Multiprocessor System-on-Chips based Wireless Sensor Network Energy Optimization

Haider Ali

College of Engineering and Technology
University of Derby, Derby
United Kingdom

A thesis in fulfillment of the requirements for the degree of Ph.D in Computer Science and Engineering

October 2020

Supervised By: Dr. John Panneerselvam and Prof. Yong Xue

Declaration

The study outlined in this dissertation were carried out in the Department of Engineering and Technology at University of Derby, under the supervision of Dr. John Panneerselvam and Professor Yong Xue. This is to declare that the work stated in this thesis was done by the author, and no part of the thesis has been submitted in a thesis form to any other university or similar institution. No human or animal participation have been included in this research and the research presented in this thesis has been ethically approved. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others. Parts of this thesis have previously appeared in the papers listed in the list of publications.

Haider Ali
Department of Electronics, Computing and Mathematics
University of Derby
Derby, United Kingdom
January 2020

Acknowledgement

First of all, I thank **Allah** Almighty for His countless blessings. It was through His kindness that I was able to overcome all the challenges that I faced during my research.

I would like to thank my supervisors **Dr. John Panneerselvam** and **Professor Yong Xue**: it is difficult to express my gratitude to them for giving me the opportunity to join the research group at the University of Derby and supporting me in my research pursuits.

I express my deepest appreciation and gratitude to **Professor Lu Liu**. It would have never been possible for me to grow as a researcher without the continuous and extensive support of **Professor Lu Liu**.

I would also like to thank my dearest friend **Dr. Umair Ullah Tariq** for his help, valuable comments and constructive feedback on my research. I also appreciate the support from **Dr. Xiaojun Zhai** and **Dr. Yongjun Zheng**.

I would immensely thank to **Dr. Muhammad Kazim** who acted as a huge inspiration and spent quality time with me during my PhD. I also want to extend my gratitude to my friends specially **Mr. Shahroz Nadeem** and colleagues within the college of Engineering and Technology at the University of Derby.

I extend my profound gratitude to my parents **Mr. Akbar Khan** and **Mrs. Zainab Bibi** for their prayers, love and constant support throughout my PhD. Without their support, my dream of getting PhD was not possible.

Last but not least, I am indebted to my wife **Jiya Haider** and my three beautiful children **Umer Haider**, **Ebad Haider**, **Arsh Haider**, their continuous love and encouragement throughout my journey has been indispensable.

Abstract

Wireless Sensor Network (WSN) is an integrated part of the Internet-of-Things (IoT) used to monitor the physical or environmental conditions without human intervention. In WSN one of the major challenges is energy consumption reduction both at the sensor nodes and network levels. High energy consumption not only causes an increased carbon footprint but also limits the lifetime (LT) of the network. Network-on-Chip (NoC) based Multiprocessor System-on-Chips (MPSoCs) are becoming the de-facto computing platform for computationally extensive real-time applications in IoT due to their high performance and exceptional quality-of-service. In this thesis a task scheduling problem is investigated using MPSoCs architecture for tasks with precedence and deadline constraints in order to minimize the processing energy consumption while guaranteeing the timing constraints. Moreover, energy-aware nodes clustering is also performed to reduce the transmission energy consumption of the sensor nodes. Three distinct problems for energy optimization are investigated given as follows:

First, a contention-aware energy-efficient static scheduling using NoC based heterogeneous MPSoC is performed for real-time tasks with an individual deadline and precedence constraints. An offline meta-heuristic based contention-aware energy-efficient task scheduling is developed that performs task ordering, mapping, and voltage assignment in an integrated manner. Compared to state-of-the-art scheduling our proposed algorithm significantly improves the energy-efficiency.

Second, an energy-aware scheduling is investigated for a set of tasks with precedence constraints deploying Voltage Frequency Island (VFI) based heterogeneous NoC-MPSoCs. A novel population based algorithm called ARSH-FATI is developed that can dynamically switch between explorative and exploitative search modes at run-time. ARSH-FATI performance is superior to the existing task schedulers developed for homogeneous VFI-NoC-MPSoCs.

Third, the transmission energy consumption of the sensor nodes in WSN is reduced by developing ARSH-FATI based Cluster Head Selection (ARSH-FATI-CHS) algorithm integrated with a heuristic called Novel Ranked Based Clustering (NRC). In cluster formation parameters such as residual energy, distance parameters, and workload on CHs are considered to improve LT of the network. The results prove that ARSH-FATI-CHS outperforms other state-of-the-art clustering algorithms in terms of LT.

List of Publications

J = Journal Paper; **C** = Conference Proceeding; **B** = Book Chapter

J1: **Haider Ali*,** Umair Ullah Tariq*, Lu Liu, John Panneerselvam, and Xiaojun 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)** 10, no. 6 (2019): 66. ** Umair Ullah Tariq, Haider Ali contributed equally to this research.*

J2: **Haider Ali,** Umair Ullah Tariq, Yongjun Zheng, Xiaojun Zhai, and Lu Liu."Contention Energy-Aware Real-Time Task Mapping on NoC Based Heterogeneous MPSoCs." **IEEE Access** 6 (2018): 75110-75123.

J3: Lei-Lei Shi, Lu Liu, Yan Wu, Liang Jiang, Muhammad Kazim, **Haider Ali,** and John Panneerselvam. "Human-Centric Cyber Social Computing Model for Hot-Event Detection and Propagation." **IEEE Transactions on Computational Social Systems (2019)**.

J4: Umair Ullah Tariq, **Haider Ali,** Lu Liu· John Panneerselvam , Xiaojun Zhai. "A Novel Meta-heuristic based Energy-aware Computing in Edge-devices using VFINoC-HMPSoC for Streaming Applications" Submitted to S.I. : IoT for Everyday Living: Ubiquitous Intelligence at Scale in **Journal of Ambient Intelligence and Humanized Computing (AIHC)**.

J5: **Haider Ali,** Umair Ullah Tariq, John Panneerselvam, Mubashir Hussain, Lu Liu. "ARSH-FATI a Novel Meta-heuristic for Cluster Head Selection in Wireless Sensor Networks" Accepted in **IEEE Systems**.

J6: Umair Ullah Tariq, **Haider Ali,** Muhammad Kazim, Waqar Ahmed. "Energy-aware Scheduling of Streaming Applications in IoT based Healthcare using VFI-NoC-HMPSoC as an Edge-device" Submitted in **Future Gen ration Computer Systems Journal**.

C1: **Haider Ali,** Xiaojun Zhai, Umair Ullah Tariq, and Lu 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), pp. 1099-1104. IEEE, 2018.

C2: **Haider Ali,** Umair Ullah Tariq, Xiaojun Zhai, and Lu Liu. "Energy Efficient Task Mapping Scheduling on Heterogeneous NoC-MPSoCs in IoT Based Smart City." In 2018 IEEE 20^{th} International Conference on High Performance Computing and Communications; IEEE 16^{th} International Conference on Smart City; IEEE 4^{th} International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1305-1313. IEEE, 2018.

C3: Hengjian Wang, John Pannereselvam, Lu Liu, Yao Lu, Xiaojun Zhai, and **Haider Ali**. "Cloud Workload Analytics for Real-Time Prediction of User Request Patterns." In 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16^{th} International Conference on Smart City; IEEE 4^{th} International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1677-1684. IEEE, 2018.

C4: **Haider Ali**, Umair Ullah Tariq, Lu Liu, John Panneerselvam, and Xiaojun 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), pp. 1297-1304. IEEE, 2019.

C5: Umair Ullah Tariq, **Haider** Ali, Lu Liu, and Xiaojun Zhai. "A Novel Meta-Heuristic for Green Computing on VFI-NoC-HMPSoCs." 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), pp. 1545-1552. IEEE, 2019.

B1: Yongjun Zheng, **Haider Ali,** and Umair Ullah Tariq. "Big Data Security in Internet of Things". Pulished in Security and Privacy for Big Data, Cloud Computing and Applications (Computing and Networks), IET, UK.

List of Abbreviations

ACO	Ant Colony Optimization
ARSH-FATI	Algorithm named after my daughter and niece
ARSH-FATI-CHS	ARSH- FATI based Cluster Head Selection
ATR	Automatic Target Recognition
CATMES	Contention and Energy-aware Task Mapping and Edge Scheduling
CH	Cluster Head
CITM-VA	Contention-aware Integrated Task Mapping and Voltage Assignment
CMOS	Complementary Metal Oxide Semi-Conduction
DAG	Directed Acyclic Graph
DE	Differential Evolution
DPM	Dynamic Power Management
DR	Dimensional Rate
DVFS	Dynamic Voltage and Frequency Scaling
EDF	Earliest Deadline First
EECDF	Edge Consistent Deadline First
EG	Energy Gradient
ELFTF	Earlier Latest Finish Time First
E3S	Embedded Systems Synthesis
FFT	Fast Fourier Transform
FND	First Node Death
FoV	Field-of-View
GA	Genetic Algorithm
GALS	Globally Asynchronous Locally Synchronous
IC	Integrated Circuit
ICT	Information and Communication Technology
IDCT	Inverse Discrete Cosine Transform
IFFT	Inverse Fast Fourier Transform

ILP	Integer Linear Programming
IoT	Internet-of-Things
IP	Intellectual property
ISA	Instruction Set Architecture
ITRS	International Technology Road-map for Semiconductors
LEACH	Low-Energy Adaptive Clustering Hierarchy
LFT	Latest Finish Time
LP	Linear Program
LT	Lifetime
MILP	Mixed Integer Linear Programming
MPSoC	Multiprocessor System-on-Chip
NLP	Non-Linear Programming
NoC	Network-on-Chip
NRC	Novel Ranked based Clustering
PEs	Processing Elements
PSO	Particle Swarm Optimization
PSO-ECHS	PSO based Energy-efficient CH Selection
QoS	Quality-of-Service
SA	Simulated Annealing
SC	Smart City
SN	Sensor Node
SoC	System on Chip
VFI	Voltage Frequency Island
VCT	Virtual cut-through
VLCs	Voltage Level Converters
WH	Wormhole
WSN	Wireless Sensor Network

Contents

Abstract **i**

1 Introduction **1**

1.1 Internet-of-Things (IoT) 1

1.2 Wireless Sensor Network (WSN) 4

1.3 MPSoCs in IoT 6

1.3.1 Multimedia Surveillance 6

1.3.2 Healthcare and Automated Assistance 8

1.3.3 Environment Monitoring 8

1.3.4 Industrial Applications 8

1.4 Challenges 10

1.4.1 Research Approach 10

1.5 Aim and Objectives 13

1.6 Contributions 13

1.7 Thesis Organization 15

2 Background Study **17**

2.1 Sensor Nodes Architectures 17

2.1.1 Architecture 19

2.1.1.1 Homogeneous MPSoCs 20

2.1.1.2 Heterogeneous MPSoCs 20
2.1.2 Interconnect 22
2.1.2.1 Bus 22
2.1.2.2 NoC 23
2.1.2.3 Bus based MPSoCs 27
2.1.2.4 NoC based MPSoCs 27
2.1.2.5 Voltage Frequency Islands based MPSoCs 28
2.2 Task Scheduling 29
2.2.1 System Level Energy Management Techniques 34
2.2.2 Other Energy Reduction Techniques 38
2.3 Summary 43

3 Literature Review **44**

3.1 Scheduling using NoC-MPSoCs 44
3.1.1 Computational Energy Reduction 44
3.1.2 Inter-processor Communication Energy Reduction 46
3.1.3 Total Energy Savings 47
3.2 Scheduling using VFI-NoC-MPSoC 48
3.2.1 Independent Task Scheduling 49
3.2.2 Single-processor Per VFI 50
3.2.3 VFI based MPSoC 50
3.3 Sensor Nodes Clustering 52
3.4 Summary 55

4 Research Methodology **57**

4.1 Preliminaries . . . 57

4.1.1 Extended Graph . . . 57

4.1.2 Multiprocessor Computing Platforms . . . 58

4.1.2.1 Non-VFI based NoC-MPSoC . . . 59

4.1.2.2 VFI based NoC-MPSoC . . . 62

4.1.3 Wireless Sensor Network . . . 65

4.2 Research Method . . . 67

4.2.1 Scheduling . . . 67

4.2.1.1 Task Mapping . . . 68

4.2.1.2 Task Ordering . . . 68

4.2.1.3 Voltage Scaling . . . 68

4.2.2 Clustering . . . 69

4.3 Experimental Setup and Data Collection . . . 69

4.3.1 Experimental Setup . . . 69

4.3.1.1 Node Level Experimental Setup . . . 70

4.3.1.2 Network Level Experimental Setup . . . 71

4.3.2 Data Collection . . . 71

4.4 Summary . . . 73

5 Energy-aware Static Task scheduling on Heterogeneous NoC-MPSoCs **74**

5.1 Contention and Energy-aware Approach . . . 76

5.2 Results and Discussions . . . 83

5.2.1 CITM-VA Energy performance . . . 83

5.2.1.1 5×4 NoC . . . 84

5.2.1.2 6×4 NoC . . . 86

5.2.1.3 7×4 NoC 86
5.2.1.4 Robustness and QoS 87
5.2.1.5 Energy and MM 88
5.3 Summary 89

6 Energy-aware Static Task Scheduling on Heterogeneous VFI-NoC-MPSoCs 90
6.1 Static Contention-aware Energy-efficient Scheduling 92
6.1.1 Earliest Edge Consistent Deadline First ($EECDF$) Algorithm 95
6.1.1.1 Energy gradient descent (EGD) 99
6.2 Experimental Results and Discussions 101
6.2.1 Scenario 1 103
6.2.2 Scenario 2 105
6.2.3 Scenario 3 106
6.2.4 Scenario 4 108
6.3 Summary 109

7 Energy-aware Clustering for Enhancing Wireless Sensor Network Lifetime 111
7.1 ARSH-FATI based Cluster Head Selection 112
7.1.1 ARSH-FATI-CHS 114
7.1.2 Cluster Formation 117
7.2 Experimental Results and Discussions 119
7.2.1 Scenario 1(NoR) 120
7.2.2 Scenario 2(NoANs) 121
7.2.3 Scenario 3(IoBSL and IoNCHs) 123
7.3 Summary 125

8 Conclusion and Future Work **127**

8.1 Conclusion 127

8.2 Future Work 132

8.3 Summary 134

List of Tables

1.1 Smart city services 4

2.1 MPSoCs architectures used in smart-phones 21

2.2 Heterogeneous MPSoCs for IoT based applications 22

2.3 Different DVFS-enabled NoC based MPSoCs from Tilera^{TM} 28

2.4 Two different processors power consumption modes 36

2.5 The 70 nm processor technology parameter values 37

2.6 Transmeta crusoe processor power consumption at different supply voltage levels 37

4.1 Operating frequency and power consumption of type 1 and type 2 processors . . 70

4.2 Processors power consumption modes 70

4.3 Various parameters used in simulation 71

5.1 Terms and notations 77

5.2 Real benchmarks description 83

5.3 Real benchmarks dynamic energy consumption in Joule (J) 84

5.4 Real benchmarks dynamic+static energy consumption in Joule (J) 84

5.5 Real benchmarks dynamic+static energy in joule (J) at $0.9 \times (makespan)_{ECM}$. 85

5.6 Energy consumption (J) at different MM values using 28 tiles 85

6.1 List of parameters used in results 102

6.2 Real benchmarks energy consumption in joule (J) at $NVFI = 4$ and $PPI = 2 \times 2$ 103

6.3 ARSH-FATI energy performance summary 109

7.1 List of abbreviations used in results 120

7.2 ARSH-FATI-ECHS performance improvement comparison 125

List of Figures

1.1 IoT connecting different devices and users for various applications 2

1.2 Wireless sensor network structure 5

1.3 MPSoCs applications in IoT 7

2.1 MPSoC systems categorization 19

2.2 Homogeneous MPSoCs (a) shared memory (b) distributed memory 20

2.3 Generic heterogeneous MPSoC architecture 21

2.4 Bus communication architecture 23

2.5 Bus matrix communication architecture 23

2.6 A typical 2D-mesh NoC architecture 24

2.7 Xilinx Zynq Ultrascale+MPSoC using ARM AMBA-AXI4 27

2.8 A typical representation of NoC based MPSoC architecture 28

2.9 A generic representation of VFI based NoC-MPSoC architecture 29

2.10 Directed acyclic graph 32

2.11 Task scheduling algorithms 33

2.12 Coarse-grained software pipelining (a) a simple DAG with three task nodes representing a workload (b) tasks mapping without re-timing (c) task mapping with re-timing 38

2.13 Sensor nodes clustering 41

4.1 Directed acyclic graph 58

4.2 Extended graph 58
4.3 2D-mesh topology based NoC-MPSoC 60
4.4 VFI based heterogeneous NoC-MPSoC architecture 63
4.5 2D-mesh topology NoC interconnect 64

5.1 5×4 NoC containing 20 tiles 85
5.2 6×4 NoC containing 24 tiles 86
5.3 7×4 NoC containing 28 tiles 87
5.4 Robustness 88
5.5 Energy and makespan multiplier relation 88

6.1 Dimensional rate parameter variations 102
6.2 Energy consumption at $NVFI = 4$ and $PPI = 2 \times 2$ 104
6.3 Energy consumption using $NVFI = 4$ at different PPI 105
6.4 Set-1 energy consumption using $NVFI = 4$ and $PPI = 2 \times 2$ 106
6.5 Set-2 energy consumption using $NVFI = 4$ and $PPI = 2 \times 2$ 107
6.6 CCR impact on ARSH-FATI at $VFI = 4$ and $PPI = 2 \times 2$ 109

7.1 Performance analysis in terms of NoR after FND using different NSNs 122
7.2 NoAN using different NSNs and NCHs = 25% at BS = (50, 50) 123
7.3 Impact of base station and cluster heads on LT of the network 125

Chapter 1

Introduction

In this chapter we briefly introduce Internet-of-Things (IoT) and Wireless Sensor Network (WSN). We further explain the applications of Multiprocessor Systems-on-Chips (MPSoCs) in IoT and introduce the context of energy-efficiency in WSNs. We also list the objectives and contributions of this research. The organization of this thesis is presented at the end of this chapter.

1.1 Internet-of-Things (IoT)

IoT is a technological communication revolution that bridges a plethora of modern digital devices, users, and smart things to the Internet for numerous applications. Thus, IoT is transforming the Internet into a more pervasive and immersive model [1, 2]. The literature demonstrates that the emergence of IoT has initiated Smart City (SC) concept, a paradigm that particularly concentrates on reconciling and enhancing both ecology and economy of city modernization [3, 4]. An ultimate goal of the IoT technology for the SC is to optimize and efficiently control the city systems. More precisely the fundamental aim is to boost the effectiveness of city governance by establishing a communicating link between the human users and smart technology. IoT is gaining popularity for smart cities in order to develop efficient and low-cost applications for purposes such as monitoring, control and automation.

Figure 1.1: IoT connecting different devices and users for various applications

In IoT based services, the information is collected from the users and/or smart things via wearable devices, sensors, and cameras in Figure 1.1, these are represented by red, blue, and green colours respectively. These digital devices gathering information from a Field-of-View (FoV) are commonly known as Sensor Nodes (SNs). A process called clustering is performed on the SNs for achieving better energy efficiency, where several SNs are grouped into one cluster and one specific sensor node often referred as Cluster Head (CH) is selected. The CH gathers the data from each node within the cluster and performs further processing to compress the data and then transmits the aggregated data wirelessly to the Base Station (BS) [5, 6]. The BS is usually a conventional computer with powerful processing capabilities and provides a platform where the data can be displayed to the professionals and/or other users through a user interface [6, 7]. The data gathered in BS is transferred, processed, and stored in the cloud for further post processing to enable visualization and recommendations to be made, as shown in Figure 1.1 [8, 9]. All the data from different CHs is accumulated in the cloud. The cloud provides a massive data/information storage and processing infrastructure [10, 11]. The cloud promises high scalability, reliability, speed, performance, autonomy, and low-cost for the IoT applications. In simple words cloud delivers computing services, storage, networking, databases, software, intelligence, and analytics. The cloud is an essential part of the IoT system [12] and the collected data arrives in big amounts often reaches at real-time. Amazon EC2, Microsoft Azure, Google App Engine (GAE), Nimbus, IBM Blue Cloud, and 3Tera are some of the

examples of cloud computing systems [10].

There are three types of cloud (1) private, (2) public, and (3) hybrid explained as follows:

1. **Private Cloud:** The private cloud is defined as a computing service offered either over the Internet or a private internal network and to selected users only instead of the general public. The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise. Processes and data are managed within the same organization. Some of the benefits of a private cloud service include improved insurance, unlimited bandwidth, and low risk of security issues [13].

2. **Public Cloud:** A public cloud is a type of computing in which a service provider makes resources available to the public via the internet. Resources may include storage capabilities, applications or virtual machines. The cloud infrastructure is available to the general public and/or a large industry group and is owned by an organization providing cloud services. It provides services to each user using the same infrastructure. Basically the users pay for the resources they use. Some benefits of utilizing public cloud service include simple scalability, high reliability, and cost potency [13, 14].

3. **Hybrid Cloud:**A hybrid cloud is a combination of a private cloud combined with the use of public cloud services where one or several touch points exist between the environments. The goal is to combine services and data from a variety of cloud models to create a unified, automated, and well-managed computing environment. Hybrid clouds are more complex than the other deployment models, since they involve a composition of two or more clouds (private, community, or public). By allowing workloads/jobs to move between the public and private clouds as computing costs and needs change subsequently, hybrid cloud can give businesses more data deployment options and greater flexibility [13, 15].

Italy is one of the first countries to offer large scale smart services [16]. Recently in China, a boom in IoT based services occurred as listed in Table 1.1 to promote green, low-carbon, harmonious, and a sustainable development for 1.3 billion people [17]. IoT is opening new opportuni-

Table 1.1: Smart city services

Urban Function	Smart Applications
Reproduction	Public safety, environment, energy, healthcare, household, and urban management.
Economic Development	Manufacturing, industry, logistics, and city planning.
Social Interactions	Public transportation, online shopping, and general social management.
Culture Enjoyment	Education, tourism, and outdoor stream media.

ties to develop efficient, reliable, and low-cost applications that aim to enhance Quality-of-Life (QoL) in the cities [18]. As a result, various Information and Communication Technology (ICT) companies including Cisco, Samsung, IBM, HP, and Google are launching and promoting IoT based SC initiatives [19]. These initiatives encompass different domains such as environmental monitoring, utility monitoring, public transportation scheduling, advanced healthcare, incident reporting, and surveillance [16, 20]. A report on the IoT Market (2011-2018) indicates that the global market exceeded US$ 1 trillion in 2017 and is anticipated to reach US$ 1,266 billion by 2019 [21].

1.2 Wireless Sensor Network (WSN)

WSN shown in Figure 1.2, is an integrated part of the IoT [22, 23] and provides information about the physical world collected by dedicated sensors/cameras to the cloud from the Base Station (BS). Thus, WSN is a vital resource necessary to implement the vision of IoT paradigm [24–26]. In simple terms, WSN is an ad hoc network composed of resource-constrained digital devices known as SNs that are used to collect information from a FoV [27, 28]. Advancements in modern technologies have had a significant positive impact on the availability of high-quality SN for numerous applications with superior technical features, low cost, minimal power consumption, and small physical size. The technological improvement in SN technology for multimedia data has enhanced the growth in WSN [29, 30]. Multimedia content such as audio and video streams processing a promising technology utilized by numerous applications [31].

A typical SN indicated (highlighted by blue colour in Figure 1.2) used in a network contains four logical component blocks [6] each component is explained as follows:

1. **Sensing Unit:** Single or multiple sensors and Analogue to Digital Converters (ADC). The physical information is detected and captured in analogue form which the ADC converts to a digital format as required for the processing unit.

2. **Processing Unit:** Responsible for intelligent processing of the data received from the target area, this is a microprocessor and/or microcontroller with integrated memory. Digital Signal Processor (DSPr) and Application Specific Instruction-set Processor (ASIP) could also be component parts of the processing unit.

3. **Communication Unit:** Short-range transceiver system commonly based on standards such as IEEE 802.14.3, IEEE 802.15.4 or ZigBee™ although other protocols e.g. IEEE 802.1 can also be utilised for specialised applications such as Industrial IoT (IIoT).

4. **Power Source:** Regulated supply for the data collection, processing, and transmission subsystems. Batteries with limited residual energy are often deployed.

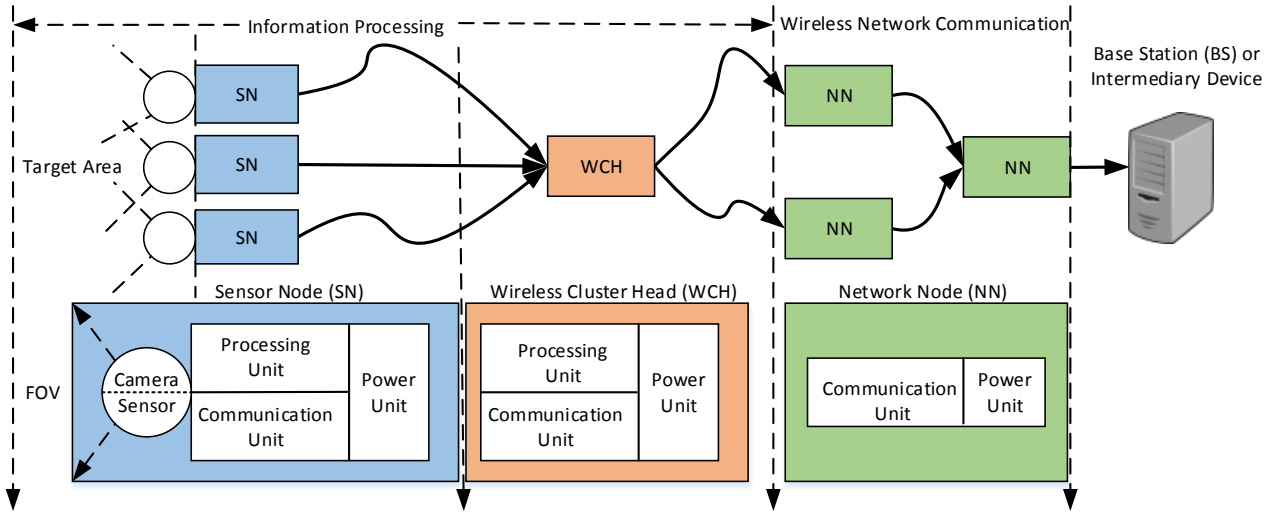


Figure 1.2: Wireless sensor network structure

The continuously expanding need for computationally extensive real-time applications has influenced the growth in the usage of Multiprocessor-System-on-Chips (MPSoCs) in modern embedded systems [32]. MPSoCs provide high performance, exceptional Quality-of-Service (QoS),

and overwhelming reliability [33]. These qualities have contributed to the wide deployment of MPSoC as SNs for numerous real-time IoT applications. Some of the real-time applications involve multimedia content which is computationally extensive operations [34]. Subsequently, the number of processors are growing on the MPSoC for example Tilera Tile64 MPSoC contains 64 processors. According to International Technology Roadmap for Semiconductors (ITRS) there will be hundreds of processors in the MPSoC by 2025 [35] therefore, traditional bus based communications on chip architecture will become bottle neck due to the poor scalability and limited band width. NoC based communication has several advantages over hierarchical (STBus, Advance Microcontroller and Bus Architecture) as well as traditional bus architecture in terms of scalability, flexibility and performance [35].

The BS is responsible to collect the information from NNs. BS is usually a conventional computer with powerful processing capabilities i.e. multiprocessors. The BS provides WAN connectivity and data logging. The base station connects to database replicas across the internet. Finally, the data is displayed to the professionals and/or other users through a user interface [6, 7]. The BS has an unlimited power supply, while the SNs are mostly battery operated digital devices with limited residual energy. In modern WSNs edge-computing is deployed before transferring all the data to BS to reduce load on the BS [36].

1.3 MPSoCs in IoT

The MPSoCs have become a de-facto computing platform and they can be used in various computationally extensive real-time applications as shown in Figure 1.3. Few examples are discussed as follows:

1.3.1 Multimedia Surveillance

MPSoCs integrated with video and audio sensors are used for target detection and tracking, border protection, public event monitoring [37–39], video/image enhancement [40], person

Figure 1.3: MPSoCs applications in IoT

tracking [41], people and object identification [42, 43]. These multimedia surveillance applications involve compression/decompression, encoding/decoding and different conversions techniques which are computationally extensive operations. A few highly complex MPSoCs such as Xetal-I (128 processors) [34] and Xetal-II (320 processors) [44] are also used for surveillance.

Video surveillance to monitor on-road situation is a prime example of the multimedia application in IoT. In this video streaming of on-road situation, the information is collected regarding vehicles positions, traffic jams and road accident severity. In the video streaming the multimedia data content is streamed over the network in an encoded form while the video is displayed to the end user and/or professional either in a recorded or pre-recorded manner. In IoT based applications video streams are usually compressed to reduce the video size and achieve better load balancing in the video communications. The MPEG-encoder is executed numerous times for the whole video stream [45].

1.3.2 Healthcare and Automated Assistance

Multiprocessor systems are widely adopted by remote medical centres for various advanced healthcare related applications such as patient monitoring [46], drug administration, diagnostics [47,48], human gait analysis [49], care assistance, and motoring [50]. These servces using MPSoC help to reduce the frequency of patient's visit to the hospitals while enhancing the Quality-of-Life (QoL). STMicroelectronics MPSoC is one of the popular multiprocessor platform that is widely adopted in advanced healthcare [46].

1.3.3 Environment Monitoring

MPSoCs are also used in applications such as animal and bird tracking as well as condition monitoring for irrigation, livestock, crops, and air pollution [44, 51]. Monitoring systems are vital in time-critical applications, such as wild fire containment, flood detection, and disaster management [52, 53]. High-performance computing platforms provided by MPSoC produce smart technology to monitor and detect natural and anthropogenic emergencies.

1.3.4 Industrial Applications

MPSoCs are deployed to extract and analyze information regarding civil structures e.g. nuclear power plants, pipelines, and large bridges especially during and after earthquakes, high winds, or environmental changes [54]. In industry MPSoCs are used for automation and manufacturing process control for example, Xilinx Zynq® UltraScale™ are deployed in robots for operations such as supervision, control, and automation that increase repeatability while reducing human efforts while maintaining continuous operation [55, 56].

Among the applications of MPSoCs in IoT, the most popular is surveillance where video analytics is performed for different purposes. Video analytics also called Video Content Analysis (VCA), it involves different techniques to monitor, extract and analyze the information from video streams [57]. Closed-circuit television (CCTV) cameras are the main contributors of com-

puterized video analysis. In video analytics a key challenge is the size of the video data. In one second of high-definition video there is approximately equal to 2000 pages of text. Video analytics in IoT is widely used in the recent years for surveillance and automated security. Automated surveillance systems are cost effective, cheaper, and remain focused as compared to labor-based surveillance systems. Video analytics can be used for human recognition, face recognition, object detection, recognizing suspicious activities, and detecting breach of restricted zones [58]. In terms of the IoT architecture two known approaches can be adopted for performing video analytics, namely (1) server-based architecture and (2) edge-based architecture. Each architecture is explained as follows: [59].

Server-based Architecture: In this approach, captured video using cameras is transmitted to the centralized and dedicated server where video analytics is performed. The generated video is usually compressed to reduce the frame rates or the image resolution due to limited bandwidth availability. In this configuration the compressing may result the loss of information which can adversely affect analysis overall accuracy. However, the server-based approach facilitates easier maintenance.

Edge-based Architecture: In this configuration, analytics are applied at the SN level or 'edge' of the system. In other words video analytics is performed on the raw data gathered from the camera in the SNs. In this approach the entire content/data of the video stream remains available for the video analysis. Therefore, no loss of information occurs and enables efficient and effective content analysis. However, edge-based systems are more costly to maintain and posses lower processing power capability compared to server-based systems.

Briefly, server-based approach is easier to be implemented and maintained while edge-based system is costly though the entire video stream is available for performing video analytics on gathered data.

1.4 Challenges

One of the major technological challenges for IoT is energy consumption optimization at the SN and network level. High energy consumption of embedded systems at SN level not only reduces the lifetime of the network but also results in an increased carbon footprint. Approximately 4.7% of global electrical energy is consumed by Information and Communication Technology (ICT) and results in the releases of 2.0% of overall atmosphere carbon footprint [60]. Energy savings and green computing are therefore highly important. Moreover, SN mostly operates on embedded battery sources with limited residual energy, replacement of the batteries is usually challenging, difficult, and expensive. Consequently, there is high demand for energy-efficient techniques for reducing the energy consumption [61, 62].

1.4.1 Research Approach

In this thesis we perform task scheduling to reduce the processing energy consumption considering NoC-MPSoC and VFI-NoC-MPSoC architectures. Then we implement clustering to minimize the transmission energy consumption.

Scheduling used for real-time applications is an effective energy management mechanism when combined with Dynamic Voltage and Frequency Scaling (DVFS). This reduces the processing energy consumption of the MPSoC computing architectures. Task scheduling is a process of properly allocating an application containing a set of tasks on the processors such that specific obligations are fulfilled e.g. energy consumption optimization and/or execution time reduction. Proper task mapping and scheduling approach can drastically influence an embedded system's performance and reliability [63]. DVFS is a useful technique whereby the processor supply voltage and clock frequency are dynamically reduced as a means to reduce overall power consumption without negatively impacting performance and deadline completion [64,65]. Modern MPSoCs includes DVFS-enabled processors. As an example, the Samsung Mongoose M2 has 4, DVFS-enabled homogeneous processors with an operating frequency range of 2.3 GHz to 2.8 GHz. Reducing the supply voltage can significantly decrease the energy consumption in

MPSoCs because voltage has a quadratic law relationship with energy consumption [66]. The DVFS not only reduces dynamic power, P_D but also minimizes static power, P_S, since it is also dependent on V_{dd}. Thus DVFS is an efficient technique to reduce the total power (dynamic and static power) consumption.

Though DVFS is an effective technique to reduce the energy consumption of the tasks scheduled on MPSoCs however, there are other parameters that may affect the overall energy-efficiency. For example, task ordering plays significant role for achieving higher energy savings. If tasks with longer deadline are blocked by tasks with shorter deadline then DVFS may not have enough slack available to reduce the energy consumption. Similarly, in heterogeneous MPSoCs there are not only processors with different voltage scaling capabilities but also they contain processors with distinctive energy performance profiles. Therefore, mapping the tasks on heterogeneous influences the capability of DVFS to minimize the total energy consumption. If the tasks are not prioritized to be mapped on high energy-efficient processor such that the deadline is not missed then the overall energy-efficiency may be adversely affected. Moreover, energy consumption is also affected by the communication overhead due to the energy consumed on NoC links and router for inter-processor communications. Thus, our motivation for reducing the energy consumption of the MPSoC computing platform is to consider energy performance profiles of the processors, communication overhead, task ordering, and voltage scaling.

The total energy consumption of a network is the combination of sensing, processing and communication energy dissipation given as follows:

$$E_{network} = E_{sensing} + E_{processing} + E_{communication} \tag{1.1}$$

The sensing energy consumption reduction is performed at the fabrication level which is not within the scope of our thesis. However, the communication/transmission energy can be reduced by formulating it as a scheduling problem. The communication unit often consumes higher energy compared to the sensing and processing units [67,68]. Therefore, it is also important to minimize the communication energy consumption apart from reducing the processing energy.

The transmission energy of the communication unit for l, bits is given as follows:

$$E_{TX}(l,d) = \begin{cases} l \times E_{elec} + l \times e_{fs} \times d^2 & d \leq d_o \\ l \times E_{elec} + l \times e_{mp} \times d^4 & d \geq d_o \end{cases} \tag{1.2}$$

where E_{elec} is the energy consumed per bit for running the transmitter circuit, e_{fs} and e_{mp} are the amplification energy for free space model and multi-path model respectively while d_o shows the threshold transmission distance.

Equation 1.2 shows that the transmission energy of the communication unit increases significantly when it transmits the data to the BS beyond its threshold transmission distance. Clustering is an effective technique deployed to increase the energy-efficiency of the network [69]. Clustering not only provides data aggregation, scalability, and bandwidth conservation but also prolongs the network Lifetime (LT) by decreasing the communication energy consumption of the SNs. In clustering process, the SNs are partitioned into groups called clusters. Where each cluster has its own leader known as CH. The CH collects the data within the cluster from its member SNs and transmits it to the BS.

