Energy-efficient scheduling of streaming applications in VFI-NoC-HMPSoC based edge devices

Umair Ullah Tariq¹ · Haider Ali² · Lu Liu³ · John Panneerselvam³ · James Hardy²

Received: 6 March 2020 / Accepted: 23 November 2020 / Published online: 14 December 2020
© The Author(s) 2020

Abstract

Energy-aware high-performance computing is becoming a challenging facet for streaming applications at edge devices in Internet-of-Things (IoT) due to the high computational complexity involved. Therefore, the number of processors has increased significantly on the multiprocessor system subsequently, Voltage Frequency Island (VFI) recently adopted for an effective energy management mechanism in the large scale multiprocessor chip designs. In this paper, energy-aware scheduling of real-time streaming applications on edge-devices is investigated. First, an innovative re-timing based technique is developed to transform the dependent workload into an independent task model to avail resources and the wasted slack in the processors with a possible minimal prologue. Moreover, unlike the existing population-based optimization algorithms, a novel population-based algorithm, ARSH-FATI is proposed that can dynamically switch between explorative and exploitative search modes at run-time for performance trade-off. Finally, a communication contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling algorithm is presented. Our static scheduler ARHS-FATI collectively performs task mapping and ordering. Consequently, its performance is superior to the existing state-of-the-art approach proposed for homogeneous VFI based MPSoCs.

Keywords ARSH-FATI · Computing · DAG, Energy-aware · Edge-devices · IoT · MPSoCs · Mapping · Scheduling · Re-timing · Streaming-applications

1 Introduction

Internet-of-Things (IoT) paradigm is a 21st century technological revolution to interconnect plethora of edge-devices to the Internet for monitoring and automation without human intervention Ali et al. (2018a). IoT is positively influencing

human lives through interconnection and automation used for applications including environmental monitoring, health monitoring, home appliances, automated devices, surveillance and security. The literature suggests that there will be over 50 billion interconnected IoT edge-devices by the 2020. The multimedia content was estimated to be around 80% of the overall Internet data traffic in 2019. Video surveillance used to monitor traffic conditions is a prime example of a multimedia application in IoT. In the example of traffic monitoring, video streaming is used to collect information regarding vehicle positions, traffic congestion and road ac severity. The data content is streamed over the network in an encoded form, the e live or recorded video is presented to the end-user or professional for analysis Aliyu et al. (2018). Meaningful information is extracted from the video using machine learning techniques Yu and Wang (2019).

Streaming applications can be modelled using periodic dependent task models because theses applications repeatedly execute to service data stream when arrived Wang et al. (2013). For example the MPEG-encoder is executed many times for the video stream. Streaming applications

✉ Haider Ali
H.Ali@derby.ac.uk
Umair Ullah Tariq
u.tariq@cqu.edu.au
Lu Liu
l.liu@leicester.ac.uk
James Hardy
j.hardy@derby.ac.uk

¹ School of Engineering and Technology, Central Queensland University, Norman Gardens, Australia

² Department of Electronics, Computing and Mathematics, University of Derby, Derby, UK

³ Department of Informatics, University of Leicester, Leicester, UK

are computationally extensive as they service continuous streams of data. These properties make streaming services suitable for execution on MPSoCs (Multiprocessor-System-on-Chips) due to their high performance, and exceptional Quality-of-Service (QoS) Han et al. (2015); Ali et al. (2018a); Tariq and Wu (2017). The MPSoCs integrate processors, I/O units, and memory on a single silicon die. MPSoCs are widely deployed in IoT applications and consumer electronics such as smart-phones, personal computers and tablets Ali et al. (2018a); Miao et al. (2018); Zhao et al. (2018). The MPSoC systems also play a vital role in mobile systems such as autonomous cars and robots Tariq et al. (2018). Xilinx Zynq[®] UltraScale+[™] MPSoCs and Renesas R-Car H3 nona-core are the popular examples of the multiprocessor systems commercially available for both robotic and IoT based applications. Similarly Apple A11 Bionic, Xilinx Zynq[®] UltraScale+[™] RFSocS, Samsung Exynos 9810, Intel[®] Stratix[®] 10, Tiler TILE-Gx72[™], and EZchip TILE-Mx100[™] are high-performance MPSoCs used in consumer digital systems Ali et al. (2018a); Tariq and Wu (2016).

To efficiently use the multiprocessor architecture of MPSoCs techniques are required that can increase the degree of parallelism of streaming applications Wang et al. (2013). In this article, we explore task level software pipelining to maximize the degree of parallelism of the periodic dependent task set. Re-timing is a powerful technique applied at the application level in order to transform the intra-period dependencies between tasks by regrouping the tasks from different periods. Re-timing reschedules a parent task few periods ahead of its child task so that the data needed by the child task is available at the start of each period. Consequently, the start time of the child task is not constrained by the finish time of the parent task. Stated simply, re-timing converts the dependent task model to independent task model by transforming intra-period data dependencies into inter-period data dependencies.

The complexity of the data extensive applications in both IoT and consumer digital devices is proliferating subsequently, the number of processors on MPSoC computing systems is increasing Ali et al. (2018a). According to the International Technology Roadmap for Semiconductors (ITRS), soon, MPSoCs will have hundred of processors so, traditional bus-based System-on-Chips (SoCs) will become a bottleneck due to their poor scalability and congestion issues. Network-on-Chip (NoC) based communication alternatively offers an improved scalability with higher flexibility over the traditional bus structures Guindani and Moraes (2013); Tariq and Wu (2016, 2017). Various IoT based real-time streaming applications evidently deploy MPSoCs few of them include human gait analysis Nguyen (2015), human recognition Safaei et al. (2018); Meng et al. (2008), video surveillance Yan et al. (2009); Sasagawa and Mori (2016),

video/image enhancement Saponara et al. (2013), healthcare Djelouat et al. (2018); Zhai et al. (2018), and person tracking Ahmed et al. (2018).

Voltage Frequency Island (VFI), Globally Asynchronous Locally Synchronous (GALS) is recently introduced to NoC based MPSoCs in which tiles are divided into islands. Each island possesses its own operating frequency, supply voltage, and threshold voltage Jang and Pan (2011); Ullah Tariq et al. (2019). The VFI based NoC-MPSoC (VFI-NoC-HMPSoC) is an ideal choice for data extensive applications due to its higher throughput, lower complexity, and superior energy-efficiency Lackey et al. (2002). VFI-MPSoCs require lower number of multiple clock first-in-first-out (MCFIFO) and voltage level converters (VLCs) Mahabadi et al. (2013). Green computing in modern embedded systems is a challenging facet because higher energy consumption reduces the lifetime of an edge-device in IoT and causes an increased carbon dioxide CO_2 emission i.e. carbon footprint Rault et al. (2014); Huang et al. (2014). Proper task scheduling using search-based algorithms on multiprocessor architectures can significantly improve the performance and energy-efficiency of a battery-constrained edge-device. Re-timing is another technique that drastically increases the energy-efficiency Ali et al. (2018a). Re-timing is another technique used to transform intra- data dependencies into inter-data dependencies Wang et al. (2011).

The rest of the paper is organized as follows: Section 2 discusses existing scheduling approaches and/or algorithms. Section 3 presents preliminaries and Sect. 4 explains our re-timing technique. In Sect. 5 we discuss our proposed static contention-aware energy-efficient task scheduling. We present simulation results on different benchmarks in Sect. 6 while Sect. 7 offers the conclusion.

