous cores, aimed at enhancing energy efficiency by amalgamating task mapping, scheduling, and voltage scaling. Designed specifically for tasks with individual deadlines, this technique employs an NLP-based DVFS algorithm to assign continuous frequencies and voltages to tasks and communications, later converted into valid discrete levels through either ILP or heuristic methods. BESS [16] designed for optimizing task mapping, ordering, and dynamic voltage/frequency scaling (DVFS) on heterogeneous multi-core systems, specifically tailored for Conditional Task Graphs where tasks share a common deadline. It employs a mapping algorithm designed to evenly distribute latency and energy dissipation across cores, thereby maximizing available slack time without notably increasing energy consumption. The scheduling strategy utilizes workload statistics to minimize average power use while maintaining adherence to deadlines. While the BESS algorithm exhibits pseudo-linear complexity for smaller benchmarks with few OR-FORK nodes, its time complexity generally grows exponentially with the number of conditions in the CTGs. Malani et al. [17] presented an online scheduling algorithm termed CAP-Online, tailored for CTGs with a shared deadline, under the dynamic power model. This algorithm dynamically calculates the critical path when scheduling a task, determines the available slack time, and extends it to maximize utilization of the slack time. Just like BESS, CAP-Online time complexity generally grows exponentially with the number of conditions in the CTGs.

These existing techniques represent the current benchmarks in the field of energy-efficient scheduling. Our evaluation process aimed at thoroughly analyzing the strengths and weaknesses of EESEDF and its performance in comparison to established methods. By conducting evaluations on both synthetic and real benchmarks, we ensured a robust and comprehensive assessment of our proposed approach's capabilities. Overall, the evaluation results provided valuable insights into the effectiveness of EESEDF and its potential to outperform existing approaches in terms of energy efficiency and scheduling performance. These findings contribute significantly to the advancement of energy-efficient scheduling techniques and can guide further research in this important domain.

A. EXPERIMENTAL SETUP

In our experimental setup, we employed the 70 nm technology as outlined in TABLE 5 adopted from the work by Chen et al. [4] to conduct our evaluations on benchmarks listed in TABLE 4 and TABLE 6. The continuous voltage range was set to $0.65 \leq V_{dd} \leq 0.85$, and the discrete voltage levels used are $[0.65, 0.70, 0.75, 0.80, 0.85]$. The maximum frequency f_{max} was set to 3.1 GHz, corresponding to the maximum supply voltage $V_{dd} = 0.85$. The transition overhead in switching processor frequency is set to 100 cycles [46]. To implement our offline scheduling approach (EESEDF), the online DVS heuristic, and other approaches for comparison, we employed Matlab version R2020a. Additionally, we utilized the fmincon solver to solve the NLP problems arising in the NLP-based approach of EESEDF. The hardware platform

TABLE 4: Benchmarks Characteristics.

Benchmarks	$x/y/z$	T	vol	len	D	Benchmarks	$x/y/z$	T	vol	len	D	Benchmarks	$x/y/z$	T	vol	len	D
CTG1	15/1/3	185	151.27	71.7	175	CTG2	18/1/2	185	186	98.6	176	CTG13	34/4/10	274	279.6	142	260
CTG3	20/3/7	370	185	108	351	CTG4	20/3/6	370	197.6	94.6	352	CTG14	35/4/10	274	280	143	260
CTG5	18/2/5	170	139.8	64.3	161	CTG6	17/1/3	170	170.5	107.3	162	CTG15	15/1/3	548	151.2	71.7	520
CTG7	17/1/3	340	171.8	63.1	323	CTG8	20/1/2	340	195.3	81.2	325	CTG16	32/2/4	548	257	148.8	522
CTG9	15/1/3	184	155.7	71.7	175	CTG10	18/1/2	184	185.6	98.6	176	CTG17	16/1/3	548	151.2	71.7	530
CTG11	20/3/7	368	185	108	349	CTG12	20/3/6	368	372	135.8	350	CTG18	34/2/5	345	346	174	339
CTG19	35/6/14	690	252.1	110.6	655	CTG20	34/3/7	690	332.3	216	657	CTG21	34/2/5	345	345.31	174.64	339
CTG22	20/1/2	690	195.3	81.2	680	CTG23	30/2/6	305	305.58	188.7	298	CTG24	34/2/6	305	306	189	299
CTG25	33/4/9	610	272.5	148.8	598	CTG26	32/4/10	610	227.4	154.1	597	CTG27	15/1/3	610	155.7	71.7	599
CTG28	34/3/6	324	325	137.9	322	CTG29	32/6/14	648	188.4	87.8	640	CTG30	32/2/4	648	246.1	104.7	643
CTG31	34/3/6	324	324.5	137.9	321												

TABLE 5: 70 nm processor technology parameters

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	1.5
I_j	4.80×10^{-10}	L_g	4.00×10^{6}
V_{bs}	0	V_{th}	0.244

TABLE 6: Large Benchmarks Characteristics

Becnhmarks	x	y	z	T
CTG-32	100	4	8	680
CTG-33	110	5	10	650
CTG-34	120	6	13	690
CTG-35	130	7	14	685
CTG-36	150	8	16	705
CTG-37	200	20	50	745
CTG-38	250	30	60	770
CTG-39	300	35	72	1000
CTG-40	350	40	83	1010
CTG-41	400	50	130	1030

used for our experiments featured an Intel(R) Core(TM) with CPU, i5-4570 of clock frequency, 3.20 GHz, 8.00 GB of memory, and a 3 MB cache. Notably, the benchmarks listed in TABLE 4 are the same ones employed in the work by Lombardi et al. [13]. We utilized these benchmarks to ensure consistency with the existing literature and made use of the available input data, including the probabilities of each condition, worst-case execution times, and periods for the respective tasks.