Though, clustering increases energy-efficiency of the the network however there are other parameters such as residual energy and workload on the CHs which must be observed. For instance if a CH with lower residual energy is selected then its energy would be depleted after few rounds. Subsequently it would adversely affect overall LT of the network. Similarly unsupervised workload on the CHs significantly can reduce the energy-efficiency of the network. Moreover higher the distance of a CH from BS can result an increased transmission energy consumption and subsequently may reduce the LT of the network. Therefore, we consider different distance parameters, residual energy, and workload on the CHs during our clustering approach in order to achieve maximum energy savings.

Briefly, in order to increase the energy-efficiency both task scheduling and clustering are the important mechanisms to be applied at the SN and network level respectively. Scheduling and clustering reduce the processing and transmission energy consumptions respectively which significantly increases the energy savings of the network while reducing the carbon footprint.

1.5 Aim and Objectives

This thesis aims to increase the lifetime of an MPSoC based wireless network by reducing the processing and transmission energy consumptions by performing proper task scheduling and nodes clustering respectively. In order to accomplish this aim, the following research objectives have been formalized.

1. To design an energy-efficient and contention-aware task scheduling algorithms deploying heterogeneous NoC-MPSoC computing architectures for real-time tasks with deadline and precedence constraints.

2. To design energy and contention-aware scheduling algorithms for a set of real-time tasks with precedence and deadline constraints targeting Voltage Frequency Island (VFI) based heterogeneous NoC-MPSoCs system.

3. To design algorithms for energy-efficient cluster heads selection in WSN to increase the overall LT of the network while efficiently reducing the transmission energy consumption of the sensor nodes distributed over a field.

1.6 Contributions

Contributions and innovations of this thesis are summarized as follows:

1. **Contribution of Chapter 5:** In this chapter, we investigate contention-aware and energy-efficient static scheduling using heterogeneous NoC-MPSoC for real-time tasks with an individual deadline and precedence constraints. Unlike other schedulers task ordering, mapping, and voltage assignment are performed in an integrated manner to minimize the processing energy while explicitly reduce contention between the communications and communication energy. Furthermore, both dynamic voltage and frequency scaling and dynamic power management are used for energy consumption optimization. The developed Contention-aware Integrated Task Mapping and Voltage Assignment (CITM-VA)

static scheduler performs tasks ordering using Earliest Latest Finish Time First (ELFTF) strategy that assigns priorities to the tasks having shorter Latest Finish Time (LFT) over the tasks with longer LFT. It remaps every task to a processor and/or discrete voltage level that reduces processing energy consumption. Similarly, the communication energy is minimized by assigning discrete voltage levels to the NoC links. Further, total energy efficiency is achieved by putting the processor into a low-power state when feasible. Moreover, this algorithm resolves the contention between communications that traverse the same link by allocating links to communications with higher priority. The results obtained through extensive simulations of real benchmarks demonstrate that CITM-VA outperforms state-of-the-art scheduling algorithms and achieves an average $\sim$ 30% total energy improvement. Additionally, it maintains high Quality-of-Service (QoS) and robustness for real-time applications.

2. **Contribution of Chapter 6:** In this chapter, we study energy-efficient and contention-aware static scheduling for tasks with precedence and deadline constraints on heterogeneous VFI based NoC-MPSoCs (VFI-NoC-HMPSoC) with DVFS-enabled processors. Unlike the existing population-based optimization algorithms, we proposed a novel population based algorithm called ARSH-FATI that can dynamically switch between explorative and exploitative search modes at run-time. Our static scheduler ARHS-FATI collectively performs task mapping, task ordering, and voltage scaling. Consequently, its performance is superior to the existing state-of-the-art approach proposed for homogeneous VFI based NoC-MPSoCs. We also developed a communication contention-aware Earliest Edge Consistent Deadline First (EECDF) task ordering algorithm and gradient descent inspired voltage scaling algorithm called Energy Gradient Decent (EGD). We introduced a notion of Energy Gradient (EG) that guides EGD in its search for islands voltage settings and minimize the total energy consumption. Conducted the experiments on 8 real benchmarks adopted from Embedded Systems Synthesis Benchmarks (E3S). Our static scheduling algorithm, ARSH-FATI outperformed state-of-the-art heuristics and achieved an average energy-efficiency of $\sim$ 24% and $\sim$ 30% over CA-TMES-Search and CA-TMES-Quick respectively.

3. **Contribution of Chapter 7:** Wireless Sensor Network (WSN) consists of a large number of sensor nodes distributed over a certain target area. The WSN plays a vital role in surveillance, advanced healthcare, and commercialized industrial automation. Enhancing energy-efficiency of the WSN is a prime concern because higher energy consumption restricts LT of the network. Clustering is a powerful technique widely adopted to increase LT of the network and reduce the transmission energy consumption. In this chapter we develop a novel ARSH-FATI based Cluster Head Selection (ARSH-FATI-CHS) algorithm integrated with a heuristic called Novel Ranked based Clustering (NRC) in order to reduce the communication energy consumption of the sensor nodes while efficiently enhancing LT of the network. Unlike other population based algorithms ARSH-FATI-CHS dynamically switches between exploration and exploitation of the search process during run-time to achieve higher performance trade-off and maximally increase network LT. ARSH-FATI-CHS considers the residual energy, communication distance parameters, and workload during CHs selection. We simulate our proposed ARSH-FATI-CHS and generate various results to determine the performance of the WSN in terms of network LT. We compare our results with state-of-the-art Particle Swarm Optimization (PSO) based clustering and we prove that our developed ARSH-FATI-CHS energy-efficient sensor nodes clustering approach improves the network LT by $\sim 25\%$. ARSH-FATI-CHS also outperforms LEACH and PSO-C while achieving an average LT improvements of $\sim 60\%$ and $\sim 40\%$ respectively.

1.7 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 discusses the background. Chapter 3 examines existing algorithms for task scheduling and nodes clustering. Chapter 4 presents our research methodology. Chapter 5 focuses on the development of contention-aware and energy-efficient scheduler for real-time tasks with precedence and deadline constraints on NoC-MPSoC system. Chapter 6 presents static algorithms for scheduling real-time tasks with precedence and deadline constraints on VFI based heterogeneous NoC-MPSoC architecture. Chapter 7

investigates the problem of energy-aware CHs selection in WSN for improving overall LT of the network. Finally, Chapter 8 concludes the thesis and discusses future research direction.

Chapter 2

Background Study

In this chapter we present an overview of the MPSoCs considering different architectures and explain task mapping and scheduling in the viewpoint of energy-efficiency. We further discuss various energy management techniques applied at the SN level. Moreover we also explain SNs clustering process utilized to reduce the communication energy dissipation of the sensor network.

2.1 Sensor Nodes Architectures

In this section we discuss different architectures used in SNs for WSNs considering the energy-efficiency and performance requirements [70].

Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of high performance, low-cost, low-power, multi-functional SNs that are small in size and communicate untethered in short distances. They are the endpoints of the WSN and each SN comprises of sensor, processing unit and a communication unit [6, 67]: Different architectures used as SNs in WSNs are discussed as follows:

Microcontroller: A microcontroller consisting of antenna (for wireless communication), pro-

cessor, memory, sensor, and DC battery has been widely used as SN in the WSNs. Atmel ATMega 128L, MSP430, and Mica2 are the popular examples used as SNs. Though the energy-efficiency of these microcontroller based SNs was phenomenal however, the processing capability was limited and there was a latency issue because of the slower response. Thus, new architectures have been developed to overcome the limitations of microcontrollers and one of the known example is Digital Signal Processors (DSPr).

Digital Signal Processors: Some applications in WSNs may require to perform digital filtering, Fourier analysis, and encoding. Microcontrollers are generally not optimized for such operations therefore, DSPr are used to perform these data extensive mathematical operations. Specifically DSPr optimizes handles and optimizes digital signal processing related tasks. Though DSP is a suitable solution for many applications but there are limitations such as lower speed and bandwidth. However DSPr can be an integrated part of the MPSoCs though some studies suggested ASIC (Application Specific Integrated Circuit).

ASIC: ASIC is an electronic circuit that integrates all components on a single chip required for performing a special tasks. ASIC has high performance, decreased circuit's congestion, and low power consumption. These qualities make them ideal for being deployed in WSNs. Though ASIC provide an energy-efficient and robust computing platform for data extensive applications in WSNs however, time-to-market, price, lack of scalability and flexibility are the disadvantages of ASIC. Thus Field Programmable Gate Arrays (FPGAs) have replaced ASICs as an alternative.

FPGA: It is an integrated circuit that is designed to be configured by the customers or designers after manufacturing. FPGAs have higher adaptability and flexibility compared to ASIC. The FPGA architectures are re-configurable but complex to design though, they are useful for complex applications in WSNs. Xilinx Virtex-4 is an example of FPGA based architecture used in WSNs for video processing related applications (compression, decoding, image processing). FPGAs have gained drastic acceptance in WSNs to fulfill the requirements such as performance and flexibility.

Multiprocessor System-on-Chips (MPSoCs): Multiprocessor systems are beneficial for

developing high performance and energy-efficient systems such as green computing. MPSoC is set of independent and interconnected processors integrated on a silicon chip. These processors cooperate and communicate with each other to execute applications [71,72]. MPSoC is a single chip system that integrates all or most of the functions of an electronic system including I/O units with analog and mixed-signal components, memory, instruction-set processors, buses, specialized logic, and digital signal processing functions [73,74]. MPSoC has set a new direction to the field of the embedded system. Modern MPSoC architectures also integrate Graphics Processor Unit (GPU), USB controller, Ethernet and/or wireless radios (3G, 4G, WiFi, 4G LTE), power management circuits, and multi-core functions [75–77]. MPSoCs have been pioneered by CPU manufacturing companies such as Xilinx, Tilera, IBM, Motorola, Intel, Samsung, and Apple [78,79].

MPSoC systems can broadly be categorized into three types as illustrated in Figure 2.1 based on their architecture, communication infrastructure, and number of processors deployed within each island of the MPSoC system. Each category of the MPSoC systems is explained with suitable examples given as follows:

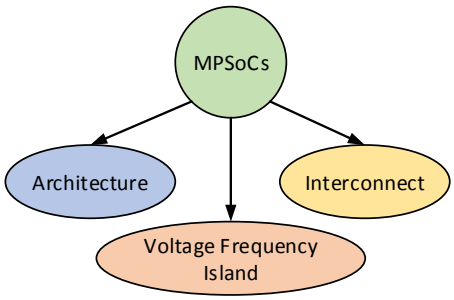


Figure 2.1: MPSoC systems categorization

2.1.1 Architecture

In terms of the types of the processors used, MPSoCs can be divided into two general groups of homogeneous MPSoCs and heterogeneous MPSoCs. Each group of MPSoCs is explained as follows:

2.1.1.1 Homogeneous MPSoCs

Homogeneous MPSoCs are symmetric multiprocessing systems where identical processing elements with the same Instruction Set Architecture (ISA) are used [80]. Therefore, in homogeneous MPSoC, a single thread task will have the same power consumption and complete in the same time irrespective of which processor is utilised. Examples of commercial homogeneous MPSoC include Samsung Mongoose M2 with 4, Cortex-A53 processors and EZchip TILE-Mx100^{TM} with one hundred Cortex-A53 processors. Homogeneous MPSoCs are suitable computing platforms for applications where communication to computation ratio is higher [81].

Homogeneous MPSoCs can be further classified into two categories based on memory organization. Memory can be either shared shown in Figure 2.2(a) or distributed mirrored in Figure 2.2(b).

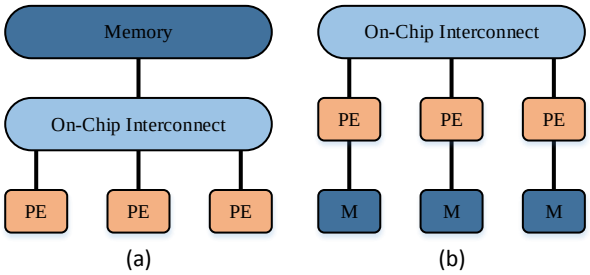


Figure 2.2: Homogeneous MPSoCs (a) shared memory (b) distributed memory

2.1.1.2 Heterogeneous MPSoCs

Heterogeneous MPSoC include multiple types of different processing elements. Heterogeneous MPSoCs can either be functional asymmetric or performance asymmetric.

1. **Functional Asymmetric:** They contain a set of architecturally different processing units, as a consequence with different ISA. Example architecture can include Specific Instruction-set Processor (ASIP), Field Programmable Gate Array (FPGA) fabric tiles, and dedicated microcontroller [82]. Figure 2.3 shows a generic functional asymmetric heterogeneous MPSoC consisting of architecturally different processing units i.e. general-purpose processor(CPU), video accelerator, and audio accelerator [83].

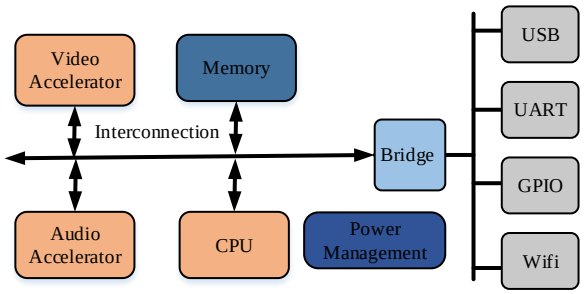


Figure 2.3: Generic heterogeneous MPSoC architecture

2. **Performance Asymmetric:** The other type of heterogeneous MPSoCs is performance asymmetric where ISA remains the same but performance and power consumption of the processing units are different. In other words, some processing units may have higher performance capabilities but lower energy-efficiency compared to others [84]. Samsung Exynos 5 Octa (big.LITTLE) also known as Exynos 5410 used in Samsung Galaxy S4 is an example of performance asymmetric heterogeneous MPSoC. It has 4 Cortex-A15 processors and 4 Cortex-A9 processors. The Cortex-A15 processors offer about 50% higher performance compared to Cortex-A9 [85]. Concisely, Cortex-A9 processor is energy-efficient but low performance while Cortex-A15 is low energy-efficient but high performance processor. Thus energy performance profile of the processors is different in heterogeneous MPSoCs.

Heterogeneous MPSoCs are widely adopted for various processing extensive applications due to their superior performance and lower energy consumption compared to homogeneous MPSoCs [86, 87]. Examples of heterogeneous MPSoC include, Samsung Exynos 9810 used in Samsung Galaxy S9+, S9++, and Note 9+. Few other heterogeneous MPSoCs are listed in Table 2.1 and Table 2.2.

Table 2.1: MPSoCs architectures used in smart-phones

Model	Type	Architecture
Samsung Exynos 9810	Heterogeneous	4 Mongoose 3 big cores and 4 Cortex-A55 little cores
Apple A11 Bionic	Heterogeneous	2 ARMv8-A monsoon and 4 ARMv8-A Mistral

Table 2.2: Heterogeneous MPSoCs for IoT based applications

MPSoC Model	Architecture
Xilinx Zynq® UltraScale+™ MPSoCs	1. Quadcore ARM Cortex-A53 2. Dualcore ARM Cortex-R5 3. Dynamic power management unit 4. ARM Mali™-400MP graphics processor
Renesas R-Car H3 nona-core	1. Quadcore ARM Cortex-A53 2. Quadcore ARM Cortex-A57 3. Dualcore ARM Cortex-R7 4. Integrated power VR GX6650
Xilinx Zynq® UltraScale+™ RFSoCs	1. Quad-core ARM Cortex-A53 2. Dual-core Cortex-R5 3. Dynamic power management unit

2.1.2 Interconnect

The second broad categorization of MPSoC is based on the interconnect, the communication infrastructure used for inter-processor communication [88]. Subcategories are (1) Bus based MPSoC and (2) NoC based MPSoC. The type of inter-processor communication network plays a vital role in achieving energy-efficiency and avoiding communication congestion/contention. Loss of data in the communication network reduces the overall system performance and energy-efficiency [89].

2.1.2.1 Bus

The bus-based architecture is probably the oldest on-chip interconnect in the computer industry and it is used in many MPSoCs [90]. The bus provides a communication mechanism which interconnects different components (processing units, memory, I/O units) of the MPSoC architecture. The bus interconnect shown in Figure 2.4 is an easier approach to integrating a small number of components due to its simple protocol design and silicon cost. However, it offers limited bandwidth and increased delays when used for a large number of components.

Matrix bus interconnect offers a solution for bandwidth as it offers multiple communication paths. Figure 2.5. demonstrates an example of 3 master and 5 slave Advanced Microcontroller

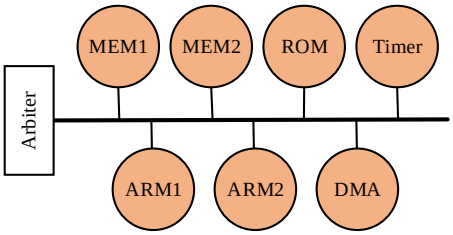


Figure 2.4: Bus communication architecture

Bus Architecture (AMBA) bus matrix communication subsystem architecture for dual ARM processor-based MPSoC. A bus matrix (crossbar switch) has several parallel wires (busses) which offer a suitable backbone to support concurrent data streams. The input stage handles interrupted bursts if receiving slaves are unable to accept them immediately. Decode generates a signal for proper slave selection. The component arbiter collects requests from all masters and allows only one module to have access to the slave at a time [91]. The evolution of bus interconnect is a progression started from AMBA, Advanced System Bus (ASB) to High-Performance Bus (AHB) then AMBA AHB-Lite and finally AMBA AXI (Advanced Extensible Bus). AXI4 is the latest example of MPSoC interconnect [92].

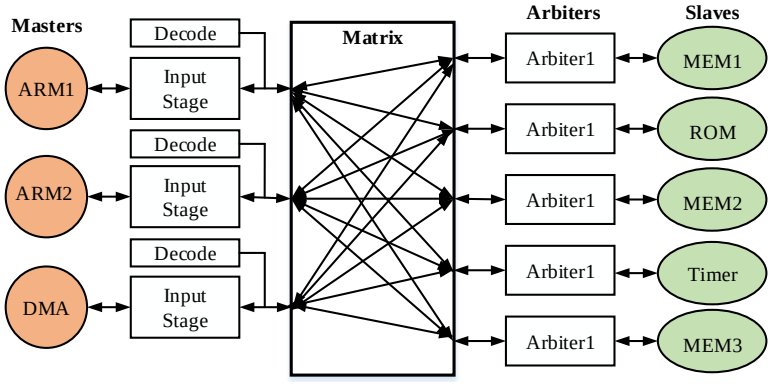


Figure 2.5: Bus matrix communication architecture

2.1.2.2 NoC

NoC is a a typical example of network based communication subsystem on a chip. NoC technology applies the method of computer networking and improves the communication mechanism compared to conventional crossbar communication architectures. NoC increases the scalability, flexibility, and power-efficiency of MPSoC. Figure 2.6 shows a typical 2D-mesh NoC that consists $N_R = 3$ rows and $N_C = 3$ columns i.e. a total of 9 routers. Each router in a NoC

has five ports associated with buffers, four ports are used to communicate with the neighbour routers and one dedicated for the purpose of communicating with the processing unit. NoC architecture basically consist of three major components explained as follows.

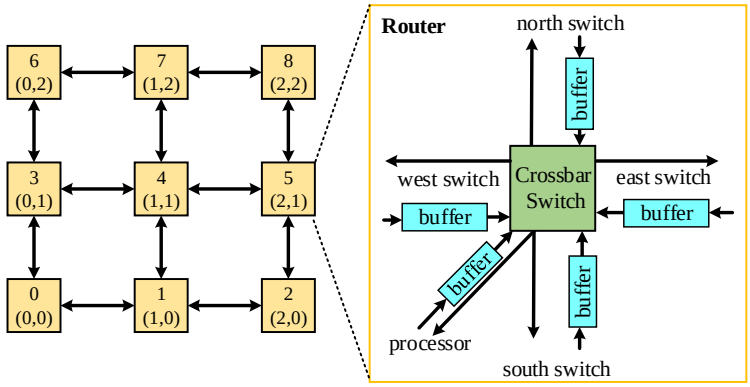


Figure 2.6: A typical 2D-mesh NoC architecture

Communication links: A communication link is used to connect two routers and a router with a processor. It contains one or more channels while each channel consists a set of wires. The links can be half duplex or full duplex and have a bandwidth, b_w.

Router: NoC router consists a set of global output and input ports which are connected to other routers. A switching matrix connects input ports to output ports and the processor associated with the router. The buffers used in routers are to host the incoming data when immediate transfer to next processor and/or Intellectual Property (IP) is not possible because of the congestion.

Network Interface: It provides physical connections and logical connection between the processor and network.

In NoC communication is performed by transmitting message from the source processor to the associated router through network interface and the message is stored in the buffer temporarily for servicing. The router performs routing decision based on its policy regarding the path to be used for message delivery. The communication message is delivered to another router and this process continues until the communication message reaches to its destination i.e. the required processor. The communication latency depends on parameters such as message data size, network bandwidth, and buffer size [93]. Two major policies are used in NoC to deliver a

communication message (1) switching and (2) routing.

Switching: Switching determines when and how a data is transmitted in the NoC. The switching mechanism can be divided into two subcategories i.e. circuit switching and packet switching. A routing algorithm is used in the circuit switching to determine the communication path and thus the a single communication message follows the same route [93]. In packet switching a communication message is often subdivided into several small fixed size packets called flits. Separate routing decisions are made for each packet or flit [94]. Some of the known packet switching strategies are (1) store-and-forward, (2) virtual cut-through, and (3) wormhole.

1. **Store-and-forward:** In store-and-forward strategy of data communication a complete message is stored by the router before sending it to the next router. Buffers of sufficient size are required in this mechanism to hold the whole packet otherwise it may cause the packet stalled.

2. **Wormhole (WH):** In the network when a flit traverses, the WH immediately determines its next hop, forwards it and then the subsequent flits worm their way through the network. The disadvantage of this strategy is that in case of contention occurrence, a stalling packet may blocks all the communication links [33, 95].

3. **Virtual cut-through (VCT):** In VCT routing the buffer size is large and the entire packet is sent to the next router. Thus, VCT has lower latency, higher link utilization, and lesser packet blocking probability. Though WH switching is simple and possesses higher efficiency of flow control over VCT but in case of congestion occurrence, the stalling packet can block all the links and result a low link utilization [96].

Routing: A routing algorithm determines the path of a communication message from source processor to the destination processor. Routing schemes can be either deterministic or adaptive.

1. **Deterministic:** In deterministic approach the packet always follows the same path between source and destination. XY is the most popular and deterministic routing technique

used in NoC. In XY routing the packets are traversed in the X-direction first and then in the Y-direction to reach the target router.

2. **Adaptive:** Adaptive routing scheme dynamically performs link load evaluation and applies a dynamic load balancing strategy. In other words, this scheme may use different path between source and destination if there is congestion on the original path.

There are also some other networks that are deployed as a interconnect for the inter-processors communications on chip. They are explained as follows:

Torus: Torus is the modified version of the basic mesh network. In this network the left sides of all the rows are connected to their right sides and the top of the columns are connected to the bottom of the columns. Torus performs better than mesh-network in terms of hop count but the wire length of mesh-network is significantly smaller than torus [97].

Star: In star network a central router in the middle of the star is responsible for inter-processor communications and the processors are on spikes of the network. There is always a higher chance of congestion occurrence as all the communication messages are going through the central router to reach the destination [98].

Polygon (Hexagon): It is basically a circular network in which the communication messages travel in loop from one router to another. One of the popular example for polygon network is hexagon topology. Hexagon network consist of six routers connected to five other routers while one link is associated for the computational resource [99].

Tree: It uses a tree structure in which the nodes show the routers while leaves represent the computational resources. The routers that are above the leaf are called leaf's ancestors while the leaves below the ancestor are its children. In a fat tree topology for each node there are replicated ancestors thus many alternative routes are available between the nodes [99].

2.1.2.3 Bus based MPSoCs

Bus based MPSoCs use a bus as the medium of communication between processors and other components. Fig. 2.7 shows the architecture of Xilinx Zynq Ultrascale+MPSoC widely used for IoT applications. It deploys ARM AMBA-AXI4 bus to interconnect Quad-core ARM Cortex-A53, Dual-core ARM Cortex-R5 real-time processors, and other controls/peripherals [100,101]. ARM AMBA-AXI4 bus also establishes communication between a memory-mapped master device and a single or multiple memory-mapped slave devices [101]. The peripherals are interconnected on a chip via the Security Policy Engine (SPE) to ensure security. Tightly Coupled Memory (TCM) segregates the untrusted applications. Traditional bus communication architectures support only limited bandwidths and are not scalable for high-performance designs leading to the development of NoC based MPSoC. NoC based communication has several advantages over hierarchical methods (STBus, Advance Micro-controller and Bus Architecture) as well as traditional bus architecture in terms of scalability, flexibility, power-efficiency, and performance [102,103].

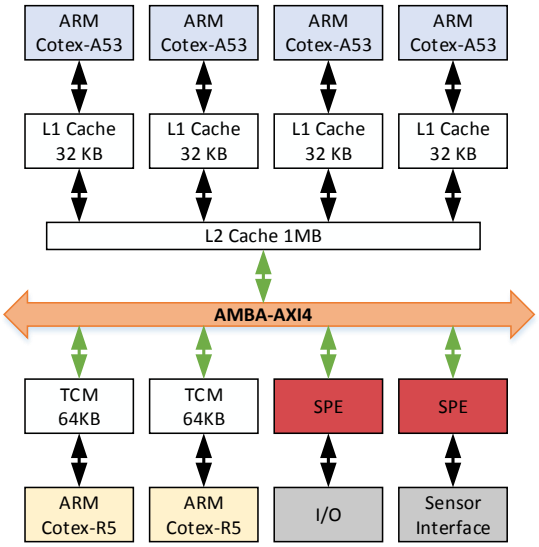


Figure 2.7: Xilinx Zynq Ultrascale+MPSoC using ARM AMBA-AXI4

2.1.2.4 NoC based MPSoCs

In NoC based MPSoC, the various modules such as processors (tiles), IP blocks, and memory elements exchange the data through a network. In NoC data or messages can be relayed

from source to destination module over many links depending upon the routing decision at the switches [82, 104, 105]. NoC based MPSoCs can either be homogeneous or heterogeneous depending upon the nature of the processors used [82]. A generic NoC based MPSoC is illustrated in Figure 2.8.

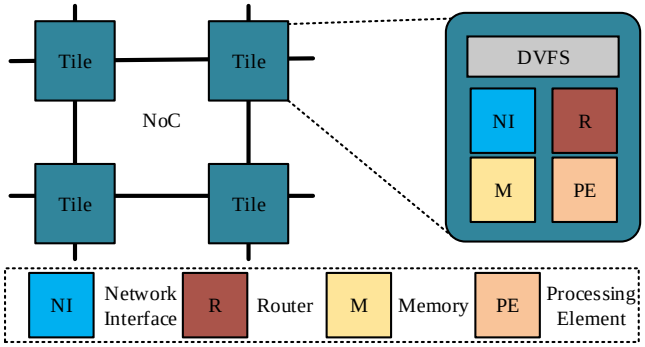


Figure 2.8: A typical representation of NoC based MPSoC architecture

Samsung Exynos 7 Octa (7420) with 2.1 GHz Quad-Core Cortex®-A57 and 1.5GHz Quad-Core Cortex®-A53, HiSilicon Kirin 960 with 4× Cortex-A73, 4× Cortex-A53 are examples of NoC based MPSoC. The Tilera TILEPro64™ is an example that uses 64 identical processing units and iMesh NoC to interconnect tiles with each other and to the external resources. The iMesh NoC comprises of six 2D mesh-networks to support multi-hop routing. Various other NoC based MPSoCs by the Tilera corporation are listed in Table 2.3 [106, 107].

Table 2.3: Different DVFS-enabled NoC based MPSoCs from Tilera™

MPSoC Model	Network Topology	Max: CPU Clock Rate
Tile64 ™	Mesh	600-900 MHz
TilePro64 ™	Mesh	600-866 MHz
TilePro36 ™	iMesh	500 MHz
Tile-Gx ™	iMesh	1.2 GHz
Tile-Gx36 ™	iMesh	1.2 GHz
Tile-Gx16 ™	iMesh	1.2 GHz
Tile-Gx ™	iMesh	1-1.2 GHz

2.1.2.5 Voltage Frequency Islands based MPSoCs

More recently, Voltage Frequency Island (VFI), Globally Asynchronous Locally Synchronous (GALS) model is introduced to NoC interconnect, where the tiles are partitioned into islands

while each island is optimized with its own threshold voltage, operating frequency, and supply voltage. MPSoC systems implemented with GALS have a reduced number of voltage level converter and mixed-clock/mixed-voltage FIFO requirements [95,108,109]. However, the power consumption and complexity of the VFI based MPSoC systems increase when the number of VFIs are increased. Figure 2.9 shows a generalized VFI based NoC-MPSoC. It consists of four VFIs represented by different colours (yellow, green, violet, and blue) and six tiles per VFI. Each VFI contains an independent voltage supply and a local clock. Inter-VFI communication is established through mixed-voltage FIFO, mixed frequency clocks, and voltage converters. Moreover, each tile in every VFI has a local memory, network interface, and a processor [110]. It is worthy of note that state-of-the-art commercially available multiprocessor systems e.g. Intel Itanium i7 and IBM Power 7 series use VFI based MPSoC architectures [111, 112].

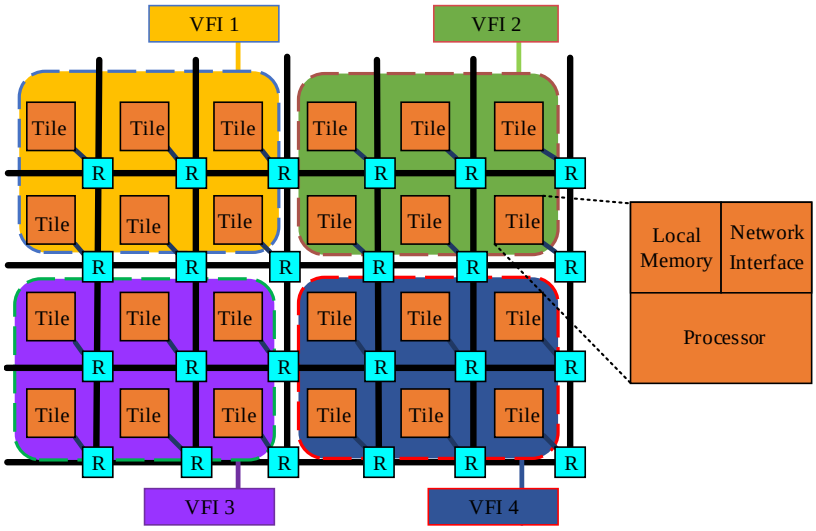


Figure 2.9: A generic representation of VFI based NoC-MPSoC architecture

2.2 Task Scheduling

Task scheduling is a process of properly allocating a set of tasks on the processors such that specific obligations are fulfilled e.g. power consumption optimization and/or execution time reduction. Scheduling real-time applications on a multi-processor system can be considered as solving two problems, 1) task mapping and (2) task ordering.

1. **Task mapping:** It specifies where each task is executed on the multiprocessor computing system.

2. **Task ordering:** This specifies the order and time in which each task or communication message would be executed on processors.

Task scheduling can be broadly divided into two categories (1) Preemptive Scheduling (2) Non-preemptive Scheduling [113].

Preemptive Scheduling: In this type of scheduling task execution can be interrupted and the unfinished portion of the task is re-allocated to a different processor. Preemptive scheduling is more complex than non-preemptive scheduling. Preemptive scheduling need individual task stacks leading to high memory utilization. Interrupting a task while transferring it to another processor on the MPSoC leads to a significant processing overhead and communication delays.

Non-preemptive Scheduling: This scheduling allows a task to execute until its completion on a single processor. Non-preemptive schedulers share a common stack thus results in vastly reduced memory requirements. Non-preemptive schedulers are far easier to be implemented compared to preemptive schedulers. Moreover, they exhibit lower run-time overheads. They guarantee an exclusive access to the resources and eliminate the need for complex resource.

There are different approaches for determining the task priorities for example Longest Processing Time (LPT) which sorts the tasks by their processing time and assigns the tasks to the processors with the earliest finish time. [114]. Highest Level First with Estimated Time (HLFET) schedules a task from set of nodes to a processor in multiprocessor system that allows earliest start time [115]. Dynamic Heterogeneous Earliest Finish Time (dHEFT) scheduler detects the processor in MPSoC that finishes a task at the earliest possible time [116], Earliest Edge Consistent Deadline First (EECDF) is a list scheduler that prioritizes nodes with shorter edge consistent deadline (ECD) over nodes with longer ECD [117]. Earliest Time First (ETF) computes the earliest start times of all ready tasks and selects the task node with the smallest start time [118]. Earliest Completion Time (ECT) priorities a task having lowest earliest start time and execution time over other task nodes. List schedulers for instance DLS (Dynamic

Level Scheduling) [119] are dynamic priority based i.e. after mapping each task the priority is recomputed for unscheduled tasks.

Task can be defined as a sequential program triggered for execution when a particular event occurs. There are several task models explained as below:

Independent Task Model: Independent task model $T = \{T_1, T_2, T_3 \dots T_n\}$ is a collection of tasks with no inter-task data dependencies i.e tasks are not related by precedence relations. In the task model, n shows the total number of tasks. Each task T_i has an execution time, t. Within the independent task model each task T_i contains all the necessary data required to execute on a processor [120–122]. Independent tasks can be executed on the processors in any order by the scheduler. Scheduling independent task is often used to allocate the tasks on available distributed processors such that the overall makespan is reduced. Specifically, scheduler priorities some tasks over others to minimize the finishing time of the last task T_n [123]. Different applications running concurrently such as video streaming, target tracking, and image enhancement can be modelled as independent tasks [124].

Dependent Task Model: The dependent or interacting task model is mostly represented by a Directed Acyclic Graph (DAG) shown in Figure 2.10. This popular representation of a real-time application comprises characteristics such as inter-task communication data size, tasks deadlines, and task dependencies. A DAG can be mathematically represented as $G(V, E, \tau)$ where, $V = \{v_1, v_2, v_3 \dots, v_n\}$ shows the tasks in an application/workload, $E \subseteq V \times V$ denotes data dependencies between the tasks while τ shows edge weights. The edge weight is basically the data transferred (represented by the numbers on each edge) in units of bits between two nodes v_i and v_j [125–127]. Conditional Task Graph (CTG) is another-type-of dependent task model. In the CTG, an edge is called a conditional edge if it is associated with a condition representing that the following task is executed only if the condition holds. An edge is an unconditional edge if no condition is associated with it [35].

There are some typical parameters which are considered during the scheduling process when a real-time application is represented by a task model.

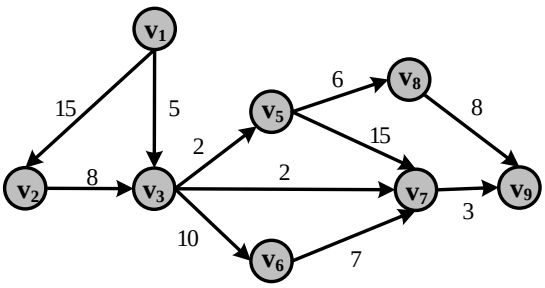


Figure 2.10: Directed acyclic graph

Release Time: As there is a set of task to represent an application so this parameter represents the time at which a task is ready to be executed.

Deadline: This parameter shows the maximum time by which a task should have completed its execution.

Worst-Case Execution Time: It shows the estimated maximum execution time needed to complete a task using maximum available frequency.

Proper task scheduling approach can drastically influence an embedded system energy-efficiency, performance, and reliability [63]. Task scheduling on MPSoCs is NP-hard problem [125,128]. It means that task scheduling problem is intractable and there is no algorithm that can determine an optimal solution for it within polynomial time. Therefore, various scheduling algorithms are proposed by the researchers to determine the near optimal solution using Non-Linear Programming (NLP) [129], Mixed Integer Linear Programming (MILP) [130], and Integer Linear Programming (ILP) [131]. Numerous other search-based approaches are also investigated using Differential Evolution (DE) [132], randomization [133], Simulated Annealing (SA) [32], Particle Swarm Optimization (PSO) [134], and Genetic Algorithm (GA) [33, 135]. These algorithms are categorized in Figure 2.11. Dependent task scheduling can be divided into two other types dynamic and static explained as follows:

1. **Static Scheduler:** It assigns task priorities before the embedded system runs. Static or Offline scheduling simplifies optimization complexity, although this technique is inefficient regarding resources utilization [136]. It is easy to be implemented and guarantees to meet the tasks deadline [137].