2 Literature review and contributions

Task scheduling on MPSoC architecture is a well-known NP-hard problem. Different heuristics have been developed using mathematical formulation such as Non Linear Programming (NLP), Mixed Integer Linear Programming (MILP), Integer Linear Programming (ILP), and Linear Programming (LP). Similarly, search based heuristics using selection, crossover, mutation, and elitism have been deployed etc. The popular examples of search based heuristics include Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO). Among the aforementioned algorithms, GA is widely used for task scheduling Ali et al. (2018a, 2018b); Mirjalili and Lewis (2016). These evolutionary algorithms are stochastic generate and test algorithms based on exploration and exploitation. Exploration is the capability of a heuristic to discover the unseen regions while exploitation primarily

shows the ability of an algorithm to proceed in the desired direction for improvement. For example, in GA the crossover and mutation are hypothetically considered to perform exploitation and exploration respectively Črepinšek et al. (2013); Vafae and Nelson (2010). However, there is strong criticism that mutation possessed no competitive advantage over crossover Vafae and Nelson (2010). Nevertheless, these stochastic heuristics fail to exploit the available chunk of information (schemata) efficiently. Moreover, exploration and exploitation are totally opposing forces therefore, a well found balance determines the success of a heuristic.

Olafsson (1995) introduced one of the first dynamical model for the scheduling of tasks on heterogeneous multi-processor systems. Aydin et al. introduced DVFS technique to determine the optimal voltage levels for the tasks and developed an algorithm called Earliest Deadline First (EDF) to generate feasible tasks schedule Aydin et al. (2001). Aydin et al. (2004) further investigated the energy-aware scheduling considering periodic tasks and designed Dynamic Reclaiming Algorithm (DRA) to efficiently utilize the available idle slack. Shin and Kim (2003) investigated non-population based scheduling and developed a Non Linear Programming (NLP) based heuristic for assigning optimal discrete voltage levels to each task for reducing the computational energy consumption. Tariq and Wu (2016) scheduled the tasks on homogeneous MPSoCs and assigned continuous voltages to the tasks for executions. Tariq et al. (2018) formulated the scheduling problem as NLP and integrated task ordering and voltage scaling to maximally reduce the energy consumption. They developed an algorithm called Iterative Offline Energy-aware Task and Communication Scheduling (IOETCS) using Earliest Successor-Tree-Consistent Deadline First (ESTCDF) heuristic for generating an initial task schedule and then applying voltage scaling. However, these researchers only investigated energy-aware task scheduling on single processor per VFI.

Research studies further explored task scheduling deploying VFI based MPSoCs using bus as a communication interconnect. For example, Jang and Pan (2011) investigated energy optimization of VFI based NoC-MPSoC by reducing power overheads due to VFIs. Similarly Shin et al. (2011) considered a VFI based NoC-MPSoC and developed inter-VFI communication aware mapping algorithm to decrease the communication energy consumption. Han et al. (2014) mapped the tasks on the processors of the VFIs and communications on the NoC to reduce the overall makespan and inter-VFI communication. The authors developed two contention and energy-aware task mapping and edge scheduling heuristics called CA-TMES-Quick and CA-TMES-Search for assigning tasks to the processors and edges on NoC. Pagani et al. (2014) presented a Single Frequency Approximation (SFA) algorithm for optimal

voltage assignment to the processor islands in MPSoC architecture. SFA is integrated with Dynamic Programming Mapping Algorithm (DPMA) to increase the energy-efficiency and minimize the running time. Liu and Guo (2016) developed an algorithm called Voltage Island Largest Capacity First (VILCF) for task scheduling. The VILCF reduces the energy consumption by fully utilizing an island that is already active before activating other islands. Gamoudi et al. (2018) scheduled periodic tasks on homogeneous NoC-VFI-MPSoC architectures deploying well-known EDF task ordering policy. Tariq et al. (2019) developed a meta-heuristic for energy-efficient and contention-aware dependent tasks with precedence constraints. Though these investigations reduced the energy consumption by proper task mapping and scheduling however they did not consider re-timing at the task level to further decrease the total energy consumption.

In this paper, we investigate energy-efficient and contention-aware static scheduling for a set of tasks with precedence and deadline constraints representing a real-time periodic streaming application on VFI based NoC-HMPSoC. Our contributions and innovations include as follows:

1. We present an innovative re-timing based algorithm to transform the intra-period data dependencies into inter-period data dependencies of the streaming application to maximize the degree of parallelism and at the same time reduce the re-timing latency.
2. We propose a novel meta-heuristic called ARSH-FATI for task scheduling. ARSH-FATI considers energy performance profiles of the processors, contention at NoC links, and inter-VFI communications during task mapping.
3. We also develop a novel contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling scheme applicable on both the task and communication nodes.
4. Our energy management scheme achieves better average energy-efficiency of $\sim 15\%$ and $\sim 20\%$ over CA-TMES-Search and CA-TMES-Quick Han et al. (2015) respectively for 8 real benchmarks. The energy savings further increases to $\sim 35\%$ and $\sim 40\%$ when ARSH-FATI is combined with our proposed re-timing technique. Furthermore, the prologue reduces by $\sim 50\%$ when we compare our re-timing technique with R-DAG Liu et al. (2009).

3 Preliminaries

In this section we explain relevant models that are used in our simulations to generate different results. In this paper, we use the term processor and tile interchangeably.

3.1 Application model

We characterize a real-time periodic application by DAG illustrated in Fig. 1, $G(V, E, X)$. Where $V = \{v_1, v_2, v_3, \dots, v_n\}$ shows a set of tasks and $E \subseteq V \times V$ represents directed edges set where each edge $(v_i, v_j) \in E$ shows data dependency between two tasks. For example, an edge from v_i to v_j indicates that v_i is the predecessor of v_j and outputs the data to v_j , where v_j is the successor of v_i and accepts input data from v_i . Moreover, X denotes a set of directed edge weights and $\chi_{(i,j)}$ is edge-weight showing the volume of data represented in bits sent from v_i to v_j . We consider soft periodic applications and all tasks in the application have a common period D .

3.2 System architecture

We consider a VFI-NoC-HMPSoC architecture shown in Fig. 2 with M processors $P = \{pe_1, pe_2, pe_3, \dots, pe_M\}$. Each tile contains a processor, network interface (NI) card, and local memory. We partition the processors of the target multiprocessor architecture into a set $C = \{c_1, c_2, c_3, \dots, c_{m'}\}$ of m' heterogeneous VFIs. Where each VFI, $c_i \in C$ consist of a set of k homogeneous processors. We assume that processors within an island are of the same type while processors across different VFIs may be of different types. Furthermore, we assume a 2D-mesh topology NoC, XY routing and virtual cut-through (VCT) switching.

3.3 Communication architecture

We consider a 2D-mesh topology NoC interconnect for inter-processor and/or inter-VFI communications. Each processor of the VFI-NoC-MPSoC associates with a router for communicating with other processors. 2D-mesh NoC primarily contains N_R rows and N_C columns with number of processors equal to $N_R \times N_C$. There are five ports in each router, four of them are deployed to communicate with other neighbor routers while one router is dedicated for communicating with the associated processor. A link connects two routers and a processor a router. We assume that all links are indistinguishable and full-duplex with the same bandwidth, b_w .

