ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

ALZEMIRO HENRIQUE LUCAS DA SILVA

# DYNAMIC THERMAL MANAGEMENT FOR NOC-BASED MANY-CORE SYSTEMS

Porto Alegre
2021

**PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL**
**SCHOOL OF TECHNOLOGY**
**COMPUTER SCIENCE GRADUATE PROGRAM**

# DYNAMIC THERMAL MANAGEMENT FOR NOC-BASED MANY-CORE SYSTEMS

**ALZEMIRO HENRIQUE LUCAS DA SILVA**

Doctoral Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. Fernando Gehm Moraes
Co-Advisor: Prof. André Luís del Mestre Martins

**Porto Alegre**
**2021**

## Ficha Catalográfica

S586d Silva, Alzemiro Henrique Lucas da

Dynamic thermal management for NoC-based many-core systems / Alzemiro Henrique Lucas da Silva. – 2021.
115.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Fernando Gehm Moraes.
Co-orientador: Prof. Dr. André Luís del Mestre Martins.

1. Many-core systems. 2. Temperature monitoring. 3. Dynamic thermal management. 4. Lifetime reliability. 5. Mapping. I. Moraes, Fernando Gehm. II. Martins, André Luís del Mestre. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

**ALZEMIRO HENRIQUE LUCAS DA SILVA**

# DYNAMIC THERMAL MANAGEMENT FOR NOC-BASED MANY-CORE SYSTEMS

This Doctoral Thesis has been submitted in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science, of the Computer Science Graduate Program, School of Technology of the Pontifical Catholic University of Rio Grande do Sul

Sanctioned on August 12, 2021.

## COMMITTEE MEMBERS:

# GERENCIAMENTO TÉRMICO DINÂMICO EM SISTEMAS MANY-CORE BASEADOS EM REDES INTRA CHIP

## RESUMO

Nodos tecnológicos recentes permitem fabricar bilhões de transistores em uma pequena área de silício, replicando estruturas idênticas, resultando em sistemas many-core. No entanto, a densidade de potência pode limitar a quantidade de potência que o sistema pode consumir. Um many-core em seu desempenho máximo pode levar a violar temperatura segura e, consequentemente, resultar em problemas de confiabilidade. Técnicas de gerenciamento térmico dinâmico (DTM) foram propostas para garantir que sistemas many-core funcionem com bom desempenho sem comprometer a confiabilidade. Técnicas DTM dependem de dados precisos de monitoramento de temperatura. Esta Tese revisa trabalhos recentes de DTM e propõe um novo método para permitir o monitoramento da temperatura em tempo de execução em um sistema many-core, novas heurísticas tendo por função custo a temperatura, bem como métodos de atuação, mapeamento e migração de tarefas e controle dinâmico de frequência e tensão (DVFS). Os trabalhos do estado-da-arte sobre técnicas de DTM apresentam heurísticas complexas de atuação em tempo de execução, com foco principalmente no mapeamento de tarefas, e não se apresentam métodos de monitoramento de temperatura, comprometendo a aplicabilidade em sistemas reais. O estado da arte também apresenta trabalhos voltados ao gerenciamento dinâmico de confiabilidade (DRM), onde o objetivo principal é garantir maior confiabilidade ao sistema, utilizando as mesmas técnicas de atuação para controlar a temperatura. Esta Tese também revisa alguns dos efeitos de envelhecimento em circuitos integrados e analisa resultados de confiabilidade relacionados ao tempo de vida para as heurísticas propostas. As principais contribuições desta Tese incluem: (*i*) um acelerador de hardware para estimativa térmica (TEA), (*ii*) uma heurística de gerenciamento de temperatura proporcional, integral e derivativa (PIDTM); (*iii*) uma heurística de gerenciamento de temperatura tendo por função custo restrições de energia (TMEC). TEA possibilitou o monitoramento preciso da temperatura em tempo de execução no many-core de referência, permitindo a validação das propostas de DTM. O PIDTM reduziu em até 7,15% a temperatura de pico em um cenário de alta carga de trabalho, enquanto o TMEC melhorou em até 82,9% a vida útil esperada do sistema.

**Palavras-Chave:** Sistemas Many-core, Monitoramento de Temperatura, Gerenciamento Dinâmico da Temperatura, Confiabilidade de Tempo de Vida, Mapeamento.