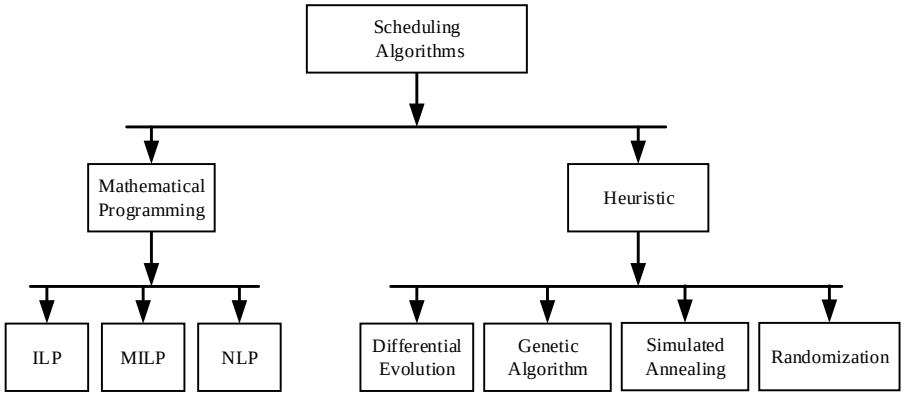


Figure 2.11: Task scheduling algorithms

2. **Dynamic Scheduler:** It executes the tasks on the processor in real-time. The dynamic scheduler considers the available resources and can rearrange the tasks list during run-time [136, 138]. The disadvantage of dynamic scheduling is the runtime overhead and there is no guarantee that all the tasks to be executed can meet their deadline [137].

Static scheduling can be categorized into (1) list-based scheduling (2) clustering-based scheduling (3) duplication-based scheduling, and (4) guided random search-based scheduling. Each of the type is explained as follows [128]:

List-based Scheduling: In this type of scheduling a heuristic prioritizes all the tasks and maintains a task list with their priorities. The tasks are scheduled on the processor based on priority assigned to the tasks and their readiness.

Clustering-based Scheduling: In this static scheduling the tasks are grouped into clusters and each cluster is mapped on the same processor. The tasks are then ordered based on their respective processors.

Duplication-based Scheduling: This scheduling is basically used for reducing the inter-processor communications for energy consumption reduction. It redundantly allocates some of the tasks on more than one processor to minimize the communication overhead.

Guided Random Search-based Scheduling: In this scheduler, algorithm iteratively improves the scheduling result until a stopping criteria is satisfied. It adopts the knowledge gained from previous steps and generates new improved solutions according to random features.

2.2.1 System Level Energy Management Techniques

In this section, we present an overview of the system level power management techniques widely adopted for reducing the energy consumption of multiprocessor system.

Energy is basically "the capacity to do work". Mathematically, $Energy = Power \times Time$, where power is the rate of consumption of energy. The unit of power is watt represented by w and time unit is second denoted by s. The unit of energy is joule denoted by J. Generally the residual energy of the batteries are rated in milliamp-hours (mAh). Theoretically 1000 mAh battery can support a processor for 100 hours when it is consuming 10 mA. Where "A" represents ampere which is the unit of current. [139].

There are many motivations behind reducing the energy consumption of the multiprocessor systems using scheduling. First, the battery life, embedded systems used in WSN are mostly battery powered with limited residual energy. Large amount of energy is consumed by digital chips in the embedded systems. Furthermore, applications are data extensive these days and a lot of processing is required. Second, if a chip consumes high amount of energy then there is a possibility that it can become very hot which consequently may cause system's failure. Moreover, excessive heating causes to increase the production costs of Integrated Circuits (IC) because a special ceramic packaging is required for cooling purposes. Thirdly, reducing the energy consumption of an embedded systems minimizes the carbon footprint and subsequently, results in to achieve green computing objectives. Due to these factors researchers are motivated to perform task scheduling combined with energy management techniques explained below.

Dynamic Voltage and Frequency Scaling (DVFS): MPSoCs are CMOS devices, the overall power consumption in CMOS circuits is due to dynamic and static power dissipation. Dynamic power dissipation occurs due to transistors switching and dominates the total power consumption in CMOS technology. The dynamic power consumption of each processor executing a task at a certain discrete voltage and frequency level (V_{dd}, f) is given as follows [33, 130]:

$$P_D = C_{eff} \times V_{dd}^2 \times f \tag{2.1}$$

where V_{dd} represents the supply voltage and f denotes the operating frequency while C_{eff} is the effective switching capacitance. The cycle length, t_{cycle} of the clock for executing a task assigned to a processor using a certain speed/voltage level can be represented as $t_{cycle} = \frac{L_d \times K_6 \times V_{dd}}{(V_{dd} - V_{th})^{\alpha}}$. Where K_6 denotes technology dependent constant, L_d shows the average logic depth of the processor's critical path, while $1.4 \leq \alpha \leq 2$, and V_{th} is the threshold voltage which can be calculated as $V_{th} = (V_{th_1} - K_1) \times (V_{dd} - K_2) \times V_{bs}$, where V_{th_1}, K_1, K_2 are technology dependent constants and V_{bs} represents body bias voltage.

Equation 2.1 shows that there is a quadratic law relationship between supply voltage and power consumption. In other words reducing the supply voltage minimizes the power consumption quadratically. Thus DVFS is an effective technique to be combined with scheduling for enhancing the energy-efficiency of the multiprocessor platform. DVFS is a system level technique whereby the processor supply voltage and clock frequency are dynamically reduced as a means to reduce overall power consumption without negatively impacting performance and deadline completion [64, 65]. Modern MPSoCs includes DVFS-enabled processors. As an example, the Samsung Mongoose M2 has 4 DVFS-enabled homogeneous processors with an operating frequency range of 2.3 to 2.8 GHz. Moreover, the DVFS-enabled processors include Intel Speedstep, Marvells XScale and Transmeta Crusoe.

Dynamic Power Management (DPM): Now, suppose I_{subn} denotes subthreshold leakage current while I_j shows the junction current, and L_g represents the total number of CMOS devices connected in the circuit, then the static power (P_s) can be expressed as follows:

$$P_S = L_g \times (V_{dd} \times I_{subn} + |V_{bs}| \times I_j) \tag{2.2}$$

where $I_{subn} = K_3 \times e^{K_4 V_{dd}} \times e^{K_5 V_{bs}}$ and K_3, K_4 and K_5 represent processor technology specific parameters (constants). Thus, the total power (P) consumption of each processor in the MPSoC can be computed as follows:

$$P = P_D + P_S \tag{2.3}$$

In CMOS technology the power consumption is due to both dynamic (electronic switching) and static (electronic leakage) components evident from Equation 2.3. Thus the objective of DPM is to reduce static power consumption while DVFS is majorly used to minimize dynamic power consumption [140]. DPM is an effective energy saving technique that can be employed to reduce the static power consumption without significant loss of performance. The objective of DPM is to switch the system to low-power mode when idle, returning to full power mode when required [141]. In other words, the DPM technique switches the processor to an inactive state for as long as possible, while ensuring that all viable tasks finish within their deadlines.

Table 2.4 shows different power consumption modes of Transmeta Crusoe and PXA-250 processors. Power consumption in sleep (low-power) mode is evidently smaller than idle mode, therefore, a significant amount of energy can be saved using DPM technique to switch an idle processor into a low-power mode whenever possible. Though some of the power is dissipated on switching the processor from idle mode into sleep mode. This power consumed is called sleep overhead power denoted by P_{soh}.

Table 2.4: Two different processors power consumption modes

Transmeta Crusoe		PXA-250	
Parameter	Value	Parameter	Value
P_{idle}	276 mW	P_{idle}	555 mW
P_{sleep}	80.0 μ W	P_{sleep}	180 μW
P_{soh}	385 μ W	P_{soh}	483 μ w

As a motivational example in order to understand the working principle of DVFS and its impact on power-efficiency we consider 70 nanometer (nm) technology parameters listed in Table 2.5 [130]. According to the specification, a Transmeta Crusoe processor [142,143] operates at the five discrete voltage levels of $\{0.65, 0.7, 0.75, 0.8, 0.85\}$. The value for body bias voltage is $V_{bs} = -0.70$ V. Given this data we calculate the corresponding frequency, dynamic power P_D, and static power P_S for the different supply voltage V_{dd} levels summarized in Table 2.6. Both P_D and P_S considerably reduce with the decrease in the V_{dd} but the dynamic power reduces more than static power.

Coarse Grained Software Pipelining: Coarse grained software pipelining also known as

Table 2.5: The 70 nm processor technology parameter values

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

Table 2.6: Transmeta crusoe processor power consumption at different supply voltage levels

Parameters	Value				
V_{dd}(V)	0.85	0.80	0.75	0.70	0.65
f(GHz)	2.10	1.81	1.53	1.26	1.01
P_D(mW)	655.5	498.9	370.4	266.7	184.9
P_S(mW)	462.7	397.6	340.3	290.1	246.0

re-timing is another task level technique used to decrease the wasted slack by regrouping nodes from different periods [144]. Consequently, the intra-period precedence constraints are transformed into inter-period precedence constraints. Figure 2.12(c) shows the schedule where the execution of v_1 and v_2 is delayed by one period. Since v_1 executes one period ahead of v_3, it can start executing early as shown in Figure 2.12(c). However, re-timing has a side effect i.e. it adds a prologue and memory overhead. Re-timing can be applied only to applications such as streaming, executing repeatedly. Such applications and are known as periodic applications. Periodic applications are associated with an integer value called the period that defines the time interval after which the application executes again.

We provide an example to better understand the working principle of the re-timing technique. Consider a periodic application modelled by a DAG as shown in Fig. 2.12(a). Figure 2.12(b) shows the schedule for the first three periods. Notice that v_2 cannot execute until v_1 completes execution because v_2 is dependent on v_1. Thus the processor where v_2 is scheduled remains idle (assuming the idle interval is shorter than break even time) during the time interval v_1 executes. In other words, the available slack (on the processor where v_2 scheduled) is wasted. The wasted slack is efficiently minimized by using re-timing.

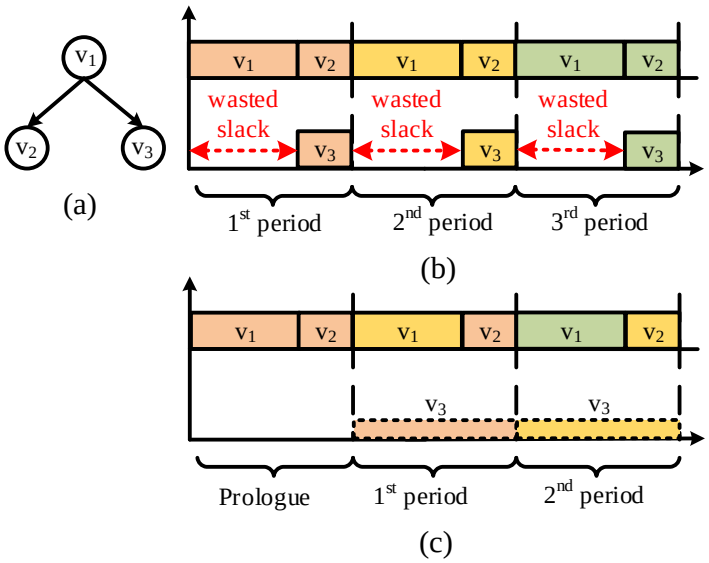


Figure 2.12: Coarse-grained software pipelining (a) a simple DAG with three task nodes representing a workload (b) tasks mapping without re-timing (c) task mapping with re-timing

2.2.2 Other Energy Reduction Techniques

In this section we discuss other energy reduction techniques used at the circuit level of the processors.

Gate Sizing: The dynamic power consumption in CMOS circuit is due to effective switched capacitance C_{eff} which is dependent on load capacitance given as follows:

$$C_{eff} = \gamma C_L \tag{2.4}$$

where γ shows average switching activity and C_L represents the load capacitance, where the load capacitance can be represented as follows:

$$C_L = C_I + C_W + C_O \tag{2.5}$$

where C_I shows the input gate capacitance, C_W denotes capacitance of the connecting wire, and C_O represents output gate capacitance. Thus dynamic power can be minimized by reducing the gate size to decrease the load capacitance. However gate size reduction causes delay and affects circuits timing response.

Input Vector Control (IVC): IVC is a well known technique used for reducing the leakage power. It utilizes transistor stack effect by applying Minimum Leakage Vector (MLV) to the inputs of the combinational circuit during standby mode. In other words a CMOS gate's sub-threshold leakage current depends upon the applied inputs which are also called Input Vector (IV) to the transistor. Subsequently, the circuit can be forced to operate at low leakage condition by using the appropriate IV [145]. One of the approach is to determine IV by enumerating all possible combinations but such method involves exponential time complexity and becomes intractable for a circuit with many primary inputs. another possible approach is to find the near optimal solution of the IVC problem by using probabilistic meta-heuristics. As the leakage power has started to dominate the total power consumption in modern embedded systems therefore, researchers have developed numerous IVC approaches for intance the schemes proposed in [145–147].

Power Gating: It is another effective technique to reduce CMOS circuits leakage power by turning off the supply voltage when blocks are are in idle mode. The power is tuned on once the block is required. Thus power gating is the process of putting some blocks into off state when not required and vice versa. Shutting down the blocks can be accomplished either by software or hardware. Often a dedicated power management controller is used to shut down the blocks [148]. Some techniques integrate power gating with clock gating to reduce both the static power and dynamic power respectively. Clock gating is a widely adopted technique in synchronous circuits to add up more logic to the circuit in order to prune the clock tree. Pruning basically disables portions of the circuitry such that the flip-flops switch states because switching states causes more power consumption [149].

Multi-threshold Design: Multi-threshold CMOS circuits combine low-threshold and high-threshold transistors on a single chip. Low-threshold transistors are used to provide higher performance while high-threshold transistors generates lower leakage current. The circuit paths in CMOS which require high speeds are built by using low-threshold transistor and the rest of the paths are designed by deploying high-threshold transistors. Multi-threshold circuits are build using different approaches. First, initially an entire circuit is built using low-threshold transistors if circuit path delay is comparatively lower than the required clock period the some

lower-threshold transistors are replaced high-threshold transistors on the path. Second, building the complete circuit such that it consists of high-threshold transistors and then estimate if the circuit can't operate effectively at the required clock speed then replace high-threshold transistors on the circuit path with low-threshold transistors. Third, method is to replace groups of cells by low-threshold or high-threshold transistors at once [150].

Apart from the aforementioned techniques which are deployed at the system and circuit level on SNs to reduce the processing energy consumption there is an effective and powerful mechanism called clustering to reduce the transmission energy consumption. It is applied at the network level to increase the energy-efficiency of the sensor network. Clustering in WSN not only provides data aggregation, scalability, bandwidth conservation but also prolongs the network LT by decreasing the communication energy consumption reduction of the SNs. In clustering process, the SNs are partitioned into groups called clusters. Where each cluster has its own leader known as CH. The CH collects the data within the cluster from its member SNs and transmits it to the BS. The information from the BS is transmitted to the Cloud for further processing and visualization purposes as demonstrated in Figure 2.13 [69]. Like scheduling, the CH selection is also a well known NP-hard problem [151]. Subsequently, various searching based heuristics for clustering have been developed using GA [152], ACO [153], PSO [151], DE [154], SA [155]. Among these heuristics PSO is widely utilized for clustering purpose in WSN to enhance the energy-efficiency. Low Energy Adaptive Clustering Hierarchy (LEACH) [156] is one of the earliest and very popular clustering heuristic that reduces the transmission energy dissipation of SNs in wireless networks. LEACH randomly selects CHs, so that the energy dissipation in communicating with the BS is spread over all SNs in the sensor network.

The existing clustering algorithms in WSN fall into these groups, (1) homogeneous and heterogeneous network, (2) centralized or distributed algorithms, (3) static and dynamic clustering, and (4) uniform and nonuniform clustering [157].

1. **Homogeneous and Heterogeneous Network:** The nature and capability of the SNs used in the network influence the CHs selection. For example if a homogeneous network is used where all the SNs have the same processing capability then the CHs are selected

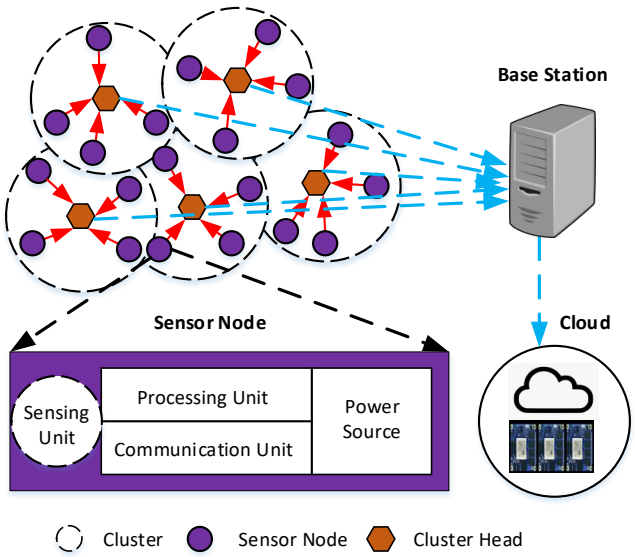


Figure 2.13: Sensor nodes clustering

based on parameters such as residual energy and distance from the BS while any node can become a CH. In heterogeneous network which consists of SNs with different processing capabilities, often a SN with higher processing capability is selected as CH.

2. **Centralized or Distributed Algorithms:** The BS or CH is mainly responsible for cluster formation and network partitioning in the centralized approach. On the other hand in distributed approach cluster formation and CH selection are performed by the SNs themselves. Mostly distributed algorithms approach is used in the homogeneous environment. Hybrid techniques are also deployed where the advantages of both the distributed and centralized algorithms are utilized.

3. **Static and Dynamic Clustering:** In static clustering the cluster formation and CHs selection are performed once and they remain fixed for a long time. Dynamic clustering efficiently group the SNs into clusters dynamically i.e. there is no fixed cluster. Dynamic clustering is used in an environment where where the SNs are mobile that is changing their positions.

4. **Uniform and Nonuniform Clustering:** In uniform clustering the SNs are evenly distributed among the clusters to achieve energy-efficiency. However, this approach is effective where SNs are static. The SNs in non-uniform clustering are not distributed uniformly among the clusters. Non-uniform clustering is satiable approach for dynamic

SNs where the nodes change their position.

There are some terms that are frequently used when algorithms are developed to perform energy-aware clustering.

Intracluster Communication: It represents the communication of SNs with its selected CH within a cluster. Mostly SNs directly communicate with their CH depending upon the distance between them. In large scale networks, multi-hop communications are performed.

Network Lifetime (LT): LT of a network is the number of rounds until the first SN death. Some other literature define network life time as the number of rounds when all the SNs dissipate their residual energy.

Hop Count: Hop count is basically the number of steps/hopes a communication message takes from source to its destination, or the number of nodes it traverses. Increase in the hope count increases the latency and affects the overall performance of the network.

Scalability: The number of SNs distributed over the target area may be ranging from hundreds to thousands. Routing scheme must be able to support this huge number of SNs. Moreover, sensor network clustering algorithms should be scalable enough to respond to events in the environment [158].

Coverage: Coverage is one of important evaluation metric for WSN. It is always advantageous to distribute SNs over a larger physical area. It is worth noticing that network coverage different than the range of the wireless communication links being used. Multi hop communication extends the network coverage beyond the range of radio technology [139].

Quality-of-Service (QoS): In some time-critical applications, data is supposed to be delivered within a particular period of time the moment when it is sensed, otherwise it will be useless. However, in most of the applications, energy savings is directly related to LT of the network and considered the most important parameter than QoS [139].

Data Aggregation: It is the combination of data gathered from different sources using certain aggregation function e.g. minima, maxima, duplicate suppression, and average etc. This

technique is deployed to achieve energy-efficiency. Other signal processing based techniques are also used for data aggregation [139].

2.3 Summary

Summarizing this chapter, MPSoCs are widely adopted in WSN for numerous computational extensive applications due to their ability of high performance and QoS. The energy consumption reduction plays a significant role in modern embedded systems to reduce the carbon footprint and enhance LT of the network. It is important to minimize the energy consumption both at the SN and network level. At SN level DVFS and DPM are two effective techniques that can be intelligently combined with scheduling to increase the energy-efficiency. Proper static scheduling can decrease both the processing energy and the energy due to inter-processor communications traversing the NoC links. Similarly, clustering is another simple but effective technique at the network level to minimize the transmission energy of the distributed SNs.

Chapter 3

Literature Review

Energy-efficiency plays a vital role in WSN to enhance the its LT therefore, different algorithms for energy-aware scheduling and energy-efficient sensor nodes clustering have been investigated in the literature. In this chapter we discuss these heuristics for scheduling and clustering.

3.1 Scheduling using NoC-MPSoCs

Motivated by the fact that MPSoCs are high-performance green computing platform several studies have investigated the static task scheduling problem. These investigations are categorically explained as follows:

3.1.1 Computational Energy Reduction

Computational extensive applications can consume significant energy of the power source and therefore, efficient task scheduling is required increase the overall energy-efficiency.

Srinivasan and Chatha [159] developed a SA based energy optimization scheduling approach called $(LPPWU)_{sa}$ for MPSoC with DVFS-enabled homogeneous processors using bus-based interconnect for inter-processor communications. The $(LPPWU)_{sa}$ reduced an overall run time

and combined both DVFS and DPM for achieving maximum energy savings. However, this approach first only combines DVFS with the scheduling and then separately deploys DPM in the final step to minimize the overall energy consumption.

Tosun [160] mapped periodic independent tasks on heterogeneous MPSoC architecture to reduce the computational energy consumption. A heuristic is developed by using the Earliest Deadline First (EDF) strategy for task mapping while the voltage levels are assigned using DVFS technique. This investigation assumes independent task model while fail to perform experiments on dependent tasks.

Chen et al. [130] formulated the energy optimization problem as MILP using homogeneous NoC-MPSoC architecture for dependent real-time tasks represented by DAG. A Non-preemptive schedule is generated and discrete voltage level is assigned to each task for energy consumption reduction. However, this study does not explicitly consider communication energy overhead i.e. inter-processor communications and heterogeneous MPSoC architecture for achieving higher energy-efficiency.

An ILP based meta-heuristic called Shuffled Frog Leaping Algorithm (SFLA) is proposed by Zhang et al. [134] to map real-time tasks on bus-based interconnect MPSoC system that consists of heterogeneous processors with different energy performance profiles. The study focuses only on real-time applications represented by independent periodic tasks while does not consider tasks with precedence constraints and deadline constraints.

Tariq and Wu [161] investigated the problem of energy-aware scheduling of tasks with conditional precedence constraints on shared-memory homogeneous MPSoC. Their objective was to minimize the total computational energy. In their approach they first map the tasks on processors and then separately perform voltage scaling. They proposed an NLP based algorithm that assigns optimal continuous voltage level to each task given in the initial schedule. The major drawback of this approach is that the processors are assumed to operate at any voltage value between minimum and maximum of the voltage and frequency levels. It is not a practical scenario as the processors in MPSoCs only operate at discrete voltage and frequency levels.

3.1.2 Inter-processor Communication Energy Reduction

Other researchers studied the communication-aware task mapping and scheduling on MPSoCs using heuristics and mathematical programming. Some of the popular approaches are explained as follows:

Shin and Kim [162] considered NoC with voltage scalable links and proposed a GA based approach to reduce the communication energy by finding the optimal voltage levels for communications on the NoC links. Chou and Marculescu [163] improved contention of the NoC for tasks represented by a Directed Acyclic Graph (DAG). Mapping problem is formulated as an ILP for improving congestion and providing best-effort communication in the network. However, these studies have not explicitly considered the reduction of the computational energy consumed by the processors of the MPSoCs architecture.

Carvalho et al. [88] scheduled the dynamic workload on heterogeneous NoC-MPSoC architecture using Nearest Neighbour Heuristic (NNH) algorithm. The authors mainly focused on improving the NoC energy-efficiency by reducing average link occupation and minimized the link congestion using Path Load (PL) algorithm. Though, this scheduling approaches reduced the communication overhead but could not consider the energy performance profiles of the processor for attaining higher energy-efficiency.

Maqsood et al. [164] mapped the tasks on NoC-MPSoCs using Communication-aware Packing based Nearest Neighbour (CPNN) algorithm. The reduction in communication energy was attained by migrating the tasks from the processors with a light load to other appropriate processors that can actively accommodate those tasks as a consequence the inter-processor communication workload is reduced. The work based on CPNN is further improved by Chatterjee et al. [165] and performed communication-aware energy-efficient dynamic task scheduling on homogeneous NoC-MPSoC for real-time applications. However, task migration causes power-overhead that can significantly reduce the over all energy-efficiency.

Wang et al. [166] minimized the inter-processor communication overhead and improved memory utilization using traditional bus-based interconnect infrastructure for homogeneous MP-

SoC platform. First, intra-data dependencies are transformed into inter-data dependencies in the DAG. Second, Heuristic Memory-Aware Task Scheduling (HMATS) is developed for task scheduling . Since the scheduling heuristic algorithm focus only on traditional bus based communication, therefore cannot be extended to NoC based multiprocessor systems.

Singh et al. [167] proposed a Contention-aware and Energy Efficient, Duplication based Mixed Integer Programming (CEEDMIP) formulation for scheduling tasks on heterogeneous NoC-MPSoC architecture. This approach duplicates some tasks on the processor to reduce the inter-processor communication energy and avoid traffic congestion. The study fails to minimize processing energy consumption. Furthermore, duplication of tasks adversely affects overall systems energy savings.

3.1.3 Total Energy Savings

Wang et al. [168] deployed homogeneous MPSoC architecture for real-time streaming applications and integrated task level coarse-grained software pipelining with DPM and DVFS to reduce the total energy consumption. A two-phase energy optimization algorithm is developed where in the first phase DAG is transformed into independent task model using re-timing. In the second phase GA based algorithm called GeneS is used to find a feasible schedule and DVFS and DPM are used to reduce computation and inter-processor communication energy consumption. Though, processing and communication energies are minimized but fails to consider NoC based communication. Furthermore, in this study all processors are assumed to be homogeneous.

Tariq et al. [35] used NoC based heterogeneous MPSoC and minimized the total energy consumption of tasks having conditional precedence constraints. They proposed an Iterative Offline Energy-aware Task and Communication Scheduling (IOETCS) Algorithm that collectively performs scheduling and voltage scaling. They developed an NLP based algorithm assigning each task and communication optimal voltage and frequency levels within a continuous voltage and frequency range provided an initial schedule by Earliest Successor-Tree-Consistent Deadline First (ESTDF) algorithm. The optimal continuous voltage and frequency levels are then

mapped to valid discrete voltage and frequency levels using either an ILP or Heuristic based algorithms. However, in their approach DPM with DVFS has not been integrated moreover, no energy dissipation occurs has been assumed during the idle time slots which is not a practical scenario.

Gosh et al. [169] presented a unified approach for task mapping problem on heterogeneous NoC-MPSoC. The tasks are mapped such that links congestion does not occur and tasks are executed on optimal speed levels using MILP. Huang et al. [32] used an extended ILP formulation for optimizing both the communication and processing energy on heterogeneous NoC-MPSoC architectures. Moreover, task scheduling is accelerated using Simulated Annealing with Timing Adjustment (SA-TA) heuristic algorithm. The SA-TA algorithm basically optimises the energy consumption by reaching near to the global optimum under even tight timing constraints. However, links voltage scaling and energy performance profiles of the processors are not considered.

Unlike the aforementioned studies we consider a NoC based heterogeneous MPSoC architecture with distributed memory for real-time dependent tasks represented by a DAG. Moreover, we reduce both the computational and communication energy consumptions and efficiently integrate DVFS and DPM within our developed CITM-VA meta-heuristic while explicitly considering NoC links contention and energy performance profiles of the processor during task mapping.

3.2 Scheduling using VFI-NoC-MPSoC

Task mapping and scheduling on multiprocessor architectures is an NP-hard problem and different heuristics have been proposed based on mathematical formulation such as ILP, NLP, LP, and MILP. Similarly, search based heuristic algorithms using selection, crossover, mutation, and elitism are also widely deployed. The popular examples of these search based algorithms are ACO, GA, and PSO. Among these algorithms, GA is widely adopted for task mapping and scheduling [33, 170]. These evolutionary algorithms belong to stochastic generate and test algorithms which are based on (1) exploration of the search space and (2) exploitation of the promising information already found. Exploration primarily describes the ability of an algo-

rithm to discover the unseen regions while exploitation demonstrates the capability to proceed in the desired direction for improvement. For example in GAs, mutation and crossover are hypothetically considered to perform exploration and exploitation respectively [171,172]. However, there is strong criticism that crossover does not possess a competitive advantage over mutation [172]. Nevertheless, these search based algorithms fail to efficiently exploit the available chunk of information i.e. schemata. Moreover, exploration and exploitation are the two opposing forces and a well-found balance between them determines the success of a search based algorithm. The task scheduling algorithms developed in literature using VFI-NoC-MPSoCs are broadly categorized below.

3.2.1 Independent Task Scheduling

As a matter of fact multiprocessor systems are reliable and high-performance computing platforms for edge devices and edge computing in WSN. Thus, several researchers have investigated task mapping and scheduling. One of the earliest work in scheduling includes a scheme developed by Olafsson [173] in order to efficiently distribute the tasks i.e. workload on heterogeneous multiprocessor system. Aydin et al. [174] provided energy-efficient scheduling algorithm for independent real-time on multiprocessors system and deployed Earliest Deadline First (EDF) for task ordering. Other energy management studies used DVFS technique for energy optimization. For example, Zhang et al. [134] presented a meta-heuristic scheduling algorithm called Shuffled Frog Leaping Algorithm (SFLA) by integrating the gains of PSO and Memetic algorithms while compared the energy-efficiency of SFLA with GA. Kumar and Vidyarth [135] integrated task mapping and voltage assignment in a single optimization loop of GA. They used DVFS technique to assign voltages to the tasks such that the dynamic energy consumption is reduced with an acceptable performance trade-off. Wang et al. performed preemptive periodic independent tasks scheduling using Discrete Event System (DES) supervisory control [175]. Liu and Qi mapped the tasks using Weighted Earliest Finish Time (WEFT) algorithm and executed the tasks with lowest possible earliest completion time [176]. These investigations reduced energy consumption of the independent tasks running on MPSoC architectures without explicitly

considering the precedence constraints.

3.2.2 Single-processor Per VFI

Huang et al. [32] used an extended ILP formulation for energy optimization on heterogeneous NoC based MPSoC systems and developed a heuristic called Simulated Annealing with Timing Adjustment (SA-TA). SA-TA optimizes energy consumption by reaching near to the global optimum under timing constraints. Gammoudi et al. scheduled real-time periodic tasks on homogeneous NoC based MPSoC in order to meet deadline, energy and communication constraints using a heuristics manipulated by deterministic strategy [177]. Ali et al. performed integrated task mapping, scheduling, and voltage assignment on NoC based heterogeneous MPSoC (HMPSoC) systems using a heuristic called EIMSVS for reducing processing and communication energies [178]. Ishak et al. investigated a non-preemptive scheduling for tasks with precedence constraints and individual deadlines. They used NLP and ILP to assign optimal voltages to the tasks and communications on NoC links [87]. A similar ILP based approach is followed by Tariq et al. [35] using Iterative Offline Energy-aware Task and Communication Scheduling (IOETCS) algorithm for total energy consumption reduction. Ali et al. developed an energy-efficient task scheduling approach using Contention-aware Integrated Task Mapping and Voltage Assignment (CITM-VA) meta-heuristic. CITM-VA integrated DVFS and DPM to achieve maximum energy savings by reducing both static and dynamic power consumptions while considering the contention at NoC links [33]. Ding et al. presented a Hybrid Heuristic Genetic Algorithm with Adaptive Parameter (HGAAP) for task mapping on heterogeneous multiprocessors [179]. However, these studies consider MPSoC systems for tasks with precedence constraints but perform mapping and scheduling on single processor per VFI.

3.2.3 VFI based MPSoC

Ninomiya et al. [180] developed a task scheduling scheme for VFI based MPSoC architecture using SA based algorithm for energy consumption reduction and generated a schedule for

set of tasks under deadline constraints. Pagani et al. [181] presented a scheduling scheme called Single Frequency Approximation (SFA) to map the tasks and assign optimal voltage and frequency levels to each VFI. Liu and Guo [182] developed a Voltage Island Largest Capacity First (VILCF) algorithm for mapping the tasks on active VFI first in order to fully utilize it before activating other inactive VFIs. Shin et al. [183] studied communication-aware VFI partitioning approach and developed a task mapping, voltage assignment algorithm for reducing inter-VFI communications. These investigations in [180–183] deploy bus-based VFI-MPSoC systems for independent tasks mapping and scheduling. Some other researchers considered NoC based VFI-MPSoC systems for instance, Jang and Pan [184] performed energy-aware scheduling for dependent tasks by reducing VFI's power overheads. Digalwar et al. [185] presented a scheduling algorithm in order to optimize the total energy consumption for periodic tasks with hard deadline. Han et al. [95] developed a contention-aware static mapping and scheduling scheme for a set of tasks with precedence constraints in order to minimize the make-span and inter-VFI communications. They developed a contention and energy-aware task mapping and edge scheduling (CATMES) heuristics to assign tasks to processors while scheduling the edges on NoC. Two approaches CA-TMES-Quick and CA-TMES-Search were developed to select the processor for a task where it can start earliest among all processor. CA-TMES-Quick first performs task assignment and then determines routes for the communications that are sent to this task. CA-TMES-Search calculates start time for each task while considering communication contention. The processor offering earliest start time for a task is selected by CA-TMES-Search. Specifically, CA-TMES-Search relatively performs better than CA-TMES-Quick because it coordinates the task mapping in an exhaustive way therefore, make-span significantly reduces.

Though these state-of-the-art studies [95, 180, 183–185] addressed the energy-efficiency on VFI based NoC-MPSoC systems but none of them performed investigation on heterogeneous computing platform while considering processor energy performance profiles for achieving higher energy savings. Specifically, to the best of our knowledge none of the prior work focused on contention-aware and energy-efficient task scheduling on VFI-NoC-HMPSoC using DVFS technique for dependent tasks with precedence and deadline constraints.

3.3 Sensor Nodes Clustering

One of the most popular and widely used sensor nodes clustering algorithm is Low-Energy Adaptive Clustering Hierarchy (LEACH) [156]. LEACH is a probabilistic approach that randomly selects CH in each round. Though, LEACH attains a significant energy consumption reduction while prolonging network LT compared to Static Clustering and Minimum Transmission Energy (MTE) algorithm however, some disadvantages are associated with it. For example, LEACH can select a SN with least residual energy as a CH subsequently, which adversely affects the network life. Numerous other clustering algorithms have been developed by the researchers to improve the efficiency of LEACH. For instance Power-Efficient GAthering in Sensor Information Systems (PEGASIS) [186] that is a chain based approach. PEGASIS organizes SNs into a chain while each SN communicates only with neighbour SNs. Each SN in the chain takes turns in every round to transmit data to the BS. PEGASIS saves more energy when compared to LEACH but it is not suitable for large sized networks due to its instability and high delays. Loscri et al. [187] presented a two-level hierarchical approach called TL-LEACH. TL-LEACH efficiently distributes the energy load among the SNs for large size networks. It significantly enhances the network LT compared to LEACH but causes an extra overhead for selecting secondary CHs. Xiangning et al. [188] improved LEACH by considering the residual energy of the SNs during CH selection process and developed a protocol called Energy-LEACH (E-LEACH). However, this protocols only minimizes the average distance between CHs and the non-CH nodes while fails to consider the distance between the BS and CHs. Yassein et al. [189] developed a Voice-LEACH (V-LEACH) protocol. V-LEACH also selects a voice-CH besides a CH in the cluster. The voice-CH acts as a CH when there is no residual energy left in the CH. Though, V-LEACH performs better than LEACH in terms of energy-efficiency however, extra amount of energy is utilized for selecting the voice-CHs during clustering phase. Al-Baz et al. [190] improved the performance of LEACH and developed an algorithm called Node Ranked-LEACH (NR-LEACH). The NR-LEACH selects CH by considering cost and number of links between the sensor nodes. This enhances the network LT and distributes the energy load among the sensor nodes. However, this approach is not suitable for large scale networks.