We deploy a well-known XY deterministic routing which is a choice of selection for 2D-mesh NoC topology. In XY deterministic routing the packets are first routed in x-direction then in the y-direction. We consider a Virtual Cut-through (VCT) packet switching in this paper. wormhole (WH) and VCT are the popular and widely adopted packet switching techniques in NoC interconnects for communications. In WH each packet is divided into small pieces called FLITS. When a packet traverse in the network, WH immediately determines its next hop and forwards it then the subsequent FLITS worm their way through the network. Though, WH switching is a simple approach having higher flow control efficiency over VCT however, in case of congestion occurrence, the stalling packet may block all the links subsequently, results a lower link utilization. In VCT the entire packet is sent to next node while in WH a portion of the message is stored. Thus, VCT

Fig. 1 DAG representation of periodic application

Fig. 2 VFI-NoC-HMPSoC architecture

shows higher link utilization, lower latency and packet blocking probability therefore, we assume VCT switching Ali et al. (2018a); Tariq et al. (2019).

3.4 Energy model

We adopted the energy model presented in Ogras et al. (2009), where the total energy consumption is given as follows:

$$E = \sum_{\forall v_i \in V} E_i + \sum_{\forall v_u \in V^*} E_u \quad (1)$$

where V and V^* show task nodes and communication nodes respectively. E_i is the energy consumed in execution of a task v_i mapped on processor pe_k that belongs to VFI c_j :

$$E_i(V_{ddj}) = (C_{effk} V_{ddj}^2 NC_{i,k} + L_g(V_{ddj} K_3 e^{K_4 V_{ddj}} e^{K_5 V_{bs}} + |V_{bs}| I_{jn})) \Delta t_i \quad (2)$$

where C_{effk} is the effective switched capacitance of pe_k , V_{ddj} represents supplied voltage, L_g denotes the number of logic gates, K_3, K_4, K_5 represents technology specific parameters, v_{bs} and I_{jn} shows body-bias voltage and leakage current respectively. E_u is the energy consumed in execution of communication node v_u that traverses the route $R_u = \langle L_1, L_2, \dots, L_l \rangle$. The parent and child task nodes of v_u are v_p and v_c respectively. E_u is calculated as follows:

$$E_u = \sum_{\tau=1}^{l-1} \chi_{p,c} E_{bit}(\tau, \tau+1) \quad (3)$$

where $E_{bit}(src, dest)$ is the energy consumed to transmit one bit from source tile to destination $E_{bit}(src, dest)$ is discussed in detail in Ogras et al. (2009).

4 Offline pipelined scheduling

In this section we discuss our coarse-grained task level pipelining or re-timing approach. Before we start explaining our proposed re-timing approach we briefly explain re-timing.

The notion of re-timing was originally introduced by Leiserson and Saxe (1991) to reduce the synchronous circuit cycle period. Recently, Wang et al. (2011, 2013, 2014) extended re-timing to schedule applications represented by a classical DAG (Directed Acyclic Graph) task model on MPSoCs and is defined as follows:

Definition 1 Given a DAG G , re-timing is a function $RT : V \mapsto \mathbb{Z}$, that maps each node $v_i \in G$ to an integer $RT(v_i)$, where $RT(v_i)$ is the number of periods of v_i reschedule in prologue. Re-timing v_i once if it is legal, reschedule one period of v_i into prologue.

From the point of view of the program, re-timing regroup loop body such that some or all dependencies within a period are transformed into inter-period dependencies. The re-timing function is valid if no reference is made to the data from the future period. The definition of a valid re-timing function is:

Definition 2 Given a DAG G , for each edge $(v_i, v_j) \in G$ and $v_i, v_j \in G$, the re-timing function RT is said to be valid if $RT(v_i) - RT(v_j) \geq 0$.

If $RT(v_i) - RT(v_j) < 0$, the re-timing function is illegal because this condition implies a reference to unavailable data from future.

Figure 3(a) shows the schedule of first three periods of the DAG shown in Fig. 4(a). Figure 4(b) and Fig. 4(c) showing the re-timing values of nodes generated RDAG and our approach respectively. Table 1 demonstrates the execution time and energy consumption of the tasks of the CTG in Fig. 4(a). The applications is scheduled on MPSoC that consists of two processors pe_1 and pe_2 . Table 1 shows the energy consumption and execution time of each task on the two processors. Compared to pe_1 , pe_2 is more energy-efficient.

Figure 3(a) illustrates a schedule generated by CA-TMES-Search. Notice that CA-TMES-Search fails to efficiently utilize the more energy-efficient processor pe_2 because it favours the processor on which the task can start the earliest. pe_2 remains idle until v_1 completes execution on pe_1 . This is because of the intra-period dependency between v_1 and v_2 . CA-TMES-Search cannot utilize this slack.

Figure 3(c) shows the schedule generated by our approach. Notice that if intra-period dependencies can be transformed into inter-period dependencies then the wasted

slack can be utilized. This can be obtained by regrouping tasks from different periods with computation and communication node rescheduling. As each task is periodic, in Fig. 3(c), we reschedule periodic task v_1 and execute it one period before v_2 and v_3 . The newly added period is called the prologue. In this way the data required by v_2 and v_3 is available at the start of each period, therefore v_2 can start early on pe_2 . Consequently task v_4 can be scheduled on pe_2 . Since our approach can utilize the available resources more efficiently hence it is able to generate more energy-efficient schedule. The energy consumption of the schedule in Fig. 3(a) is $7.5nJ$ where as the schedule in Fig. 3(c) consumes $7nJ$. Our approach can further reduce the energy to $6nJ$ if the MPSoC has another energy efficient processor like pe_2 .

Although re-timing is effective in reducing energy consumption, but there is a cost associated with it. One drawback of re-timing is that it adds prologue. The prologue latency is the number of periods in prologue times the period D . The number of periods in the prologue is equal

Fig. 3 Periodic scheduling **a** Scheduling without re-timing, **b** R-DAG based re-timing, **c** re-timing using our approach

Fig. 4 Periodic application **a** DAG, **b** DAG with re-timing values using R-DAG, **c** DAG with re-timing values using our approach

Table 1 Energy consumption and execution time of tasks of DAG in Fig. 4a

Task	pe1		pe2	
	Energy (nJ)	Execution Time	Energy (nJ)	Execution Time
V_1	2	1	1.5	1.5
V_2	2	1	1.5	1.5
V_3	2	1	1.5	1.5
V_3	2	1	1.5	1.5

to the maximum re-timing value RT_{max} of the nodes in G , $RT_{max} = \max\{RT(v_i) : \forall v_i \in G\}$. Thus the prologue latency is:

$$prologLatency = RT_{max} \times D \quad (4)$$

Besides energy reduction we also want to minimize the *prologLatency*. Figure 3(b) shows the re-timed schedule generated by R-DAG. Compared to this the prologue latency of the re-timed schedule generated by our approach is half. We are able to reduce the prologue latency because we take a

different approach compared to R-DAG. We first transform the DAG into independent task set by relaxing the precedence constraints between the nodes and then we schedule the independent task model onto the MPSoC. Hence the MPSoC resources are maximally utilized and do not remain idle due to precedence constraints between nodes. Finally we calculate the re-timing values of the nodes and generate the re-timed schedule.