We have also created 10 large benchmarks, detailed in TABLE 6, specifically to demonstrate the scalability of our approach and the limitations of existing single CTG schedulers found in the literature. For a fair comparison, we have set the task deadlines in the first five benchmarks (CTGs 32 - 36) within a range of $0.65T$ to T. For the remaining five benchmarks, all tasks share a common deadline equal to their period because the approaches we are comparing against are designed for the task model with common deadlines.

In our experimental analysis, we utilized eight real benchmarks sourced from the Embedded System Synthesis Benchmarks Suite (E3S). This suite is widely recognized in task mapping and scheduling research [7]. Robot benchmark represents tasks used by industrial robots to automate or perform processes/controls. ATR (Automatic Target Recognition) serves as a real-time streaming application utilized for pattern recognition. MP3-decoder benchmark involves Huffman decoding and Inverse Discrete Transform (IDCT). Office benchmark comprises tasks for text processing, image rotation, and gray-scale to binary conversion. Consumer-1 and Consumer-2 benchmarks encompass tasks related to JPEG decompression or compression, along with conversions from RGB to YIQ and RGB to CMYK. This rigorous and well-defined experimental setup allowed us to conduct comprehensive evaluations and draw meaningful conclusions about the performance and efficiency of our proposed EESEDF approach and the comparison against other state-of-the-art techniques.

B. RESULTS AND DISCUSSION

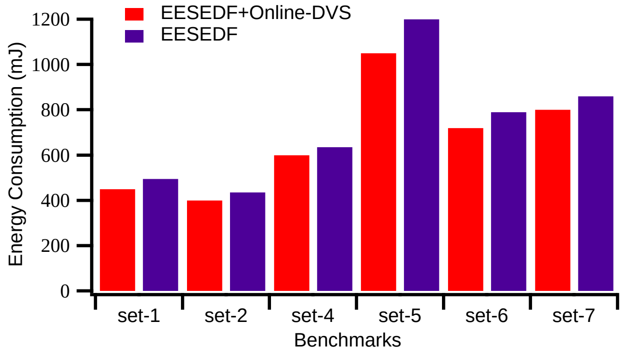


FIGURE 5: Energy Consumption on 4 Processors

In our comprehensive research, we initiated two distinct sets of experiments to elucidate the advantages of integrating a low-time complexity online DVS algorithm with an offline NLP-based DVS algorithm, specifically EESEDF, across different computational models. The first set of experiments focused on CTGs, while the second set targeted Task Graphs (TGs), recognizing that TGs represent a special case of CTGs. This bifurcation was pivotal in comprehensively evaluating the efficacy of our approaches in varied contexts.

1) Experiments on CTGs

First we compare the performance of our standalone EESEDF method against our combined EESEDF + Online-DVS strategy. For detailed benchmark information, please

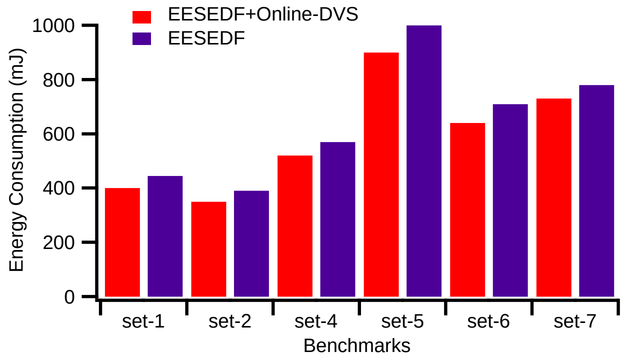


FIGURE 6: Energy Consumption on 8 Processors

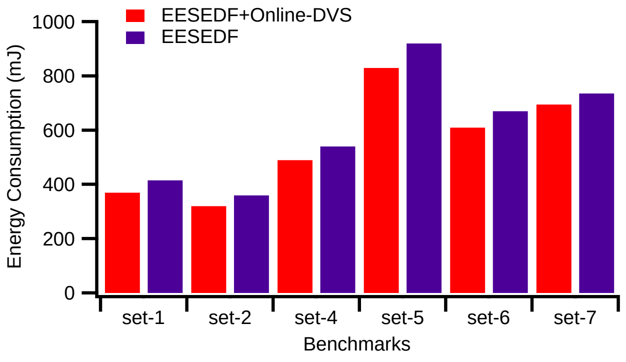


FIGURE 7: Energy Consumption on 12 Processors

refer to TABLE 4. The simulations spanned seven sets of CTGs, as presented in TABLE 3, enabling a thorough analysis of the proposed methodologies in addressing diverse computational challenges.

FIGURES 5, 6, and 8 demonstrate a comparison between these two approaches when executed on MPSoCs with 4, 8, and 12 processors respectively. We evaluated the average energy consumption of our NLP approach and our online DVS heuristic on the seven sets of CTGs specified in TABLE 3. Based on our experiments, we found that the combined EESEDF+Online-DVS approach significantly outperformed the standalone EESEDF approach.