Other investigations integrated evolutionary algorithms with LEACH in order to improve its performance. Some of these popular examples include, LEACH-C [191], LEACH-GA [192], LEACH-C [193], and DE-LEACH [194].

In the analytical study presented by Zungeru et al. [195], it has been discussed that evolutionary meta-heuristic algorithms based sensor nodes clustering provide a significant energy savings. Since finding m-optimal clusters is NP-hard problem [196] therefore, in the last decade numerous search based meta-heuristics have been proposed in the literature for efficient CHs selection. Subsequently, researchers developed meta-heuristic based algorithms for efficiency cluster formation in WSN to enhance the network LT. Centralized LEACH (LEACH-C) [193] is among the earliest meta-heuristic based clustering approach where SA is used for solving the clustering problem. In LEACH-C, initially all sensor nodes transmit their location and energy information to the BS where SA is applied to determine the clusters. In terms of energy savings LEACH-C outperforms LEACH [156]. Guru et al. [197] proposed a cluster formation scheme using PSO using a fitness function that includes the sink distance and intra-cluster distance. This approach fail to consider the residual energy of the SNs. Latiff et al. presented an algorithm called PSO-C for energy-efficient clustering in WSN. PSO-C considers an average intra-cluster distance and total initial energy of all SNs to the total current energy of all CHs. However, PSO-C assigns non-CH nodes in the cluster formation to the nearest CH which can potentially decrease the energy-efficiency of the network. Singh et al. [198] presented a PSO based energy-efficient CH selection algorithm called PSO-Semi Distributed (PSO-SD) and used a fitness function that considers residual energy, distance, and node density. The disadvantage of PSO-SD fail to consider cluster formation phase which consequently can reduce the energy efficiency of the network. Rao et al. developed energy-aware clustering approach using novel Chemical Reaction Optimization (nCRO) heuristic. The nCRO significantly prolongs the network LT but the main drawback is that the selected CHs directly communicating to the BS, which is not desirable for large scale network [199]. Kuila and Jana [154] used a novel DE algorithm for SNs clustering to enhance the network energy-savings. Though, this scheme efficiently prolongs LT of the SNs but does not consider sink distance in the process of clustering moreover, CHs are selected randomly. Kuila et al. [200] developed a load balancing algorithm

using DE for better energy-savings. The demerit of this approach includes selecting the CHs randomly while not considering energy and distance parameters. Gupta and Jana [152] presented a GA based clustering approach while considering the residual energy of the SNs and the distance from their corresponding CHs. Zhang et al. [155] clustered the SNs using SA and GA while CHs are selected on the basis of estimating the average energy of the cluster. A CH is selected if a SN has higher residual energy than the average energy of the cluster. Hoang et al. [201] developed Harmony search algorithm based clustering protocol (HSACP) to decrease the intra-cluster distances between the cluster members and their CHs. However energy-efficiency can decrease if the workload on CHs is not considered. Rao et al. [69] developed a PSO based energy-efficient CH selection (PSO-ECHS) algorithm to increase the SNs LT. The fitness function in PSO-ECHS considers parameters including sink distance, intra-cluster distance, and residual energy of the SNs. The clustering algorithms in [69, 152, 155, 200] increased the LT of the network but there are numerous number of parameters involved required to tune these heuristics for achieving a balance between exploitative and exploitative search modes in order to attain higher energy-efficiency. Cisse et al. [202] developed an Energy Aware Neighbor Oriented Clustering (EANOC) to select the CH on the basis of Received Signal Strength Indicator (RSSI) value. However, sensor nodes consume energy due to propagating RSSI in multi-directions which consequently affects the overall energy-efficient of the network.

There is abundant literature on energy savings for WSNs and various methods and/or algorithms have been presented in the last few years but there is still much ongoing research to optimize the energy consumption of the battery-limited wireless networks. Our proposed CHs selection integrated with clustering technique using a novel meta-heuristic for enhancing network LT possesses the following advantages over other existing approaches:

1. Unlike, these sensor nodes clustering algorithms in [151, 155, 156, 186, 188, 201], we use a fitness function that considers the residual energy, distance parameters, and workload on the selected CHs to prolong the overall LT of the network.

2. The state-of-the-art searching based algorithms in [69, 152, 155, 200] are complex and require different parameters tuning to attain energy-efficient solution. Our novel meta-

heuristic ARSH-FATI based Cluster Head Selection (ARSH-FATI-CHS) dynamically switches between exploitative and exploitative modes at run-time for better and efficient performance trade-off.

3. Heuristic based approaches such as [69,202] develop energy-aware sensor nodes clustering using the signal strength values. However, it may affect the overall network LT. Thus, we use sensor nodes positions in the clustering process.

4. Dissimilar to the sensor nodes clustering approaches in [186, 190], ARSH-FATI-CHS is applicable to large-sized networks. Furthermore, ARSH-FATI-CHS does not produce any power overhead like the clustering approach presented in [187].

In order to compare the the performance of ARSH-FATI-CHS, we select three well known clustering algorithms such as LEACH [156]. The reason behind selecting LEACH is that it is one of the classical clustering approach while PSO-C [151] and PSO-ECHS [69] are selected because these algorithms are the latest meta-heuristics for the same network scenario as we proposed in this chapter.

Briefly we investigate an efficient and energy-aware sensor nodes clustering approach applicable to any-sized network based on ARSH-FATI-CHS meta-heuristic which is a population based algorithm. The ARSH-FATI-CHS is guided by A heuristic called Novel Ranked Based Clustering (NRC) to enhance the overall network LT. Our developed fitness function considers residual energy, different distance parameters, and workload on the CHs during cluster formation.

3.4 Summary

We examined energy-aware scheduling and energy-efficient approaches and/or algorithms used at SN level, particularly targeting MPSoC architectures. We identified challenges to the existing energy-aware approaches/algorithms. For example (1) inefficiency to address the problem of communication contention at the NoC links for heterogeneous MPSoC systems, (2) deficiency to schedule task on processors while considering processors energy performance profiles, (3)

inadequacy to establish an optimal balance between DPM and DVFS during task scheduling, (4) ineffective search based algorithms that can dynamically switch between explorative and exploitative modes at run time. Similarly we examined energy-efficient clustering algorithms in the literature at network level and identified the research gaps such as (1) inefficiency of the population based algorithms that require a lot of parameters tuning for achieving higher energy savings, (2) deficiency of the fitness function considering residual energy, distance parameters, and workload on the CHs.

Chapter 4

Research Methodology

In this chapter we first present the relevant application model and its extended format, multiprocessor architectures, network and energy models (both for node and cluster levels) deployed for simulation to produce quantitative results. Then we present our research method adopted for scheduling and clustering. Finally, we also explain how the data is collected from different simulation scenarios and discuss how this generated data is analyzed using certain metrics.

4.1 Preliminaries

Before starting to explain our two step research approach that is (1) scheduling and (2) clustering for reducing the total energy consumption, we explain an extended graph, multiprocessor computing platforms, and energy models that are used in the simulations.

4.1.1 Extended Graph

In an application demonstrated in Figure 4.1, there are two kinds of events communications and tasks. In order to schedule communications using traditional DAG based scheduling approaches we transform a DAG i.e. G into an extended graph G_e shown in Figure 4.2. Given a task to processor mapping an extended graph G_e is constructed by inserting an additional node v_s to

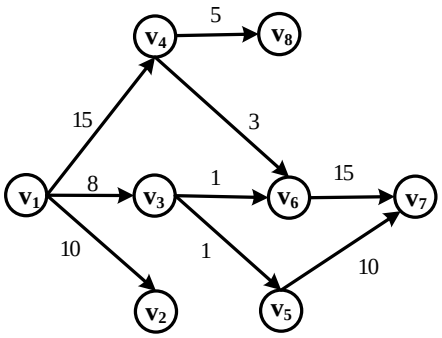


Figure 4.1: Directed acyclic graph

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 is a set of the task nodes, V^* is a set of the communication nodes, and E is a set of the edges.

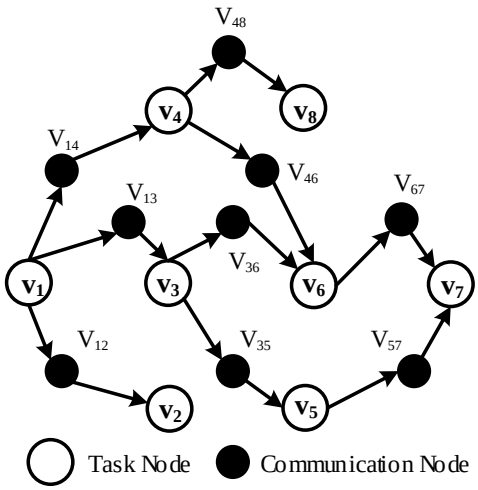


Figure 4.2: Extended graph

4.1.2 Multiprocessor Computing Platforms

In this research two different architecture of the multiprocessor systems are used. Each architecture and its energy model is explained as follows:

4.1.2.1 Non-VFI based NoC-MPSoC

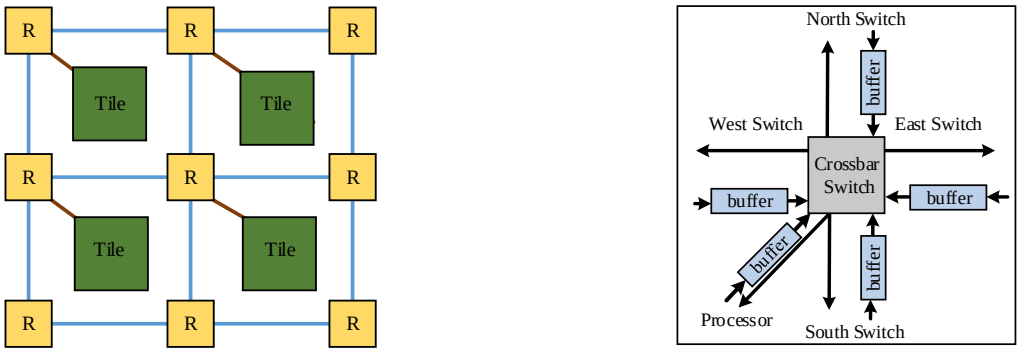
We consider a NoC-MPSoC consisting k number of heterogeneous tiles as demonstrated in Figure 4.3(a). Each tile is comprised of local memory, processor (pe) and a network interface. Therefore, MPSoC contains a set $P = \{pe_1, pe_2, \ldots, pe_k\}$ of k DVFS-enabled processors. The links that provide connection between the router (represented by R) and tile are termed as *Local Links* while *Global Links* interconnecting the routers with each other for communications. Each processor $pe_i \in P$ can operate at $\{(V_{dd_1}, f_1), (V_{dd_2}, f_2), \ldots, (V_{dd_n}, f_n)\}$ set of n discrete voltage and frequency levels. Moreover, each processor supports DPM and can be switched into different power modes.

1. **Communication Interconnect Model:** A 2D-mesh topology NoC architecture is assumed for inter-processor communication. It consists N_x rows and N_y columns of the routers therefore, k number of routers are equal to $N_x \times N_y$. Each router is comprised of five ports to communicate with neighbor routers and a processor as shown in Figure 4.3(b). Each port has a link and buffer. All links are identical and full duplex with band width, b_w. Similar to the processors the links in NoC can operate at n set of discrete voltage/frequency levels i.e. $\{(V_{dd_1}, f_1), (V_{dd_2}, f_2), \ldots, (V_{dd_n}, f_n)\}$.

   We assume a simple and energy-efficient Wormhole (WH) packet switching technique for NoC communication. WH splits the data packet into small pieces called flits and they are delivered in a pipelined fashion in the network. Furthermore, we assume widely used deterministic XY routing scheme. The distance between two processors pe_i and pe_j in 2D-mesh is given by the Manhattan distance $\eta_{(i,j)} = |x_i - x_j| + |y_i - y_j|$, where (x_i, y_i) are the coordinates of processor pe_i and (x_j, y_j) are the coordinates of processor pe_j in the mesh.

2. **Energy Model:** We consider the energy consumption due to processors, routers, and network links in our energy model. The dynamic power P_{d_i} dissipated in executing a task v_i on a pe_j is given by the following equation [161, 203]:

$$P_{d_i} = C_{eff_i} V_{dd_j}^2 f_j \tag{4.1}$$



(a) NoC-MPSoC architecture (b) NoC router structure

Figure 4.3: 2D-mesh topology based NoC-MPSoC

where, V_{dd_j}, f_j, and C_{eff_i} denote supply voltage, operating frequency, and effective load switching capacitance respectively. This mathematical relation shows that decrease in dynamic power occurs when the supplied voltage is reduced thus, Equation 4.1 serves as the baseline for DVFS.

The total power (P_{T_i}) dissipated in executing a task on pe_j is the sum of dynamic power, static power and inherent power, P_{on} required to keep the processor on i.e. idle power when no task is running on the processor. At V_{dd_j} and f_j, the P_{T_i} is calculated as follows [161]:

$$P_{T_i} = P_{d_i} + L_g(V_{dd_j} K_3 e^{K_4 V_{dd_j}} e^{K_5 V_{bs}} + |V_{bs}| I_{j_n}) + P_{on} \tag{4.2}$$

where K_3, K_4 and K_5 are technology dependent constants, L_g is the number of logic gates in the circuit and I_j and V_{bs} are junction leakage current and body-bias voltage, respectively.

As $E = P \times t$, where t shows time so, the energy E_i consumed in executing a task v_i on p_j is given as follows [35, 130]:

$$E_i = C_{eff} V_{dd_i}^2 NCC_i + L_g(V_{dd} K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} + |V_{bs}| I_j) t_i \tag{4.3}$$

where t_i is the execution time of v_i and is given by $t_i = \frac{NCC_{(i,j)}}{f_j}$. The operating frequency, f and supply voltage, V_{dd} are related by the following equation [203]:

$$f = (V_{dd} + V_{th})^{\alpha} / K_6 L_d V_{dd} \tag{4.4}$$

where V_{th} is the threshold voltage, K_6 is the process dependent constant, L_d is the logic depth of the processor critical path and α reflects velocity saturation ($1.4 \leq \alpha \leq 2$). So, according to Equation 4.4 energy consumption rely on the voltage and frequency level assigned to the tasks.

The energy consumed by the processor when it is idle in an active mode is given as follows:

$$E_{idle} = P_a \times t_{idle} \tag{4.5}$$

where P_a is the power consumed by the processor in active mode and t_{idle} is the time period for which the processor is idle. Similarly, the energy consumed by a processor in the sleep mode is calculated by:

$$E_{sleep} = P_{sleep} \times t_{sleep} \tag{4.6}$$

where P_{sleep} is the power dissipated by the processor in sleep mode and t_{sleep} is the duration for which processor stays in sleep mode. Since $P_a > P_{sleep}$, energy efficiency can be achieved by switching the processor into a sleep mode. However, there are switching costs associated in transitioning the processor between active and sleep modes. The processor break even time, t_{BET} represents the shortest duration of idle time interval that justifies processor's transition from active mode to sleep mode. Thus, if this interval is shorter than t_{BET} the mode switch overheads are larger than the energy saving and therefore, transition to low power mode should be avoided. The definition of t_{BET} is given as follows:

$$t_{BET} = \max\left\{t_{sw}, \frac{E_{sw}}{P_a - P_{sleep}}\right\} \tag{4.7}$$

where t_{sw} and E_{sw} are total switching time interval and total switching energy overhead respectively.

Under the fixed operational frequency f_l of the links the time taken to transmit the message ($\chi_{i,j}$) of a communication is in general dominated by the serialization delay. The execution time $t_{i_{com}}$ of transmitting a message for communication $e_i = (v_s, v_d)$ is given as

follows [35]:

$$t_{i_{com}} = \frac{\chi_{(i,j)}}{f_l \times b_w} \tag{4.8}$$

Let the parent node v_s of e_i be mapped on pe_i and its child node v_d be mapped on pe_j. Then the hop count between pe_i and pe_j is $\eta_{(i,j)}$. The energy consumed in transmitting one bit of a message e_i is $E_{bit} = E_{Rbit}(\eta_{(i,j)} + 1) + \eta_{(i,j)}E_{lbit}$, where E_{Rbit} is the energy consumed by a router in transmitting one bit and E_{lbit} is the energy consumed by links in transmitting a bit. The energy E_{c_i} consumed in transmitting $\chi_{(i,j)}$ volume of data is calculated as follows [35, 204]:

$$E_{c_i} = \chi_{(i,j)}E_{Rbit}(\eta_{(s,d)} + 1) + \chi_{(i,j)}E_{lbit}\eta_{(s,d)} \tag{4.9}$$

$$E_{lbit} = \frac{P_i}{f_j \times b_w} \tag{4.10}$$

where P_i shows the total power consumption for one bit on the links that e_i traverses at f_j. Thus, P_i is the sum of the static power (P_s) and dynamic power (P_d) i.e. $P_i = P_{d_i} + P_{s_i}$ [203].

Inserting Equation 4.10 in Equation 4.9 yields the following equation:

$$E_{c_i} = \chi_{(i,j)}((\eta_{(s,d)} + 1)E_{R_{bit}} + \eta_{(s,d)}\frac{P_i}{f_j \times b_w}) \tag{4.11}$$

Equation 4.11 indicates that inter-processor communication energy can be reduced by assigning optimal discrete frequency or voltage levels (as voltage and frequency are interchangeable according to Equation 4.4) to mesh topology NoC links for transmitting the data.

4.1.2.2 VFI based NoC-MPSoC

Second, we consider a VFI based NoC-MPSoC architecture with M processors, $P = (pe_1, pe_2, pe_3, \ldots pe_M)$ demonstrated in Figure 4.4. Each tile consists of a processor, local

memory, and network interface card. Processors of the target architecture are partitioned into a set $C = \{c_1, c_2, c_3, \ldots c_m\}$ of m heterogeneous VFIs. Where each VFI, $c_i \in C$ consists a set of k homogeneous processors. We assume processors within an island (VFI) are of same type. Processors across different VFIs may be of different types i.e. inter-VFI processors may be heterogeneous. Each VFI can operate independently at a set $\{(V_{dd_1}, f_1), (V_{dd_2}, f_2), (V_{dd_3}, f_3), \ldots, (V_{dd_n}, f_n)\}$ of n discrete voltage and frequency levels while a common supply voltage is shared by intra-VFI processors and routers.

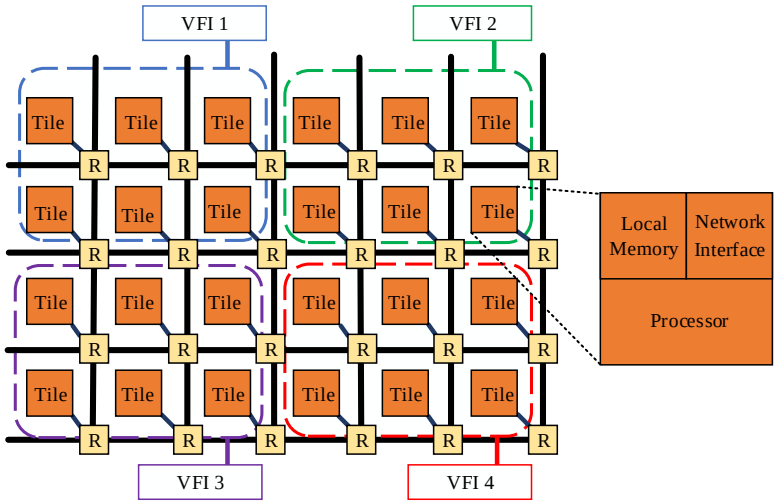


Figure 4.4: VFI based heterogeneous NoC-MPSoC architecture

1. **Communication Model:** We assume a 2D-mesh topology NoC as a communication interconnect for our VFI-NoC-HMPSoC shown in Figure 4.5. Each tile of the computing system is associated with a router to communicate with other processors. In NoC buffers are used in routers to host the incoming flits when immediate transfer to next processor and/or Intellectual property (IP) is not possible because of the congestion. NoC mesh consists of N_R rows and N_C columns therefore, number of processors in VFI-NoC-HMPSoC is equal to $N_R \times N_C$. Each router has five ports, four ports are used to communicate with the neighbor routers and one is dedicated for the purpose of communicating with the processor. A link is used to connect two routers and/or a router with a processor. We consider that all links are identical, full duplex, and have the same bandwidth, b_w.

2. **Switching and Routing Techniques:** VCT and WH are the two most popular

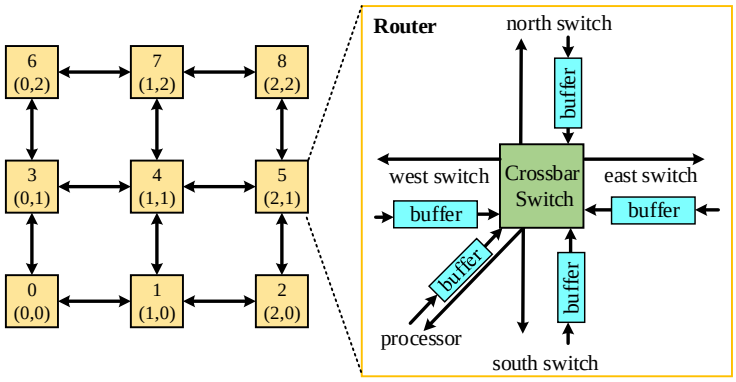


Figure 4.5: 2D-mesh topology NoC interconnect

packet switching techniques for NoC interconnects. In WH each packet is split into small pieces known as flits. When in the network a packet traverses the WH immediately determines its next hop, forwards it and then the subsequent flits worm their way through the network. In VCT routing the buffer size is large and the entire packet is sent to the next node. Thus, VCT has lower latency, higher link utilization, and lesser packet blocking probability. Though WH switching is simple and possesses higher efficiency of flow control over VCT in case of congestion occurrence, the stalling packet can block all the links and produces a low link utilization. Therefore, we consider VCT packet switching technique in this chapter. Routing technique in a network decides the path of a packet from source to the destination router. We use a well known XY deterministic routing on NoC which is most suitable option for 2D-mesh topology networks. Moreover, XY routing is a simple but yet effective approach. In XY routing the packets at the routers are routed in X-direction first and later on in the Y-direction.

3. **Energy Model:** We adopt the energy model described by Ogras et al. [205]. The total energy consumption due to an application running on multiprocessor systems is the sum of processing, E_p and communication energy consumptions E_c. The parameter E_p is the energy consumed in the execution of tasks on the processor, whereas inter-processor communication energy is consumed in transmission of communications on the network that includes routers, links, and buffers. E_p and E_c are discussed in detail in [205].

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

$$E = E_p + E_c \tag{4.12}$$

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

4.1.3 Wireless Sensor Network

Offline/static SNs clustering is performed considering the following energy and network scenarios:

1. **Energy Model:** We adopt the energy model used in [69, 151, 156, 206]. In this energy model the total energy consumption (E) is due to the energy dissipated by transmitter denoted by $E_{TX}(l,d)$ and receiver represented by $E_{RX}(l)$ given as follows.

$$e_{total}(l,d) = E_{TX}(l,d) + E_{RX}(l) \tag{4.13}$$

where $E_{TX}(l,d)$ occurs to run the power amplifier and radio electronics. Similarly, $E_{RX}(l)$ is the energy consumption due to running the radio electronics. For each sensor node in the network, the transmitter energy for transmitting a data of l bit is given as follows:

$$E_{TX}(l,d) = \begin{cases} l \times E_{elec} + l \times e_{fs} \times d^2 & d \leq d_o \\ l \times E_{elec} + l \times e_{mp} \times d^4 & d \geq d_o \end{cases} \tag{4.14}$$

where E_{elec} shows the energy consumed per bit for running the transmitter or receiver circuit, e_{fs} and e_{mp} represent the amplification energy for free space model and multi-path model while d_o denotes the threshold transmission distance and generally its value is $d_o = \sqrt{\frac{e_{fs}}{e_{mp}}}$. The transmitter energy consumption is dependent on the

distance parameter d and the amount of data to be sent. If the data transmission distance is within the threshold range then transmittance energy is proportional to d^2 otherwise this relationship increases to d^4. Thus distance and workload play a significant role in the sensor nodes clustering for network LT improvement.

Now the energy dissipation of the receiver to receive l bit of data is estimated as follows:

$$E_{RX}(l) = l \times E_{elec} \tag{4.15}$$

where E_{elec} is dependent on factors such as modulation, filtering, digital coding, and signal spreading.

2. **Network Model:** We consider a WSN that has n number of sensor nodes and a BS. We adopt the wireless network model presented in [69, 199, 207, 208] and this scenario of the WSN possesses the following properties.

   i. The sensor nodes are randomly distributed in 2D sensing field that is xy plane such that there are no more than one sensor node existing on a single point.

   ii. All the sensor nodes are homogeneous and possess similar processing and communication capabilities while consume same amount of energy for processing and communicating a data of l bit.

   iii. After deploying the sensor nodes they are considered to be stationary with respect to the BS while each sensor node in the sensing field has the capability and equal probability to operate as a non-CH (normal node) or CH.

   iv. Each sensor node has always a data ready to be sent to its CH. The number of sensor nodes in the sensing field is always higher than the number of CHs.

   v. In the clustered based WSN architecture, we assume that each CH aggregates the collected data and transmits this aggregated data to the base station.

   vi. The sensor nodes can use various transmission power levels depending upon the distance to which the collected data to be sent.

4.2 Research Method

The total energy consumption of a wireless network is due to sensing, processing and communication energy dissipation given as follows:

$$E_{network} = E_{sensing} + E_{processing} + E_{communication} \tag{4.16}$$

The sensing energy consumption can be reduced at the fabrication level which is out of the scope of this thesis. However processing energy dissipation can be minimized by proper task scheduling at the node level. Similarly transmission energy consumption can be significantly reduced by nodes clustering at the network level. Clustering may be formulated as a scheduling problem and population based algorithms/heuristics developed for scheduling can be extended to perform nodes clustering.

In this thesis we first, perform task scheduling to reduce the processing energy consumption at the node level considering NoC-MPSoC and VFI-NoC-MPSoC heterogeneous computing architectures. Second, we implement clustering at the network level to minimize the transmission energy consumption. Thus these two steps significantly reduce the overall network energy consumption while prolongs the LT of the network. The methodology of the task scheduling and nodes clustering is explained as follows:

4.2.1 Scheduling

A real-time application in IoT with precedence and deadline constraints represented by DAG is scheduled on multiprocessor system considering two different architecture i.e. NoC-MPSoC and VFI-NoC-MPSoC heterogeneous computing systems. The energy-efficiency of scheduling is dependent on task mapping, task ordering, and voltage scaling. Therefore, all these parameters are considered during scheduling to achieve maximum energy savings. Unlike other scheduling we perform task mapping, task ordering, and voltage assignment in an integrated manner to minimize the processing energy while explicitly reducing contention between the communications and inter-processor communication energy. Different novel meta-heuristics are

developed to perform task scheduling on the MPSoCs. The important steps involved in our task scheduling on MPSoCs are explained as follows:

4.2.1.1 Task Mapping

Heterogeneous MPSoCs consist of processors that have different energy-performance profiles and may operate at different voltage and frequency levels. Thus tasks are mapped on the processors of the MPSoCs by considering the energy performance profiles of the processors such that the individual deadline of each task is not violated. Moreover, dependent tasks are either mapped on the same processor or close to each other in order to reduce the inter-processor communication energy consumption.

4.2.1.2 Task Ordering

During task scheduling as deadline and precedence constraints of the tasks must be observed. Thus, the order in which tasks and communications execute may significantly impact the overall energy consumption. Therefore we save considerable amount of energy by prioritizing tasks and communications with shorter deadlines over tasks and communications with longer deadlines, because the slack available for tasks can be efficiently utilized by DVFS technique to assign low voltages and frequencies to them.

4.2.1.3 Voltage Scaling

We assign near optimal voltage to tasks on the processors in order to efficiently avail the processor slack and reduce the overall processing energy consumption. Similarly, we minimize the communication energy by assigning discrete voltage levels to the NoC links. Further, total energy efficiency is achieved by putting the processor into a low-power state when feasible. Moreover, we resolve the contention between communications that traverse the same link by allocating links to communications with higher priority.

4.2.2 Clustering

We reduce the transmission energy consumption by performing clustering applicable to any-sized network based using a meta-heuristic which is a population based algorithm. The steps involved in clustering are explained as follows:

1. The suitable positions for a particular set of CHs are determined using our proposed population based algorithms. Our clustering algorithm considers residual energy, different distance parameters, and workload on the CHs during cluster formation.

2. In the this step each SN that has not chosen its CH tentatively selects one by one each CH and calculates the rank (Rank describes the LT of the network, explained in details in Chapter 7).

3. The SNs join a particular CH based on the value of the ranking.

Concisely, we formulate Nodes clustering in itself as a scheduling problem and reduce the overall transmission energy consumption of the network.

4.3 Experimental Setup and Data Collection

In this section we briefly explain our experimental setup and discuss how quantitative data is produced. Moreover, we also discuss how heterogeneous computing platform is generated then we describe the tools and resources used for data generation.

4.3.1 Experimental Setup

The experimental setup used in the simulations for both node and network levels is explained as follows:

Table 4.1: Operating frequency and power consumption of type 1 and type 2 processors

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

Table 4.2: Processors power consumption modes

Type 1		Type 2	
Parameters	Value	Parameters	Value
P_a	276 mW	P_a	555 mW
P_{sleep}	80.0 μ W	P_{sleep}	180 μW
P_{sw}	385 μ W	P_{sw}	483 μ w

4.3.1.1 Node Level Experimental Setup

We use Samsung Exynos 5422 chip power and energy model for our simulations adopted from [209] moreover, we use two types of processors i.e. type 1: high-performance Cortex A15 (big) and type 2: low-power Cortex A7 (little). The Cortex A7 consumes $\sim 6-12$ times less power compared to Cortex A15 [210]. The operating frequencies and relative power consumption of both types are given in Table 4.1. The power and energy consumption for different power modes of these processors are listed in Table 4.2. Moreover, we adopt 70 nanometers (nm) processor technology parameters from Ali et al. [33] already listed in Table 2.5.

We build the simulation environment in Matlab version R2016a moreover, we conduct the experiments using hardware platform of Intel (R) Xeon (R), i5-3570 CPU with the clock frequency of 3.50 GHz and 16.00 GB memory, 10 MB cache. We use different real benchmarks in our experimental analysis for task scheduling. The real benchmarks are adopted from Embedded System Synthesis Benchmarks Suite (E3S) which is a widely used benchmark suit in the task mapping and scheduling research. The benchmarks are selected with different complexities i.e. different number of tasks while covering various applications in IoT.

4.3.1.2 Network Level Experimental Setup

We build the simulation environment in Python version 3.7.3 moreover, we conduct the experiments using hardware platform of Intel (R) Xeon (R), i5-3570 CPU with the clock frequency of 3.50 GHz and 16.00 GB memory, 10 MB cache. We perform the simulation for different number of sensor nodes various percentage of CHs at different locations of BS in the sensing field of area $200 \times 200\ m^2$. The sensor nodes change within a range of 100 to 400 and the number of selected CHs vary from 15% to 30% while BS is positioned at (50, 50), (100, 100), (150, 150). We assume that each sensor node carries equal initial residual energy of 2.0 J. Other constant parameters used in this chapter for simulations are enlisted in Table 4.3 adopted from [69, 151, 156].

Table 4.3: Various parameters used in simulation

Parameter	Value
Area of the Network	$200 \times 200\ m^2$
Base Station Location	(50, 50), (100, 100), (150, 150)
Number of Sensor Nodes	100 to 400
Initial Sensor Node Energy	2.0 J
E_{elec}	50 nJ/bit
e_{fs}	10 $pJ/bit/m^2$
e_{mp}	0.0013 $pJ/bit/m^4$
d_{max}	100 m
d_o	30 m
Packet size	4000 bits
Percentage of CHs	10% to 30%

4.3.2 Data Collection

We generate energy consumption values (represented in joule) for different real benchmarks at the node level. We generate results for different scenarios considering different metrics such as number of processors, deadline, homogeneous MPSoC platform, heterogeneous multiprocessing computing system etc. It is worth noticing that we randomly select the type of processor for generating heterogeneous computing platform in order to ensure unbiased experimentation.

We compare the the results of our scheduling algorithms with state-of-the-art Energy-efficient Contention-aware Mapping (ECM) for NoC-MPSoC architecture developed by Li and Wu

in [204]. ECM is a two-step scheduling heuristic. First, ECM formulated the application mapping problem as a quadratic binary programming problem to minimize the overall inter-processor communication energy consumption by deploying a relaxation-based iterative rounding algorithm. Once task mapping is performed then ECM determines the optimal voltage level for each task and optimal frequency level for each communication in the application in order to minimize the overall system energy consumption considering the application deadline.

Similarly, we use CA-TMES-Quick [95] and CA-TMES-Search [95] energy management algorithms as baseline to determine the performance of our static task scheduling developed for VFI-NoC-MPSoCs. CA-TMES-Quick and CA-TMES-Search select the processor for a task where it can start earliest among all processor. CA-TMES-Quick first performs task assignment and then determines routes for the communications that are sent to this task. CA-TMES-Search calculates start time for each task while considering communication contention. The processor offering earliest start time for a task is selected by CA-TMES-Search. Specifically, CA-TMES-Search relatively performs better than CA-TMES-Quick because it coordinates the task mapping in an exhaustive way therefore, make-span significantly reduces.

At the network level we consider different scenarios for our simulations to generate various results. The parameter such as number of SNs is changed in the sensing field/target area and results are produced. We also empirically observe impact of BS location and number of CHs in the results.

We compare the results of our proposed clustering algorithm with a PSO based state-of-the-art technique as a baseline developed by Rao et al. [69]. In their energy-aware clustering approach they consider parameters such as intra-cluster distance, residual energy of the sensor node, and sink distance in the fitness function. The network LT is increased by developing a PSO based energy-efficient CH selection (PSO-ECHS). The PSO-ECHS determines the optimal positions of a pre-determined numbers of CHs for a set of sensor nodes distributed over a specific area of the sensing fields. The non-CH sensor nodes in the field join their elected CHs based on a derived weight function. We also compare the performance of our algorithm with the widely used LEACH [156] and PSO-C [151]. As a performance metric we consider LT to compare

our clustering algorithm with other existing approaches. LT is the measure of the number of rounds when the first sensor nodes dissipates all of its energy.

4.4 Summary

In this chapter we presented the relevant application model and explained its extended format. Then we explained our multiprocessor architectures, network, and energy models used in our simulations for generating quantitative data to compare the energy performance. We also explained our research approach for scheduling and clustering to decrease the processing and transmission energy consumption both at the node and network levels. Furthermore, we explained that different simulation scenarios have been used to generate and analyze that data using metrics while comparing it with state-of-the-art heuristics.

Chapter 5

Energy-aware Static Task scheduling on Heterogeneous NoC-MPSoCs

In this chapter, we investigate contention-aware and energy-efficient static scheduling using heterogeneous NoC-MPSoC for real-time tasks with an individual deadline and precedence constraints. Unlike other schedulers task ordering, mapping, and voltage assignment are performed in an integrated manner to minimize the processing energy while explicitly reduce contention between the communications and communication energy. Furthermore, both dynamic voltage and frequency scaling and dynamic power management are used for energy consumption optimization. The developed Contention-aware Integrated Task Mapping and Voltage Assignment (CITM-VA) static scheduler performs tasks ordering using Earliest Latest Finish Time First (ELFTF) strategy that assigns priorities to the tasks having shorter Latest Finish Time (LFT) over the tasks with longer LFT. It remaps every task to a processor and/or discrete voltage level that reduces processing energy consumption. Similarly, the communication energy is minimized by assigning discrete voltage levels to the NoC links. Further, total energy efficiency is achieved by putting the processor into a low-power state when feasible. Moreover, this algorithm resolves the contention between communications that traverse the same link by allocating links to communications with higher priority. The results obtained through extensive simulations of real benchmarks demonstrate that CITM-VA outperforms state-of-the-art scheduling algo-

rithms and achieves an average $\sim$ 30% total energy improvement. Additionally, it maintains high Quality-of-Service (QoS) and robustness for real-time applications.