Algorithm 1 describes our offline pipelined scheduling approach. Our approach has two main steps. Firstly we generate the relaxed schedule of DAG G on the MPSoC by relaxing the precedence constraints between nodes in G . In Sect. 5 we discuss in detail our offline scheduling approach.

Secondly we calculate the re-timing values of the nodes. Given a node v_i and its child node v_j , our re-timing function is defined as follows:

$$RT(v_i) = \begin{cases} 0 & \text{if } v_i \text{ is sink node} \\ \max\{RT(v_i), RT(v_j) + 1\} & \text{if } \zeta(v_i) < \rho(v_j) \\ \max\{RT(v_i), RT(v_j)\} & \text{otherwise} \end{cases} \quad (5)$$

where the start and finish times of a task node v_i are respectively denoted by $\rho(v_i)$ and $\zeta(v_i)$.

Algorithm 1: Offline pipelined scheduling

input : A DAG G , tasks Deadlines, an MPSoC, total number of iterations Ω and total number of initial solutions κ

output: Re-timed schedule

- 1 Relax the Generate the relaxed schedule of DAG G on the MPSoC using Algorithm 2;
 - 2 **for each** node v_i **in the reverse topological order of** G **do**
 - 3 **for each** parent v_j **of** v_i **do**
 - 4 **if** $\rho(v_i) < \zeta(v_j)$ **then**
 - 5 $RT(v_i) \leftarrow \max\{RT(v_i), RT(v_j) + 1\}$;
 - 6 **else**
 - 7 $RT(v_i) \leftarrow \max\{RT(v_i), RT(v_j)\}$;
 - 8 Generate the retimed schedule;
-

5 Contention-aware static energy optimization approach

The details of our proposed ARSH-FATI meta-heuristic are given in Algorithm 2. Before we start explain our algorithms first we define an extended graph G_e illustrated in Fig. 5. In an application, two kinds of events exist i.e. communications and tasks. In order to schedule the communications using traditional DAG based scheduling approaches, we transform a DAG i.e. G into an extended graph G_e . Given

a task to processor mapping an extended graph we construct G_e by inserting an additional node v_s to graph G for each edge (v_i, v_j) whose tail node v_i and head node v_j are mapped on different processors. The edge (v_i, v_j) in extended graph is replaced by two edges (v_i, v_s) and (v_s, v_j) . The additional inserted nodes are called communication nodes. The extended graph is represented by $G(V + V^*, E)$, where V shows task nodes, V^* represents the communication nodes, and E denotes edges.

Fig. 5 Extended graph

Algorithm 2: ARSH-FATI

input : A DAG G , tasks Deadlines, an MPSoC, maximum generations $maxGens$, β , T_1 , T_2 and DR_0 the initial value of DR

output: Task to processor mapping map .

- 1 Construct a matrix Π of zeros having dimensions $\mu \times n$ and a vector f of zeros having dimension $\mu \times 1$;
- 2 $DR \leftarrow DR_0$;
- 3 **for** $\eta \leftarrow 1$ **to** κ **do**
- 4 **for each** $v_i \in G$ **do**
- 5 $\Pi[\eta][i] \leftarrow \lceil rand()(|P| - 1) + 1 \rceil$;
- 6 Generate the extended graph G_e for task to processor mapping determined by $\Pi[\eta][:]$;
- 7 Relax the precedence constraints between the node of G_e ;
- 8 $[m, e] \leftarrow EECDF(G, G_e, \Pi[\eta][:])$; $f[\eta] \leftarrow fitness(m, e)$;
- 9 Find the generation best π_b and worst π_w members;
- 10 $gBest' \leftarrow \pi_b$; $T \leftarrow T_2$;
- 11 $gBestFit' \leftarrow f_b$; // where f_b is the fitness value of π_b
- 12 **repeat**
- 13 **for** $\eta \leftarrow 1$ **to** μ **do**
- 14 $\pi \leftarrow \Pi[\eta][:]$;
- 15 **for** $i \leftarrow 1$ **to** n **do**
- 16 $r \leftarrow rand(0, 1)$;
- 17 $r_1 \leftarrow rand(0, 1)$;
- 18 $r_2 \leftarrow rand(0, 1)$;
- 19
$$\pi[i] \leftarrow \begin{cases} \min(abs(\lceil \pi[i] \rceil + r_1(gBest[i] - \pi[i]) - r_2(\pi_w[i] - \pi[i])), M) & \text{if } r \leq DR \\ \pi[i] & \text{otherwise} \end{cases}$$
- 20 Generate the extended graph G_e for task to processor mapping determined by π ;
- 21 Relax the precedence constraints between the node of G_e ;
- 22 $[m, e] \leftarrow EECDF(G, G_e, \pi)$;
- 23 $f \leftarrow fitness(m, e)$;
- 24 **if** $f > f[\eta]$ **then**
- 25 $f[\eta] \leftarrow f$;
- 26 $\Pi[\eta, :] \leftarrow \pi$;
- 27 **if** $f > gBestFit'$ **then**
- 28 $gBestFit' \leftarrow f$;
- 29 $gBest' \leftarrow \pi$;
- 30 **else**
- 31 **if** $e^{\frac{(f-f[\eta])}{T}} > rand(0, 1)$ **then**
- 32 $f[\eta] \leftarrow f$;
- 33 $\Pi[\eta, :] \leftarrow \pi$;
- 34 $DR \leftarrow \begin{cases} \min(\frac{DR}{\lambda}, DR_{max}) & \text{if } gBestFit > gBestFit' \\ \max(\lambda DR, DR_{min}) & \text{otherwise} \end{cases}$;
- 35 $gBestFit \leftarrow gBestFit'$; $gBest \leftarrow gBest'$;
- 36 Update the generation worst π_w ;
- 37 $T \leftarrow \lambda_1 T$;
- 38 **until** termination;
- 39 $map \leftarrow gBest$;

The robustness of ARSH-FATI algorithm lies in the notion of updating the parameter dimensional rate (DR) at run-time during the searching process. Our algorithm attains a satisfactory trade-off between the exploration and exploitation attributes of the search process. We define the parameter DR as the percentage of tasks that are probabilistically re-mapped to generate a new solution (mapping) from the current (best and worst) solutions. The need for

only re-mapping a percentage of tasks and not all the tasks stems from the sensitivity of energy consumption to task mapping in this (energy optimization) problem. In other words re-mapping, even a small subset of tasks can generate a schedule with energy consumption significantly different than the schedule generated by the original mapping. Thus, the role of DR is to adjust at run-time the exploration and exploitation features of ARSH-FATI meta-heuristic that we explain in the following.