The EESEDF+Online-DVS approach demonstrated a minimum improvement of 12% on one particular set, a maximum improvement of 17% on another set, and an average improvement of 15% compared to the EESEDF approach alone. This outcome was expected, and we attribute it to two primary reasons:

1) The EESEDF approach cannot leverage slack, freed up due to the early completion of jobs. The offline schedule is constructed based on the assumption of worst-case execution time for each job. In reality, the actual execution time is often significantly lower than the worst-case scenario. Our low-time complexity online algorithm is designed to effectively distribute this slack, resulting in improved performance.
2) Not all jobs are executed in every scenario. The offline schedule must allocate time slots to all tasks since the determination of which tasks will execute can only be made at runtime. The online DVS algorithm can efficiently distribute the freed-up slack caused by jobs not being executed in certain scenarios.

Overall, these results confirm our expectations and validate the effectiveness of the combined EESEDF+Online-DVS approach in optimizing energy consumption and performance in the context of CTGs.

We next conduct experiments on 10 benchmarks chosen randomly from TABLE 6. We compare our approach against IOETCS-Heuristic. Although IOETCS-Heuristic is specifically designed for heterogeneous systems, it is applicable to homogeneous systems as well, because a homogeneous system is a special case of a heterogeneous system. Our comparisons and results are only applicable to this special case. FIGURE 8 shows the comparison of our approach against IOETCS-Heuristic in terms of expected energy consumption. IOETCS-Heuristic, an offline scheduler designed for CTGs to be scheduled on heterogeneous systems, performs better than our offline scheduler, EESEDF, achieving an average improvement of 7% over our method. This is due to their heuristic for discretizing task voltages, which is efficient in distributing slack and reducing energy consumption. In the offline phase, we use a simpler technique to discretize task voltages, employing a rounding technique that rounds the invalid task voltages to the nearest higher voltage levels. However, the primary purpose of conducting these experiments is to demonstrate the necessity and effectiveness of our two-phase approach, at least in the context of CTG. Our EESEDF approach, combined with the online-DVFS algorithm, achieves an average improvement of 13% over IOETCS-Heuristic. This is because IOETCS-Heuristic is an offline scheduler, but in the context of CTGs, online schedulers are necessary because of the following two reasons:

1) Offline schedulers assume worst-case execution times for tasks. However, the actual execution time is usually lower than these estimates. Due to this difference, the slack needs to be efficiently redistributed by the online scheduler.
2) In the case of CTGs, not all tasks execute in all scenarios. The offline schedulers that generate a single global schedule for all scenarios typically do not account for this. Hence, an efficient online algorithm is required to redistribute the slack available due to the non-execution of tasks in some scenarios.

To show the effectiveness of our approach in efficiently redistributing slack released at runtime due to reason two, we ensure that the actual execution time of tasks is the same as the worst-case execution time of tasks.

Despite numerous methods being developed to schedule Conditional Task Graphs (CTG), these existing approaches fall short of efficiently managing PCTG. We proceed to conduct experiments to showcase the limitations of current energy-aware CTG schedulers, elucidating their deficiencies and explaining their unsuitability for application to PCTGs.

To underscore the advantages of our approach in energy optimization and scalability, we contrast it with the CAP-Online and BESS approaches, focusing on homogeneous systems. All of these approaches are applicable to homogeneous systems. The comparison makes use of benchmarks outlined in TABLE 6, which are specifically chosen to replicate PCTGs and the supergraph. Consequently, these benchmarks feature a substantial number of nodes and OR-FORK nodes. Both CAP-Online and BESS exhibit exponential time complexity in the number of scenarios within Conditional Task Graphs (CTGs), rendering them less effective except in situations with a manageable number of scenarios. However, PCTGs, which require scheduling over a hyperperiod, introduce a substantial number of OR-Fork nodes and, consequently, a significant increase in scenario count. The benchmarks in TABLE 6 aim to mirror these conditions.

In the first five benchmarks presented in TABLE 6, our method demonstrates markedly lower energy consumption than both the BESS and CAP-Online approaches. Against BESS, our approach shows a 24% to 29% improvement, averaging around 25%. When compared to CAP-Online, the improvement ranges from 34% to 37%, with an average enhancement of 35% as shown in FIGURE 9. These gains are attributed to several key factors:

1) EESEDF, our proposed solution, outperforms both CAP-Online and BESS in generating energy-efficient task ordering and balanced mapping. It strategically arranges tasks to prevent longer-deadline tasks from being impeded by those with shorter deadlines, facilitating optimal task execution. This sequencing, combined with the application of NLP and online Dynamic Voltage and Frequency Scaling (DVFS), allows for the assignment of task speeds that further reduce energy consumption compared to the alternatives.
2) CAP-Online incurs a considerably higher online running overhead than our method, leading to increased energy consumption. This additional consumption underscores the efficiency of our approach in optimizing energy.

The subsequent five benchmarks in TABLE 6, from CTG-37 to CTG-41, feature a much larger number of OR-FORK nodes, mirroring the complexity observed in a super-graph for PCTGs. For these benchmarks, both BESS and CAP-Online struggle to converge within a practical timeframe. This challenge stems from the overwhelming number of scenarios present in these benchmarks. Despite pruning, the sheer volume of unique scenarios remains so extensive that both approaches fail to achieve convergence.