In this offline task scheduling investigation, NoC based heterogeneous MPSoC architecture with DVFS-enabled processors is considered while contention and energy-aware static scheduling for real-time DAG tasks with individual deadlines and precedence constraints is performed. Our major contributions include as follows:

1. Task mapping, task ordering, and voltage scaling are performed in an integrated manner. Unlike other approaches which map the tasks first and then assign voltage levels separately. Our proposed Contention-aware Integrated Task Mapping and Voltage Assignment (CITM-VA) algorithm guides the tasks and communications mapping to a more energy efficient solution. Moreover, both DVFS and DPM are integrated to reduce the total energy consumption.

2. The proposed CITM-VA static scheduling algorithm saves communication energy by minimizing the communication over the NoC. It further reduces the communication energy by scaling the voltage levels of the NoC links and assigns communications voltage level such that communication energy is minimized. Hence the available slack is efficiently shared between the communications and tasks. Furthermore, contentions among concurrent tasks and communications are resolved by prioritizing the execution of high priority tasks and communications over low priority ones.

3. Our experimental results are generated from simulations conducted on five real-world benchmarks adopted from Embedded Systems Synthesis Benchmarks (E3S) [211]. The results are compared to state-of-the-art Energy-efficient Contention-aware Mapping (ECM) static scheduler developed by Li and Wu in [204]. The proposed CITM-VA meta-heuristic outperforms ECM in terms of energy savings and QoS. Compared to ECM, CITM-VA reduces the average total energy consumption by $\sim$ 30%.

The rest of the chapter is organized as follows: Section 5.1 presents our proposed static contention and energy-aware scheme. The results examined in Section 5.2, and Section 5.3 con-

cludes this chapter.

5.1 Contention and Energy-aware Approach

In this section we discuss our proposed contention-aware energy-efficient task scheduling for a set of tasks with precedence and deadline constraints represented by DAG. The heterogeneous MPSoC architecture presented in Section 4.1.2.1 is adopted.

Heterogeneous-MPSoCs consist of processors that have different power-performance profiles and may operate at different voltage and frequency levels. Moreover, deadline and precedence constraints of the tasks must be observed. Thus, the order in which tasks and communications execute may significantly impact the overall energy consumption. Considerable amount of energy may be saved by prioritizing tasks and communications with shorter deadlines over tasks and communications with longer deadlines, because the slack available for tasks can be efficiently utilized by DVFS algorithm to assign low voltages and frequencies to them. Hence, the quality of solution obtained by energy efficient scheduling of tasks with precedence and individual deadline constraints on heterogeneous NoC-MPSoC is influenced by three factors: task mapping, ordering, and voltage assignment. Furthermore, DPM also plays a vital role when processor in idle mode because it increases the energy-efficiency by switching the processors into a low-power mode. The state-of-the-art approach presented in [204] performs task orderings and voltage scaling in an integrated manner and performs task mapping separately. However, we believe that task and communication ordering and voltage scaling can be helpful in steering the task mapping optimization process towards a more energy efficient solution. This is one of the major factors that we consider in design of our CITM-VA algorithm. The CITM-VA algorithm considers mapping, scheduling, and voltage scaling in an integrated manner while also deploys DPM for achieving maximum energy savings.

Table 5.1 defines some terms and notations used by our algorithms. The details of CITM-VA are given in Algorithm 1. There are four steps in the implementation of our energy optimization CITM-VA algorithm outlined as follows:

Table 5.1: Terms and notations

Term/ Notation	Definition
κ	Total number of mapping solutions to the problem.
Π	A matrix of κ mapping. Each row represents processor to which task v_i is mapped, $1 \leq i \leq \|V\|$.
Ψ	A matrix of κ voltage assignments. Each row represents the voltage level of task or communication node v_i, $1 \leq i \leq \|V\|\|E\|$.
$\Pi[\eta][i]$	Processor where task v_i is mapped for η mapping.
$\Psi[\eta][i]$	Voltage level assigned to task v_i for η voltage assignment.
$rand()$	Returns a uniform random number in interval $(0, 1)$.
Ω	Maximum number of iterations (Pre-defined).
$Stagnation$	No improvement achieved in energy reduction for a predefined number of iterations
$Solution$	A row from Π and the same row from Ψ is together is a solution.
G'_e	A copy of the extended graph G_e
$cSet(v_i)$	Set of concurrent nodes to v_i
$ISuc(v_i)$	Set of immediate successors of v_i.
PRI	It shows priority of the tasks
$NULL$	Processor is idle

Step 1 Initial Solutions: We first generate two matrices Π and Ψ of dimensions $\kappa \times |V|$ and $\kappa \times |V||E|$ where κ is an input parameter of $CITM - VA$ algorithm. Each row of matrix Π is generated by randomly mapping a task node v_i to a random processor, $1 \leq i \leq |V|$ (Line 5). Similarly a row of matrix Ψ is generated by assigning maximum processor voltage where a task v_i is mapped, in case if v_i is a task node and maximum link voltage if v_i is a communication node $1 \leq i \leq |V||E|$ (Lines 6-7). Each row of matrix Π and Ψ together forms a solution. For each solution we next compute its fitness value by constructing a schedule using Earliest Latest Finish Time First $ELFTF$ algorithm (Lines 8-11). The $ELFTF$ algorithm returns the energy and feasibility of each solution. Given the feasibility and energy of a solution, its fitness value is computed as follows:

$$Fitness(feasible, e) = \begin{cases} \frac{1}{e} & \text{if feasible is true} \\ -\infty & \text{otherwise} \end{cases} \tag{5.1}$$

Step 2 Initial Solutions Refinement: In each iteration of the while loop (Lines 12-40) we sort the solutions in non-increasing order of fitness value and delete $\lfloor \frac{1}{3}\kappa \rfloor$ solutions with smaller fitness value. We then randomly select $\lfloor \frac{1}{3}\kappa \rfloor$ solutions from the remaining solutions to generate $\lfloor \frac{1}{3}\kappa \rfloor$ new solutions (Lines 32-40). We generate a new solution from an existing solutions by re-mapping a task node to a processor and/or voltage level or a communication to a voltage level such that total energy is minimized and deadlines are satisfied by using $ReMap$ algorithm (Line 38). If such a task or communication node is found then the solution is updated to from new solution otherwise the original solution is the new solution (Lines 2-40 Algorithm $ReMap$). We calculate the fitness value of each solution and the new solutions are merged with the existing solutions to form the κ solutions for the next iteration.

Step 3 Stagnation Control: Notice that $CITM-VA$ algorithm follows an elitist approach as the best solution in each iteration is always kept. Like all elitist approaches our $CITM-VA$ is also prone to stagnation. We assume that $CITM-VA$ has converged to a stagnation state if no improvement in energy reduction is observed for a predefined number of iterations. The $CITM-VA$ algorithm is checked against stagnation. If stagnation state is detected then we use the following stagnation control technique to climb out of the local optimum (Lines 20-31). We first delete $\lfloor \frac{1}{2}\kappa \rfloor$ solutions with smaller fitness value (Line 22). From the remaining solution we generate $\lfloor \frac{1}{2}\kappa \rfloor$ new solutions. We generate each new solution from an existing solution by selecting $\phi\%$ of tasks and randomly remap them to other processors at maximum voltage levels (Lines 25-28). The fitness value of each newly generated solution is computed and the newly created solutions are merged with the existing solution to form κ solutions for next generation (Lines 29-31).

Step 4 Termination: $CITM-VA$ algorithm terminates when a predefined maximum number of iterations Ω have been reached. Before it terminates we check if the solutions with the maximum fitness is feasible. If it is feasible then it is the solution to the given problem otherwise $CITM-VA$ cannot find a solution (Lines 15-20).

Algorithm 2 illustrates the Earliest Latest Finish Time First ($ELFTF$) algorithm that we utilize to construct a schedule given task and voltage mapping. For each task and communication node

Algorithm 1: CITM-VA

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

output: Task to processor mapping *map* and voltage levels *vol* corresponding to tasks and communications.

Construct two matrices Π and Ψ of zeros having dimensions $\kappa \times |V|$ and $\kappa \times |V||E|$ respectively;
for $\eta \leftarrow 1$ *to* κ **do**
  for ***all*** $v_i \in V$ **do**
    $\Pi[\eta][i] \leftarrow \lceil rand()(|P|-1)+1 \rceil$;
    $\Psi[\eta][i] \leftarrow$ Maximum Voltage Level;
  for ***all*** $e_i \in E$ **do**
    $\Psi[\eta][i] \leftarrow$ Maximum Link Voltage;
for $\eta \leftarrow 1$ *to* κ **do**
  Construct an extended graph G_e given a mapping ;
  $[f, e] \leftarrow ELFTF(G_e, \Pi, \Psi, \eta)$;
  Compute the $fitness(f, e)$ value of the solution according to equation (5.2) ;
while 1 **do**
  Sort solutions in descending order of their fitness values;
  if *stopping criteria satisfied* **then**
    if *the solution with highest fitness feasible* **then**
      Copy first row of Π to map and Ψ to *vol*;
    else
      No feasible solution found;
    break;
  if *stagnation detected* **then**
    Sort solutions in descending order of their fitness values;
    Delete $\lfloor \frac{1}{2}\kappa \rfloor$ solutions with smaller fitness value;
    Construct two matrices Π' and Ψ' of zeros having dimensions $\lfloor \frac{1}{2}\kappa \rfloor \times |V|$ and $\lfloor \frac{1}{2}\kappa \rfloor \times |V||E|$;
    for $j \leftarrow 1$ *to* $\lfloor \frac{1}{2}\kappa \rfloor$ **do**
      $\phi \leftarrow rand()$;
      Copy j^{th} row of Π and Ψ to j^{th} row of Π' and Ψ' respectively;
      Randomly re-map $\lfloor \phi |V| \rfloor$ tasks in j^{th} row of Π' to other processors at maximum voltage level;
      Construct the extended graph G_e for mapping given by j^{th} row of Π';
      $[f, e] \leftarrow ELFTF(G_e, \Pi', \Psi', j)$;
      Compute the fitness $fitness(f, e)$ of the j^{th} solution;
      Merge the matrix Π' with Π and Ψ' with Ψ;
  Delete $\lfloor \frac{1}{3}\kappa \rfloor$ solutions with smaller fitness values;
  Construct two matrices Π' and Ψ' of zeros having dimensions $\lfloor \frac{1}{3}\kappa \rfloor \times |V|$ and $\lfloor \frac{1}{3}\kappa \rfloor \times |V||E|$;
  for $j \leftarrow 1$ *to* $\lfloor \frac{1}{3}\kappa \rfloor$ **do**
    $\eta \leftarrow \lceil rand()(\kappa - \lfloor \frac{1}{3}\kappa \rfloor - 1)+1 \rceil$;
    Copy the η row of Π and Ψ to j^{th} row of Π' and Ψ' respectively;
    Construct the extended graph G_e for mapping given by j^{th} row of Π';
    $[f, e] \leftarrow ReMap(G_e, \Pi', \Psi', j)$;
    Compute the fitness $fitness(f, e)$ of the j^{th} solution;
  Merge the matrix Π' with Π and Ψ' with Ψ;

we first compute the latest finish time (Lines 2-12). We then schedule nodes based on their priorities represented by PRI where PRI is equal to the sum of latest finish time and the earliest start time (16-33). Nodes with smaller value of PRI have higher priority over nodes with larger value of PRI. The nodes with higher priorities are scheduled earlier in time compared to nodes with lower priorities. To achieve this goal we first define a set named $cSet(v_i)$. If v_i is a task node $cSet(v_i)$ is a set of task nodes concurrent to v_i (task nodes not reachable from v_i in G_e) and mapped on the same processor where v_i is mapped otherwise it is a set of communication nodes concurrent to v_i and have conflict with v_i (communication nodes have conflicts if they use same NoC links). Therefor, after we have scheduled a node the earliest start time of all the nodes that belong to $cSet(v_i)$ and $ImSuc(v_i)$ are updated to finish time of v_i. This ensures that no two nodes that are concurrent are scheduled at the same time and the nodes with smaller priority are scheduled later than nodes with higher priority. The schedule feasibility is determined and its energy is also computed in the process.

Algorithm 3 shows the $ReMap$ algorithm that we use to choose the best node to reassign and also the processor and voltage to reassign it to for a given solution. The algorithm consider all the nodes in the extended graph. If the node is a task node we tentatively remap it to all the processors and on all voltage levels on those processor (6-20) and in case of a communication node we tentatively remap it on all the voltages of the links (Lines 22-34). Each time we remap a node to a new processor and or voltage level we invoke $ELFTF$ algorithm to determine the feasibility and energy consumption of the schedule. We calculate the $Rank(v_i)$ of the new schedule generated by remapping the node v_i if it is feasible as follows (Line 16 or 29):

$$Rank(v_i) = \begin{cases} \frac{e - E^{new}}{\frac{NC(i,k)}{f_i^{new}} - \frac{NC(i,k)}{f_i^{cur}}} & \text{if } v_i \text{ is task node} \\ \frac{e - E^{new}}{\frac{x_j}{bw f_i^{new}} - \frac{x_j}{bw f_i^{cur}}} & \text{otherwise} \end{cases} \tag{5.2}$$

The node with the maximum rank is selected and the solution is updated. Note that we don't invoke $ELFTF$ algorithm for all voltage levels of a processor or link. Rather we first find a maximum voltage level V_{dd}^L such that energy consumption at this voltage is lower than the current energy consumption (Lines 6 and 22). The voltage level higher than V_{dd}^L are not

Algorithm 2: ELFTF

input : An extended DAG G_e, matrix Π, matrix Ψ and current chromosome index η
output: Schedule feasibility indicated by binary variable $Feasible$, total energy e_T.

$Feasible \leftarrow True$; $e_T \leftarrow 0$;
for *each* v_i *in reverse topological order of* G_e **do**
  if v_i *is a task node* **then**
    for *each* $v_j \in ISuc(v_i)$ **do**
      Compute frequency f_j of node v_j corresponding to voltage level $\Psi[\eta][j]$;
      if v_j *is a task node* **then**
        $LFT_i \leftarrow \min\{\min\{LFT_i, d_i\}, LFT_j - \frac{NCC(j,\Pi[\eta][j])}{f_j}\}$;
      else
        $LFT_i \leftarrow \min\{\min\{LFT_i, d_i\}, LFT_j - \frac{x_j}{f_j b_w}\}$;
  else
    Compute frequency f_j of task $v_j \in ISuc(v_i)$ corresponding to voltage level $\Psi[\eta][j]$;
    $LFT_i \leftarrow LFT_j - \frac{NCC(j,\Pi[\eta][j])}{f_j}$;
Create a copy G'_e of G_e;
$\forall v_i \in G_e\ est_i \leftarrow 0$;▷ Set the earliest start time (est) of all the tasks and communication nodes to zero
Insert all the source nodes in G'_e to ready queue R;
while R *is not empty* **do**
  for *each* $v_i \in R$ **do**
    $PRI_i \leftarrow est_i + LFT_i$;
  Select the node v_i with the smallest value of PRI from R;
  if v_i *is a task node* **then**
    $e_T \leftarrow e_T + E(i, \Psi[\eta][i], \Pi[\eta]][i])$; $t_i^{start} \leftarrow est_i$;
    $t_i^{finish} \leftarrow t_i^{start} + \frac{NC(i,\Pi[\eta][i])}{f_i}$;
    for *each* *unscheduled node* $v_j \in cSet(v_i) \cup ISuc(v_i)$ **do**
      $est_j \leftarrow \max\{est_j, t_i^{finish}\}$;
  else
    $e_T \leftarrow e_T + E_c(i, \Psi[\eta][i])$;$t_i^{start} \leftarrow est_i$;
    $t_i^{finish} \leftarrow t_i^{start} + \frac{vol_i}{l_w f_i}$;
    for *each* *unscheduled node* $v_j \in cSet(v_i) \cup ISuc(v_i)$ **do**
      $est_j \leftarrow \max\{est_j, t_i^{finish}\}$;
  if $t_i^{finish} > d_i$ **then**
    $Feasible \leftarrow False$;
  Delete v_i and all its outgoing edges from G'_e;
  Insert all the source nodes in G'_e to R;

considered.

There may be slack available after voltage scaling. As the processor can remain in active state during the idle period or it can switch to sleep state. This is determined by Algorithm 4, Slack

Algorithm 3: ReMap

input : A DAG G_e, matrix Π, matrix Ψ, index η.
output: Schedule feasibility indicated by f, Energy e.

$Rank^{select} \leftarrow -\infty; \tau \leftarrow -1; \rho \leftarrow -1; \Gamma \leftarrow -1; e \leftarrow \sum_{\forall v_i \in V} E(i, \Pi[\eta][i], \Psi[\eta][i]) + \sum_{\forall v_i \in V^*} E_c(i, \Psi[\eta][i])$;
$f \leftarrow true$ if current solution if feasile otherwise $false$;
for ***each*** $v_j \in G_e$ **do**
  if *v_i is a task node* **then**
    for ***each*** $pe_k \in P$ **do**
      Find the highest voltage level V_{dd}^L on pe_k that minimize total energy consumption;
      $\Upsilon \leftarrow \{V_{dd}^{min}, \ldots, V_{dd}^L\}$; ▷ V_{dd}^{min}: lowest voltage, voltage levels higher than V_{dd}^L not considered
      for ***each*** $V_{dd_s} \in \Upsilon$ **do**
        $tempProc \leftarrow \Pi[\eta][i]$; $\Pi[\eta][i] \leftarrow k$; ▷ Remap task to processor pe_k
        Update G_e according to new mapping;
        $tempVol \leftarrow \Psi[\eta][i]$; $\Psi[\eta][i] \leftarrow s$;
        $[Feasible, E^{new}] \leftarrow ELFTF(G_e, \Pi, \Psi, \eta)$;
        $\Psi[\eta][i] \leftarrow tempVol$; $\Pi[\eta][i] \leftarrow tempProc$;
        if $Feasible$ **then**
          Compute frequencies f_i^{cur} and f_i^{new} corresponding to voltages $\psi[\eta][i]$ and V_{dd_s} respectively;
          $Rank(v_i) \leftarrow \frac{e - E^{new}}{\frac{NC(i,\Pi[\eta][i])}{f_i^{new}} - \frac{NC(i,\Pi[\eta][i])}{f_i^{cur}}}$;
          if $Rank(v_i) > Rank^{select}$ **then**
            $Rank^{select} \leftarrow Rank(v_i); \tau \leftarrow i; \rho \leftarrow s$; $e \leftarrow E^{new}; \Gamma \leftarrow k$;
        else
          break;
  else
    Find the highest link voltage level V_{dd}^L that minimizes the total energy consumption;
    $\Upsilon \leftarrow \{V_{dd}^{min}, \ldots, V_{dd}^L\}$;
    for ***each*** $V_{dd_s} \in \Upsilon$ **do**
      $tempVol \leftarrow \Psi[\eta][i]$; $\Psi[\eta][i] \leftarrow s$;
      $[Feasible, E^{new}] \leftarrow ELFTF(G_e, \Pi, \Psi, \eta)$;
      $\Psi[\eta][i] \leftarrow tempVol$;
      if $Feasible$ **then**
        Compute frequencies f_i^{cur} and f_i^{new} corresponding to voltages $\psi[\eta][i]$ and V_{dd_s} respectively;
        $Rank(v_i) \leftarrow \frac{e - E^{new}}{\frac{x_i}{b_w f_i^{new}} - \frac{x_i}{b_w f_i^{cur}}}$;
        if $Rank(v_i) > Rank^{select}$ **then**
          $Rank^{select} \leftarrow Rank(v_i); \tau \leftarrow i; \rho \leftarrow s$; $e \leftarrow E^{new}$;
      else
        break;
if $\tau \neq -1$ **then**
  if *v_i is a task node where $i = \tau$* **then**
    $\Pi[\eta][\tau] \leftarrow \Gamma; \Psi[\eta][\tau] \leftarrow \rho; f \leftarrow true$;
  else
    $\Psi[\eta][\tau] \leftarrow \rho; f \leftarrow true$;

Power Management (SPM). After scheduling a node we call SPM (in ELFTF, Line 22). We first determine the length of idle time slot, t_{idle} (Line 1) and then calculate break even time t_{BET} (Line 2). If idle interval is greater than or equal to break even time then the processor is switched to sleep mode otherwise the processors stays in active mode (Lines 3-6).

Algorithm 4: SPM

input : Nodes v_i and v_j
output: Energy E consumed during the idle interval.

1 $t_{idle} \leftarrow \rho_j - \zeta_i$; ▷ If v_i is NULL then $\zeta_i \leftarrow 0$
2 Calculate t_{BET} using equation (4.7);
3 **if** $t_{idle} \geq t_{BET}$ **then**
4 $\quad E \leftarrow (t_{idle} - t_{sw})P_{sleep} + E_{sw}$;
5 **else**
6 $\quad$ **return** $E \leftarrow E_{idle}$;

5.2 Results and Discussions

In this section of this chapter we generate and discuss different results to compare with state-of-the-art energy management techniques. Five real benchmarks listed in Table 5.2 are selected from E3S. The experimental setup is explained in details in Section 4.3.1.1.

Table 5.2: Real benchmarks description

Real-world benchmark	No.of Tasks
MP3-encoder	16
Consumer	7
Networking	4
Office-automation	5
Auto-industry	9

5.2.1 CITM-VA Energy performance

Different results have been generated considering the parameters such as dynamic energy (d), static energy (s), number of tiles, and makespan. Table 5.3 shows the dynamic energy consumption of the real benchmarks using a different number of tiles. Dynamic plus static energy consumption is summarized in Table 5.4. Where the energy consumptions at $0.9\times(Makespan)_{ECM}$

Table 5.3: Real benchmarks dynamic energy consumption in Joule (J)

Application	20-Tiles		24-Tiles		28-Tiles	
Benchmark	$E_{CITM-VA}$	E_{ECM}	$E_{CITM-VA}$	E_{ECM}	$E_{CITM-VA}$	E_{ECM}
MP3-encoder	1.139	1.524	1.074	1.470	0.971	1.520
Consumer	0.309	0.382	0.302	0.377	0.296	0.371
Networking	0.401	0.533	0.384	0.517	0.358	0.522
Office-automation	0.257	0.346	0.249	0.341	0.244	0.337
Auto-industry	0.448	0.599	0.438	0.590	0.433	0.586

Table 5.4: Real benchmarks dynamic+static energy consumption in Joule (J)

Application	20-Tiles		24-Tiles		28-Tiles	
Benchmark	$E_{CITM-VA}$	E_{ECM}	$E_{CITM-VA}$	E_{ECM}	$E_{CITM-VA}$	E_{ECM}
MP3-encoder	1.415	1.994	1.402	1.960	1.361	1.941
Consumer	0.389	0.497	0.374	0.488	0.363	0.483
Networking	0.496	0.679	0.480	0.665	0.465	0.658
Office-automation	0.328	0.463	0.315	0.454	0.306	0.450
Auto-industry	0.549	0.778	0.527	0.762	0.511	0.753

and Makespan Multiplier (MM) ranging from 0.9 to 1.1 with 0.05 step are listed in Table 5.5, and Table 5.6 respectively.

5.2.1.1 5×4 NoC

Figure 5.1 shows the energy performance comparison of 5 real benchmarks using 5×4 NoC, i.e. 20 tiles. The x-axis represents the benchmarks (task graphs) and y-axis denotes the energy consumption in joules (J). The proposed $CITM - VA$ performs better than ECM in terms of energy savings. Numerically $(CITM - VA)_d$ achieves $\sim$ 23% total energy-efficiency over $(ECM)_d$. This energy performance improvement is because unlike $(ECM)_d$ our approach, $(CITM - VA)_d$ deploys heterogeneous MPSoC architecture while explicitly considers energy performance profiles of the processors and generates a feasible solution with lower energy consumption. Moreover, $(CITM - VA)_d$ performs mapping, scheduling, and voltage scaling in an integrated manner to guide the given task mapping problem to a more energy-efficient solution. Some task nodes after execution on MPSoC platform may produce idle slack on the processors which is not addressed by $(ECM)_{d+s}$. Contrarily, beside the mapping and voltage

Table 5.5: Real benchmarks dynamic+static energy in joule (J) at $0.9 \times (makespan)_{ECM}$

Application	20-Tiles		24-Tiles		28-Tiles	
Benchmark	$E_{CITM-VA}$	E_{ECM}	$E_{CITM-VA}$	E_{ECM}	$E_{CITM-VA}$	E_{ECM}
MP3-encoder	1.563	—	1.507	—	1.479	—
Consumer	0.411	—	0.401	—	0.409	—
Networking	0.527	—	0.510	—	0.496	—
Office-automation	0.347	—	0.332	—	0.324	—
Auto-industry	0.583	—	0.561	—	0.550	—

Table 5.6: Energy consumption (J) at different MM values using 28 tiles

	Benchmarks									
MM	MP3-encoder		Consumer		Networking		Office-automation		Auto-industry	
	$CITM-VA$	ECM	$CITM-VA$	ECM	$ECITM-VA$	ECM	$CITM-VA$	ECM	$CITM-VA$	ECM
0.90	1.479	—	0.409	—	0.496	—	0.324	—	0.550	—
0.95	1.403	—	0.388	—	0.481	—	0.317	—	0.534	—
1.00	1.361	1.941	0.363	0.483	0.465	0.658	0.306	0.450	0.511	0.753
1.05	1.258	1.873	0.357	0.471	0.450	0.642	0.289	0.435	0.490	0.732
1.10	1.187	1.805	0.345	0.458	0.441	0.624	0.278	0.420	0.473	0.718

scaling $(CITM-VA)_{d+s}$ also deploys DPM strategy to minimize the energy consumption during processor idle slack period. An average $\sim 25\%$ increase in the total energy consumption $(CITM-VA)_{d+s}$ occurs compared to $(CITM-VA)_{d}$ because of the leakage power consumption. Compared to $(ECM)_{d+s}$ the proposed energy management approach $(CITM-VA)_{d+s}$ saves an average $\sim 27\%$ total energy.

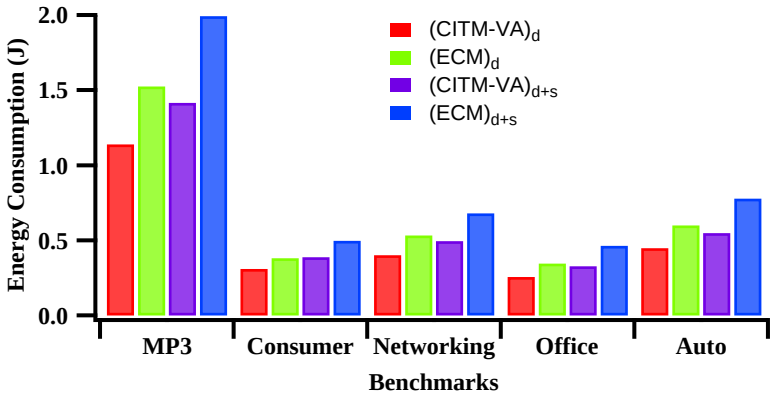


Figure 5.1: 5×4 NoC containing 20 tiles

5.2.1.2 6×4 NoC

The energy-efficiency of $CITM-VA$ is further investigated for all 5 benchmarks when number of tiles are increased to 24 (from previously 20 tiles) as shown in the Figure 5.2. A reduction in the overall energy consumption occurs when number of tiles are changed from 20 to 24. For example, MP3 benchmark consumed 1.139 J, 1.415 J for $(CITM-VA)_d$ and $(CITM-VA)_{d+s}$ respectively using 20 tiles NoC. These energy values reduce to 1.074 J and 1.402 J for $(CITM-VA)_d$ and $(CITM-VA)_{d+s}$ respectively when 6×4 NoC with 24 tiles is deployed. This energy reduction occurred due to the availability of more number of low energy performance processors for task mapping. Relatively a smaller decrease in energy consumption resulted for ECM because it uses MPSoC platform containing homogeneous processors with the same energy performance profiles. The proposed approach $(CITM-VA)_d$ achieves an average $\sim$ 25% energy improvement compared to $(ECM)_d$. Moreover, $\sim$ 30% energy-efficiency is attained by $(CITM-VA)_{d+s}$ over $(ECM)_{d+s}$.

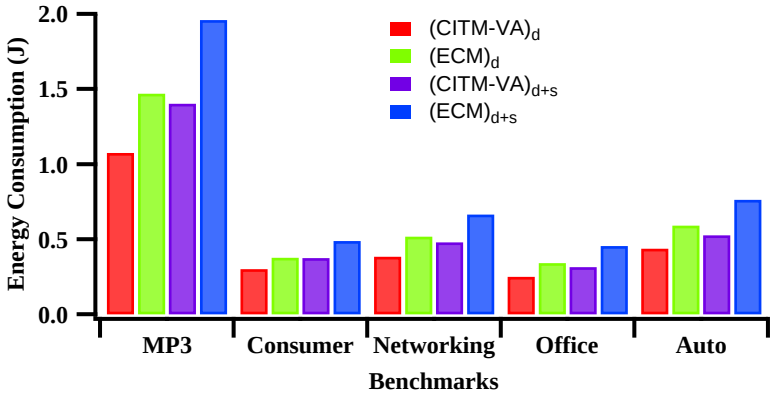


Figure 5.2: 6×4 NoC containing 24 tiles

5.2.1.3 7×4 NoC

Figure 5.3 demonstrates the energy consumption comparison when 7×4 NoC with 28 heterogeneous processors on MPSoC architecture is used. A similar pattern like Figure 5.2 is followed by the energy consumption in this case though, the energy savings increases for both $(CITM-VA)_d$ and $(CITM-VA)_{d+s}$ respectively. For instance the energy consumption of

$(CITM-VA)_d$ for MP3 reduces to 0.971 J from 1.074 J and $(CITM-VA)_{d+s}$ also minimizes to 1.361 J from 1.402 J when number of tiles are increased from 24 to 28. A similar decrease in the energy consumption is observed for the rest of the real benchmarks. The energy consumption reduction using $CITM-VA$ is steeper than ECM. Moreover, $(CITM-VA)_d$ and $(CITM-VA)_{d+s}$ improve the energy-efficiency by $\sim 27\%$ and $\sim 33\%$ respectively over state-of-the-art $CITM-VA$.

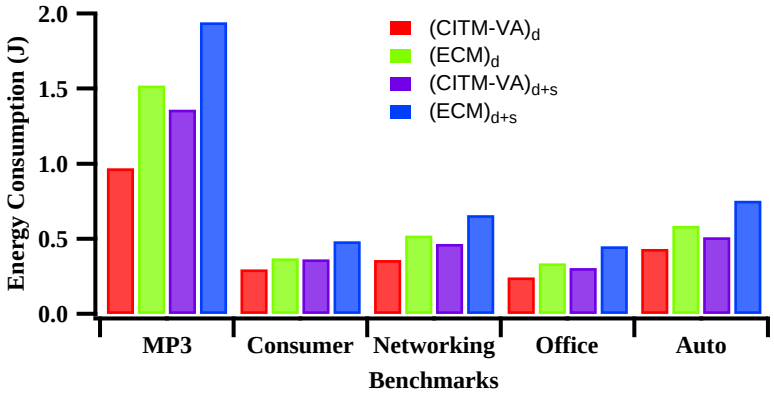


Figure 5.3: 7×4 NoC containing 28 tiles

5.2.1.4 Robustness and QoS

The robustness and QoS of both the schemes are analyzed when the makespan generated by ECM is considered as baseline and multiplied with MM of 0.9 as demonstrated in Figure 5.4. The ECM approach can not produce energy consumption approximation at $0.9 \times makespan$ because it does not implement ELFTF strategy to re-arrange the task nodes order according to the deadline set. Therefore, QoS degrades at strict deadline and ECM fails to efficiently perform task mapping and voltage scaling at strict deadlines for real applications. Thus, ECM shows no robustness and exibits poor QoS. Contrarily, $CITM-VA@(0.9 \times makespan)_{ECM}$ converges at strict deadlines and produces energy consumption values. Though the energy-efficiency reduces to $\sim 18\%$, $\sim 24\%$ for $(CITM-VA)_d$ and $(CITM-VA)_{d+s}$ compared to the normal makespan.

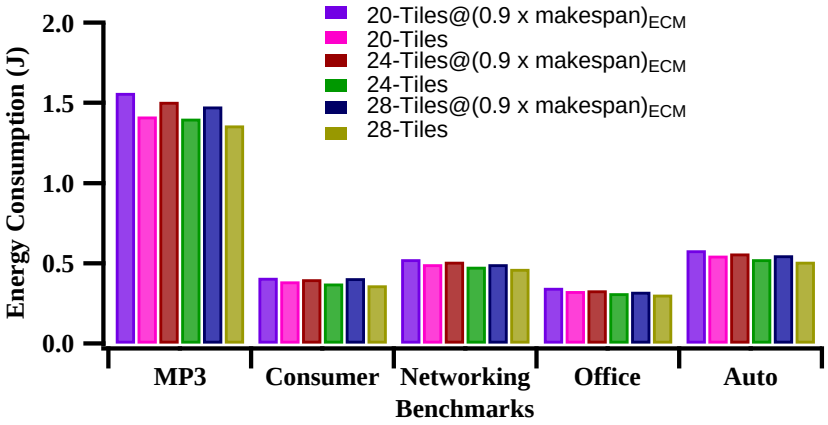


Figure 5.4: Robustness

5.2.1.5 Energy and MM

Figure 5.5 illustrates the total energy consumption of real benchmarks for different makespan when 7×4 NoC with 28 tiles is used. The total energy consumption of both $CITM - VA$ and ECM decreases when MM value is increased from 0.9. This reduction of energy consumption is due to the expansion in the common deadline and lower discrete voltage levels can be applied to the tasks and communication nodes. It is worth noticing that $CITM - VA$ generates energy consumption values even at strict deadlines while ECM does not converge and fails to produce output below 1.0 value of MM. Moreover, the total energy consumption of ECM for all benchmarks is higher than $CITM - VA$. So, $CITM - VA$ performs better than ECM at different values of the MM.

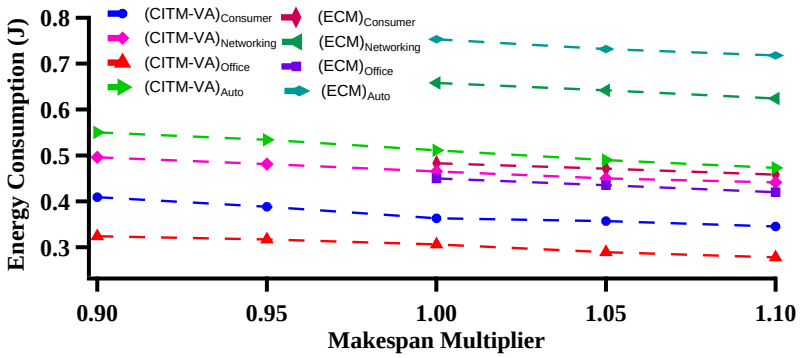


Figure 5.5: Energy and makespan multiplier relation

5.3 Summary

In this chapter, we performed investigation on contention-aware static mapping for real-time task set with precedence constraints and individual deadlines using NoC-MPSoC architecture with DVFS-enabled heterogeneous processors. We proposed CITM-VA meta-heuristic that optimizes the inter-processor communications, NoC links energy consumption, and computational energy. The CITM-VA performs task mapping, scheduling, and voltage scaling in an integrated manner to achieve higher energy-efficiency. It adopts ELFTF strategy and generates a prioritized task schedule to adequately utilize the available slack and links. ReMap algorithm is used to efficiently map the task and communication nodes to the resources and discrete voltage levels such that overall energy consumption is reduced. To further improve the energy savings we deploy DPM when an idle processor is in a high-power consumption state. Contention between the communications traversing the same link is eliminated by dedicating the links to higher priority communications. The extensive evaluation results illustrate that compared to state-of-the-art technique ECM, our proposed approach CITM-VA achieves better energy-efficiency. CITM-VA attains an average energy savings of $\sim 30\%$ for 5 real benchmarks. Moreover, it also maintains high QoS and robustness at strict task deadlines with significant energy-efficiency.