# DYNAMIC THERMAL MANAGEMENT FOR NOC-BASED MANY-CORE SYSTEMS

## ABSTRACT

Recent technology nodes enable to deploy billions of transistors in a small silicon area by replicating identical structures, resulting in many-core systems. However, power density may limit the amount of power the system can consume. A many-core at its maximum performance may lead to violate safe temperature definition and, consequently, result in reliability issues. Dynamic Thermal Management (DTM) techniques have been proposed to guarantee that many-core systems run at good performance without compromising reliability. DTM techniques rely on accurate temperature monitoring data. This Thesis reviews recent DTM works and proposes a new method to enable runtime temperature monitoring in a many-core system and new heuristics for thermal-aware application mapping, migration, and dynamic frequency and voltage scaling (DVFS) actuation, considering temperature and energy consumption. The state-of-art study on DTM techniques presents complex mechanisms for runtime actuation, focusing mainly on task mapping, and shows no concern about temperature monitoring methods, compromising the applicability in real systems. The state-of-art also presents works targeting dynamic reliability management (DRM), where the main objective is to ensure longer lifetime for the system, using the same actuation knobs used to control the temperature. This thesis also reviews some of the aging effects and analyses the lifetime reliability results for the proposed heuristics. The main contributions of this Thesis include: (*i*) a thermal estimation accelerator (TEA), (*ii*) a proportional, integral and derivative temperature management (PIDTM) heuristic; (*iii*) a temperature management heuristic having as cost function energy constraints (TMEC). TEA enabled accurate runtime temperature monitoring in the reference many-core, enabling the validation of the DTM proposals. PIDTM reduced up to 7.15% the overall peak temperature in a high workload scenario, while TMEC improved up to 82.9% in the system's expected lifetime.

**Keywords:** Many-core Systems, Temperature Monitoring, Dynamic Thermal Management, Lifetime Reliability, Mapping.

# LIST OF FIGURES

Figure 2.1 – (a) Arrangement of 64 cores on physical floorplan.(b) Folded torus.(c) Physical and logical views [Yang et al., 2017b]. . . . . . . . . . . . . . . . . . . . . . . . 22
Figure 2.2 – Heterogeneous Quad-core cluster architecture [Yang et al., 2017a]. . 23
Figure 2.3 – (a) Thermal-aware mapping strategy.(b) Contention in SMART NoC [Liu et al., 2018]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
Figure 2.4 – (a) Uniform TSP illustration for 6 active cores.(b) TSP illustration with different power constraints [Pagani et al., 2014]. . . . . . . . . . . . . . . . . . . . . . . 26
Figure 2.5 – Heterogeneous many-core setup used to evaluate seBoost proposal [Pagani et al., 2015]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
Figure 2.6 – Effect of temperature (°C) over the time in a PE and its neighbors [Castilhos et al., 2016]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
Figure 2.7 – MORM overview [Martins et al., 2019a]. . . . . . . . . . . . . . . . . . . . . . 32
Figure 2.8 – DTaPO overview [Mohammed et al., 2020]. . . . . . . . . . . . . . . . . . . 35
Figure 2.9 – LifeGuard overview [Rathore et al., 2019]. . . . . . . . . . . . . . . . . . . . 36
Figure 2.10 – Thermal-cycling-aware resource manager overview [Haghbayan et al., 2020]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
Figure 2.11 – (a) Task graph. (b) Execution time. (c) Possible schedules for the example task graph [Liu et al., 2019]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
Figure 3.1 – NoC-based many-core system with peripherals (adapted from [Ruaro et al., 2019]). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
Figure 3.2 – Application graph model examples. . . . . . . . . . . . . . . . . . . . . . . . . 45
Figure 3.3 – Inter task communication interface (adapted from [Ruaro et al., 2019]). 46
Figure 3.4 – Quantum-based simulation paradigm. . . . . . . . . . . . . . . . . . . . . . . . 48
Figure 3.5 – Assembly code snippet used for processor characterization [Martins, 2018]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
Figure 3.6 – 3x3 NoC traffic scenario for the 5-port router characterization [Martins, 2018]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
Figure 3.7 – Hierarchical organization of the PEs [Martins, 2018]. . . . . . . . . . . . . . 55
Figure 3.8 – Hierarchical energy monitoring of the reference architecture [Martins et al., 2015]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
Figure 4.1 – Architectural partitioning of the system. (a) Top view with heat sink resistances. (b) One block with its lateral and vertical thermal resistances. (c) Silicon die layer divided into three blocks [Huang et al., 2006]. . . . . . . . . . 57
Figure 4.2 – Comparison of temperature estimation models in a 6x6 system. . . . . 58