1. **Setting the initial value of DR:** We set DR to an initial value DR_0 (Line 3). DR_0 can take on any value between the range $0 < DR_0 \leq 1$. The higher DR value means more explorative search that leads to large and unconstrained step sizes. Compared to this a small value of DR motivates a more exploitative search by allowing small and conservative steps in the search space. Therefore we set $DR_0 = 0.4$.
2. **Population Generation:** We initially generate a matrix Π of dimensions $\mu \times n$ of zeros (Line 1), where μ is the size of the population and n is the total number of tasks. Each member of the population reflects a possible task to processor mapping and the entire population reflects μ different mappings. The value of the element $\Pi[i][j]$ represents the processor where the j^{th} task is mapped in the i^{th} member of the population. This value is a positive integer that determines the processor where the task is mapped. For example if the value of the third element is 2 this reflects that task v_3 is mapped on pe_2 . We use the notation $\Pi[i][:]$ to access the i^{th} member of the population. We generate an initial value of the member of the population by randomly mapping tasks to processors. For each member we calculate the fitness value by the following fitness function:

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

where e is the total energy consumption and m is the make-span of the schedule. e and m are obtained by executing algorithm 3. Before we proceed with our discussion we define the following two terms:

- (a) **Global Best, $gBest$,** is a member of the population that has the highest fitness value $gBestFit$ in all generations.
 - (b) **Generation worst π_w** is a member of the population that has the lowest fitness value in a generation.
3. **Population Refinement:** We refine the initial population until the termination criteria satisfy (Lines 12-36). In each generation we update each member of the population. The j^{th} element of the i^{th} member is updated as follows:

$$\Pi[i, j] \leftarrow \begin{cases} \min(abs(\lceil \Pi[i, j] + r_1(gBest[j] - \Pi[i, j]) - r_2(\pi_w[j] - \Pi[i, j]) \rceil), M) & \text{if } r \leq DR \\ \Pi[i, j] & \text{otherwise} \end{cases}$$

where r_1 and r_2 are the random numbers. The term $r_1(gBest[i] - \Pi[i, j])$ reflects the likelihood that the member moves closer to global best and the term

$r_2(\pi_w[i] - \Pi[i, j])$ reflects the likelihood that it moves away from the worst member of the population. We use an acceptance probability function $P(\Delta f, T)$ to adopt or reject the new value of the member $\Pi[i, :]$ (Lines 18-24). The parameter Δf is the difference of the new and old fitness value of $\Pi[i, :]$, $\Delta f = f_{new} - f_{old}$. The parameter T is referred to as temperature. We define function $P(\Delta f, T)$ as follows:

$$P(\Delta f, T) = \begin{cases} 1 & \text{if } f_{new} > f_{old} \\ e^{-\frac{\Delta f}{T}} & \text{otherwise} \end{cases} \quad (6)$$

when the new mapping reduces the energy consumption it is always accepted. If the new mapping is worse than the current mapping, probability still exists that may be accepted. We have included this feature in ARSH-FATI to prevent it getting stuck in a local optimum. The value of temperature T reduces in each iteration by multiplying it with a cooling factor λ_1 , ($0 < \lambda_1 < 1$) (Line 27). The value of λ_1 is calculated from $maxGens$:

$$\lambda_1 = \left(\frac{T_1}{T_2} \right)^{\frac{1}{maxGens}} \quad (7)$$

where T_2 is a very large number and T_1 is a very small number. Initially the value of temperature is set to T_2 and it reduces to T_1 as optimization finishes. In each generation we update DR (Line 25):

$$DR = \begin{cases} \min \left\{ \frac{DR}{\lambda}, DR_{max} \right\} & \text{if } gBest \text{ improves} \\ \max \{ \lambda DR, DR_{min} \} & \text{otherwise} \end{cases} \quad (8)$$

where the λ is dimensional rate adaption parameter and its value lies within the range $0 < \lambda < 1$. In this work λ is set to 0.98. The parameter λ sets the new value of the DR during the optimization process. The values of DR_{min} and DR_{max} are respectively the upper and lower bounds on DR . The values of DR_{min} and DR_{max} must be set subject to constraint $0 < DR_{min} < DR_{max} < 1$. We avoid excessive exploration and exploitation by setting DR_{min} and DR_{max} to 0.2 and 0.6 respectively.

4. **Termination criteria:** ARSH-FATI terminates if either the generation count reaches maximum generation $maxGens$ or no improvement is observed in $gBest$ for consecutive β generations.

5.1 Earliest edge consistent deadline first (EECDF) algorithm

Before we describe *EECDF* we define some notations. The worst case execution time of a task node v_i mapped on processor pe_k operating at frequency f_j is $et(v_i) = \frac{NCC(v_i, k)}{f_j}$,

where $NCC(v_i, k)$ is the worst case clock cycles of v_i on processor pe_k . The start and finish times of a task node v_i are respectively denoted by $\rho(v_i)$ and $\zeta(v_i)$. Similarly for a communication node v_j (corresponding to edge (v_a, v_b)) the transmission time on a link L between processors pe_s and pe_d is $et(v_j, L) = \frac{\chi_{a,b}}{b_w \min\{f_s, f_d\}}$, where b_w is the link width, f_s and f_d are the frequencies of pe_s and pe_d respectively. The start and finish time of v_j on link L are respectively denoted by $\rho(v_j, L)$ and $\zeta(v_j, L)$, where $\zeta(v_j, L) = \rho(v_j, L) + et(v_j, L)$.

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

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

$$d'_i = \min\{d_i, \min\{d'_j - et_j : \forall v_j \in (ISucc_i)\}\} \quad (9)$$

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

The $EECDF$ algorithm is described in Algorithm 3. We performs four major steps.

1. Calculate the ECD of each task $v_i \in G$ (Line 1).
2. Relax the precedence constraints between the nodes in G_e . We generate the schedule under no precedence constraints so no wasted slack exist due to the precedence constraints.
3. Create a ready queue R and insert all the source nodes in G_e to R (Line 2).
4. Find a node v_i that has minimum ECD in R and schedule it. Then delete v_i from R and insert all the ready nodes in G_e to R . Repeat this until R is empty (Line 3-10).
5. Calculate the energy E and make-span m of the schedule.

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

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

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

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

Consider a communication node v_j whose source is mapped on pe_{src} and destination is mapped on pe_{dest} , the routing algorithm used by the network generates the route R_j from pe_{src} to pe_{dst} . The route $R_j = \langle L_1, L_2, \dots, L_l \rangle$ is an ordered list of links, where L_1 is the first link and L_l is the last link on the route. The schedule of v_j is obtained by applying the following rules collectively in order:

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

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

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

- **R6:** The finish time of v_j is equal to the finish time of v_j on the last link L_l , $\zeta(v_j) = \zeta(v_j, L_l)$.

Table 2 Real Benchmarks and Description

Real Benchmark	Description
Automatic Target Recognition (ATR)	Performs pattern recognition in real-time streaming applications
MP3-decoder	Performs Inverse Discrete Transform (IDCT) and Huffman decoding
JPEG-encoder	Performs Discrete Cosine Transform (DCT) and Huffman encoding
Office	Performs text processing, gray-scale to binary conversion, and image rotation
Auto Industry-1	Performs Fast Fourier Transform (FFT), IDCT, finite/infinite impulse response filtering, matrix arithmetic.
Auto Industry-2	Performs FFT, IDCT, filtering, matrix arithmetic, table lookup, Inverse FFT, interpolation and road speed calculation
Consumer-1	Performs JPEG compression and decompression and different conversions
Consumer-2	Performs JPEG decompression and compression and RGB to CMYK and RGB to YIQ conversions

Table 3 Processors operating frequency and power consumption

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

Algorithm 3: *EECDF*

input : A DAG G , an extended DAG G_e , task to processor mapping vector α
output: Energy e and make-span m of schedule

- 1 Calculate the ECD , d'_i of each task in $v_i \in G$;
- 2 Relax the precedence constraints in G_e ;
- 3 Insert all source node in a ready queue R ;
- 4 **while** there are ready nodes in R **do**
- 5 Find a node v_i in R with smallest d'_i while ties are broken in favour of smallest index;
- 6 **if** v_i is a task node **then**
- 7 Schedule v_i subject to rules **R1**, **R2** and **R3**;
- 8 **else**
- 9 Schedule v_i subject to rules **R4**, **R5**, **R6** and **R7**;
- 10 Delete v_i from R ;
- 11 Insert all ready nodes in R ;
- 12 Calculate the energy e and make-span m of the schedule;

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

Concisely, we presented a contention-aware population based searching algorithm for task scheduling that can switch into explorative and exploitative search modes for better performance trade-off. We also developed a novel energy-aware re-timing approach that efficiently avails the idle slack in the processors while significantly reduces the prologue for real-time streaming applications. Moreover, we performed task ordering, and mapping in an integrated manner.