Regarding running time, our approach significantly runs faster than both BESS and CAP-Online. For the benchmarks listed in TABLE 6, our approach consistently achieves convergence in under 30 minutes. In contrast, both BESS and CAP-Online were allowed up to seven hours of run time for benchmarks CTGs 37-41 but both failed to converge

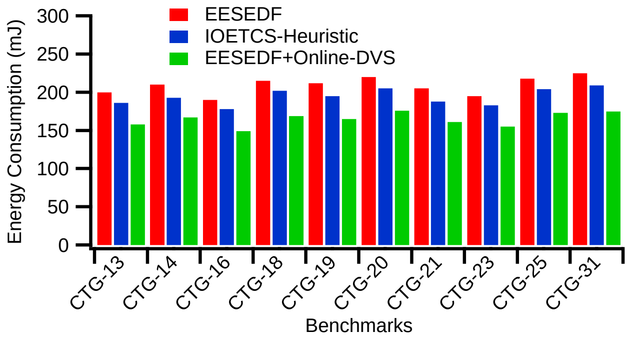


FIGURE 8: Energy Consumption on 24 Processors

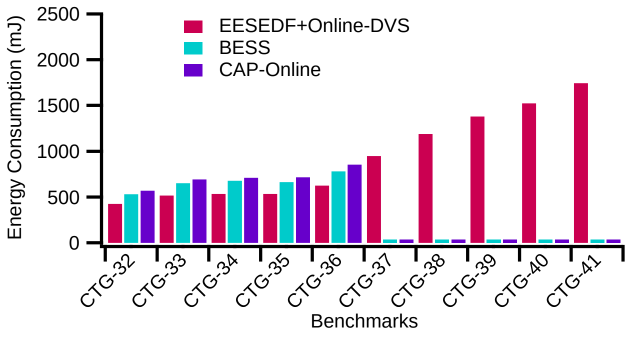


FIGURE 9: Energy Consumption on 24 Processors

as demonstrated in FIGURE 9. This demonstrates that our approach excels not only in energy efficiency but also in handling larger and more complex problem instances. Specifically designed for PCTGs, our approach is optimized for energy efficiency and scalability. These findings highlight the necessity of our method, given that existing solutions for CTGs fall short when it comes to energy-aware scheduling of PCTGs.

2) Experiments on TGs

As we have previously emphasized, Task Graphs (TGs) are a special case of Conditional Task Graphs (CTGs), underscoring the importance of evaluating our approaches across both domains to demonstrate their effectiveness comprehensively. We have conducted a second set of experiments using 4, 8, and 12 processors on the MPSoC computing platform as demonstrated in FIGURE 10, FIGURE 11, and FIGURE 12 respectively. Real benchmarks with different scenarios are considered to compare EESEDF with LESA [14] and NCM [15]. Our energy-efficient approach, EESEDF, incorporating NLP, achieves average energy savings of 25% and 20% over LESA [14] and NCM [15] respectively.

1) Unlike LESA [14] and NCM [15] our novel two-phase offline scheduling approach, EESEDF constructs a single global schedule for all the scenarios while convex NLP assigns an optimal speed to each task of

conditional task graphs to achieve maximum energy efficiency.
2) Our low time complexity online-DVS algorithm assigns each task a speed online for reducing the overall energy consumption of each task and achieves higher energy savings for PCTGs on multi-core computing architectures.

It's imperative to highlight that our research findings and claims are specifically tailored to homogeneous systems. This distinction is crucial, especially when considering our comparison with the LESA algorithm, which is designed for heterogeneous systems. Given that a homogeneous system can be viewed as a special case of a heterogeneous system where all processors are of the same type, our comparison is both relevant and insightful. Additionally, our analysis extends to a comparison with the NCM approach, focusing on a more specialized scenario within homogeneous systems: environments consisting of single processors per cluster. This nuanced comparison framework allows us to demonstrate the effectiveness and applicability of our approaches in specific system configurations.

Our study is dedicated to enhancing energy efficiency, an endeavor we approach by segmenting our approach into offline and online phases. This segmentation is strategic, addressing the critical issue of runtime overhead from online algorithms, which can adversely affect energy consumption. To counteract this, we introduce an online algorithm designed for low time complexity, aiming to minimize energy expenditure associated with processing tasks in real-time.

For offline algorithms generally, higher time complexity is acceptable as long as they operate within polynomial time, ensuring they can converge in a timely and efficient manner. Our approach is designed with this balance in mind. Our online algorithm stands out for its O(1) time complexity, offering rapid execution that complements the inherently longer, yet reasonable, convergence times of our offline algorithms. This design philosophy ensures that, across these TGs, our algorithms can achieve convergence in under 15 minutes.