Figure 4.3 – TEA Architecture Overview. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
Figure 4.4 – Hierarchical temperature monitoring strategy. . . . . . . . . . . . . . . . . . . . . 65
Figure 4.5 – PE and TEA area comparison. Library C28SOI_SC_12 (28nm). . . . . 67
Figure 4.6 – Typical scenario temperature results (8x8 system) for (a) HotSpot and (b) TEA. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
Figure 5.1 – Observe-Decide-Act paradigm is the methodology for the development of Dynamic Thermal Management applied to the reference platform. . . 70
Figure 5.2 – Application Admission and Task Mapping protocol. . . . . . . . . . . . . . . 72
Figure 5.3 – Task migration protocol. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
Figure 5.4 – CPU Power States Illustration. Adapted from [Mohammed et al., 2020]. 75
Figure 5.5 – Relationship between the algorithms. Continuous lines: event that triggers an action. Dashed lines: input for a given algorithm. . . . . . . . . . . . . . 80
Figure 5.6 – Heatmaps extracted from scenario 1. (a) Original MORM. (b) Temperature cluster selection. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
Figure 5.7 – Peak and average temperatures (top) and power consumption (bottom): (a) MORM (200mW power budget), (b) Temeprature cluster selection. 86
Figure 5.8 – MORMxTPES temperature results. . . . . . . . . . . . . . . . . . . . . . . . . . . 87
Figure 5.9 – MORMxTPES energy consumption and execution time. . . . . . . . . . . . 87
Figure 5.10 – MORMxTPESM temperature results. . . . . . . . . . . . . . . . . . . . . . . . . 88
Figure 5.11 – MORMxTPESM energy consumption and execution time. . . . . . . . . . 88
Figure 5.12 – MORMxPIDTM temperature results. . . . . . . . . . . . . . . . . . . . . . . . . . 89
Figure 5.13 – MORMxPIDTM energy consumption and execution time. . . . . . . . . . . 89
Figure 5.14 – MORMxTMEC temperature results. . . . . . . . . . . . . . . . . . . . . . . . . . 90
Figure 5.15 – MORMxTMEC energy consumption and execution time. . . . . . . . . . . 90
Figure 5.16 – MORM, Patterning, TPESM, PIDTM, TMEC peak temperature results [da Silva et al., 2020]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
Figure 5.17 – Heatmaps comparison between different mapping and migration strategies [da Silva et al., 2020]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
Figure 5.18 – Spiral, Patterning and PIDTM peak temperature results [da Silva et al., 2021a]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
Figure 5.19 – Comparison between peak temperatures in 2 scenarios: (a) High Workload (HW); (b) Dynamic Workload (DW1) [da Silva et al., 2021a]. . . . . . 94
Figure 6.1 – Scanning electron micrographs of the open circuit induced by EM: (a) normal topographic image; (b) voltage contrast image [Read et al., 2011]. 96
Figure 6.2 – RAMP 2.0 usage flow. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101

Figure 6.3 – Expected MTTF comparison between MORM, patterning, PIDTM and TMEC for the HW scenario. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
Figure 6.4 – Spiral, Patterning, PIDTM and TMEC normalized MTTF results. . . . . 103

# LIST OF TABLES

Table 2.1 – State-of-art summary. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
Table 3.1 – RTL simulation results obtained from instruction classes (adapted from [Martins et al., 2014]). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
Table 3.2 – Power characterization results and energy estimation for each instruction class of the processor. Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC (T=1ns) (adapted from [Martins et al., 2014]). . . . . . . . . . . . . . . . . . . . . 51
Table 3.3 – Processor energy estimation error (adapted from [Martins et al., 2014]). 52
Table 3.4 – Router average power. Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC (T=1ns) (adapted from [Martins, 2018]). . . . . . . . . . . . . . . . . . . . . . . . 53
Table 3.5 – CACTI-P Report for a scratchpad Memory (28nm, 1.0V, 25℃) . . . . . . 54
Table 4.1 – *TEA* memory and register sizes. . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
Table 4.2 – Power and Performance results for TEA and software implementation of the temperature model. TEA implementation: Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC (T=1ns). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
Table 4.3 – Temperature error results for different scenarios. Reference: HotSpot. 68
Table 5.1 – DVFS levels used in the reference platform. Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
Table 5.2 – Thermal evaluation scenarios. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
Table 7.1 – Summary of Publications. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