Chapter 6

Energy-aware Static Task Scheduling on Heterogeneous VFI-NoC-MPSoCs

In Chapter 5 we performed task scheduling considering NoC-MPSoC i.e. Non-VFI based MPSoC architecture. We developed a meta-heuristic, CITM-VA for task scheduling that changes only one task at a time within a population member. Moreover, CITM-VA does not switch between an explorative and exploitative search modes at run-time. Now in this chapter, we study energy-efficient and contention-aware static scheduling for tasks with precedence and deadline constraints on heterogeneous VFI based NoC-MPSoCs (VFI-NoC-HMPSoC) with DVFS-enabled processors. Unlike the existing population-based optimization algorithms, we proposed a novel population based algorithm called ARSH-FATI that can dynamically switch between explorative and exploitative search modes at run-time. Our static scheduler ARHS-FATI collectively performs task mapping, task ordering, and voltage scaling. Consequently, its performance is superior to the existing state-of-the-art approach proposed for homogeneous VFI based NoC-MPSoCs. We also developed a communication contention-aware Earliest Edge Consistent Deadline First (EECDF) task ordering algorithm and gradient descent inspired voltage scaling algorithm called Energy Gradient Decent (EGD). We introduced a notion of Energy Gradient (EG) that guides EGD in its search for islands voltage settings and minimize the total energy consumption. Conducted the experiments on 8 real benchmarks adopted from Embedded Sys-

tems Synthesis Benchmarks (E3S). Our static scheduling algorithm, ARSH-FATI outperformed state-of-the-art heuristics and achieved an average energy-efficiency of $\sim 24\%$ and $\sim 30\%$ over CA-TMES-Search and CA-TMES-Quick respectively.

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

1. We performed task mapping, task ordering, and voltage scaling in an integrated way using a novel search based meta-heuristic called ARSH-FATI. Our static scheduler also considers energy performance profiles of the processors, voltage levels within each processor, communication contention at the NoC links, and inter-VFI communications during the task scheduling.

2. Our meta-heuristic ARSH-FATI can dynamically switch between different search modes to achieve a satisfactory trade-off between explorative and exploitative search during runtime. Moreover, we presented a new a contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling algorithm and gradient descent inspired Energy Gradient Decent (EGD) voltage scaling technique.

3. We compared the energy performance of our static scheduler ARSH-FATI with state-of-the-art CA-TMES-Search [95] and CA-TMES-Quick [95] energy management algorithms using 8 real benchmarks adopted from E3S benchmark suit. Our meta-heuristic based static task scheduler achieved an average energy-efficiency of $\sim 24\%$ and $\sim 30\%$ over CA-TMES-Search and CA-TMES-Quick respectively.

We organize the rest of the chapter as follows: Section 6.1 discusses our novel static contention-aware energy optimization algorithm. The simulation results on different benchmarks are discussed in Section 6.2, while in Section 6.3 we conclude this chapter.

6.1 Static Contention-aware Energy-efficient Scheduling

In this section we discuss our proposed contention-aware energy-efficient task scheduling for a set of tasks with precedence and deadline constraints represented by DAG. The heterogeneous MPSoC architecture presented in Section 4.1.2.2 is adopted.

Algorithm 5: ARSH-FATI

input : A DAG G, tasks Deadlines, an MPSoC, total number of iterations Ω and population size μ

output: Task to processor mapping *map* and islands voltage levels *vol*

1 Construct two matrices Π and Ψ of zeros having dimensions $\mu \times |V|$ and a vector f of zeros having dimension $\mu \times 1$;
2 **for** $\eta \leftarrow 1$ *to* μ **do**
3 **for** ***each*** $v_i \in G$ **do**
4 $\Pi[\eta][i] \leftarrow \lceil rand()(|P|-1)+1 \rceil$;
5 **for** ***each*** $c_j \in C$ **do**
6 $\Psi[\eta][j] \leftarrow$ maximum island voltage;
7 **for** $\eta \leftarrow 1$ *to* μ **do**
8 Generate the extended graph G_e;
9 $[m, e] \leftarrow EGD(G, G_e, \Pi, \Psi, \eta)$;
10 $f[\eta] \leftarrow fitness(m, e)$;
11 **while** *stopping criteria is not satisfied* **do**
12 Find the best solution π_b and the worst solution π_w;
13 $f'_b \leftarrow -\infty$;
14 **for** $\eta \leftarrow 1$ *to* μ **do**
15 **for** ***each*** $v_i \in G$ **do**
16 $r \leftarrow rand()$;
17 $\theta \leftarrow \Pi[\eta][i]$;
18 $$\Pi[\eta][i] \leftarrow \begin{cases} \Upsilon(\theta, \pi_b[i], \pi_w[i]) & \text{if } r \leq DR \\ \Pi[\eta][i] & \text{otherwise} \end{cases}$$
19 Construct an extended graph G_e given a mapping ;
20 $[m, e] \leftarrow EGD(G, G_e, \Pi, \Psi, \eta)$;
21 $f[\eta] \leftarrow fitness(m, e)$;
22 **if** $f'_b < f[\eta]$ **then**
23 $f'_b \leftarrow f[\eta]$;
24 $$DR \leftarrow \begin{cases} \frac{DR}{\lambda} & \text{if } f'_b > f_b \\ \lambda DR & \text{otherwise} \end{cases}$$
25 Set *map* and *vol* to mapping and islands voltage settings respectively with the highest fitness in the population;

The details of ARSH-FATI are given in Algorithm 5. ARSH-FATI is a population-based algo-

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

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

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

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

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

We define the following two terms:

1. **Best solution:** It is a member of the population that has the highest fitness value.

2. **Worst solution:** It is a member of the population that has the lowest fitness value.

Step3. Setting parameter DR: We set the value of DR to 0.3 for the initial population.

This value is determined empirically after extensive experiments. The DR value for the other populations generated during the optimization process is determined as follows:

$$DR = \begin{cases} \frac{DR}{\lambda} & \text{if the best solution is improved} \\ \lambda DR & \text{otherwise} \end{cases} \tag{6.1}$$

According to equation (6.1) if the best solution found so far is improved in the previous iteration then the value of DR is increased by dividing it by $0 < \lambda < 1$ otherwise we decrease DR by multiplying it with λ. We refer to λ as the dimensional rate adaption parameter as it determines the new value of DR during the optimization process. The larger the value of DR the more explorative the search is as this enables the moves in the search space by re-mapping many tasks at the same time, thereby leading to large and unconstrained step sizes. Compared to this a small value of DR motivates a more exploitative search by allowing small and conservative steps in the search space. The motivation behind Equation (6.1) is to encourage the re-mapping of more tasks and thus, support more explorative search if the energy consumed by the schedule generated by the mapping in the previous iteration reduces. On the other hand if the energy does not reduce then the explorative search is rather restricted and ARSH-FATI takes small steps near the current mapping.

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

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

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

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

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

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

$$\upsilon(\theta, \pi_b[i], \pi_w[i]) = \lceil \theta + r_1(\pi_b[i] - \theta) - r_2(\pi_w[i] - \theta) \rceil \tag{6.2}$$

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

6.1.1 Earliest Edge Consistent Deadline First ($EECDF$) Algorithm

Before we describe $EECDF$ given in Algorithm 6 we define some notations. The worst case execution time of a task node v_i mapped on processor pe_k operating at frequency f_j is $et(v_i) = \frac{NCC(v_i,k)}{f_j}$, where $NCC(v_i, k)$ is the worst case clock cycles of v_i on processor pe_k. The start and finish times of a task node v_i are respectively denoted by $\rho(v_i)$ and $\zeta(v_i)$. Similarly for a communication node v_j (corresponding to edge (v_a, v_b)) the transmission time on a link L between processors pe_s and pe_d is $et(v_j, L) = \frac{\chi_{a,b}}{b_w \min\{f_s, f_d\}}$, where b_w is the link width, f_s and f_d are the frequencies of pe_s and pe_d respectively. The start and finish time of v_j on link L are

respectively denoted by $\rho(v_j, L)$ and $\zeta(v_j, L)$, where $\zeta(v_j, L) = \rho(v_j, L) + et(v_j, L)$.

Algorithm 6: *EECDF*

input : A DAG G, an extended DAG G_e, matrix Π, matrix Ψ and current chromosome index η
output: Energy e and make-span m of schedule

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

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

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

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

$$d'_i = \min\{d_i, \min\{d'_j - et_j : \forall v_j \in ISucc_i\}\} \tag{6.3}$$

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

The $EECDF$ algorithm is described in Algorithm 6. We performs four major steps.

1. Calculate the ECD of each task $v_i \in G$ (Line 1).

2. Create a ready queue R and insert all the source nodes in G_e to R (Line 2).

3. Find a node v_i that has minimum ECD in R and schedule it. Then delete v_i from R and insert all the ready nodes in G_e to R. Repeat this until R is empty (Line 3-10).

4. Calculate the energy E and make-span m of the schedule.

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

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

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

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

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

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

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

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

$$\rho(v_j, L_1) \leq \rho(v_j, L_k) \tag{6.4}$$

$$\zeta(v_j, L_{k-1}) \leq \zeta(v_j, L_k) \tag{6.5}$$

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

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

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

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

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

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

Algorithm 7: Energy Gradient Descent (EGD)

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

```
1  [e, m] ← EECDF(G, G_e, Π, Ψ, η);
2  if m ≤ D then
3      while there are extensible islands do
4          Γ ← −1;
5          for each extensible island c_j ∈ C do
6              EG^best ← −∞;
7              Find the voltage V_dd^L of island c_j exactly one level lower than current voltage level
                Ψ[η][j];
8              for each V_dd ∈ {V_dd^min, ..., V_dd^L} do
9                  temp ← Ψ[η][j];Ψ[η][j] ← V_dd^L;
10                 [e', m'] ← EECDF(G_e, Π, Ψ, η);
11                 Ψ[η][j] ← temp;
12             if m ≤ D then
13                 Calculate EG(c_j);
14                 if EG^best < EG(c_j) then
15                     EG^best ← EG(c_j);
16                     Γ ← V_dd;τ ← j;e'' ← e';m'' ← m';
17         if Γ ≠ −1 then
18             Ψ[η][τ] ← Γ;e ← e'' ;m ← m'';
```

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, $\zeta(v_j) = \zeta(v_j, L_l)$.

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

6.1.1.1 Energy gradient descent (EGD)

EGD in Algorithm 7 is inspired by gradient descent. Given task mapping and the initial islands operating voltages, EGD explores the solution space to find voltage settings for islands such that total energy consumption is minimized and the resulting schedule under these settings is feasible.

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

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

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

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

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

EGD repeats the following two steps until there are no extensible islands:

Step 1: First find a set of extensible islands. Then for each extensible island c_j do the following:

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

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

EGD may repeat the above mentioned two steps several times before it converges. In each iteration *EGD* can find many extensible islands and can adjust their operating voltages to many different levels. Each of these island voltage pairs may lead to some reduction in energy consumption. *EGD* chooses the pair that maximizes the *EG*. This is because for each island voltage pair the energy consumption of the schedule under the new voltage settings reduces without or with an increase in the make-span of the schedule. Both of these cases are reflected

in the *EG* function. The first case is an ideal one because energy is reduced without any reduction in the available slack. Hence, the *EG* gives more weight to island voltage pairs that lie in case 1 by multiplying the energy difference with a large integer γ. In the second case energy reduces but with an increase in schedule make-span. In this case *EG* is the ratio between energy difference and the make-span difference. Higher the ratio the better the island voltage pair. A large value of this ratio is an indication of a large numerator and a small denominator. A large numerator reflects a big energy difference. This is desirable because it indicates that by changing the voltage level the schedule under new voltage settings reduces energy significantly. A small denominator reflects a small make-span difference. This is also desirable as this indicates more slack will be available for the nodes in the subsequent iterations.

6.2 Experimental Results and Discussions

In this section, we explain the experimental set up used for our simulations. We also generate energy consumption values for different real benchmarks and explicitly discuss the results. The experimental setup is explained in details in Section 4.3.1.1.

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

Figure 6.1 shows the impact of DR on $ARSH - FATI$ performance. We set $DR = 0.3$ initially though it can acquire values $0.1 \leq DR \leq 0.5$ with small impact on the total energy performance for static task scheduling. The results indicate that the energy performance of the $ARSH - FATI$ slightly decreases when $DR = 0.1$ and $DR = 0.5$. However, our algorithm automatically sets the DR value to produce maximum energy-efficiency but initially setting $DR = 0.10$ means $ARSH - FATI$ performs an insufficient exploration while $DR = 0.5$ leads to an excessive exploration. Thus, $DR = 0.3$ is the nominal initial value for our meta-heuristic. $ARSH - FATI$ converges i.e. DR value relatively stabilizes at 200 number of iterations (NI) and a minute variation occurs till 500 while no variations occur when $NI > 500$. Therefore, we consider $NI = 500$, $\mu = 5$, and $\lambda = 0.98$ for our experiments.

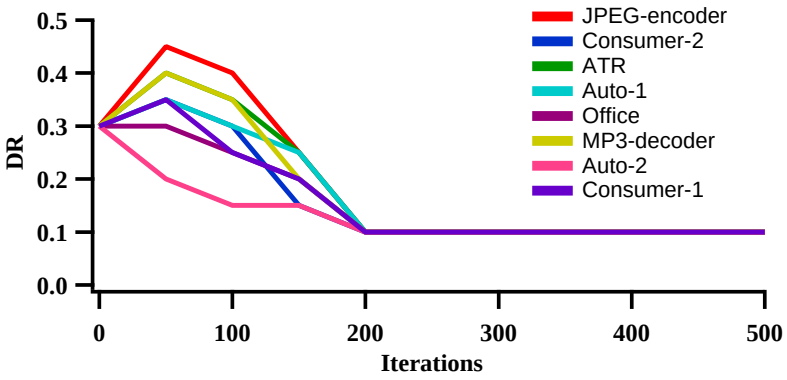


Figure 6.1: Dimensional rate parameter variations

Table 6.1: List of parameters used in results

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

We generate results for four scenarios considering different metrics such as homogeneous MPSoC platform, heterogeneous multiprocessing computing system, PPI, deadline, and CCR. In this section, we refer to different parameters listed in Table 6.1.

Table 6.2: Real benchmarks energy consumption in joule (J) at $NVFI = 4$ and $PPI = 2 \times 2$

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

6.2.1 Scenario 1

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

We use CA-TMES-Quick [95] and CA-TMES-Search [95] energy management schemes as baseline in order to determine the performance of our static task scheduling approach, $ARSH - FATI$. First, we consider a homogeneous VFI-NoC-MPSoC system where all the processors are of type 1. We set the operating frequencies of the processors to their maximum ($f_{max} =$ 2.0 GHz). Second, we use a VFI-NoC-HMPSoC deploying both type 1 and type 2 processors without voltage scaling technique. We randomly select the type of processor for each VFI to generate a heterogeneous computing platform in order to ensure unbiased experimentation. Third, we consider a VFI-NoC-HMPSoC computing architecture and use EGD in order to efficiently avail the slack in the processors. Table 6.2 summarizes the energy consumption values for these three cases on 8 real benchmarks.

Figure 6.2 demonstrates the energy performance of our static task scheduler $ARSH - FATI$ compared to CA-TMES-Search and CA-TMES-Quick. The x-axis denotes real benchmarks while y-axis represents energy consumption in joule (J). Not surprisingly when all the processors are of type 1, $(ARSH - FATI)_{homogeneous}$ consumes lower energy because our population based meta-heuristic performs better solution space exploration during task mapping

and subsequently, reduces communication energy. In other words $(ARSH - FATI)_{homogeneous}$ schedules dependent tasks closer to each other in order to avoid energy dissipation occurring due to the utilization of links, switches, and buffers for communications. Specifically, $(ARSH - FATI)_{homogeneous}$ achieves an average energy-efficiency of $\sim$ 15%, $\sim$ 8% over CA-TMES-Quick and CA-TMES-Search respectively.

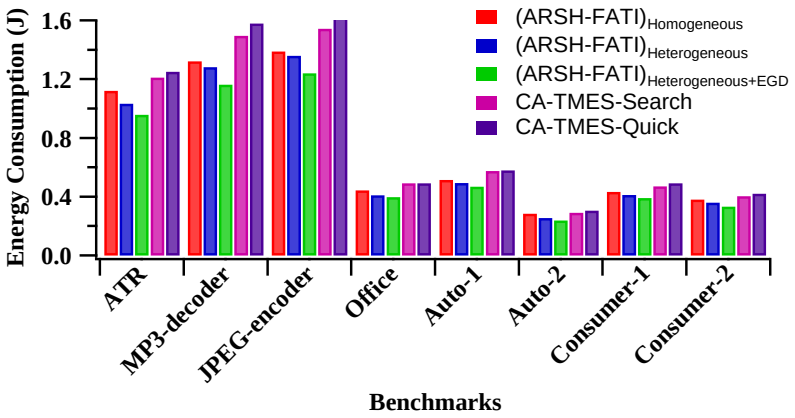


Figure 6.2: Energy consumption at $NVFI = 4$ and $PPI = 2 \times 2$

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

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

Summarizing the observations in these experiments in scenario 1, $ARSH - FATI$ reduces both the communication and computation energy consumptions while does not sacrifice the

constraints. Our novel approach $ARSH-FATI$ for static task scheduling on VFI-NoC-MPSoC architecture outperforms both CA-TMES-Search and CA-TMES-Quick.

6.2.2 Scenario 2

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

Figure 6.3 illustrates that except MP3-decoder, JPEG-encoder, and Robot other benchmarks do not show a significant decrease in the energy consumption when PPI is gradually increased. These benchmarks contain relatively higher number of task nodes and degree of parallelism. MP3-decoder consumes 1.1627 J energy at $PPI = 2 \times 2$ while it decreases to 1.0936 J and 1.0728 J for $PPI = 4 \times 2$ ($\Delta E = 0.0.0691$ J), and $PPI = 4 \times 3$ ($\Delta E = 0.0899$) respectively. Similarly, JPEG-encoder depletes 1.2396 J energy at $PPI = 2 \times 2$ and this energy consumption reduces to 1.1722 J ($\Delta E = 0.0674$ J), 1.1517 J ($\Delta E = 0.0879$ J) at $PPI = 4 \times 2, 4 \times 3$ respectively. We also evaluate the performance of our static scheduler $ARSH-FATI$ on more complex real benchmark, Robot containing 88 tasks. Compared to $PPI = 2 \times 2$ (M = 16), $ARSH-FATI$ achieves energy savings of $\sim$ 13% and $\sim$ 18% for robot when $PPI = 4 \times 2$ (M = 32) and $PPI = 4 \times 3$ (M = 64) are used respectively.

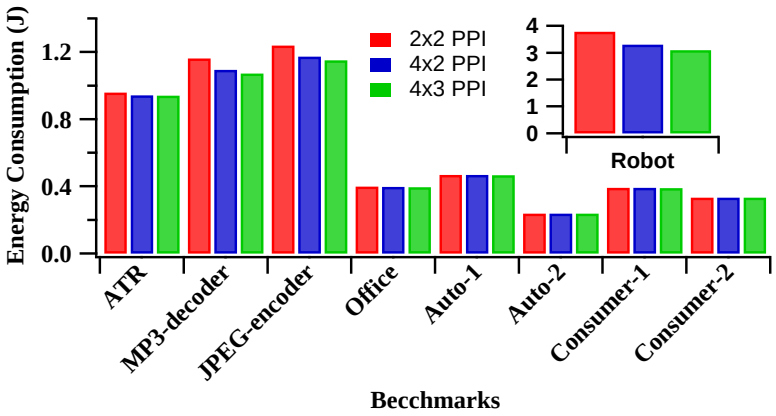


Figure 6.3: Energy consumption using $NVFI = 4$ at different PPI

These results demonstrate that our meta-heuristic $ARSH - FATI$ is scalable and it can efficiently utilize the resources and degree of parallelism in the benchmarks to reduce the total energy consumption.

6.2.3 Scenario 3

We now conduct experiments to analyze the robustness of $ARSH - FATI$ under deadline variations and compare its performance with CA-TMES-Search. We consider voltage scalable heterogeneous computing architecture with $NVFI = 4$, $PPI = 1 \times 2$ and $M = 8$. We set the baseline deadline for each benchmark in set-1 and set-2 (described in Table 6.1) to the makespan of the schedule generated by CA-TMES-Search under the condition of all VFIs operating at maximum frequencies.

Figure 6.4 and Figure 6.5 show the energy consumption of $ARSH - FATI$ and CA-TMES-Search for set-1 and set-2 respectively. In Figure 6.4 blue, green, and red colors represent ATR, MP3-decoder, JPEG-encoder respectively. Similarly, blue, green, and red colors denote Consumer, Office, Auto benchmarks respectively in Figure 6.5. The MULT represents the factor multiplied to the baseline deadline. For example, $MULT = 1.00$ at the horizontal axis in Figure 6.4 and Figure 6.5 indicates the deadline of each benchmark is set to $1.00 \times$ baseline deadline. The dotted lines represent our $ARSH - FATI$ while the straight lines show CA-TMES-Search. The condition $MULT < 1$ indicates a strict deadline while $MULT > 1$ shows a relaxed deadline.

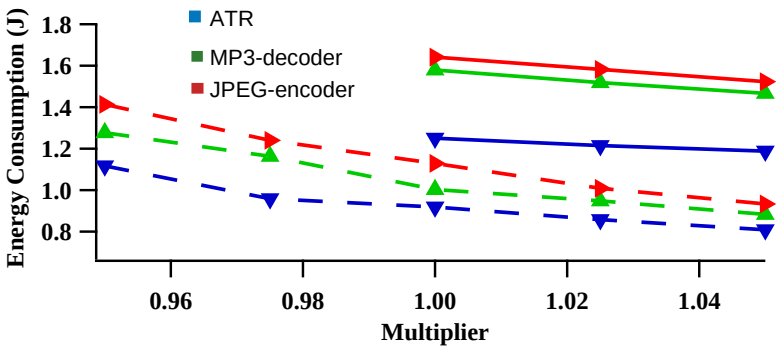


Figure 6.4: Set-1 energy consumption using $NVFI = 4$ and $PPI = 2 \times 2$

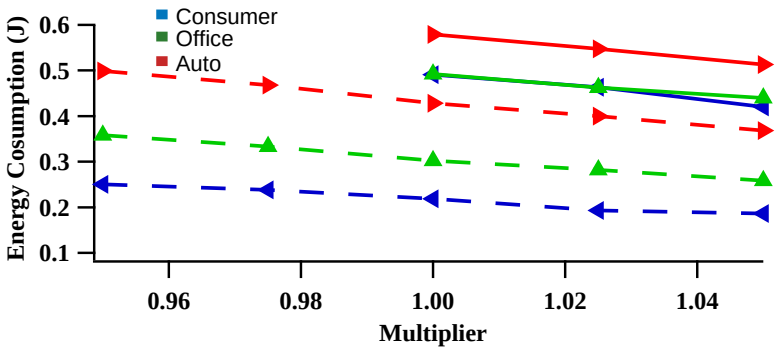


Figure 6.5: Set-2 energy consumption using $NVFI = 4$ and $PPI = 2 \times 2$

The energy-efficiency of $ARSH - FATI$ gradually reduces starting from $MULT < 1$ to $MULT = 0.95$. This increase in energy consumption occurs due to the reduction in slack. Though, energy consumption slightly increases under the strict deadline conditions (of $MUL < 1$), $ARSH - FATI$ can still successfully generate a feasible schedule. Moreover, as deadline decreases ARSH-FATI tends to schedule more tasks on high-performance processors. These processors reduce task execution time at a cost of higher energy consumption. This is another reason in the increase of energy consumption along with the reduction in slack. The same is not true for CA-TMES-Search because it neglects to consider the energy performance profiles of the processors during the task mapping phase. The $EECDF$ prioritizes nodes with shorter ECD thereby, increasing the chance of generating a feasible schedule. This is because ECD of a node depends on the pre-assigned deadline. As the deadline varies so does ECD consequently, the relative urgency of nodes may change. This additional information reflected by ECD can be exploited by $EECDF$. On the contrary, CA-TMES-Search uses b-level (different basic scheduling attributes of list scheduling are discussed with details in [212]) to reflect the relative urgency of tasks. The metric b-level is independent of the application deadlines hence, the CA-TMES-Search is unaware of the deadline variations and is unable to respond accordingly.

Under the condition $MULT > 1$ the energy-efficiency of $ARSH - FATI$ rapidly increases. $ARSH - FATI$ being aware of processor energy performance profiles tends to map more tasks on lower performance but energy-efficient processors. Contrarily CA-TMES-Search is inadequate to avail energy performance profiles consequently, it maps more tasks on high performance, lower energy-efficient processors. The $ARSH - FATI$ schedules nodes in $EECDF$

manner hence EGD can efficiently utilize the slack because nodes with longer ECD are not blocked by nodes with shorter ECD. The same is not true for CA-TMES-Search. Furthermore, uniform voltage scaling used by CA-TMES-Search is inefficient technique for a heterogeneous system.

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

6.2.4 Scenario 4

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

Figure 6.6 illustrates the impact of CCR on $ARSH - FATI$ energy performance while CA-TMES-Quick (represented by blue line) being used as a baseline. Evidently, $ARSH - FATI$ static scheduler consumes lesser energy compared to CA-TMES-Search due to performing task mapping, scheduling, and voltage scaling in an integrated manner. With the increase in communication volume, the energy consumption of $ARSH - FATI$ reduces and reaches to its minimum value at $CCR = 1.0$. $ARSH - FATI$ maps the dependent tasks (parent and child nodes) on the same processor when $0.2 \leq CCR \leq 1.0$ in order to decrease the communication energy. $ARSH - FATI$ at $CCR > 1$ tends to map all the dependent tasks on the closest possible processors which leads to a slight increase in energy consumption. Our static scheduler $ARSH - FATI$ performs relatively better when $0.5 \leq CCR \leq 2.0$ i.e. when network contention is medium. Our static scheduler $ARSH - FATI$ outperforms CA-TMES-Search in terms of energy-efficiency for $0.2 \leq CCR \leq 3.0$.

Table 6.3 summarizes the energy performance of $ARSH - FATI$ compared to the base-line state-of-the-art $CA - TMES - Search$ and $CA - TMES - Quick$ energy management approaches when $NVFI = 4$ and $PPI = 2 \times 2$ are deployed in the multiprocessor computing system. Energy consumption of the dependent DAG tasks decreases when computing platform is changed from homogeneous to heterogeneous. The energy-efficiency further improves

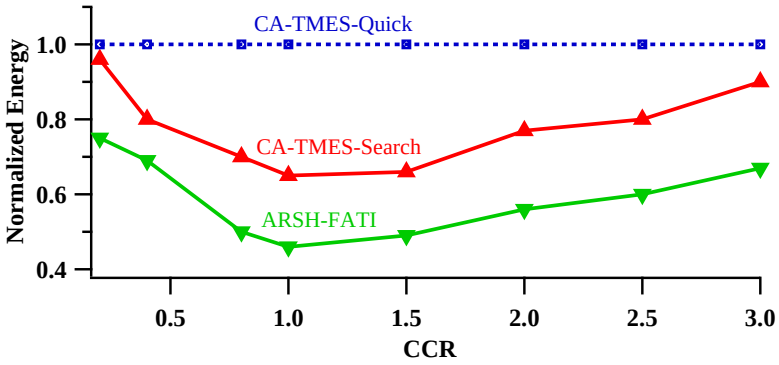


Figure 6.6: CCR impact on ARSH-FATI at $VFI = 4$ and $PPI = 2 \times 2$

Table 6.3: ARSH-FATI energy performance summary

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

when voltage scaling technique EGD is deployed. Concisely, $ARSH - FATI$ outperforms CA-TMES-Search and CA-TMES-Quick in terms of energy savings while maintains higher robustness.

6.3 Summary

In this chapter unlike other scheduling techniques presented in [95,173,175,213], we investigated a harder scheduling problem i.e. contention-aware and energy-efficient DAG tasks scheduling on heterogeneous VFI based NoC-MPSoC (VFI-NoC-HMPSoC) computing architecture with DVFS-enabled processors. We proposed a novel static task scheduler ARSH-FATI that performs task mapping, scheduling, and voltage scaling in an integrated manner while considering the energy performance profiles of the processors and contention at the NoC links. Our meta-heuristic ARSH-FATI can intelligently switch at run-time between explorative and exploitative search modes for performance trade-off. We also integrated communication contention-aware Earliest Edge Consistent Deadline First (EECDF) scheduling approach and Energy Gradient Decent (EGD) algorithm for voltage scaling in ARSH-FATI to reduce the computation en-

ergy consumption. We performed experiments on eight real benchmarks considering different scenarios. Our static scheduler outperformed state-of-the-art CA-TMES-Search [95] and CA-TMES-Quick [95] energy management approaches. Our task scheduling approach ARSH-FATI achieved an average energy-efficiency of $\sim$ 24% and $\sim$ 30% over CA-TMES-Search and CA-TMES-Quick respectively. Moreover, our static scheduling approach showed robustness while maintained higher QoS and energy-efficiency at restricted deadlines.

Chapter 7

Energy-aware Clustering for Enhancing Wireless Sensor Network Lifetime

In this chapter we develop a novel ARSH-FATI based Cluster Head Selection (ARSH-FATI-CHS) algorithm integrated with a heuristic called Novel Ranked based Clustering (NRC) in order to reduce the transmission energy consumption of the sensor nodes while efficiently enhancing Lifetime (LT) of the network. Unlike other population based algorithms ARSH-FATI-CHS dynamically switches between exploration and exploitation of the search process during run-time to achieve higher performance trade-off and maximally increase network LT. ARSH-FATI-CHS considers the residual energy, communication distance parameters, and workload during Cluster Heads (CHs) selection. We simulate our proposed ARSH-FATI-CHS and generate various results to determine the performance of the WSN in terms of network LT. We compare our results with state-of-the-art PSO based clustering and we prove that our developed ARSH-FATI-CHS energy-efficient sensor nodes clustering approach improves the network LT by $\sim 25\%$. ARSH-FATI-CHS also outperforms LEACH and PSO-C while achieving an average LT improvements of $\sim 60\%$ and $\sim 40\%$ respectively. Concisely in this chapter we reduce the transmission energy consumption of the sensor nodes in WSN while in chapter 6 and chapter 5 we decreased their processing energy by proper tasks scheduling and mapping using DVFS and DPM.

In this chapter we investigate to reduce the transmission energy consumption of the sensor nodes in WSN and present ARSH-FATI [96] based Cluster Head Selection (ARSH-FATI-CHS) meta-heuristic integrated with Novel Ranked based Clustering (NRC) heuristic. The major innovations and contributions of this chapter are given as follows:

1. We develop ARSH-FATI-CHS algorithm that can dynamically switch between exploration and exploitation search modes during run-time of the CHs selection process for better performance trade-off.

2. We present an algorithm NRC that estimate the LT of the sensor nodes and guides the ARSH-FATI-CHS to a better and energy-efficient clustering to enhance the LT of the network.

3. We propose a fitness function that considers various parameters such as residual energy, the communications distance of the sensor nodes, and workload on the selected CHs in the network.

4. We compare our clustering scheme with state-of-the-art PSO based approach presented by Rao et al. [69]. We generate different results and achieve an overall LT improvement of $\sim 25\%$ when simulated for various number of sensor nodes. Similarly, ARSH-FATI-CHS outperforms LEACH [156] and PSO-C [151] while attains an average LT improvement of $\sim 40\%$ and $\sim 60\%$ respectively.

We organize the rest of the chapter as follows; Section 5.1 discusses the related work performed so far. Section 5.2 presents the preliminaries. Section 5.3 explains our novel CHs selection and clustering algorithms. Experimental results are presented in Section 5.4 followed by conclusion of this chapter in Section 5.5.

7.1 ARSH-FATI based Cluster Head Selection

In this section we explain our CHs selection approach, ARSH-FATI-CHS for improving LT of the network and reducing sensor nodes transmission energy consumption.

The ARSH-FATI is a population based algorithm, hence first a fitness function is required to determine the quality of each member of the population. In this chapter our objective is to maximize the minimum LT or in simpler words maximize the LT of the network. Therefore, the fitness function is the LT of the network:

$$f = \min\{LT(sn_i, ch_j) : \forall sn_i \in SNs\} \tag{7.1}$$

where the ch_j is the CH of sensor node sn_i.

Before we proceed with our discussion we define the following two terms and then explain briefly explain the an important term LT:

1. **Global Best**, $gBest$, is a member of the population that has the highest fitness value $gBestFit$ in all generations.
2. **Generation worst,** π_w is a member of the population that has the lowest fitness value in a generation.

Before we describe the LT of the sensor network first we define the LT of a sensor node and a CH. The LT of a sensor node $sn_i \in SN$ when it chooses ch_j as its CH is defined as follows:

$$LT(sn_i, ch_j) = \left\lfloor \frac{e^i_{residual}}{e_{total}(l, d_i)} \right\rfloor \tag{7.2}$$

where d_i is the euclidean distance between the sn_i and ch_j and $e^i_{residual}$ is the residual energy of sn_i.

The LT of a CH, ch_j is defined as follows:

$$LT(ch_j) = \left\lfloor \frac{e^j_{residual}}{E_{TX}(l, d_j) + \sum_{sn_i \in \psi} E_{RX}(l)} \right\rfloor \tag{7.3}$$

where d_j is the euclidean distance between the ch_j and Base Station (BS) and ψ is the set of sensor nodes whose CH is ch_j.

Though there are are different definitions to describe the LT of the wireless network but we use the definition of LT given as follows:

LT Definition: According to this definition the LT of the network is the number of rounds until the First Node Dies (FND).

We use FND as a metric to describe the LT of a network and produce various simulations results based on this definition.

7.1.1 ARSH-FATI-CHS

The objective of ARSH-FATI-CHS described in Algorithm 8 is to find a set of CHs among the sensor nodes such that energy consumption reduces and LT of the network maximize. ARSH-FATI-CHS improves the network LT by maximizing the minimum LT in the network. The steps followed by ARSH-FATI-CHS are explained as follows:

1. **Setting the initial value of DR:** We set DR to and 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 $\mu \times m$ of zeros (Line 1), where μ is the size of the population and m is the total number of CHs. We use the notation $\Pi[i,:]$ to access the i^{th} member of the population. Each member of the population has m elements and the notation $\Pi[i,j]$ is used to access the j^{th} element of i^{th} member. The value of an element is a positive integer that indicates a sensor node chosen as a CH. For example if the value of an element is 2 this reflects that sensor node sn_2 is selected as a CH. Each member of the population reflects one possible CH selection and the entire population reflects μ different CH selections.

Algorithm 8: ARSH-FATI-CHS

input : Set $SNs = \{sn_1, sn_2, \ldots, sn_n\}$ of n sensor nodes, total number of CHs m, maximum generations $maxGens$, β, T_1, T_2 and DR_0 the initial value of DR

output: a set $CHs = \{ch_1, ch_2 \ldots ch_m\}$ of m CHs

1 Generate a matrices Π of zeros having dimensions $\mu \times m$;
2 Generate a vector f of zeros having dimension $\mu \times 1$;
3 Set DR to DR_0;
4 **for** $\eta \leftarrow 1$ *to* μ **do**
5 **for** $i \leftarrow 1$ *to* m **do**
6 $\Pi[\eta][i] \leftarrow \lceil rand(0,1)(n-1)+1 \rceil$;
7 $f[\eta] \leftarrow Clustering(SNs, \Pi[\eta, :])$;
8 Find the generation best π_b and worst π_w members;
9 $gBest' \leftarrow \pi_b$; $T \leftarrow T_2$;
10 $gBestFit' \leftarrow f_b$; ; `// where` f_b `is the fitness value of` π_b
11 **repeat**
12 **for** $\eta \leftarrow 1$ *to* μ **do**
13 $\pi \leftarrow \Pi[\eta, :]$;
14 **for** $i \leftarrow 1$ *to* m **do**
15 $r \leftarrow rand(0,1)$; $r_1 \leftarrow rand(0,1)$; $r_2 \leftarrow rand(0,1)$;
16 $\pi[i] \leftarrow \begin{cases} \min(abs(\lceil \pi[i] + r_1(gBest[i] - \pi[i]) - \\ r_2(\pi_w[i] - \pi[i]) \rceil), m) & \text{if } r \leq DR \\ \pi[i] & \text{otherwise} \end{cases}$
17 $f \leftarrow Clustering(SNs, \Pi[\eta, :])$;
18 **if** $f > f[\eta]$ **then**
19 $f[\eta] \leftarrow f$; $\Pi[\eta, :] \leftarrow \pi$;
20 **if** $f > gBestFit'$ **then**
21 $gBestFit' \leftarrow f$; $gBest' \leftarrow \pi$;
22 **else**
23 **if** $e^{\frac{(f - f[\eta])}{T}} > rand(0,1)$ **then**
24 $f[\eta] \leftarrow f$; $\Pi[\eta, :] \leftarrow \pi$;
25 $DR \leftarrow \begin{cases} \min(\frac{DR}{\lambda}, DR_{max}) & \text{if } gBestFit > gBestFit' \\ \max(\lambda DR, DR_{min}) & \text{otherwise} \end{cases}$;
26 $gBestFit \leftarrow gBestFit'$; $gBest \leftarrow gBest'$;
27 Update the generation worst π_w; $T \leftarrow \lambda_1 T$;
28 **until** *termination*;
29 $CHs \leftarrow gBest$;

We generate and initial value of the member of the population by randomly selecting m CHs among the sensor nodes and its fitness value is calculated by executing the clustering algorithm 9. We repeat this for the other members of the population to generate an initial population (Lines 4-7).