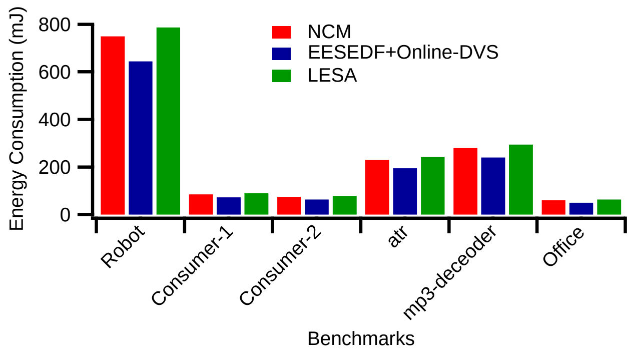


FIGURE 10: Energy Consumption Comparison on 4 Processors

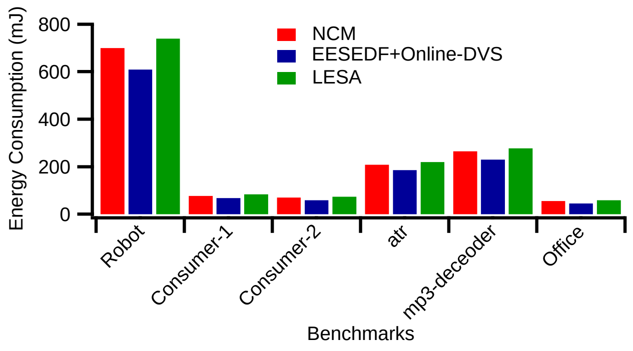


FIGURE 11: Energy Consumption Comparison on 8 Processors

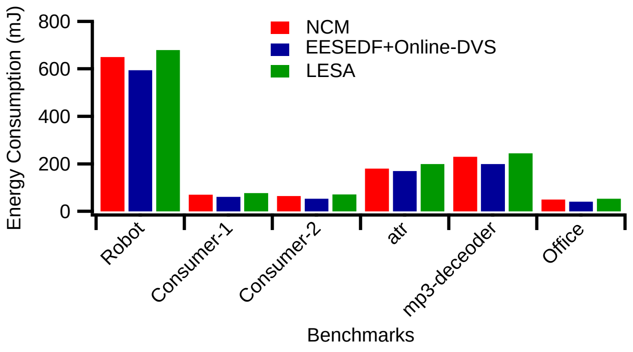


FIGURE 12: Energy Consumption Comparison on 12 Processors

However, it's important to note that when benchmarked against NCM and LESA, our approach exhibits slower execution times. This delay is primarily due to the inherent processing requirements of NLP-based approaches, which take longer to converge. Despite this, we emphasize the efficiency and practicality of our online algorithm's low time complexity, alongside the reasonable convergence times of our offline algorithms, underscoring their collective value in our overarching goal of energy optimization.

VII. CONCLUSION

We have successfully developed a pioneering two-phase offline approach, called Energy-efficient Successor Tree Consistent Earliest Deadline First (EESEDF), to tackle the challenging problem of scheduling a set of Periodic Conditional Task Graphs (PCTGs) on a group of identical processors with shared memory. Our innovative approach, EESEDF, consists of two key components: a task assignment and scheduling algorithm that efficiently assigns tasks to processors and constructs a global schedule for all scenarios, and an NLP (Non-Linear Programming) approach that optimally determines the speed for each job based on the global schedule. In our study, we have also introduced an online DVS (Dynamic Voltage Scaling) heuristic that dynamically computes the appropriate speed for each job at runtime, with the primary objective of minimizing the total energy consumption of all tasks. To evaluate the effectiveness of our contributions, we conducted comprehensive comparisons between our NLP approach and the online DVS approach. The results of our experiments have been highly encouraging. The NLP approach has demonstrated substantial improvements over the online DVS heuristic, with average improvement, maximum improvement, and minimum improvement values of 15%, 12%, and 17%, respectively. These improvements signify the potency of our NLP-based approach in achieving better energy efficiency. Furthermore, the offline scheduling aspect of our EESEDF algorithm has also delivered remarkable results. Compared to existing techniques such as LESA [14] and NCM [15], our EESEDF algorithm has outperformed with significant energy efficiency gains of 25% and 20%, respectively. In comparison to a few other state-of-the-art techniques, our suggested scheduler, EESEDF+Online-DVS, delivers notable improvements in energy efficiency. It surpasses IOETCS-Heuristic [3] by roughly 13% while outperforming BESS [16] and CAP-ONLINE [17] by impressive margins of 25% and 35%, respectively. This outcome signifies the superiority of our novel scheduling approach in minimizing total energy consumption for PCTs. It is crucial to highlight that this work represents the first-ever exploration into the domain of minimizing the total energy consumption of periodic conditional task graphs. Our approach, EESEDF, sets a new benchmark in addressing this complex problem and provides valuable insights for future research in energy-efficient scheduling techniques for parallel computing environments.

In the future, there is potential for scheduling Periodic Conditional Task Graphs (PCTGs) on Voltage Island-based (VFI) based Network-on-Chip (NoC) MPSoCs to achieve energy consumption reduction. By utilizing voltage islands, different tasks can be assigned to specific islands with varying voltage levels, allowing for dynamic power management and optimization. Additionally, the inclusion of re-timing techniques can further enhance energy consumption performance and reduce latency. Re-timing involves the adjustment of task schedules to optimize the overall timing behavior and reduce the energy required for task execution. By carefully managing voltage levels and re-timing tasks, future systems can achieve improved energy efficiency and performance in the scheduling of periodic conditional task graphs on VFI-based NoC-MPSoCs.

•••

REFERENCES

[1] P. Bauer, P. D. Dueben, T. Hoefler, T. Quintino, T. C. Schulthess, and N. P. Wedi, "The digital revolution of earth-system science," Nature Computational Science, vol. 1, no. 2, pp. 104–113, 2021.

[2] H. Ali, U. U. Tariq, M. Hussain, L. Lu, J. Panneerselvam, and X. Zhai, "Arsh-fati: A novel metaheuristic for cluster head selection in wireless sensor networks," IEEE Systems Journal, vol. 15, no. 2, pp. 2386–2397, 2020.

[3] U. U. Tariq, H. Wu, and S. A. Ishak, "Energy-efficient scheduling of tasks with conditional precedence constraints on mpsocs," in Towards Integrated Web, Mobile, and IoT Technology: Selected and Revised Papers from the Web Technologies Track at SAC 2017 and SAC 2018, and the Software Development for Mobile Devices, Wearables, and the IoT Minitrack at HICSS 2018. Springer, 2019, pp. 115–145.

[4] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination," ACM Transactions on Embedded Computing Systems (TECS), vol. 13, no. 3s, p. 111, 2014.