# LIST OF ACRONYMS

ACPI – Advanced Configuration and Power Interface

AES – Advanced Encryption Standard

ALU – Arithmetic Logic Unit

API – Application Programming Interface

AV – Audio and Video

AW – Average Workload

CBM – Cluster Based Management

CM – Cluster Manager

CMOS – Complementary Metal-Oxide-Semiconductor

CMP – Chip Multiprocessor

CPU – Central Processor Unit

CTG – Communicating Task Graph

DMNI – Direct Memory Network Interface

DPM – Dynamic Power Management

DRM – Dynamic Reliability Management

DTM – Dynamic Thermal Management

DTW – Digital Time Warping

DVFS – Dynamic Voltage and Frequency Scaling

DW – Dynamic Workload

EM – Electromigration

FIT – Failure In Time

FPGA – Field-Programmable Gate Array

FSM – Finite State Machine

GM – Global Manager

GPPC – General Purpose Processing Cores

HW – High Workload

IO – Input/Output

IP – Intellectual Property

ISA – Instruction Set Architecture

ISS – Instruction Set Simulator

ILP – Integer-Linear Programming

LLC – Last Level Cache

LUT – Look Up Table

LW – Low Workload

MAC – Multiplier ACcumulator

MILP – Mixed Integer-Linear Programing

MORM – Multi-Objective Resource Management

MPEG – Moving Picture Expert Group

MP – Manager PE

MPI – Message Passing Interface

MTTF – Mean Time To Failure

NBTI – Negative Bias Temperature Instability

NI – Network Interface

NN – Neural Network

NOC – Network-on-Chip

NOP – No OPeration

ODA – Observe-Decide-Act/

OS – Operating System

OOO – Out-of-Order

OVP – Open Virtual Platforms

PAM – Per Application Management

PE – Processing Element

PID – Proportional, Integral and Derivative

PIDTM – Proportional, Integral and Derivative Temperature Management

PLE – Physical Layout Estimation

PLL – Phase Locked Loop

PMOS – P-type Metal-Oxide-Semiconductor

PV – Process Variation

RAMP – Reliability Aware Micro-Processors

RC – Resistance and Capacitance

REST – Reliability ESTimation

RISC – Reduced Instruction Set Computer

RTL – Register Transfer Level

SDF – Standard Delay Format

SM – Stress Migration

SOFR – Sum Of Failure Rates

SOI – Silicon on Insulator

SP – Slave Processor

TC – Thermal Cycling

TCF – Toggle Count Format

TCS – Temperature Cluster Selection

TDDB – Thermal Dependent Dielectric Breakdown

TDP – Thermal Design Power

TEA – Thermal Estimation Accelerator

TMEC – Temperature Management with Energy Constraint

TPES – Temperature Processing Element Selection

TPESM – Temperature Processing Element Selection with Migration

TSP – Thermal Safe Power

UEFI – Unified Extensible Firmware Interface

VCD – Value Change Dump

VF – Voltage and Frequency

VHDL – VHSIC Hardware Description Language

VHSIC – Very High Speed Integrated Circuit

VLSI – Very Large Scale Integration

# CONTENTS

**1 INTRODUCTION . . . 17**
1.1 MOTIVATION . . . 18
1.2 PROBLEM DEFINITION . . . 19
1.3 THESIS STATEMENT . . . 19
1.4 OBJECTIVES . . . 19
1.5 ORIGINAL CONTRIBUTIONS . . . 20
1.6 DOCUMENT ORGANIZATION . . . 20

**2 STATE-OF-THE-ART . . . 21**
2.1 LEI YANG ET AL. . . . 21
2.1.1 FOTONOC - LEI YANG ET AL. . . . 21
2.1.2 QUAD-CORE CLUSTER ARCHITECTURE - LEI YANG ET AL. . . . 22
2.2 WEICHEN LIU ET AL. . . . 23
2.3 PAGANI ET AL. . . . 25
2.3.1 TSP: THERMAL SAFE POWER - PAGANI ET AL. . . . 25
2.3.2 SEBOOST - PAGANI ET AL. . . . 26
2.4 SHI SHA ET AL. . . . 28
2.5 MENGQUAN LI ET AL. . . . 29
2.6 CASTILHOS ET AL. . . . 30
2.7 MARTINS ET AL. . . . 32
2.8 ANUP DAS ET AL. . . . 33
2.9 MOHAMMED ET AL. . . . 34
2.10 RATHORE ET AL. . . . 35
2.11 HAGHBAYAN ET AL. . . . 36
2.12 WEICHEN LIU ET AL. . . . 37
2.13 RELATED WORK ANALYSIS . . . 39
2.13.1 FINAL REMARKS . . . 42