6 Experimental setup and results

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

We deploy 8 real benchmarks listed in Table 2 adopted from Embedded System Synthesis Benchmarks Suite (E3S). It is well-known and widely adopted benchmark suit in the task scheduling research Cheng et al. (2013).

We deploy an energy model of Samsung Exynos 5422 chip described in Liu et al. (2015) for our simulations while using two types of processors. Type 1 processors are Cortex

Fig. 6 Dimensional rate parameter variations

A15 (big), high-performance and type 2 are low-power, Cortex A7 (little) processors. The Cortex A15 consumes $\sim 6 - 12$ times higher power compared to Cortex A7 Lukefahr et al. (2014). The relative power consumptions and operating frequencies and of both types are given in Table 3. The 70 nanometers (nm) processor technology parameters are adopted from Ali et al. Ali et al. (2018a). We used Matlab version R2016a to build the simulation environment and conducted experiments using a hardware platform of Intel (R) Xeon (R), i5-3570 CPU with 3.50 GHz clock frequency and 16.00 GB memory, a cache of 10 MB.

We compare our ARSH-FATI results with the energy management approach presented by Han et al. (2015). The authors developed a contention-aware static mapping and scheduling scheme for a set of tasks with precedence constraints to decrease the makespan and inter-VFI communications for total energy consumption. They proposed two static heuristics CA-TMES-Quick and CA-TMES-Search. CA-TMES-Quick performs task mapping first then determines the routes for communications. CA-TMES-Search calculates start-time for the tasks and avoids communication contention. it selects a processor which offers earliest start time (EST) for a task. Concisely, CA-TMES-Search comparatively to CA-TMES-Quick performs better because it coordinates task mapping in an exhaustive way subsequently, significantly reduces the makespan.

6.1 Results and discussions

The VFI-NoC-MPSoCs are used as an edge-computing devices in IoT for processing data-extensive applications which involve multimedia data. The VFI-NoC-MPSoCs are de-facto computing architectures as they provide higher throughput and extraordinary performance as compared to the traditional controllers. Energy consumption reduction is a prime concern in edge-devices as higher energy consumption decreases lifetime of the network. Moreover, reduced

prologue i.e. latency is required during video-streaming in real-time applications such as surveillance and traffic monitoring. Therefore, we aim to reduce both the energy consumption of the edge-devices and latency of the real-time streaming application in IoT. First, we generate results for two scenarios considering homogeneous VFI-NoC-MPSoC computing platform and heterogeneous multiprocessing computing system. Second, we produce results for ARSH-FATI without re-timing and then combining our innovative re-timing approach with it deploying VFI-NoC-HMPSoC.

Before we explain the impact of ARSH-FATI meta-heuristic on energy performance, we discuss the important parameter DR impact on ARSH-FATI. Figure 6 shows the impact of DR parameter on ARSH-FATI performance. Initially we set $DR = 0.3$ though it can acquire values from 0.1 to 0.5 with negligible impact on the overall energy-efficiency during static task scheduling. The results demonstrate that energy-efficiency of the ARSH-FATI slightly decreases when $DR = 0.1$ and $DR = 0.5$ are used. However, ARSH-FATI automatically adjusts the DR value to attain maximum energy-efficiency. Setting $DR = 0.10$ initially means ARSH-FATI performs an insufficient exploration and $DR = 0.5$ leads to an excessive exploration. Therefore, $DR = 0.3$ is set which is the nominal initial value for our, ARSH-FATI meta-heuristic. ARSH-FATI converges and DR value relatively stabilizes at number of iterations (NI) equal to 200 while a infinitesimal variation occurs at 500. No further variations occur when $NI > 500$ therefore, we consider $NI = 500$, $\lambda = 0.9$, and $\mu = 5$, for our experiments.

6.1.1 ARSH-FATI without re-timing

We generate results for 8 real benchmarks considering homogeneous and heterogeneous VFI-NoC-MPSoC computing platforms with 4 VFIs, 2×2 processors per VFI (PPI). First, we consider all the processors to be homogeneous i.e. type 1 and we set the operating frequency to maximum,

Fig. 7 Real benchmarks energy consumption

$f_{max} = 2.0$ GHz. Second, we use a heterogeneous VFI-NoC-MPSoC architecture using both type 1 and type 2 processors. We randomly select the type of the processors to generate a heterogeneous multiprocessor computing platform to ensure unbiased experimentation.

Figure 7 shows the energy performance of our ARSH-FATI static task scheduler for two different scenarios when compared to CA-TMES-Search and CA-TMES-Quick, state-of-the-art task scheduling techniques. The x-axis denotes the real-world benchmarks while the y-axis represents energy consumption in joule (J). Not surprisingly $(ARSH - FATI)_{homogeneous}$ consumes lower energy because our population-based meta-heuristic ARSH-FATI performs better solution space search as it can switch between exploitation and exploration search modes during task scheduling. Our scheduler $(ARSH - FATI)_{homogeneous}$ tends to map the dependent tasks i.e. parent and child nodes on the same processor to reduce the communication energy E_c in Equation (1). Energy-efficiency further increases when we use a heterogeneous architecture for task scheduling. ARSH-FATI considers energy performance profiles of the processor during task mapping and schedules high energy consuming tasks on energy-efficient processor such that the timing constraints are not violated.

Now, we determine the impact of the PPIs on energy consumption and examining the ability of $ARSH - FATI$ to efficiently utilize the computing platform resources for real benchmarks. We set number of VFI i.e. $NVFI = 4$, heterogeneous computing system, and gradually set, $PPI = 2 \times 2, 4 \times 2, 4 \times 3$. Figure 8 illustrates that The energy consumption of MP3-decoder, JPEG-encoder, and Robot reduces however other benchmarks do not demonstrate a noticeable reduction in the energy consumption when PPI is increased gradually. MP3-decoder, JPEG-encoder, and Robot consist of relatively higher number of task nodes and offer more degree of parallelism. We also

Table 4 Maximum Re-timing Values Comparison

Benchmark	$RT_{max}@R - DAG$	$RT_{max}@Our - re - timing$
ATR	5	2
MP3-decoder	10	6
JPEG-encoder	14	7
Office	3	2
Auto-1	5	3
Auto-2	4	2
Consumer-1	4	3
Consumer-2	2	1

examine the performance of $ARSH - FATI$ on a complex real benchmark, Robot that contains 88 task nodes. Compared to $PPI = 2 \times 2$, $ARSH - FATI$ achieves energy efficiency of $\sim 13\%$ and $\sim 18\%$ for robot when $PPI = 4 \times 2$ and $PPI = 4 \times 3$ are deployed respectively. These results show that $ARSH - FATI$ utilizes the resources efficiently by availing the degree of parallelism present within the real benchmarks. However re-timing is necessary to further decrease the total energy consumption.