[5] H. Ali, U. U. Tariq, J. Hardy, X. Zhai, L. Lu, Y. Zheng, F. Bensaali, A. Amira, K. Fatema, and N. Antonopoulos, "A survey on system level energy optimisation for mpsocs in iot and consumer electronics," Computer Science Review, vol. 41, p. 100416, 2021.

[6] U. U. Tariq, H. Ali, M. Hussain, and L. Liu, "Shuffled arsh-fati: A novel meta-heuristic for lifetime maximization of range-adjustable wireless sensor networks," IEEE Transactions on Green Communications and Networking, 2023.

[7] H. Ali, U. U. Tariq, Y. Zheng, X. Zhai, and L. Liu, "Contention & energy-aware real-time task mapping on noc based heterogeneous mpsocs," IEEE Access, vol. 6, pp. 75 110–75 123, 2018.

[8] U. U. Tariq, H. Ali, L. Liu, J. Hardy, M. Kazim, and W. Ahmed, "Energy-aware scheduling of streaming applications on edge-devices in iot-based healthcare," IEEE Transactions on Green Communications and Networking, vol. 5, no. 2, pp. 803–815, 2021.

[9] H. Ali, U. U. Tariq, L. Liu, J. Panneerselvam, and X. Zhai, "Energy optimization of streaming applications in iot on noc based heterogeneous mpsocs using re-timing and dvfs," in 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, 2019, pp. 1297–1304.

[10] J. Ax, G. Sievers, J. Daberkow, M. Flasskamp, M. Vohrmann, T. Jungeblut, W. Kelly, M. Porrmann, and U. Rückert, "Coreva-mpsoc: A many-core architecture with tightly coupled shared and local data memories," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 5, pp. 1030–1043, 2017.

[11] J. Lant, C. Concatto, A. Attwood, J. A. Pascual, M. Ashworth, J. Navaridas, M. Lujan, and J. Goodacre, "Enabling shared memory communication in networks of mpsocs," Concurrency and Computation: Practice and Experience, vol. 31, no. 21, p. e4774, 2019.

[12] T. S. Darwish, K. A. Bakar, O. Kaiwartya, and J. Lloret, "Trading: Traffic aware data offloading for big data enabled intelligent transportation system," IEEE Transactions on Vehicular Technology, vol. 69, no. 7, pp. 6869–6879, 2020.

[13] M. Lombardi, M. Milano, M. Ruggiero, and L. Benini, "Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip," Journal of scheduling, vol. 13, no. 4, pp. 315–345, 2010.

[14] J. Chen, Y. He, Y. Zhang, P. Han, and C. Du, "Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems," Journal of Systems Architecture, vol. 129, p. 102598, 2022.

[15] A. K. Maurya, K. Modi, V. Kumar, N. S. Naik, and A. K. Tripathi, "Energy-aware scheduling using slack reclamation for cluster systems," Cluster Computing, vol. 23, pp. 911–923, 2020.

[16] Y. Ge, Y. Zhang, P. Malani, Q. Wu, and Q. Qiu, "Low power task scheduling and mapping for applications with conditional branches on heterogeneous multi-processor system," Journal of Low Power Electronics, vol. 8, no. 5, pp. 535–551, 2012.

[17] P. Malani, P. Mukre, Q. Qiu, and Q. Wu, "Adaptive scheduling and voltage scaling for multiprocessor real-time applications with non-deterministic workload," in Proceedings of the conference on Design, automation and test in Europe. ACM, 2008, pp. 652–657.

[18] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics," in Proceedings 13th Euromicro Conference on Real-Time Systems. IEEE, 2001, pp. 225–232.

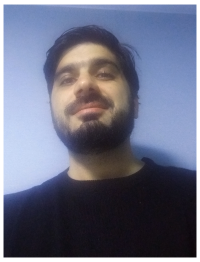
[19] W. Zhang, E. Bai, H. He, and A. M. Cheng, "Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms," Sensors, vol. 15, no. 6, pp. 13 778–13 804, 2015.

[20] N. Kumar and D. P. Vidyarthi, "A ga based energy aware scheduler for dvfs enabled multicore systems," Computing, vol. 99, pp. 955–977, 2017.

[21] X. Wang, Z. Li, and W. M. Wonham, "Optimal priority-free conditionally-preemptive real-time scheduling of periodic tasks based on des supervisory control," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 7, pp. 1082–1098, 2016.