**3 REFERENCE ARCHITECTURE AND BASIC CONCEPTS . . . 43**
3.1 REFERENCE ARCHITECTURE . . . 43
3.1.1 CLUSTER-BASED MANAGEMENT . . . 44
3.1.2 GPPC CONNECTION WITH PERIPHERALS . . . 45

3.2 APPLICATION MODEL . . . . . . 45
3.3 ARCHITECTURE MODELS . . . . . . 46
3.3.1 CLOCK CYCLE-ACCURATE MODELS . . . . . . 46
3.3.2 INSTRUCTION-LEVEL MODEL . . . . . . 47
3.3.3 CLOCK CYCLE-ACCURATE AND INSTRUCTION-LEVEL MODEL COMPARISON . . . . . . 48
3.4 POWER ESTIMATION . . . . . . 48
3.4.1 PROCESSOR CHARACTERIZATION . . . . . . 49
3.4.2 ROUTER CHARACTERIZATION . . . . . . 52
3.4.3 MEMORY CHARACTERIZATION . . . . . . 54
3.5 HIERARCHICAL MONITORING . . . . . . 54

**4 TEMPERATURE ESTIMATION . . . . . . 56**
4.1 TEMPERATURE MODELS . . . . . . 56
4.1.1 HOTSPOT . . . . . . 56
4.1.2 CASTILHOS' MODEL . . . . . . 58
4.1.3 MATEX . . . . . . 59
4.2 RUNTIME TEMPERATURE ESTIMATION . . . . . . 59
4.3 THERMAL ESTIMATION ACCELERATOR - TEA . . . . . . 62
4.4 RUNTIME TEMPERATURE MONITORING . . . . . . 64
4.5 RESULTS OF THERMAL MONITORING USING TEA . . . . . . 65
4.5.1 SOFTWARE VERSUS HARDWARE APPROACHES . . . . . . 66
4.5.2 AREA OVERHEAD . . . . . . 67
4.5.3 TEMPERATURE ESTIMATION ACCURACY . . . . . . 67
4.6 FINAL REMARKS . . . . . . 68

**5 DYNAMIC THERMAL MANAGEMENT . . . . . . 70**
5.1 ACTUATION KNOBS . . . . . . 71
5.1.1 APPLICATION ADMISSION . . . . . . 71
5.1.2 TASK MAPPING . . . . . . 73
5.1.3 MIGRATION . . . . . . 73
5.1.4 DYNAMIC VOLTAGE AND FREQUENCY SCALING . . . . . . 74
5.1.5 POWER STATES . . . . . . 75
5.2 DECISION POLICIES . . . . . . 76
5.2.1 ADMISSION . . . . . . 76

5.2.2 MAPPING . . . . . 77
5.2.3 MIGRATION . . . . . 77
5.2.4 DYNAMIC VOLTAGE AND FREQUENCY SCALING . . . . . 78
5.2.5 POWER STATES . . . . . 79
5.3 DECISION HEURISTICS FOR CLUSTER AND PE SELECTION . . . . . 79
5.3.1 TEMPERATURE CLUSTER SELECTION . . . . . 80
5.3.2 TEMPERATURE PE SELECTION AND TEMPERATURE PE SELECTION WITH MIGRATION . . . . . 81
5.3.3 PROPORTIONAL, INTEGRAL AND DERIVATIVE TEMPERATURE MANAGEMENT . . . . . 82
5.3.4 TEMPERATURE MANAGEMENT WITH ENERGY CONSTRAINT . . . . . 83
5.4 RESULTS . . . . . 83
5.4.1 EXPERIMENTAL SETUP . . . . . 84
5.4.2 TEMPERATURE CLUSTER SELECTION – TCS . . . . . 85
5.4.3 PE SELECTION . . . . . 86
5.4.4 SUMMARY OF THE RTL DTM RESULTS . . . . . 90
5.4.5 OVP DTM RESULTS . . . . . 92
5.5 FINAL REMARKS . . . . . 93