6.1.2 ARSH-FATI with re-timing

Now we generate results for 8 real benchmarks deploying VFI-NoC-HMPSoC architecture and integrate ARSH-FATI meta-heuristic with our innovative re-timing technique. We set $VFI = 4$ and $PPI = 2 \times 2$ while randomly generate a heterogeneous multiprocessor system for task scheduling.

Figure 9 illustrates the energy consumption comparison for real benchmarks of $(ARSH - FATI)_{Heterogeneous}$, $(ARSH - FATI)_{Heterogeneous-re-timing}$, CA-TMES-Quick and CA-TMES-Search. The horizontal axis and vertical axis show benchmarks and energy

Table 5 ARSH-FATI Energy Performance Summary with and without Re-timing

Static Scheduler	CA-TMES-Search	CA-TMES-Quick
$(ARSH - FATI)_{Homogeneous}$	08%	13%
$(ARSH - FATI)_{Heterogeneous}$	15%	20%
$(ARSH - FATI)_{Heterogeneous-re-timing}$	35%	40%

Fig. 8 Real benchmarks energy consumption using different PPIs

consumption respectively. The energy-efficiency further increases significantly when we integrate our re-timing technique with $(ARSH - FATI)_{Heterogeneous}$. The $(ARSH - FATI)_{Heterogeneous-re-timing}$ achieves an average energy savings of $\sim 20\%$, $\sim 35\%$, $\sim 40\%$ over $(ARSH - FATI)_{Heterogeneous}$, CA-TMES-Quick and CA-TMES-Search respectively. This increase in the energy-efficiency is due to transforming the dependent task model into an independent task model. In other words intra-dependencies in data are transformed into inter-dependencies. This transformation enables the scheduling technique

to efficiently utilize the available resources and the slack in the processors. Which consequently reduces the energy consumption for real benchmarks.

We compare the prologue latency of our re-timing approach with the R-DAG Liu et al. (2009). We conducted the experiments on 8 real benchmarks to estimate the re-timing values.

In terms of energy consumption the both of the approaches perform similarly and there is no noticeable change. However, the prologue latency of our approach is significantly lower than R-DAG. We compare the prologue latency in terms of maximum re-timing RT_{max} . The smaller the value of RT_{max} the shorter the prologue latency. The RT_{max} values are summarized in Table 4. Our re-timing technique reduces the RT_{max} by $\sim 50\%$ when R-DAG is used as baseline.

Summarizing the observations we achieve an overall average energy savings of $\sim 8\%$, $\sim 15\%$ over CA-TMES-Search for $(ARSH - FATI)_{homogeneous}$ and $(ARSH - FATI)_{heterogeneous}$ respectively. Similarly, our scheduling approach $(ARSH - FATI)_{homogeneous}$ and $(ARSH - FATI)_{heterogeneous}$ attains an average energy-efficiency of $\sim 13\%$ and $\sim 20\%$ respectively compared to CA-TMES-Quick. The energy consumption significantly further reduces when we integrate our re-timing technique with ARSH-FATI ($(ARSH - FATI)_{Heterogeneous-re-timing}$) achieving an average energy-efficiency of $\sim 20\%$, $\sim 35\%$, $\sim 40\%$ over $(ARSH - FATI)_{Heterogeneous}$, CA-TMES-Quick, and CA-TMES-Search respectively as listed in Table 5. Similarly or re-timing technique decreases the prologue by $\sim 50\%$. Concisely, our novel algorithm ARSH-FATI outperforms CA-TMES-Search and CA-TMES-Quick while our re-timing approach significantly reduces the prologue compared to R-DAG.

Fig. 9 Real benchmarks energy consumption with re-timing

7 Conclusion

Network-on-Chip (NoC) based Multiprocessor System-on-Chip (MPSoC) architecture is becoming a de-facto computing platform for real-time computational intensive applications in Internet-of-Things (IoT) due to its higher performance and Quality-of-Service (QoS). Energy consumption reduction at the edge-device level in IoT is significantly important because higher energy consumption reduces the lifetime of edge-device and network. First time ever we investigated a contention-aware and energy-efficient dependent tasks scheduling on VFI-NoC-HMPSoC computing system. We proposed a novel meta-heuristic ARSH-FATI for energy-aware task scheduling. Our algorithm ARSH-FATI can dynamically switch at run-time between exploitative and explorative search modes for better energy trade-off. We also integrated communication contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling with ARSH-FATI to minimize the total energy consumption. We performed experiments on 8 real benchmarks for different scenarios. We also transformed the periodic DAG tasks into periodic independent tasks using innovative coarse-grained software pipelining i.e. re-timing mechanism for further energy savings. Our static scheduler outperformed CA-TMES-Search and CA-TMES-Quick energy management techniques and achieved $\sim 15\%$ and $\sim 20\%$ average energy savings respectively. The energy-efficiency increased up to $\sim 35\%$ and $\sim 40\%$ when ARSH-FATI combined with our novel re-timing technique. Moreover, our re-timing technique reduces the prologue by $\sim 50\%$ compared to state-of-the-art approach R-DAG. We do not consider memory limits during our task scheduling and re-timing therefore, memory bounds can be violated. Subsequently, one of the future work could be developing a memory-aware energy-efficient task scheduling approach for real-time streaming applications. Moreover, any voltage scaling can be integrated with ARSH-FATI to further increase the energy-efficiency.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ahmed I, Ahmad A, Piccialli F, Sangaiah AK, Jeon G (2018) A robust features-based person tracker for overhead views in industrial environment. *IEEE Internet Things J* 5(3):1598–1605
- Ali H, Tariq UU, Zheng Y, Zhai X, Liu L (2018a) Contention & energy-aware real-time task mapping on noc based heterogeneous mpsoCs. *IEEE Access* 6:75,110–75,123
- Aliyu A, Abdullah AH, Kaiwartya O, Cao Y, Lloret J, Aslam N, Joda UM (2018) Towards video streaming in iot environments: vehicular communication perspective. *Comput Commun* 118:93–119
- Ali H, Zhai X, Tariq UU, Liu L (2018b) 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, pp 1099–1104
- Aydin H, Melhem R, Mossé D, Mejía-Alvarez P (2004) Power-aware scheduling for periodic real-time tasks. *IEEE Trans Comput* 53(5):584–600
- Aydin H, Melhem R, Mossé D, Mejía-Alvarez P (2001) Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In: Real-Time Systems, 13th Euromicro Conference on, 2001., IEEE, pp 225–232
- Cheng Y, Zhang L, Han Y, Li X (2013) Thermal-constrained task allocation for interconnect energy reduction in 3-d homogeneous mpsoCs. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 21(2):239–249
- Črepinšek M, Liu SH, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. *ACM Comput Surv (CSUR)* 45(3):35
- Djelouat H, Zhai X, Al Disi M, Amira A, Bensaali F (2018) System-on-chip solution for patients biometric: a compressive sensing-based approach. *IEEE Sensors J*
- Gammoudi A, Benzina A, Khalgui M, Chillet D (2018) Energy-efficient scheduling of real-time tasks in reconfigurable homogeneous multicore platforms. *IEEE Trans Syst Man Cybern Syst*
- Guindani G, Moraes FG (2013) (2013) Achieving qos in noc-based mpsoCs through dynamic frequency scaling. In: System on Chip (SoC), 2013 International Symposium on, IEEE, pp 1–6
- Han JJ, Lin M, Zhu D, Yang LT (2014) Contention-aware energy management scheme for noc-based multicore real-time systems. *IEEE Trans Parallel Distrib Syst* 26(3):691–701
- Han JJ, Lin M, Zhu D, Yang LT (2015) Contention-aware energy management scheme for noc-based multicore real-time systems. *IEEE Trans Parallel Distrib Syst* 26(3):691–701
- Huang J, Meng Y, Gong X, Liu Y, Duan Q (2014) A novel deployment scheme for green internet of things. *IEEE Internet Things J* 1(2):196–205
- Jang W, Pan DZ (2011) A voltage-frequency island aware energy optimization framework for networks-on-chip. *IEEE J Emerg Sel Topics Circuits Syst* 1(3):420–432
- Lackey DE, Zuchowski PS, Bednar TR, Stout DW, Gould SW, Cohn JM (2002) Managing power and performance for system-on-chip designs using voltage islands. In: Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on, IEEE, pp 195–202
- Leiserson CE, Saxe JB (1991) Retiming synchronous circuitry. *Algorithmica* 6(1):5–35
- Liu J, Guo J (2016) Energy efficient scheduling of real-time tasks on multi-core processors with voltage islands. *Future Gen Comput Syst* 56:202–210
- Liu H, Shao Z, Wang M, Du J, Xue CJ, Jia Z (2009) Combining coarse-grained software pipelining with dvs for scheduling real-time