[22] L. Liu and D. Qi, "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, 2018, pp. 142–146.
[23] J. Chen, P. Han, Y. Zhang, T. You, and P. Zheng, "Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems," Journal of Systems Architecture, vol. 142, p. 102938, 2023.
[24] H. Ali, X. Zhai, U. U. Tariq, and L. Liu, "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, 2018, pp. 1099–1104.
[25] S. Abd Ishak, H. Wu, and U. U. Tariq, "Energy-aware task scheduling on heterogeneous noc-based mpsocs," in 2017 IEEE International Conference on Computer Design (ICCD). IEEE, 2017, pp. 165–168.
[26] S. Ding, J. Wu, G. Xie, and G. Zeng, "A hybrid heuristic-genetic algorithm with adaptive parameters for static task scheduling in heterogeneous computing system," in 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, 2017, pp. 761–766.
[27] U. U. Tariq, H. Ali, L. Liu, J. Panneerselvam, and X. Zhai, "Energy-efficient static task scheduling on vfi-based noc-hmpsocs for intelligent edge devices in cyber-physical systems," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 6, pp. 1–22, 2019.
[28] G. Xie, G. Zeng, L. Liu, R. Li, and K. Li, "Mixed real-time scheduling of multiple dags-based applications on heterogeneous multi-core processors," Microprocessors and Microsystems, vol. 47, pp. 93–103, 2016.
[29] S. K. Roy, R. Devaraj, and A. Sarkar, "Safla: Scheduling multiple real-time periodic task graphs on heterogeneous systems," IEEE Transactions on Computers, vol. 72, no. 4, pp. 1067–1080, 2023.
[30] ——, "Contention cognizant scheduling of task graphs on shared bus-based heterogeneous platforms," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 2, pp. 281–293, 2021.
[31] R. Devaraj and A. Sarkar, "Resource-optimal fault-tolerant scheduler design for task graphs using supervisory control," IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7325–7337, 2021.
[32] Y. Sharma, S. Chakraborty, and S. Moulik, "Eta-hp: an energy and temperature-aware real-time scheduler for heterogeneous platforms," The Journal of Supercomputing, vol. 78, no. 8, pp. 1–25, 2022.
[33] S. Moulik, Z. Das, and G. Saikia, "Ceat: a cluster based energy aware scheduler for real-time heterogeneous systems," in 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2020, pp. 1815–1821.
[34] Y. Sharma and S. Moulik, "Rt-seat: A hybrid approach based real-time scheduler for energy and temperature efficient heterogeneous multicore platforms," Results in Engineering, vol. 16, p. 100708, 2022.
[35] ——, "Fats-2tc: A fault tolerant real-time scheduler for energy and temperature aware heterogeneous platforms with two types of cores," Microprocessors and Microsystems, vol. 96, p. 104744, 2023.
[36] ——, "Cetas: a cluster based energy and temperature efficient real-time scheduler for heterogeneous platforms," in proceedings of the 37th ACM/SIGAPP symposium on applied computing, 2022, pp. 501–509.
[37] D. Shin and J. Kim, "Power-aware scheduling of conditional task graphs in real-time multiprocessor systems," in Proceedings of the 2003 international symposium on Low power electronics and design. ACM, 2003, pp. 408–413.
[38] T. Walsh, "Stochastic constraint programming," in ECAI, vol. 2, 2002, pp. 111–115.
[39] D. Wu, B. M. Al-Hashimi, and P. Eles, "Scheduling and mapping of conditional task graph for the synthesis of low power embedded systems," IEE Proceedings-Computers and Digital Techniques, vol. 150, no. 5, pp. 262–273, 2003.
[40] P. Eles, K. Kuchcinski, Z. Peng, A. Doboli, and P. Pop, "Scheduling of conditional process graphs for the synthesis of embedded systems," in Proceedings of the conference on Design, automation and test in Europe. IEEE Computer Society, 1998, pp. 132–139.
[41] U. U. Tariq and H. Wu, "Energy-aware scheduling of conditional task graphs with deadlines on mpsocs," in Computer Design (ICCD), 2016 IEEE 34th International Conference on. IEEE, 2016, pp. 265–272.
[42] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design. ACM, 2002, pp. 721–725.
[43] U. U. Tariq, H. Wu, and S. Abd Ishak, "Energy-aware scheduling of conditional task graphs on noc-based mpsocs," in Proceedings of the 51st Hawaii International Conference on System Sciences, 2018.
[44] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi, "Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems," IEE Proceedings-Computers and Digital Techniques, vol. 152, no. 1, pp. 28–38, 2005.
[45] J. NESTEROV, "Interior-point polynomial algorithms," Convex Programming, 1994.
[46] A. Andrei, P. Eles, O. Jovanovic, M. Schmitz, J. Ogniewski, and Z. Peng, "Quasi-static voltage scaling for energy minimization with time constraints," IEEE transactions on very large scale integration (VLSI) systems, vol. 19, no. 1, pp. 10–23, 2009.

UMAIR ULLAH TARIQ received BSc. (Hons.) degree in Computer Engineering from COMSATS Institute of Information Technology Pakistan in 2008, an MSc. in Computer Engineering from the University of Engineering and Technology Taxila in 2013, and a Ph.D. in Computer Science and Engineering in 2018. Currently, he is working as a Lecturer-ICT at the College of Information and Communication Technology (School of Engineering and Technology), CQUniversity Sydney campus. Before this, he worked as a Digital Transformation Engineer at BlueScope Steel Port Kembla and as a Sessional Academic at CQUniversity Sydney campus. His area of research includes Evolutionary Computation, Machine Learning, Energy-Aware Scheduling, the Internet of Things, and Wireless Sensor Networks. He has published 2 book chapters and more than 20 refereed papers that include high-impact journal and conference papers.



HAIDER ALI received master degree in electronic systems design engineering from Manchester Metropolitan University (MMU), UK, and PhD degree from the University of Derby (UoD), UK. He is currently a Lecturer at the School of Computing at UoD. He served COMSATS University Islamabad, Abbottabad Campus, Pakistan as a Lecturer from 2011 to 2016. He is currently working on energy-efficient algorithms for task mappings on modern embedded systems for real-time applications. His research areas of interest are biomedical systems design, Internet-of-Things (IoT), algorithms design, and cybersecurity. He has received 2 best paper awards from international conferences. He has served as a member of the technical program committee for different workshops and serves many reputable journals and transactions as a reviewer.

MUHAMMAD SHAHROZ NADEEM received his Ph.D. in Computer Science from University of Derby in 2022. He is currently a Lecturer at the Department of Technology, business, and Arts at the University of Suffolk. Dr. Nadeem's research focuses on computer vision, Image restoration, deep learning, privacy preservation, and verification. He has worked as a data scientist on the ESF-funded project of Reskill and Recover.