**6 LIFETIME RELIABILITY . . . . . 95**
6.1 CONCEPTS . . . . . 95
6.1.1 ELECTROMIGRATION . . . . . 95
6.1.2 TIME DEPENDENT DIELECTRIC BREAKDOWN . . . . . 96
6.1.3 NEGATIVE BIAS TEMPERATURE INSTABILITY . . . . . 97
6.1.4 STRESS MIGRATION . . . . . 97
6.1.5 THERMAL CYCLING . . . . . 98
6.2 LIFETIME SIMULATION TOOLS . . . . . 99
6.2.1 LIFESIM . . . . . 99
6.2.2 REST . . . . . 99
6.2.3 RAMP . . . . . 100
6.3 RESULTS . . . . . 101
6.3.1 RTL PLATFORM . . . . . 101
6.3.2 OVP PLATFORM . . . . . 102
6.4 FINAL REMARKS . . . . . 103

**7 CONCLUSIONS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105**

7.1 FUTURE WORKS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 106

7.2 SUMMARY OF THE PUBLICATIONS PRODUCED DURING THE THESIS . . . . 107

**REFERENCES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 109**

# 1. INTRODUCTION

The steady transistor scaling and the increasing demand for performance led to the development of NoC-based many-core systems. Networks-on-chip (NoCs) provide enhanced performance and scalability for communication on systems with up to thousands of Processing Elements (PEs) [Borkar, 2007]. Many-core systems are designed to take advantage of the large availability of transistors provided by recent CMOS technologies, exploring task parallelism to increase overall system performance.

A significant challenge related to the design of many-core architectures in recent technology nodes is the increased power density which causes the effect called *Dark Silicon* [Esmaeilzadeh et al., 2011], where parts of the circuit need to be switched off or underclocked to keep the system within the physical limits of power density and safe temperature. Although the number of PEs increases in many-core designs, the power consumed by each PE remains constant since the threshold and supply voltage do not scale down with the size of the transistor as in previous technologies [Iizuka, 2015].

The increase in power density also generates the effect of localized overheating, called hotspots. The dynamic temperature behavior can create reliability threats, increase the power leakage and cooling costs [El Ahmad et al., 2018]. The literature proposes many techniques to deal with the increasing challenges imposed by the dark silicon phenomenon. System management techniques control which PEs will be active and will be dark at each moment to control the system power consumption and maintain the system under a power budget or below a critical temperature. Recent system management approaches assume that the system itself drives the adaptation to increase its performance while remaining within the physical limits of operation [Martins et al., 2019a].

The system adaptation may occur at the software or hardware levels [Martins et al., 2019b]. At the software level, the system manager may choose which PE will run each task or even migrate tasks depending on the system state. The system can apply DVFS (Dynamic Voltage and Frequency Scaling) or power gating on idle components to control its power consumption at the hardware level. These approaches require an integrated monitoring structure to allow the observation of the system state to manage the resources.

Monitoring system temperatures allow the implementation of Dynamic Thermal Management (DTM) techniques to ensure the operation within the specified limits, increasing reliability, reducing energy consumption, and possibly increasing lifetime [El Ahmad et al., 2018]. Temperature monitoring is also a challenge in many-core designs since the number of sensors required to measure the temperature in a large many-core system results in several overheads.

## 1.1 Motivation

Dark silicon emerged as a design issue in recent integrated circuits due to the increasing power density. However, dark silicon management offers research opportunities, especially in many-core designs. System management should be designed to fill the gap of performance introduced by physical constraints, monitoring power consumption and temperature distribution, acting to improve energy efficiency, reducing power consumption, and increasing chip reliability and lifetime.

According to the classification proposed by Sha et al. [Sha et al., 2016], DTMs can control the temperature of a many-core system by proactive or reactive approaches. Reactive approaches take actions when a given processor reaches a threshold temperature while proactive approaches rely on a thermal model and previous characterization of applications to ensure that a given mapping decision is thermally safe.

Many strategies are proposed in the literature to manage dark silicon using software and hardware approaches. Some works focus on proactive approaches that require extensive knowledge of the applications executed in the system. Other works use complex thermal models or require advanced monitoring mechanisms, frequently omitted in the work description. However, system management should deal with dynamic workloads to cope with unpredictable user demands, considering thermal-aware mapping, migration, and DVFS to keep the system with a uniform thermal distribution.