- periodic dependent tasks on multi-core embedded systems. *J Signal Process Syst* 57(2):249–262
- Liu D, Spasic J, Chen G, Stefanov T (2015) Energy-efficient mapping of real-time streaming applications on cluster heterogeneous mpsoCs. In: *Embedded Systems For Real-time Multimedia (ESTIMedia)*, 2015 13th IEEE Symposium on, IEEE, pp 1–10
- Lukefahr A, Padmanabha S, Das R, Dreslinski Jr R, Wenisch TF, Mahlke S (2014) Heterogeneous microarchitectures trump voltage scaling for low-power cores. In: *Proceedings of the 23rd international conference on Parallel architectures and compilation*, ACM, pp 237–250
- Mahabadi A, Zahedi S, Khonsari A (2013) Reliable energy-aware application mapping and voltage-frequency island partitioning for gals-based noc. *J Comput Syst Sci* 79(4):457–474
- Meng H, Freeman M, Pears N, Bailey C (2008) Real-time human action recognition on an embedded, reconfigurable video processing architecture. *J Real-Time Image Process* 3(3):163–176
- Miao D, Liu L, Xu R, Panneerselvam J, Wu Y, Xu W (2018) An efficient indexing model for the fog layer of industrial internet of things. *IEEE Trans Ind Inform* 14(10):4487–4496
- Mirjalili S, Lewis A (2016) The whale optimization algorithm. *Advances in engineering software* 95:51–67
- Nguyen TKH (2015) Low power architecture for fall detection system. PhD thesis, Université Nice Sophia Antipolis
- Ogras UY, Marculescu R, Marculescu D, Jung EG (2009) Design and management of voltage-frequency island partitioned networks-on-chip. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 17(3):330–341
- Olafsson S (1995) A general model for task distribution on an open heterogenous processor system. *IEEE Trans Syst Man Cybern* 25(1):43–58
- Pagani S, Chen JJ, Li M (2014) Energy efficiency on multi-core architectures with multiple voltage islands. *IEEE Trans Parallel Distrib Syst* 26(6):1608–1621
- Rault T, Bouabdallah A, Challal Y (2014) Energy efficiency in wireless sensor networks: a top-down survey. *Comput Netw* 67:104–122
- Safaei A, Wu QJ, Yang Y (2018) System-on-a-chip (soc)-based hardware acceleration for foreground and background identification. *J Franklin Inst* 355(4):1888–1912
- Saponara S, Fanucci L, Petri E (2013) A multi-processor noc-based architecture for real-time image/video enhancement. *J Real-Time Image Process* 8(1):111–125
- Sasagawa Y, Mori A (2016) High-level video analytics pc subsystem using soc with heterogeneous multicore architecture. *IEEE J Solid-State Circuits* 51(4):1051–1059
- Shin D, Kim W, Kwon S, Han TH (2011) Communication-aware vfi partitioning for gals-based networks-on-chip. *Design Autom Embed Syst* 15(2):89–109
- Shin D, Kim J (2003) 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, pp 408–413
- Tariq UU, Wu H (2016) Energy-aware scheduling of conditional task graphs with deadlines on mpsoCs. In: *Computer Design (ICCD)*, 2016 IEEE 34th International Conference on, IEEE, pp 265–272
- Tariq UU, Wu H (2017) Energy-aware scheduling of periodic conditional task graphs on mpsoCs. In: *Proceedings of the 18th International Conference on Distributed Computing and Networking*, ACM, p 13
- Tariq UU, Wu H, Abd Ishak S (2018) Energy-aware scheduling of conditional task graphs on noc-based mpsoCs. In: *Proceedings of the 51st Hawaii International Conference on System Sciences*
- Tariq UU, Ali H, Liu L, Panneerselvam J, Zhai X (2019) Energy-efficient static task scheduling on vfi-based noc-hmpsoCs for intelligent edge devices in cyber-physical systems. *ACM Trans Intell Syst Technol* 10(6):1–22
- Ullah Tariq U, Ali H, Liu L, Panneerselvam J, Zhai X (2019) 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*
- Vafae F, Nelson PC (2010) An explorative and exploitative mutation scheme. In: *IEEE Congress on Evolutionary Computation*, IEEE, pp 1–8
- Wang Y, Liu H, Liu D, Qin Z, Shao Z, Sha EHM (2011) Overhead-aware energy optimization for real-time streaming applications on multiprocessor system-on-chip. *ACM Trans Design Autom Electron Syst* 16(2):14
- Wang Y, Liu D, Qin Z, Shao Z (2013) Optimally removing intercore communication overhead for streaming applications on mpsoCs. *IEEE Trans Comput* 62(2):336–350
- Wang Y, Shao Z, Chan HC, Liu D, Guan Y (2014) Memory-aware task scheduling with communication overhead minimization for streaming applications on bus-based multiprocessor system-on-chips. *IEEE Trans Parallel Distrib Syst* 25(7):1797–1807
- Yan L, Renfa L, Cheng X, Fei Y (2009) Hw-sw framework for multimedia applications on mpsoC: practice and experience. *J Comput* 4(3):238–244
- Yu Z, Wang W (2019) Learning dalts for cross-modal retrieval. *CAAI Trans Intell Technol* 4(1):9–16
- Zhai X, Eslami M, Hussein ES, Filali MS, Shalaby ST, Amira A, Bensaali F, Dakua S, Abinahed J, Al-Ansari A et al (2018) Real-time automated image segmentation technique for cerebral aneurysm on reconfigurable system-on-chip. *J Comput Sci* 27:35–45
- Zhao Z, Min G, Gao W, Wu Y, Duan H, Ni Q (2018) Deploying edge computing nodes for large-scale iot: a diversity aware approach. *IEEE Internet Things J* 5(5):3606–3614

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.