3. **Population Refinement:** We refine the initial population until the termination criteria

satisfy (Lines 10-28). In each generation we update the members of the population. The j^{th} element of the i^{th} member is updated as follows:

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

where r_1 and r_2 are the random numbers. The term $r_1(\ gBest[i] - \Pi[\eta, i])$ reflects the likelihood that the member moves closer to global best and the term $r_2(\pi_w[i] - \Pi[\eta, i])$ 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 $\Pi[i, :]$, $\Delta f = f_{new} - f_{old}$. parameter T is referred to as temperature. We define function $P(\Delta f, T)$ as follows:

$$P(\Delta f, T) = \begin{cases} 1 & \text{if } f_{new} > f_{old} \\ e^{\frac{\Delta f}{T}} & \text{otherwise} \end{cases} \tag{7.4}$$

When the new CH selection increase the network LT then it is always accepted. If the new CH selection is worse than the current selection, probability still exists that may be accepted. We have included this feature in ARSH-FATI-CHS 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 $maxGens$:

$$\lambda_1 = \left(\frac{T_1}{T_2}\right)^{\frac{1}{maxGens}} \tag{7.5}$$

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):

$$DR = \begin{cases} \min\left\{\frac{DR}{\lambda}, DR_{max}\right\} & \text{if } gBest \text{ improves} \\ \max\{\lambda DR, DR_{min}\} & \text{otherwise} \end{cases} \tag{7.6}$$

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 bound 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-CHS terminates if either the generation count reaches maximum generation $maxGens$ or no improvement is observed in $gBest$ for consecutive β generations.

7.1.2 Cluster Formation

In this section we describe our cluster formation technique. Given a set of CHs the objective of our cluster formation approach is to maximize the network LT. Algorithm 9 describes our cluster formation approach. It is based on the ranking function $Rank(sn_i, ch_j)$ that sensor nodes use to choose a CH. The ranking function is based on the following parameters:

1. sn_i **residual energy:** Larger the residual energy of the sensor node larger its LT. Therefore,

$$Rank(sn_i, ch_j) \propto e^i_{residual} \tag{7.7}$$

2. sn_i **total energy:** The sensor node energy consumption negatively impacts its LT. The sensor node should join the CH such that its transmission energy minimizes. Therefore,

$$Rank(sn_i, ch_j) \propto \frac{1}{e_{total}(l, d_i)} \tag{7.8}$$

3. ch_j **residual energy:** The sensor node sn_i should join the CH that has large residual energy. Therefore,

$$Rank(sn_i, ch_j) \propto e^j_{residual} \tag{7.9}$$

4. ch_j **total energy:** The CH total energy consumption negatively impacts its LT. Therefore,

$$Rank(sn_i, ch_j) \propto \frac{1}{e_{total}(l, d_j)} \tag{7.10}$$

Combining equations (7.7), (7.8), (7.9) and (7.10) we get the following function:

$$Rank(sn_i, ch_j) \propto \frac{e^i_{residual} e^j_{residual}}{e_{total}(l, d_i) e_{total}(l, d_j)} \tag{7.11}$$

$$Rank(sn_i, ch_j) = \kappa \left\lfloor \frac{e^i_{residual}}{e_{total}(l, d_i)} \right\rfloor \left\lfloor \frac{e^j_{residual}}{e_{total}(l, d_j)} \right\rfloor \tag{7.12}$$

$$Rank(sn_i, ch_j) = \kappa LT(sn_i, ch_j) LT(ch_j) \tag{7.13}$$

where κ is the constant of proportionality. We assume $\kappa = 1$ here without any loss of generality and performance.

Algorithm 9 describes our cluster formation approach.It is a greedy heuristic algorithm that at each stage finds a set of valid choices and makes a locally optimal choice.

The following two steps repeat until each sensor node has chosen a CH:

1. **Step 1 (Search):** In the this step each sensor node that has not chosen its CH tentatively selects one by one each CH and calculates the rank using equation (7.13) (Lines 3-6). Then find the sensor node sn_i and CH ch_j pair that has the highest value of the rank.

2. **Step 2 (Choose):** In this step sensor node sn_i joins ch_j

Algorithm 9: Clustering

input : A set $SNs = \{sn_1, sn_2, \ldots, sn_n\}$ of n sensor nodes and a set $CHs = \{ch_1, ch_2 \ldots ch_m\}$ of m CHs

```
repeat
    Step 1: Search
    for each sn_i ∈ SNs whose CH has not been selected do
        for each ch_j ∈ CHs do
            Tentatively select ch_j as CH of sn_i;
            Calculate the rank, Rank(sn_i, ch_j) using equation (7.13);
    Find the sensor node sn_i and the CH, ch_j pair with the highest value of the rank;
    Step 2: Choose
    Choose ch_j as the CH of sn_i;
until each sensor node has selected a CH;
return min{LT(sn_i, ch_j) : ∀sn_i ∈ SNs};
```

7.2 Experimental Results and Discussions

The experimental setup is explained in details in Section 4.3.1.2. In this section we generate different results considering various scenarios. The results are compared with state-of-the-art clustering approaches to prove the superiority of ARSH-FATI-CHS over other existing technique.

We consider different scenarios for our simulations to generate various results. The list of abbreviations used in results are listed in Table 7.1. The parameter such as NSNs is changed in the sensing field/target area and results are produced. We also empirically observe IoBSL and IoNCHs in the results.

Table 7.1: List of abbreviations used in results

Parameter	Description
TA	Target Area
BS	Base Station
NSNs	Number of Sensor Nodes
NCHs	Number of Cluster Heads
FND	First Node Death
ISNE	Initial Sensor Node Energy
NoR	Number of Rounds
NoANs	Number of Alive Nodes
IoBSL	Impact of Base Station Location
IoNCHs	Impact of Number of Cluster Heads

7.2.1 Scenario 1(NoR)

In this scenario we set $TA = 200 \times 200\ m^2$, $NSNs = (100, 200, 300, 400)$, $ISNE = 2.0$ J, $BS = (50, 50)$, and $NCHs = (10\%, 15\%, 20\%, 25\%)$ while we compare the performance of ARSH-FATI-CHS combined with NRC in terms of NoR after FND in the network. This scenario basically describes the LT of the network. LT is an important metric to evaluate the performance and efficiency of an algorithm as discussed in detail by Wang et al. [214].

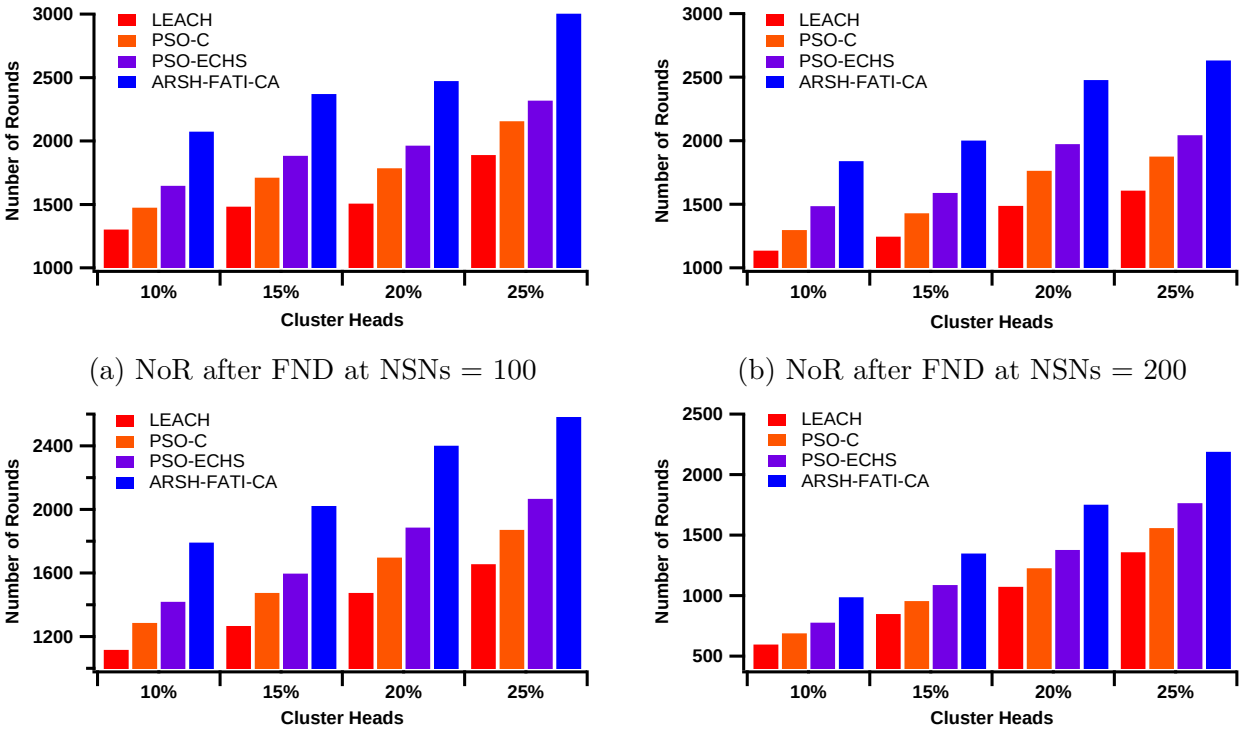
Figure 7.1 demonstrates the performance improvement of ARSH-FATI-CHS integrated with NRC over the existing approaches. A set of different sensor nodes are used for simulations such as, Figure 7.1(a) deploys $NSNs = 100$, similarly, Figure 7.1(b) uses $NSNs = 200$ while Figure 7.1(c) and Figure 7.1(d) deploy $NSNs = 300$ and $NSNs = 400$ respectively. ARSH-FATI-CHS outperforms clustering approaches such as LEACH, PSO-C, and PSO-ECHS. ARSH-FATI-CHS achieves an average improvement of $\sim$ 60%, $\sim$ 40%, and $\sim$ 25% over LEACH, PSO-C, and PSO-ECHS respectively in terms of NoR after FND. This improvement of ARSH-FATI-CHS over LEACH occurs because LEACH is a probabilistic approach and randomly selects CHs

which results into uneven distribution of clusters so, increasing the energy consumption of the network while reducing the overall performance i.e. LT of the network. Moreover, LEACH can select a sensor node with least residual energy as a CH thus, it adversely impacts the NoR after FND. Contrarily our fitness function in ARSH-FATI-CHS considers the residual energy, different distance parameters, and work load on the CHs during the cluster formation. Similarly, ARSH-FATI-CHS outperforms population based clustering approach, PSO-C. Though, PSO-C improves the intra-cluster distance i.e. the Euclidian distance of the sensor nodes to their selected CHs during cluster formation but neglects the workload on them. Compared to PSO-ECHS our proposed clustering approach ARSH-FATI-CHS achieves higher NoR after FND. ARSH-FATI-CHS also performs better solution space exploration during the cluster formation. Moreover, it is stagnation controlled meta-heuristic and determines the global optimal solution. In other words ARSH-FATI-CHS efficiently maximizes the minimum LT of a sensor node in the network. ARSH-FATI-CHS associates a sensor node having the least residual energy to a CH such that it consumes minimal possible transmission energy for data transmission. Concisely, ARSH-FATI-CHS performs better than LEACH, PSO-C, and PSO-ECHS in terms of NoR after FND when different sensor nodes and CHs are deployed in the network.

7.2.2 Scenario 2(NoANs)

In this scenario 2 we set the parameters as $TA = 200 \times 200\ m^2$, $NSNs = (100, 200, 300, 400)$, $ISNE = 2.0$ J, $BS = (50, 50)$, and $NCHs = 25\%$. We observe the performance of the ARSH-FATI-CHS in terms of NoANs. Specifically this scenario describes the amount of alive nodes after certain number of rounds.

Figure 7.2 shows the performance analysis of ARSH-FATI-CHS integrated with NRC over LEACH, PSO-C, and PSO-ECHS in terms of NoANs. The x-axis represents NoR and y-axis denotes the NoANs. Figure 7.2 (a) and Figure 7.2 (b) show the NoANs comparison for $NSNs = 100$ and $NSNs = 200$. Similarly Figure 7.2 (c) and Figure 7.2 (d) represent the simulation results for $NSNs = 300$ and $NSNs = 400$ respectively. ARSH-FATI-CHS significantly improves the NoANs when compared to LEACH, PSO-C, and PSO-ECHS. ARSH-



(a) NoR after FND at NSNs = 100

(b) NoR after FND at NSNs = 200

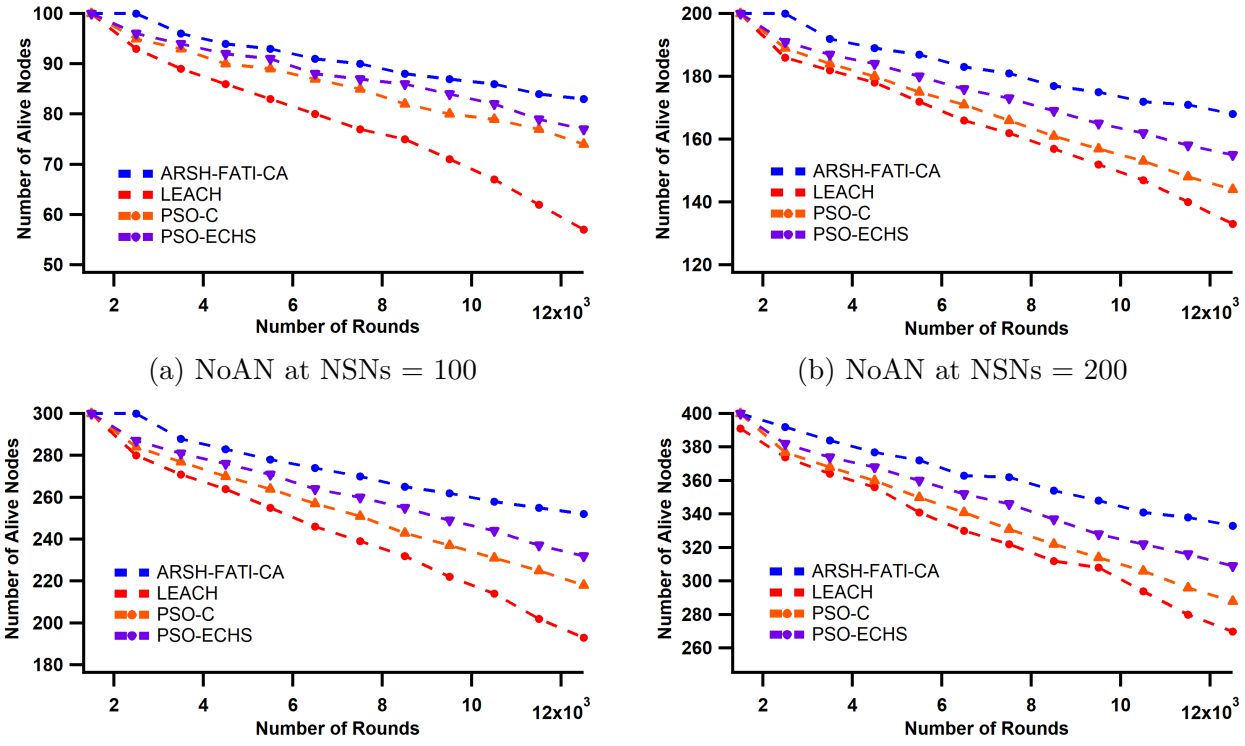
(c) NoR after FND at NSNs = 300

(d) NoR after FND at NSNs = 400

Figure 7.1: Performance analysis in terms of NoR after FND using different NSNs

FATI-CHS attains an average efficiency of $\sim 60\%$, $\sim 40\%$, and $\sim 25\%$ over LEACH, PSO-C, and PSO-ECHS respectively. Unlike LEACH, ARSH-FATI-CHS performs re-clustering when death of a CH occurs and associates the sensor nodes to other CHs in the network. On the contrary the biggest drawback of LEACH is that when a CH dies then that particular cluster becomes useless and the gathered data would never reach to the destination i.e. BS. Furthermore, LEACH may select a CH at the boundary of the network which potentially can lead to the improper clustering resulting performance degradation. Moreover, ARSH-FATI-CHS also performs uniform distribution of CHs. Furthermore, as the NoR increases the residual energy of the sensor decreases. Subsequently, effective CHs selection plays an important role to increase the LT of the network. Our meta-heuristic performs efficient CHs selection using a novel fitness function. Therefore, our algorithm ARSH-FATI-CHS significantly improves the $NoANs$ compared to LEACH. Now, compared to PSO-C our meta-heuristic produces better results because of our fitness function not only considers the intra-cluster distance but also the workload on the CHs. Similarly, it outperforms PSO-ECHS by dynamically switching between an exploitative and exploitative search modes to achieve better solution space search. Concisely,

though ARSH-FATI-CHS is a simple but yet effective approach for CHs selection and enhancing the overall LT of the network.



(a) NoAN at NSNs = 100

(b) NoAN at NSNs = 200

(c) NoAN at NSNs = 300

(d) NoAN at NSNs = 400

Figure 7.2: NoAN using different NSNs and NCHs = 25% at BS = (50, 50)

7.2.3 Scenario 3(IoBSL and IoNCHs)

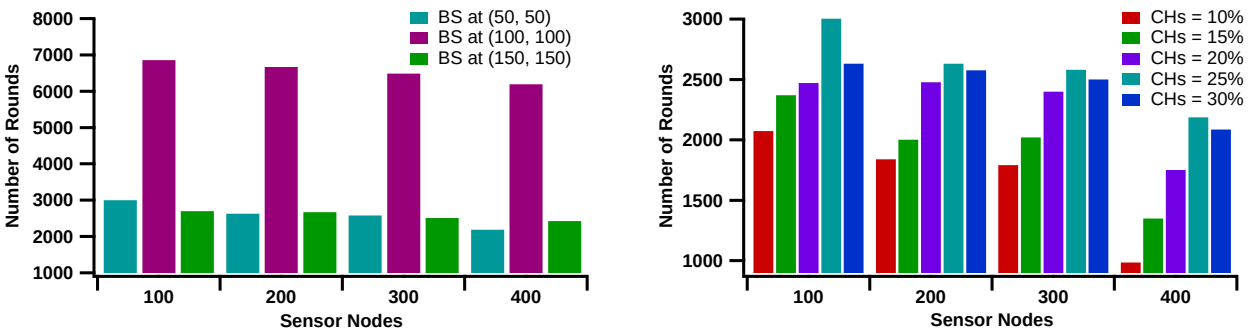
We set the parameters as $TA = 200 \times 200\ m^2$, $NSNs = (100, 200, 300, 400)$, and $ISNE = 2.0$ J. In this scenario 3 we observe the impact of the BS location and variations in CHs percentage on NoR after FND. These two specific cases are discussed as follows:

1. First we consider $NCHs = 25\%$ and $BS = (50, 50), (100, 100), (150, 150)$ for observing IoBSL on NoR after FND. Figure 7.3(a) demonstrates IoBSL on NoR after FND. The horizontal axis shows NoSNs and the vertical axis represents NoR. The NoR depends on the position of the BS. The NoR significantly increases when BS is positioned at the center of the network i.e. (100, 100). The NoR decreases when BS is moved towards positions (50, 50) and (150, 150). This performance degradation occurs because data

packets from the selected CHs travel longer distance when BS is positioned at the coroner of the network. Thus, transmitting the gathered data for a longer distance adversely affects the LT and therefore, FND occurs at lower NoR. An average $\sim 70\%$ improvement in terms of NoR after FND occurs when BS is placed at the centre compared to when BS is positioned at the corners of the network. The bottom line is that the ideal place of BS is the centre of the network.

2. Second to analyze the IoNCHs on network performance we consider $BS = (50, 50)$ and $NCHs = 10\%$, 15%, 20%, 25%, 30% as shown in Figure 7.3(b). The x-axis denotes NSNs while y-axis represents NoR after FND. A significant improvement in NoR occurs when percentage of CHs is increased. The NoR increases until reaching $CHs = 25\%$ while starts decreasing at $CHs = 30\%$. It is a proven fact that CHs usually consume more energy as compared to non-CHs because they collect data from the sensor nodes within the cluster and transmit it to the BS [69]. If lower percentage of CHs are used then clusters with higher NSNs are formed. This increases the receiving and transmission energy of the CH and there is higher chance of dying quickly. Therefore, NoR increases when higher number of CHs are used in the network. In result it reduces the workload on the CHs and prolongs the overall LT of the network. An average $\sim 25\%$ improvement in NoR is observed when $CHs = 15$ is changed to $CHs = 25$ in the network. It is also worth noticing that after certain percentage of CHs despite of improvement in NoR, it starts decreasing. This reduction in NoR is due to the fact that the number of transmissions to the BS increases which can adversely affect the NoR. Subsequently, an optimal number of CHs are required to maximize the overall NoR and LT of the network.

Concisely we achieve an average LT improvement of $\sim 25\%$, $\sim 40\%$, $\sim 60\%$ over PSO-ECHS PSO-C, and LEACH when ARSH-FATI-CHS integrated with NRC heuristic is used for cluster formation. Moreover, we also observed that LT of the network also depends on the position of the BS and the percentage of CHs deployed in the network. The LT performance of ARSH-FATI-CHS over other existing techniques is summarized in Table 7.2.



(a) Impact of base station

(b) Impact of cluster heads

Figure 7.3: Impact of base station and cluster heads on LT of the network

Table 7.2: ARSH-FATI-ECHS performance improvement comparison

Clustering Technique	ARSH-FATI-CHS
LEACH	60%
PSO-C	40%
PSO-ECHS	25%

7.3 Summary

Transmission energy consumption reduction is one of the major concerns in designing clustering algorithms for large-sized Wireless Sensor Networks (WSNs). The existing population based meta-heuristics are complex and need different parameters tuning to achieve higher energy-efficiency. Furthermore, state-of-the-art clustering approaches neglect to consider residual energy of the nodes, different distance parameters, and workload on the Cluster Heads (CHs) to enhance LT of the network. In this chapter we developed ARSH-FATI based Cluster Head Selection (ARSH-FATI-CHS) integrated with a heuristic called Novel Ranked based Clustering (NRC) for efficient cluster formation to enhance the overall LT of the network. ARSH-FATI-CHS is a simple yet effective clustering algorithm that dynamically switches between the exploitative and explorative search modes for achieving higher performance trade-off. Our fitness function considers various parameters such as residual energy, node distance, base station distance location, and work load on the CHs. The experimental results show that ARSH-FATI-CHS outperforms the existing population based clustering algorithms in terms of network LT enhancement. ARSH-FATI-CHS achieves an average LT improvement of 60%, 40%, and 25% over LEACH, PSO-C, and PSO-ECHS. Furthermore, we also investigated that the position of

Base Station (BS) and percentage of CHs in the network also play an important role in the improvement of LT.

Chapter 8

Conclusion and Future Work

Precisely, we successfully achieved our objectives to reduce the energy consumption both at the node and network level. We summarize our contributions and then discuss several future research problems related to energy-aware scheduling and nodes clustering for enhancing the overall Lifetime (LT) and energy savings of the network.

8.1 Conclusion

Wireless Sensor Network (WSN) is one of the main constituents of Internet-of-Things (IoT) and provides assistance to humans and machines for interacting with the environment and real world events. WSN is composed of a number of SNs that are the integration of digital electronics, wireless communication and micro-electro-mechanical systems. The SNs have the capability to sense the physical or environmental conditions, process the information and communicate with each other. They are assimilated through a wireless network and data synthesis is performed to acquire accurate and meaningful information. The SNs are energy constrained and normally powered by a battery source with limited residual energy.

System-on-Chip (SoC) is gaining popularity because it integrates power management circuits, multiprocessor functions, input and output peripherals on a small sized single chip. Moreover,

SoC consumes low power and exhibits high computational capability. Multiprocessor System-on-Chips (MPSoCs) emerged due to the demand for higher power computation and throughput. These MPSoCs can be either homogeneous or heterogeneous based upon the nature of the processors utilized. The heterogeneous MPSoCs contain interconnected processors with different power performance profiles. IoT applications exhibit diversity and heterogeneous MPSoCs have the advantage of adapting hardware modules required for the application additionally they achieve higher energy-efficiency compared to homogeneous processors.

The ITRS report shows that MPSoCs will have enormous number of processors in the future required to process real-time data extensive applications either at the edge-device or edge/fog level of the network. Thus, the traditional bus based interconnect for inter-processors communications on chip architecture will become bottle neck due to the poor scalability and limited band width.The Network-on-Chip (NoC) based communication has several advantages over hierarchical (STBus, Advance Micro-controller and Bus Architecture) as well as traditional bus architecture in terms of scalability, flexibility and performance [35].

Modern embedded systems use MPSoCs as the computing platform for real-time computational extensive applications in WSN due to their higher performance, exceptional Quality-of-Service (QoS), and remarkable reliability. Reducing energy consumption in these modern embedded systems used as SNs is one of the elementary objective because higher energy consumption not only increases the heat dissipation and carbon footprints but also decreases the overall network lifetime. Dynamic Voltage and Frequency Scaling (DVFS) is a useful technique applied on embedded systems for minimizing the processing energy consumption by dynamically varying the operating frequency and supply voltage. Dynamic Power Management (DPM) is another powerful technique to reduce the energy consumption. DPM shutdowns the processor or switches it to a low-power state i.e. sleeping mode when it is in an idle state and similarly wakes it up when needed.

Scheduling and mapping integrated with DVFS and/or DPM can be used to properly allocate set of tasks on the processors of MPSoC architecture in order to reduce energy consumption. Real-time applications impose some restrictions such as precedence and deadline constraints.

The consequences on the application can be observed when the task deadlines are not met. Precedence constraints enforce to perform application task ordering. Thus, these constraints complicate the task mapping and scheduling on multiprocessor system and consequently, this N-hard problem of scheduling becomes more challenging. Heuristics are required to find a near optimal solutions for energy-aware scheduling for task with precedence and deadline constraints.

The communication unit of a sensor node consumes higher energy compared to the sensing and processing units [67,68]. Therefore, sensor nodes clustering is an effective technique deployed to increase the energy-efficiency of the WSN. In clustering process, the non-cluster nodes basically join a neighboring Cluster Head (CH) to form a cluster. This process of cluster formation plays a significant role to decrease the energy consumption of a network. Clustering in WSN not only provides data aggregation, scalability, bandwidth conservation but also prolongs LT of the network by decreasing the communication energy consumption of the sensor nodes. In clustering process, the sensor nodes are partitioned into groups called clusters. Where each cluster has its own leader known as CH. The CH collects the data within the cluster from its member sensor nodes, aggregates the collected data and transmits it to the Base Station (BS). Data collected by the CH is either transmitted directly to the BS or through intermediate CHs and/or sensor nodes i.e. using multi hop communication. The information from the BS is transmitted to the Cloud for further processing and visualization purposes.

We investigate three distinct problems in this thesis. We study the contention-aware energy-efficient scheduling for set of task with deadline and precedence constraints on heterogeneous NoC-MSoCs. The problem of dependent task scheduling on heterogeneous NoC-MPSoCs with processors having different energy performance profiles is also highlighted as a future research direction by Ishak in [215]. Furthermore, we also investigate energy-efficient clustering of the sensor nodes to increase the lifetime of the network.

1. **Contribution of Chapter 5:** In this chapter, we investigate contention-aware and energy-efficient static scheduling using heterogeneous NoC-MPSoC for real-time tasks with an individual deadline and precedence constraints. Unlike other schedulers task ordering, mapping, and voltage assignment are performed in an integrated manner to mini-

mize the processing energy while explicitly reduce contention between the communications and communication energy. Furthermore, both dynamic voltage and frequency scaling and dynamic power management are used for energy consumption optimization. The developed Contention-aware Integrated Task Mapping and Voltage Assignment (CITM-VA) static scheduler performs tasks ordering using Earliest Latest Finish Time First (ELFTF) strategy that assigns priorities to the tasks having shorter Latest Finish Time (LFT) over the tasks with longer LFT. It remaps every task to a processor and/or discrete voltage level that reduces processing energy consumption. Similarly, the communication energy is minimized by assigning discrete voltage levels to the NoC links. Further, total energy efficiency is achieved by putting the processor into a low-power state when feasible. Moreover, this algorithm resolves the contention between communications that traverse the same link by allocating links to communications with higher priority. The results obtained through extensive simulations of real benchmarks demonstrate that CITM-VA outperforms state-of-the-art scheduling algorithms and achieves an average $\sim$ 30% total energy improvement. Additionally, it maintains high Quality-of-Service (QoS) and robustness for real-time applications.

2. **Contribution of Chapter 6:** In this chapter, we study energy-efficient and contention-aware static scheduling for tasks with precedence and deadline constraints on heterogeneous VFI based NoC-MPSoCs (VFI-NoC-HMPSoC) with DVFS-enabled processors. Unlike the existing population-based optimization algorithms, we proposed a novel population based algorithm called ARSH-FATI that can dynamically switch between explorative and exploitative search modes at run-time. Our static scheduler ARHS-FATI collectively performs task mapping, task ordering, and voltage scaling. Consequently, its performance is superior to the existing state-of-the-art approach proposed for homogeneous VFI based NoC-MPSoCs. We also developed a communication contention-aware Earliest Edge Consistent Deadline First (EECDF) task ordering algorithm and gradient descent inspired voltage scaling algorithm called Energy Gradient Decent (EGD). We introduced a notion of Energy Gradient (EG) that guides EGD in its search for islands voltage settings and minimize the total energy consumption. Conducted the experiments

on 8 real benchmarks adopted from Embedded Systems Synthesis Benchmarks (E3S). Our static scheduling algorithm, ARSH-FATI outperformed state-of-the-art heuristics and achieved an average energy-efficiency of $\sim 24\%$ and $\sim 30\%$ over CA-TMES-Search and CA-TMES-Quick respectively.

3. **Contribution of Chapter 7:** Wireless Sensor Network (WSN) consists of a large number of sensor nodes distributed over a certain target area. The WSN plays a vital role in surveillance, advanced healthcare, and commercialized industrial automation. Enhancing energy-efficiency of the WSN is a prime concern because higher energy consumption restricts LT of the network. Clustering is a powerful technique widely adopted to increase LT of the network and reduce the transmission energy consumption. In this chapter we develop a novel ARSH-FATI based Cluster Head Selection (ARSH-FATI-CHS) algorithm integrated with a heuristic called Novel Ranked based Clustering (NRC) in order to reduce the communication energy consumption of the sensor nodes while efficiently enhancing LT of the network. Unlike other population based algorithms ARSH-FATI-CHS dynamically switches between exploration and exploitation of the search process during run-time to achieve higher performance trade-off and maximally increase network LT. ARSH-FATI-CHS considers the residual energy, communication distance parameters, and workload during CHs selection. We simulate our proposed ARSH-FATI-CHS and generate various results to determine the performance of the WSN in terms of network LT. We compare our results with state-of-the-art Particle Swarm Optimization (PSO) based clustering and we prove that our developed ARSH-FATI-CHS energy-efficient sensor nodes clustering approach improves the network LT by $\sim 25\%$. ARSH-FATI-CHS also outperforms LEACH and PSO-C while achieving an average LT improvements of $\sim 60\%$ and $\sim 40\%$ respectively. Concisely in this chapter we reduced the transmission energy consumption of the sensor nodes in WSN while in chapter 5 and chapter 6 we decreased their processing energy consumption by proper tasks scheduling using DVFS and DPM.

8.2 Future Work

Though we have made significant advances in the field of energy-aware dependent task scheduling on Network-on-Chip based Multiprocessor System-on-Chips (NoC-MPSoCs) and energy-efficient clustering for sensor nodes in Wireless Sensor Network (WSN) however, there are several other open research problems that can be addressed in the future.

1. Coarse-grained software pipelining also known as re-timing is a powerful technique applied at the task level to transform the intra-period precedence constraints into inter-period precedence constraints of the application represented by a Directed Acyclic Graph (DAG). Re-timing primarily reduces the wasted slack and it can be applied to periodic/streaming applications. Re-timing is and effective system level energy optimization technique that can be integrated with DVFS for achieving maximum energy savings. However, unfortunately re-timing causes increased latency and memory overhead. Memory-overhead introduced due to re-timing is a considerable issue because in the worst case it may introduce violations of the memory capacity bounds. One of the future research problem can be integrating re-timing technique with our static scheduling approach while aiming to decrease the memory-overhead and prologue along with energy consumption reduction for dependent tasks with precedence and deadline constraints. Moreover, the voltage levels for both the tasks and communication messages are applied using heuristic in our scheduling approach. An Integer Linear Programming (ILP) can also be deployed in our scheduling approach to assign discrete voltage and frequency levels to the tasks for executions on the processors. Similarly this ILP based voltage scaling technique can also be deployed for reducing the NoC links energy consumption for inter-processor communications. Furthermore, the optimal voltage and frequency levels can be determined using Linear Programming (LP) for reducing both the processing and inter-processor communications. This LP based voltage scaling approach can provide maximum energy-efficiency as it would apply the voltage level at which there is no idle slack available in the processor. However, the disadvantage of this continuous frequency assignment approach is that unlimited number of frequencies would be required to execute the tasks or perform

communications on NoC. The extended our contention-aware energy-efficient static task scheduling scheme can be tested on various other types of tasks e.g. DAG, Conditional Task Graphs (CTGs) and sporadic tasks.

2. One of our future research investigation would be performing energy-aware online task scheduling for tasks with precedence and deadline constraints on heterogeneous NoC-MPSoCs. In online scheduling the application can change at run-time moreover, there no prior knowledge available about the actual execution time during run-time. Thus, online scheduling problem is even more complex and harder than offline task scheduling. A task in online scheduling can execute before its worst case execution time subsequently, generating an idle slack in the processor which can be utilized by using voltage scaling technique. Moreover, Dynamic Power Management (DPM) can be deployed to further reduce the energy consumption and minimize the idle slack. One of the biggest challenge in the designing online algorithms for energy-aware task scheduling is the overhead and degree of complexity. Subsequently, an energy-aware online scheduling approach with low degree of complexity for NoC-MPSoCs can be developed in the future. Moreover, though both the DVFS and DPM are energy saving techniques however a balanced integration of both is necessary for total energy consumption reduction. For example, a task can be run on high frequency and DPM can be used to switch the processor into sleep mode for static power consumption reduction but it will increase switching overhead and the dynamic power consumption. Similarly using lower voltage level for task execution may reduce the dynamic power but there may be still some idle slack available. Therefore, an optimal integration of both the energy savings techniques would be required to develop low complexity energy-efficient online scheduling technique for dependent tasks.

3. Often the energy consumed on information transmission from sensors nodes to the based station is higher than the processing energy. Therefore, clustering approach is adopted to intelligently divide the sensor nodes in groups for communication energy reduction. In chapter 5 we performed heuristic based energy-aware clustering to efficiently enhance the energy-efficiency and network lifetime. We performed only static energy-aware clustering, this work can be extended to perform dynamic energy-efficient clustering. Low complexity,

efficient, and fast running heuristic can be developed for dynamic clustering. Moreover, experimental results can be produced for different scenarios such as when the base station is static while sensor nodes are dynamic and when both the sensor nodes and base station are dynamic. Security and protection of privacy is also one of the major challenges in Internet-of-Things (IoT). The IoT being as a technological revolution has already started to integrate in our everyday lives where a large volume of data is collected and distributed. This has also attracted malicious intruders to access the valuable information. If the data and information are not properly handled then there is a high risk of attack in the process of data transmission and sharing which results in a serious security threat. Thus our clustering approach can also be integrated privacy protection mechanism to provide energy-aware and secure environment. Furthermore, we transmit the gathered data from CHs to the BS in one hope Thus, multi-hop routing can be adopted by using the shorted Euclidean distance between the CH and BS.