The literature also presents complex and sophisticated management models to deal with dark silicon challenges [Pagani et al., 2017, Li et al., 2018]. Although these proposals show good results, these works frequently omit many steps related to the monitoring and to the performance of the proposed algorithms. Due to the numerous challenges of power and temperature monitoring in many-core systems, the proposals are developed with the use of high-level modeling tools such as GEM5 [Binkert et al., 2011] and McPAT [Li et al., 2009]. However, these models present limitations and cannot be executed to guide runtime management strategies.

Many thermal management proposals rely only on proactive approaches to define tasks locations and the speed settings of the cores. These approaches require prior knowledge of the tasks that execute in the system and their power consumption for each speed setting or an estimated value based on the CPU average consumption, which can lead to underestimation or even safe temperature violations.

In this context, this Thesis explores hardware and software approaches to elaborate a dynamic thermal management solution, integrated with power consumption and temperature monitoring, that can manage systems with dynamic workloads and offer the best performance within the physical limits of power density and temperature.

## 1.2 Problem Definition

Recent works propose proactive DTM techniques using thermal models to estimate transient and peak temperatures during system operation to enable the application of mapping heuristics for many-core systems [Yang et al., 2017b, Liu et al., 2018]. Other proactive approaches focus on defining a schedule of high computing power and low energy periods that keep the temperature within safe bounds before the release of tasks [Pagani et al., 2015, Sha et al., 2016, Sha et al., 2018]. These techniques require the execution of a complex thermal model to provide data for system management and extensive task characterization to enable proactive decisions. The reviewed proposals often omit these requirements.

Yang et al. [Yang et al., 2017a] propose heuristics to map tasks in a heterogeneous architecture targeting the conflict between application performance and energy efficiency, also considering task migration in case of a localized high temperature. However, temperature monitoring and migration strategy are not presented in this work.

We claim that the unpredictability of dynamic workloads makes proactive approaches unfeasible for DTM strategies. On the other hand, if the DTM knows when applications start and finish (i.e., pre-characterized workload), proactive approaches relying on temperature prediction may work together with reactive DTMs as an option for enabling runtime DTMs.

## 1.3 Thesis Statement

Thermal and energy management are required in recent technology nodes to keep the operation within the limits of power density, increasing the circuit lifetime reliability. Management techniques employed in many-core designs must be aware of the power consumption and the temperature of each processing element. Thermal-aware task mapping, allied with migration and DVFS must deal with the conflicts between thermal constraints and performance demand. The hypothesis to be fulfilled by this Thesis is the demonstration that DTM techniques, applied in high and dynamic workloads, keep the temperature within safe limits and increases the system lifetime reliability.

## 1.4 Objectives

The main goal of this Thesis is the specification, development, validation, and optimization of a DTM strategy for many-core systems.

The *specific* goals of the Thesis include:

- Proposal of a thermal monitoring technique to guide management decisions;

- Characterization of the hardware components of the reference platform for recent CMOS technology nodes (28nm - C28SOI_SC_12);
- Development of a runtime thermal management for many-core systems, including proactive and reactive approaches;
- Evaluation of the proposal in large systems (64 cores), verifying if the lifetime reliability is improved;
- Comparison of the proposed technique with state-of-the-art approaches regarding the temperature distribution and the lifetime reliability.

## 1.5 Original Contributions

The main original contribution of this Thesis is a novel DTM proposal targeting NoC-based many-core systems. The management strategy relies on simple heuristics to enable thermal-aware task mapping, considering task migration and DVFS actuation to avoid hotspots and temperature violations. The proposal is scalable, able to manage the system with diverse and dynamic workload conditions with little interference in the performance of running tasks on the system.

Other contributions provided by this Thesis include:

- A Hardware-accelerated runtime temperature estimation proposal.
- An Evaluation of the proposed heuristics in different levels of abstraction.
- An analysis of the reliability improvement.

## 1.6 Document Organization

The remaining of this document is organized as follows. Chapter 2 presents the state-of-art in thermal and power management for many-core systems and an analysis of the main related works. Chapter 3 describes the reference architecture and its models in different levels of abstraction. Chapter 4 proposes a hardware-accelerated runtime thermal monitoring strategy. Chapter 5 presents the DTM proposals and their results. Chapter 6 analyses the system lifetime reliability regarding the proposed heuristics for the reference architecture. Chapter 7 concludes this Thesis and draws directions for future works.