SYED JAN is an academician and active researcher in the field of computer networks, currently working as a faculty member at the University of Suffolk, United Kingdom. He has completed his undergraduate and postgraduate studies at the University of Bedfordshire, UK. Following his studies, he completed his PhD studies titled "Energy-efficient Data Aggregation for Cluster-based Wireless Sensor Networks." During his doctoral research, he has investigated and designed innovative techniques for energy-efficient and secure data collection in wireless sensor networks (WSNs), IoT, and vehicular networks that conserve energy in these networks and enhance their performance and overall lifetime. With a remarkable publication record, he has extensively contributed to the field by actively engaging in three internationally funded research projects, organizing three IEEE international conferences, and contributing as a reviewer for esteemed journals. He has 17+ publications in top-ranked peer-reviewed journals, that received 773+ citations (Google Scholar). Some of his noteworthy projects include the International Scientific and Technological Cooperation Project of Dongguan, China, the Taif University Researchers Supporting Project, Saudi Arabia, and the Science and Technology Planning Project of Guangdong Province, China. Besides, he recently served as a web chair for EAI BigIoT-EDU2023, acknowledged for global recognition in 170 countries by the European Alliance for Innovation (EAI). His collaborative research experiences demonstrate a robust capacity for national and international partnerships in academia and research.

SRIMANNARAYANA GRANDHI is the Head of Course - Undergraduate ICT, in the College of Information and Communications Technology. He is also a member of the ICT Course Committee. Dr Grandhi received his Master of Business Administration and Master of ERP degrees from Victoria University, Master of Information Systems degree from Central Queensland University, and PhD from RMIT University in Australia. Dr. Grandhi is an active researcher with research interests in educational technology, technology adoption, sustainability, and multicriteria decision-making. He has published several papers in reputed international conferences, and refereed journals, and also participated in collaborative book chapters. He is currently teaching Blockchain technologies and Enterprise systems at the Undergraduate and Postgraduate levels and supervising research students. Over the years, Dr Grandhi has published research papers at Excellence in Research for Australia (ERA) listed conferences and journals. He is also a guest editor for the special issue (Environmental Research and Public Health) of the International Journal of Environmental Research and Public Health.

FARIZA SABRINA received her Ph.D. in Computer Science and Engineering from the University of New South Wales (UNSW), Master of Engineering (Research) from The University of Sydney (USYD) and Bachelor of Science in Engineering (Hons.) degree in Electrical and Electronics Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh. After completing her Ph.D., she worked as a Postdoctoral Research Fellow and then as a Research Scientist in the Networking Technologies Lab, Commonwealth Scientific and Industrial Research Organisation (CSIRO) ICT Centre, Sydney, and worked on a wide variety of RD and consulting projects. Currently, she is a Senior Lecturer-ICT and the Discipline Lead for Networking and Information Security in the School of Engineering and Technology at CQUniversity Sydney. Before joining CQUniversity, she worked as a Senior Lecturer and School of IT and Engineering Academic Coordinator for the Sydney Campus at the Melbourne Institute of Technology. My past industry experience also includes working as an IT Consultant on large projects. Her current research interests include Networking and information security, Artificial Intelligence, Machine Learning, Blockchain, the Internet of Things, and Cyber Security.

ZHENGLIN WANG is an accomplished academic and industry professional who has made significant contributions to the fields of computer science and engineering. He obtained his Master of Computer Science by Research and PhD degrees from the University of South Australia in 2012 and 2016, respectively. During his career, Zhenglin has worked as a software engineer at TCL, UTStarcom, and Hitachi Construction Machinery Australia for several years, where he gained extensive experience in developing software solutions for various applications. Following this, he served as a Postdoctoral Research Fellow in the Institute for Future Farming Systems at CQU for over five years. During this time, he conducted pioneering research in the areas of agriculture automation, and his team developed the world's first mango auto-harvester. In recognition of his research excellence, Zhenglin was awarded the "Advanced Queensland Industry Research Fellowship" in 2019. Zhenglin currently holds the position of lecturer-ICT with the School of Engineering and Technology at CQUniversity Sydney, where he is actively engaged in teaching and research. Zhenglin is a member of the Australian Computer Society (ACS). His research interests include computer vision, machine learning, unmanned ground vehicle, and precision agriculture.

LU LIU is the Head of School of Informatics at the University of Leicester. Prior to this, he was the Head of School of Electronics, Computing and Mathematics and Professor of Distributed Computing at the University of Derby. Professor Liu received his PhD degree from Surrey Space Centre at the University of Surrey. He had worked as a Research Fellow at the WRG e-Science Centre at the University of Leeds. Professor Liu's research interests are in the areas of data analytics, AI, Cloud computing, service computing and the Internet of Things and he has over 200 scientific publications in reputable journals, academic books and international conferences. Professor Liu has secured and participated in many research projects which are supported by research councils, BIS, Innovate UK, British Council and leading industries. He received the Vice-Chancellor's Awards for Excellence in Doctoral Supervision in 2018, BCL Faculty Research Award in 2012 and was recognised as a Promising Researcher by the University of Derby in 2011. He has been the recipient of 7 Best Paper Awards from international conferences and was invited to deliver 7 keynote speeches at international conferences/workshops. Professor Liu is a Fellow of BCS (British Computer Society) and serves as an Editorial Board member of 6 international journals and the Guest Editor for 19 journal special issues. He has chaired over 30 international conference and workshops, and presently or formerly serves as the program committee member for over 60 international conferences and workshops.