8.3 Summary

We successfully reduced the overall energy consumption of a wireless network using task scheduling and nodes clustering. Two computing architecture NoC-MPSoC and VFI-NoC-MPSoC have been deployed for task scheduling. Furthermore, clustering is formulated as a scheduling problem to reduce the transmission energy consumption. Summarizing the future work re-timing techniques can be applied at the application level to increase the degree of parallelism and efficiently utilizing the processors slack. Online algorithms for tasks mapping and scheduling can be developed for DAG tasks with deadline and precedence constraints. Similarly at the network level multi-hop routing can be used to further reduce the transmission energy of the CHs. Moreover, energy-aware clustering algorithms can be developed for scenario when sensor nodes and BS are changing their positions.

Bibliography

[1] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-things-based smart cities: Recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017.

[2] R. Dou and G. Nan, "Optimizing sensor network coverage and regional connectivity in industrial iot systems," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1351–1360, 2017.

[3] J. Colding and S. Barthel, "An urban ecology critique on the "smart city" model," *Journal of Cleaner Production*, vol. 164, pp. 95–101, 2017.

[4] M. De Jong, S. Joss, D. Schraven, C. Zhan, and M. Weijnen, "Sustainable–smart–resilient–low carbon–eco–knowledge cities; making sense of a multitude of concepts promoting sustainable urbanization," *Journal of Cleaner production*, vol. 109, pp. 25–38, 2015.

[5] Y. Shen and H. Ju, "Energy-efficient cluster-head selection based on a fuzzy expert system in wireless sensor networks," in *Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications*, pp. 110–113, IEEE Computer Society, 2011.

[6] L.-m. Ang, K. P. Seng, L. W. Chew, L. S. Yeong, and W. C. Chia, "Wireless multimedia sensor network technology," in *Wireless multimedia sensor networks on reconfigurable hardware*, pp. 5–38, Springer, 2013.

[7] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 88–97, Acm, 2002.

[8] C. Perera, C. H. Liu, S. Jayawardena, and M. Chen, "A survey on internet of things from industrial market perspective," *IEEE Access*, vol. 2, pp. 1660–1679, 2014.

[9] J. H. Kim, "A survey of iot security: Risks, requirements, trends, and key technologies," *Journal of Industrial Integration and Management*, vol. 2, no. 02, p. 1750008, 2017.

[10] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[11] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[13] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pp. 877–880, IEEE, 2012.

[14] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 1–14, ACM, 2010.

[15] J. Li, Y. K. Li, X. Chen, P. P. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1206–1216, 2015.

[16] C. I. ROTUNĂ, C. E. CÎRNU, and A. GHEORGHIȚĂ, "Implementing smart city solutions: Smart city map and city drop," *Quality of Life (1018-0389)/Calitatea Vietii*, vol. 28, no. 3, 2017.

[17] D. Li, J. Cao, and Y. Yao, "Big data in smart cities," *Science China Information Sciences*, vol. 58, no. 10, pp. 1–12, 2015.

[18] G. Deepalakshmi, K. S. Subramaniam, and S. Subramani, "A survey of iot based smart cities," *Data Mining and Knowledge Engineering*, vol. 10, no. 2, pp. 21–24, 2018.

[19] L. Coll and R. Simpson, "The internet of things and challenges for consumer protection," 2019.

[20] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[21] S. Marvin and A. Luque-Ayala, "Urban operating systems: Diagramming the city," *International Journal of Urban and Regional Research*, vol. 41, no. 1, pp. 84–103, 2017.

[22] S. . K. C. Kumar and Vijay, "A strategy for elimination of data redundancy in internet of things (iot) based wireless sensor network (wsn)," 2018.

[23] Z. Zhou, D. Zhao, L. Liu, and P. C. Hung, "Energy-aware composition for wireless sensor networks as a service," *Future Generation Computer Systems*, vol. 80, pp. 299–310, 2018.

[24] C. Alcaraz, P. Najera, J. Lopez, and R. Roman, "Wireless sensor networks and the internet of things: Do we need a complete integration?," in *1st International Workshop on the Security of the Internet of Things (SecIoT'10)*, 2010.

[25] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[26] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[27] É. L. Souza, E. F. Nakamura, and R. W. Pazzi, "Target tracking for sensor networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 30, 2016.

[28] D. C. Harrison, W. K. Seah, and R. Rayudu, "Rare event detection and propagation in wireless sensor networks," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 58, 2016.

[29] D. Kandris, M. Tsagkaropoulos, I. Politis, A. Tzes, and S. Kotsopoulos, "Energy efficient and perceived qos aware video routing over wireless multimedia sensor networks," *Ad Hoc Networks*, vol. 9, no. 4, pp. 591–607, 2011.

[30] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer networks*, vol. 51, no. 4, pp. 921–960, 2007.

[31] Z.-J. Zhang, C.-F. Lai, and H.-C. Chao, "A green data transmission mechanism for wireless multimedia sensor networks using information fusion," *IEEE Wireless Communications*, vol. 21, no. 4, pp. 14–19, 2014.

[32] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems," in *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2011*, pp. 447–454, IEEE, 2011.

[33] 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*, 2018.

[34] A. A. Abbo, R. P. Kleihorst, and B. Schueler, "Xetal-ii: A low-power massively-parallel processor for video scene analysis," *Journal of Signal Processing Systems*, vol. 62, no. 1, pp. 17–27, 2011.

[35] 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.

[36] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE personal communications*, vol. 7, no. 5, pp. 16–27, 2000.

[37] L. Yan, L. Renfa, X. Cheng, and Y. Fei, "Hw-sw framework for multimedia applications on mpsoc: practice and experience," *Journal of computers*, vol. 4, no. 3, pp. 238–244, 2009.

[38] Y. Sasagawa and A. Mori, "High-level video analytics pc subsystem using soc with heterogeneous multicore architecture," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1051–1059, 2016.

[39] M. Magno, F. Tombari, D. Brunelli, L. Di Stefano, and L. Benini, "Multimodal video analysis on self-powered resource-limited wireless smart camera," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 2, pp. 223–235, 2013.

[40] S. Saponara, L. Fanucci, and E. Petri, "A multi-processor noc-based architecture for real-time image/video enhancement," *Journal of Real-Time Image Processing*, vol. 8, no. 1, pp. 111–125, 2013.

[41] I. Ahmed, A. Ahmad, F. Piccialli, A. K. Sangaiah, and G. Jeon, "A robust features-based person tracker for overhead views in industrial environment," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1598–1605, 2018.

[42] A. Safaei, Q. J. Wu, and Y. Yang, "System-on-a-chip (soc)-based hardware acceleration for foreground and background identification," *Journal of the Franklin Institute*, vol. 355, no. 4, pp. 1888–1912, 2018.

[43] H. Meng, M. Freeman, N. Pears, and C. Bailey, "Real-time human action recognition on an embedded, reconfigurable video processing architecture," *Journal of Real-Time Image Processing*, vol. 3, no. 3, pp. 163–176, 2008.

[44] F. Karray, W. M. Jmal, M. Abid, D. Houssaini, A. M. Obeid, S. M. Qasim, and M. S. BenSaleh, "Architecture of wireless sensor nodes for water monitoring applications: From microcontroller-based system to soc solutions," in *Environmental Instrumentation and Measurements (IMEKO), 2014 5th IMEKO TC19 Symposium on*, pp. 20–24, 2014.

[45] A. Aliyu, A. H. Abdullah, O. Kaiwartya, Y. Cao, J. Lloret, N. Aslam, and U. M. Joda, "Towards video streaming in iot environments: Vehicular communication perspective," *Computer Communications*, vol. 118, pp. 93–119, 2018.

[46] I. A. Khatib, F. Poletti, D. Bertozzi, L. Benini, M. Bechara, H. Khalifeh, A. Jantsch, and R. Nabiev, "A multiprocessor system-on-chip for real-time biomedical monitoring and analysis: Ecg prototype architectural design space exploration," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 13, no. 2, p. 31, 2008.

[47] G. Kavya and V. ThulasiBai, "Wearable advanced single chip ecg telemonitoring system using sopc," *IEICE Electronics Express*, vol. 11, no. 6, pp. 20140097–20140097, 2014.

[48] A. Iranfar, A. Pahlevan, M. Zapater, M. Žagar, M. Kovač, and D. Atienza, "Online efficient bio-medical video transcoding on mpsocs through content-aware workload allocation," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 949–954, IEEE, 2018.

[49] T. K. H. Nguyen, *Low power architecture for fall detection system.* PhD thesis, Université Nice Sophia Antipolis, 2015.

[50] A. B. Ahmed, Y. Kimezawa, and A. B. Abdallah, "Towards smart health monitoring system for elderly people," in *4th International Conference on Awareness Science and Technology*, pp. 248–253, IEEE, 2012.

[51] J. Álvarez-Bermejo, D. Morales-Santos, E. Castillo-Morales, L. Parrilla, and J. López-Ramos, "Efficient image-based analysis of fruit surfaces using ccd cameras and smartphones," *The Journal of Supercomputing*, pp. 1–12, 2018.

[52] N. Jangid and B. Sharma, "Cloud computing and robotics for disaster management," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 20–24, IEEE, 2016.

[53] S. Niar, A. Yurdakul, O. Unsal, T. Tugcu, and A. Yuceturk, "A dynamically reconfigurable architecture for emergency and disaster management in its," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 479–484, IEEE, 2014.

[54] C. Tran, "Structural-damage detection with big data using parallel computing based on mpsoc," *International Journal of Machine Learning and Cybernetics*, vol. 7, no. 6, pp. 1213–1223, 2016.

[55] M. Hassan, "Heterogeneous mpsocs for mixed criticality systems: Challenges and opportunities," *IEEE Design & Test*, 2017.

[56] H. Youness, M. Moness, and M. Khaled, "Mpsocs and multicore microcontrollers for embedded pid control: A detailed study," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2122–2134, 2014.

[57] A. Abraham and S. Das, *Computational intelligence in power engineering*, vol. 302. Springer, 2010.

[58] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 6, pp. 797–819, 2011.

[59] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 2015.

[60] E. Gelenbe and Y. Caseau, "The impact of information technology on energy consumption and carbon emissions," *Ubiquity*, vol. 2015, no. June, p. 1, 2015.

[61] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of mimo and cooperative mimo techniques in sensor networks," *IEEE Journal on selected areas in communications*, vol. 22, no. 6, pp. 1089–1098, 2004.

[62] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[63] E. L. de Souza Carvalho, N. L. V. Calazans, and F. G. Moraes, "Dynamic task mapping for mpsocs," *IEEE Design & Test of Computers*, vol. 27, no. 5, pp. 26–35, 2010.

[64] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 international conference on Power aware computing and systems*, pp. 1–8, 2010.

[65] S. Mittal and J. S. Vetter, "A survey of cpu-gpu heterogeneous computing techniques," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, p. 69, 2015.

[66] T. R. da Rosa, V. Larréa, N. Calazans, and F. G. Moraes, "Power consumption reduction in mpsocs through dfs," in *2012 25th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pp. 1–6, IEEE, 2012.

[67] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[68] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041–1054, 2016.

[69] P. S. Rao, P. K. Jana, and H. Banka, "A particle swarm optimization based energy efficient cluster head selection algorithm for wireless sensor networks," *Wireless networks*, vol. 23, no. 7, pp. 2005–2020, 2017.

[70] F. Karray, M. W. Jmal, M. Abid, M. S. BenSaleh, and A. M. Obeid, "A review on wireless sensor node architectures," in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pp. 1–8, IEEE, 2014.

[71] J. Ceng and R. Leupers, "A methodology for efficient multiprocessor system on chip software development," tech. rep., Lehr-und Forschungsgebiet Software für Systeme auf Silizium, 2011.

[72] W. Wolf, "The future of multiprocessor systems-on-chips," in *Proceedings of the 41st annual Design Automation Conference*, pp. 681–685, ACM, 2004.

[73] V. Rajaraman, "Multi-core microprocessors," *Resonance*, vol. 22, no. 12, pp. 1175–1192, 2017.

[74] A. A. Jerraya and W. Wolf, "The what, why, and how of mpsocs," *Multiprocessor Systems-on-Chips*, pp. 1–18, 2005.

[75] S. Mishra, N. K. Singh, and V. Rousseau, *System on Chip Interfaces for Low Power Design*. Morgan Kaufmann, 2015.

[76] M. Bohr, "The new era of scaling in an soc world," in *Solid-State Circuits Conference-Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pp. 23–28, IEEE, 2009.

[77] E. W. Wachter, G. V. Merrett, B. M. Al-Hashimi, and A. K. Singh, "Reliable mapping and partitioning of performance-constrained opencl applications on cpu-gpu mpsocs," in *Proceedings of the 15th IEEE/ACM Symposium on Embedded Systems for Real-Time Multimedia*, pp. 78–83, ACM, 2017.

[78] T. A. Claasen, "An industry perspective on current and future state of the art in system-on-chip (soc) technology," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1121–1137, 2006.

[79] L. Chen, N. Boichat, and T. Mitra, "Customized mpsoc synthesis for task sequence," in *Application Specific Processors (SASP), 2011 IEEE 9th Symposium on*, pp. 16–21, IEEE, 2011.

[80] T. Dorta, J. Jiménez, J. L. Martín, U. Bidarte, and A. Astarloa, "Overview of fpga-based multiprocessor systems," in *2009 International Conference on Reconfigurable Computing and FPGAs*, pp. 273–278, IEEE, 2009.

[81] Z. Wang, W. Liu, J. Xu, B. Li, R. Iyer, R. Illikkal, X. Wu, W. H. Mow, and W. Ye, "A case study on the communication and computation behaviors of real applications in noc-based mpsocs," in *2014 IEEE Computer Society Annual Symposium on VLSI*, pp. 480–485, IEEE, 2014.

[82] S. Hesham, J. Rettkowski, D. Goehringer, and M. A. A. El Ghany, "Survey on real-time networks-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1500–1517, 2017.

[83] L. Torres, P. Benoit, G. Sassatelli, M. Robert, F. Clermidy, and D. Puschini, "An introduction to multi-core system on chip–trends and challenges," in *Multiprocessor System-on-Chip*, pp. 1–21, Springer, 2011.

[84] Y.-J. Chen, W.-W. Chang, C.-Y. Liu, C.-E. Wu, B.-Y. Chen, and M.-Y. Tsai, "Processors allocation for mpsocs with single isa heterogeneous multi-core architecture," *IEEE Access*, vol. 5, pp. 4028–4036, 2017.

[85] A. Pathania, A. E. Irimiea, A. Prakash, and T. Mitra, "Power-performance modelling of mobile gaming workloads on heterogeneous mpsocs," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2015.

[86] W. Quan and A. D. Pimentel, "A hybrid task mapping algorithm for heterogeneous mpsocs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 1, p. 14, 2015.

[87] S. A. Ishak, H. Wu, and U. U. Tariq, "Energy-aware task scheduling on heterogeneous noc-based mpsocs," in *2017 IEEE 35th International Conference on Computer Design (ICCD)*, pp. 165–168, IEEE, 2017.

[88] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for dynamic task mapping in noc-based heterogeneous mpsocs," in *Rapid System Prototyping, 2007. RSP 2007. 18th IEEE/IFIP International Workshop on*, pp. 34–40, IEEE, 2007.

[89] L. Benini and G. De Micheli, "Networks on chips: A new paradigm for component-based mpsoc design," *Proc. MPSoC*, 2004.

[90] E. Salminen, V. Lahtinen, K. Kuusilinna, and T. Hamalainen, "Overview of bus-based system-on-chip interconnections," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, vol. 2, pp. II–II, IEEE, 2002.

[91] S. Pasricha, N. D. Dutt, and M. Ben-Romdhane, "Bmsyn: Bus matrix communication architecture synthesis for mpsoc," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 8, pp. 1454–1464, 2007.

[92] P. Gandhani, "Moving from amba ahb to axi bus in soc designs: A comparative study," *International Journal of Computer Science & Emerging Technologies*, vol. 2, no. 4, 2011.

[93] U. Y. Ogras and R. Marculescu, *Modeling, analysis and optimization of network-on-chip communication architectures*, vol. 184. Springer Science & Business Media, 2013.

[94] É. Cota, A. de Morais Amory, and M. S. Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-chip*. Springer Science & Business Media, 2011.

[95] J.-J. Han, M. Lin, D. Zhu, and L. T. Yang, "Contention-aware energy management scheme for noc-based multicore real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 691–701, 2015.

[96] U. Ullah 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*, 2019.

[97] V. Sanju, N. Chiplunkar, M. Khalid, S. Joshi, and J. Nirmala, "A performance study of 2d mesh and torus for network-on-chip-based system," in *Proc. Int. Conf. Emerging Research in Computing, Information, Communication and Applications, Bangalore, India (Elsevier, Amsterdam, 2013)*, pp. 47–51, 2013.

[98] C. Wang, X. Li, J. Zhang, X. Zhou, and A. Wang, "A star network approach in heterogeneous multiprocessors system on chip," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1404–1424, 2012.

[99] U. V. Rane, *Network on Chip (NoC) Platform for High Performance Computing*. PhD thesis, Goa University, 2017.

[100] A. AMBA, "Axi and ace™ protocol specification," 2011.

[101] F. Siddiqui, M. Hagan, and S. Sezer, "Embedded policing and policy enforcement approach for future secure iot technologies," 2018.

[102] U. Y. Ogras, R. Marculescu, and D. Marculescu, "Variation-adaptive feedback control for networks-on-chip with multiple clock domains," in *Proceedings of the 45th annual Design Automation Conference*, pp. 614–619, ACM, 2008.

[103] H. Gu, J. Xu, and W. Zhang, "A low-power fat tree-based optical network-on-chip for multiprocessor system-on-chip," in *Proceedings of the conference on Design, Automation and Test in Europe*, pp. 3–8, European Design and Automation Association, 2009.

[104] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The nostrum backbone-a communication protocol stack for networks on chip," in *VLSI Design, 2004. Proceedings. 17th International Conference on*, pp. 693–696, IEEE, 2004.

[105] Z. Lu, *Design and analysis of on-chip communication for network-on-chip platforms*. PhD thesis, KTH, 2007.

[106] Tilera, "SoCs tilera-gx," 2019.

[107] T. Corporation, "SoCs tilera-gx," 2019.

[108] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, pp. 195–202, IEEE, 2002.

[109] S. Garg, D. Marculescu, R. Marculescu, and U. Ogras, "Technology-driven limits on dvfs controllability of multiple voltage-frequency island designs: a system-level perspective," in *Proceedings of the 46th Annual Design Automation Conference*, pp. 818–821, ACM, 2009.

[110] P. P. Pande, A. Ganguly, and K. Chakrabarty, *Design Technologies for Green and Sustainable Computing Systems*. Springer, 2013.

[111] J.-J. Han, X. Wu, D. Zhu, H. Jin, L. T. Yang, and J.-L. Gaudiot, "Synchronization-aware energy management for vfi-based multicore real-time systems," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1682–1696, 2012.

[112] M. E. Gerards, J. L. Hurink, and J. Kuper, "On the interplay between global dvfs and scheduling tasks with precedence constraints," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1742–1754, 2015.

[113] M. Short, "The case for non-preemptive, deadline-driven scheduling in real-time embedded systems," in *Lecture notes in engineering and computer science: Proceedings of the World Congress on Engineering*, 2010.

[114] D. K. Friesen, "Tighter bounds for lpt scheduling on uniform processors," *SIAM Journal on Computing*, vol. 16, no. 3, pp. 554–560, 1987.

[115] E. G. Coffman and J. L. Bruno, *Computer and job-shop scheduling theory.* John Wiley & Sons, 1976.

[116] K. Chronaki, A. Rico, M. Casas, M. Moretó, R. M. Badia, E. Ayguadé, J. Labarta, and M. Valero, "Task scheduling techniques for asymmetric multi-core systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2074–2087, 2016.

[117] 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, p. 66, 2019.

[118] F. Pop, C. Dobre, and V. Cristea, "Genetic algorithm for dag scheduling in grid environments," in *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, pp. 299–305, IEEE, 2009.

[119] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 4, no. 2, pp. 175–187, 1993.

[120] K. P. B. P. Banerjee, "An approximate algorithm for the partitionable independent task scheduling problem," *Urbana*, vol. 51, p. 61801, 1990.

[121] K. Cao, J. Zhou, M. Yin, T. Wei, and M. Chen, "Static thermal-aware task assignment and scheduling for makespan minimization in heterogeneous real-time mpsocs," in *System and Software Reliability (ISSSR), International Symposium on*, pp. 111–118, IEEE, 2016.

[122] T. Wei, J. Zhou, K. Cao, P. Cong, M. Chen, G. Zhang, X. S. Hu, and J. Yan, "Cost-constrained qos optimization for approximate computation real-time tasks in heterogeneous mpsocs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2017.

[123] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys (CSUR)*, vol. 31, no. 4, pp. 406–471, 1999.

[124] H. I. Ali, B. Akesson, and L. M. Pinho, "Combining dataflow applications and real-time task sets on multi-core platforms," in *Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems*, pp. 60–63, ACM, 2017.

[125] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682–694, 2014.

[126] L. Huang and Q. Xu, "Performance yield-driven task allocation and scheduling for mpsocs under process variation," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pp. 326–331, IEEE, 2010.

[127] J. Zhou, J. Yan, K. Cao, Y. Tan, T. Wei, M. Chen, G. Zhang, X. Chen, and S. Hu, "Thermal-aware correlated two-level scheduling of real-time tasks with reduced processor energy on heterogeneous mpsocs," *Journal of Systems Architecture*, vol. 82, pp. 1–11, 2018.

[128] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[129] U. U. Tariq and H. Wu, "Energy-aware scheduling of periodic conditional task graphs on mpsocs," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, p. 13, ACM, 2017.

[130] 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.

[131] Y. Wang, D. Liu, M. Wang, Z. Qin, and Z. Shao, "Optimal task scheduling by removing inter-core communication overhead for streaming applications on mpsoc," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pp. 195–204, IEEE, 2010.

[132] P. Krömer, J. Platoš, V. Snášel, and A. Abraham, "A comparison of many-threaded differential evolution and genetic algorithms on cuda," in *2011 Third World Congress on Nature and Biologically Inspired Computing*, pp. 509–514, IEEE, 2011.

[133] H. Orsila, "Optimizing algorithms for task graph mapping on multiprocessor system on chip," *Tampereen teknillinen yliopisto. Julkaisu-Tampere University of Technology. Publication*, vol. 972, 2011.

[134] 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. 13778–13804, 2015.

[135] N. Kumar and D. P. Vidyarthi, "A ga based energy aware scheduler for dvfs enabled multicore systems," *Computing*, pp. 1–23, 2017.

[136] S.-h. Kang, H. Yang, S. Kim, I. Bacivarov, S. Ha, and L. Thiele, "Static mapping of mixed-critical applications for fault-tolerant mpsocs," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2014.

[137] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of scheduling*, vol. 12, no. 4, p. 417, 2009.

[138] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in mpsocs," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pp. 1–6, IEEE, 2007.

[139] J. L. Hill, *System architecture for wireless sensor networks*. PhD thesis, University of California, Berkeley, 2003.

[140] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, "Energy-aware scheduling for real-time systems: a survey," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 1, p. 7, 2016.

[141] A. Das, M. J. Walker, A. Hansson, B. M. Al-Hashimi, and G. V. Merrett, "Hardware-software interaction for run-time power optimization: A case study of embedded linux on multicore smartphones," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 165–170, IEEE, 2015.

[142] Z. Najam, M. Y. Qadri, and S. Najam, "Real-time implementation of dvfs enhanced leon3 mpsoc on fpga," in *Intelligent and Advanced Systems (ICIAS), 2016 6th International Conference on*, pp. 1–6, IEEE, 2016.

[143] C. Poellabauer, L. Singleton, and K. Schwan, "Feedback-based dynamic voltage and frequency scaling for memory-bound real-time applications," in *Real Time and Embedded Technology and Applications Symposium, 2005. RTAS 2005. 11th IEEE*, pp. 234–243, IEEE, 2005.

[144] Y. Wang, D. Liu, Z. Qin, and Z. Shao, "Optimally removing intercore communication overhead for streaming applications on mpsocs," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 336–350, 2013.

[145] L. Yuan and G. Qu, "A combined gate replacement and input vector control approach for leakage current reduction," *IEEE transactions on very large scale integration (vlsi) systems*, vol. 14, no. 2, pp. 173–182, 2006.

[146] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage current reduction in cmos vlsi circuits by input vector control," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 140–154, 2004.

[147] S. Mukhopadhyay, C. Neau, R. T. Cakici, A. Agarwal, C. H. Kim, and K. Roy, "Gate leakage reduction for scaled devices using transistor stacking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 716–730, 2003.

[148] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the 2004 international symposium on Low power electronics and design*, pp. 32–37, ACM, 2004.

[149] D. Srikanth and S. S. Bindu, "Analysis of clock gating and power gating techniques on sequential circuits," 2015.

[150] P. R. Panda, A. Shrivastava, B. V. N. Silpa, and K. Gummidipudi, "Basic low power digital design," in *Power-efficient System Design*, pp. 11–39, Springer, 2010.

[151] N. A. Latiff, C. C. Tsimenidis, and B. S. Sharif, "Energy-aware clustering for wireless sensor networks using particle swarm optimization," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, IEEE, 2007.

[152] S. K. Gupta and P. K. Jana, "Energy efficient clustering and routing algorithms for wireless sensor networks: Ga based approach," *Wireless Personal Communications*, vol. 83, no. 3, pp. 2403–2423, 2015.

[153] A. Mohajerani and D. Gharavian, "An ant colony optimization based routing algorithm for extending network lifetime in wireless sensor networks," *Wireless Networks*, vol. 22, no. 8, pp. 2637–2647, 2016.

[154] P. Kuila and P. K. Jana, "A novel differential evolution based clustering algorithm for wireless sensor networks," *Applied soft computing*, vol. 25, pp. 414–425, 2014.

[155] H. Zhang, S. Zhang, and W. Bu, "A clustering routing protocol for energy balance of wireless sensor network based on simulated annealing and genetic algorithm," *International Journal of Hybrid Information Technology*, vol. 7, no. 2, pp. 71–82, 2014.

[156] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*, pp. 10–pp, IEEE, 2000.

[157] B. Jan, H. Farman, H. Javed, B. Montrucchio, M. Khan, and S. Ali, "Energy efficient hierarchical clustering approaches in wireless sensor networks: A survey," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[158] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE wireless communications*, vol. 11, no. 6, pp. 6–28, 2004.

[159] K. Srinivasan and K. S. Chatha, "Integer linear programming and heuristic techniques for system-level low power scheduling on multiprocessor architectures under throughput constraints," *INTEGRATION, the VLSI journal*, vol. 40, no. 3, pp. 326–354, 2007.

[160] S. Tosun, "Energy-and reliability-aware task scheduling onto heterogeneous mpsoc architectures," *The Journal of Supercomputing*, vol. 62, no. 1, pp. 265–289, 2012.

[161] U. U. Tariq and H. Wu, "Energy-aware scheduling of conditional task graphs with deadlines on mpsocs," in *IEEE 34th International Conference on Computer Design (ICCD), 2016*, pp. 265–272, IEEE, 2016.

[162] D. Shin and J. Kim, "Communication power optimization for network-on-chip architectures," *Journal of Low Power Electronics*, vol. 2, no. 2, pp. 165–176, 2006.

[163] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pp. 164–169, IEEE, 2008.

[164] T. Maqsood, S. Ali, S. U. Malik, and S. A. Madani, "Dynamic task mapping for network-on-chip based systems," *Journal of Systems Architecture*, vol. 61, no. 7, pp. 293–306, 2015.

[165] N. Chatterjee, S. Paul, P. Mukherjee, and S. Chattopadhyay, "Deadline and energy aware dynamic task mapping and scheduling for network-on-chip based multi-core platform," *Journal of Systems Architecture*, vol. 74, pp. 61–77, 2017.

[166] Y. Wang, Z. Shao, H. C. Chan, D. Liu, and Y. Guan, "Memory-aware task scheduling with communication overhead minimization for streaming applications on bus-based multiprocessor system-on-chips," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1797–1807, 2014.

[167] J. Singh, S. Betha, B. Mangipudi, and N. Auluck, "Contention aware energy efficient scheduling on heterogeneous multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1251–1264, 2015.

[168] Y. Wang, H. Liu, D. Liu, Z. Qin, Z. Shao, and E. H.-M. Sha, "Overhead-aware energy optimization for real-time streaming applications on multiprocessor system-on-chip," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 2, p. 14, 2011.

[169] P. Ghosh, A. Sen, and A. Hall, "Energy efficient application mapping to noc processing elements operating at multiple voltage levels," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pp. 80–85, IEEE Computer Society, 2009.

[170] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[171] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 35, 2013.

[172] F. Vafaee and P. C. Nelson, "An explorative and exploitative mutation scheme," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE, 2010.

[173] S. Olafsson, "A general model for task distribution on an open heterogenous processor system," *IEEE transactions on systems, man, and cybernetics*, vol. 25, no. 1, pp. 43–58, 1995.

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

[175] 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, 2017.

[176] 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)*, pp. 142–146, IEEE, 2018.

[177] A. Gammoudi, A. Benzina, M. Khalgui, and D. Chillet, "Energy-efficient scheduling of real-time tasks in reconfigurable homogeneous multicore platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[178] 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)*, pp. 1099–1104, IEEE, 2018.

[179] 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*, pp. 761–766, IEEE, 2017.

[180] S. Ninomiya, K. Sakanushi, Y. Takeuchi, and M. Imai, "Task allocation and scheduling for voltage-frequency islands applied noc-based mpsoc considering network congestion," in *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium on*, pp. 107–112, IEEE, 2012.

[181] S. Pagani, J.-J. Chen, and M. Li, "Energy efficiency on multi-core architectures with multiple voltage islands," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1608–1621, 2015.

[182] J. Liu and J. Guo, "Energy efficient scheduling of real-time tasks on multi-core processors with voltage islands," *Future Generation Computer Systems*, vol. 56, pp. 202–210, 2016.

[183] D. Shin, W. Kim, S. Kwon, and T. H. Han, "Communication-aware vfi partitioning for gals-based networks-on-chip," *Design Automation for Embedded Systems*, vol. 15, no. 2, pp. 89–109, 2011.

[184] W. Jang and D. Z. Pan, "A voltage-frequency island aware energy optimization framework for networks-on-chip," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, pp. 420–432, 2011.

[185] M. Digalwar, P. Gahukar, and S. Mohan, "Energy efficient real time scheduling on multi-core processor with voltage islands," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1245–1251, IEEE, 2018.

[186] S. Lindsey and C. S. Raghavendra, "Pegasis: Power-efficient gathering in sensor information systems," in *Proceedings, IEEE aerospace conference*, vol. 3, pp. 3–3, IEEE, 2002.

[187] V. Loscri, G. Morabito, and S. Marano, "A two-levels hierarchy for low-energy adaptive clustering hierarchy (tl-leach)," in *IEEE vehicular technology conference*, vol. 62, p. 1809, IEEE; 1999, 2005.

[188] F. Xiangning and S. Yulin, "Improvement on leach protocol of wireless sensor network," in *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, pp. 260–264, IEEE, 2007.

[189] M. B. Yassein, Y. Khamayseh, and W. Mardini, "Improvement on leach protocol of wireless sensor network (vleach," in *Int. J. Digit. Content Technol. Appl. 2009*, Citeseer, 2009.

[190] A. Al-Baz and A. El-Sayed, "A new algorithm for cluster head selection in leach protocol for wireless sensor networks," *International journal of communication systems*, vol. 31, no. 1, p. e3407, 2018.

[191] A. Rahmanian, H. Omranpour, M. Akbari, and K. Raahemifar, "A novel genetic algorithm in leach-c routing protocol for sensor networks," in *2011 24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 001096–001100, IEEE, 2011.

[192] J.-L. Liu and C. V. Ravishankar, "Leach-ga: Genetic algorithm-based energy-efficient adaptive clustering protocol for wireless sensor networks," *International Journal of Machine Learning and Computing*, vol. 1, no. 1, p. 79, 2011.

[193] W. B. Heinzelman, A. P. Chandrakasan, H. Balakrishnan, *et al.*, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on wireless communications*, vol. 1, no. 4, pp. 660–670, 2002.

[194] X. Li, L. Xu, H. Wang, J. Song, and S. X. Yang, "A differential evolution-based routing algorithm for environmental monitoring wireless sensor networks," *Sensors*, vol. 10, no. 6, pp. 5425–5442, 2010.

[195] A. M. Zungeru, L.-M. Ang, and K. P. Seng, "Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison," *Journal of Network and Computer Applications*, vol. 35, no. 5, pp. 1508–1536, 2012.

[196] P. K. Agarwal and C. M. Procopiuc, "Exact and approximation algorithms for clustering," *Algorithmica*, vol. 33, no. 2, pp. 201–226, 2002.

[197] S. Guru, S. Halgamuge, and S. Fernando, "Particle swarm optimisers for cluster formation in wireless sensor networks," in *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 319–324, IEEE, 2005.

[198] B. Singh and D. K. Lobiyal, "A novel energy-aware cluster head selection based on particle swarm optimization for wireless sensor networks," *Human-Centric Computing and Information Sciences*, vol. 2, no. 1, p. 13, 2012.

[199] P. S. Rao and H. Banka, "Energy efficient clustering algorithms for wireless sensor networks: novel chemical reaction optimization approach," *Wireless Networks*, vol. 23, no. 2, pp. 433–452, 2017.

[200] P. Kuila, S. K. Gupta, and P. K. Jana, "A novel evolutionary approach for load balanced clustering problem for wireless sensor networks," *Swarm and Evolutionary Computation*, vol. 12, pp. 48–56, 2013.

[201] D. C. Hoang, P. Yadav, R. Kumar, and S. K. Panda, "Real-time implementation of a harmony search algorithm-based clustering protocol for energy-efficient wireless sensor networks," *IEEE transactions on industrial informatics*, vol. 10, no. 1, pp. 774–783, 2013.

[202] C. S. M. Cisse, K. Ahmed, C. Sarr, and M. A. Gregory, "Energy efficient hybrid clustering algorithm for wireless sensor network," in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 38–43, IEEE, 2016.

[203] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. Al Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 262–275, 2007.

[204] D. Li and J. Wu, "Energy-efficient contention-aware application mapping and scheduling on noc-based mpsocs," *Journal of Parallel and Distributed Computing*, vol. 96, pp. 1–11, 2016.

[205] U. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 3, pp. 330–341, 2009.

[206] S. Bhushan, R. Pal, and S. G. Antoshchuk, "Energy efficient clustering protocol for heterogeneous wireless sensor network: A hybrid approach using ga and k-means," in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pp. 381–385, IEEE, 2018.

[207] J. Xu, W. Liu, F. Lang, Y. Zhang, and C. Wang, "Distance measurement model based on rssi in wsn," *Wireless Sensor Network*, vol. 2, no. 08, p. 606, 2010.

[208] G. P. Gupta and S. Jha, "Integrated clustering and routing protocol for wireless sensor networks using cuckoo and harmony search based metaheuristic techniques," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 101–109, 2018.

[209] D. Liu, J. Spasic, G. Chen, and T. Stefanov, "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*, pp. 1–10, IEEE, 2015.

[210] A. Lukefahr, S. Padmanabha, R. Das, R. Dreslinski Jr, T. F. Wenisch, and S. Mahlke, "Heterogeneous microarchitectures trump voltage scaling for low-power cores," in *Proceedings of the 23rd international conference on Parallel architectures and compilation*, pp. 237–250, ACM, 2014.

[211] Y. Cheng, L. Zhang, Y. Han, and X. Li, "Thermal-constrained task allocation for interconnect energy reduction in 3-d homogeneous mpsocs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 2, pp. 239–249, 2013.

[212] T. Hagras and J. Janeček, "Static vs. dynamic list-scheduling performance comparison," *Acta Polytechnica*, vol. 43, no. 6, 2003.

[213] J. D. Monte and K. R. Pattipati, "Scheduling parallelizable tasks to minimize make-span and weighted response time," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 32, no. 3, pp. 335–345, 2002.

[214] J. Wang, Y. Cao, B. Li, H.-j. Kim, and S. Lee, "Particle swarm optimization based clustering algorithm with mobile sink for wsns," *Future Generation Computer Systems*, vol. 76, pp. 452–457, 2017.

[215] S. A. Ishak, *Energy-Aware Task Scheduling for MPSoC-based Embedded Systems.* PhD thesis, University of New South Wales, Sydney, Australia, 2018.