# 2. STATE-OF-THE-ART

This Chapter presents a review and a discussion of the related works regarding dynamic thermal management strategies, dynamic reliability management, thermal modeling, and its applications. This Chapter finishes with a comparison of our proposal to the main features of the reviewed works.

## 2.1 Lei Yang et al.

Lei Yang et al. published two works in 2017, proposing different architectures to deal with dark silicon management in many-core systems. The first one uses a folded torus NoC (FoToNoC) [Yang et al., 2017b], created to provide logical clusters of physically distributed cores with reduced communication latency, enabling application tasks in distant cores to communicate with low latency for better thermal distribution. The second one adopts a quad-core cluster architecture [Yang et al., 2017a], with four core types providing the same instruction set architecture (ISA) but different performance and energy-efficiency.

### 2.1.1 FoToNoC - Lei Yang et al.

This work, [Yang et al., 2017b], presents a hierarchical management strategy for heterogeneous many-core systems based in folded torus NoC (FoToNoC). FoToNoC is a communication architecture based on a folded torus topology, on which physically distributed cores are interconnected with reduced communication latency and organized in logically condensed virtual clusters.

The architecture proposed by this work contains a 64-core NoC, organized into an 8x8 array of tiles interconnected by links. Figure 2.1(a) illustrates how cores are physically arranged on the platform numbered from $c_1$ to $c_{64}$, (b) shows how the connections between cores are arranged in the folded torus, and (c) shows the difference between the physical view and the logical view of the clusters.

In application mapping, tasks will be preferentially mapped to cores with less communication delay, such as cores 1, 3, and 17 in Figure 2.1(a). Therefore, as cores 1, 3, and 17 are physically distributed on the floorplan, the produced heat will be better dissipated, and the chip tends to have smaller peak temperatures.

In this architecture, there are four types of cores. Each logical cluster contains one of these cores, including $b$ (*big* cores with the highest performance), $M_1$, $M_2$, and L (*Little* cores with the lowest energy consumption). The cores of different types are arranged

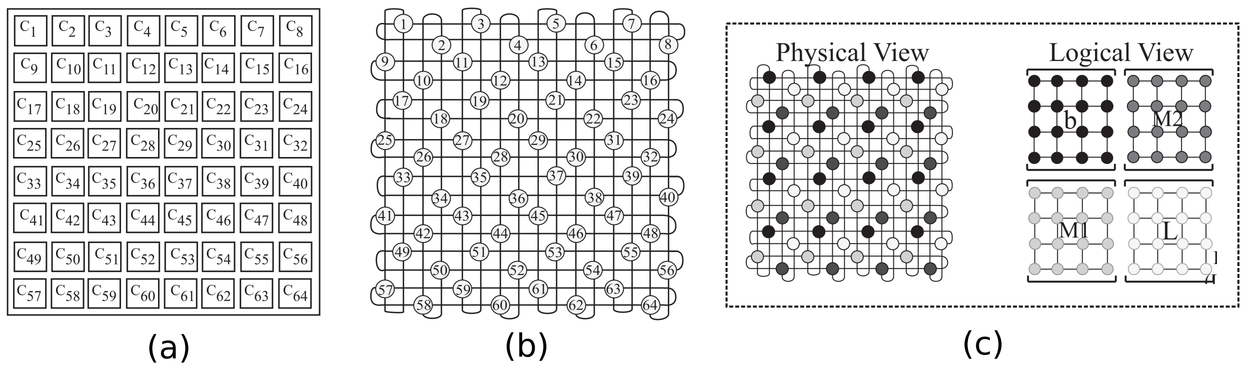


Figure 2.1 – (a) Arrangement of 64 cores on physical floorplan.(b) Folded torus.(c) Physical and logical views [Yang et al., 2017b].

physically and logically, as illustrated in Figure 2.1(c). Authors assume that only one type of core, i.e., one cluster, will be powered on at any time.

This work uses power models from the Alpha 21264 processor in 22-nm technology extracted from McPAT [Li et al., 2009], simulating the microarchitecture using GEM5 [Binkert et al., 2011]. The mapping decisions are made based on runtime transient and peak temperatures calculations provided by *MatEx* [Pagani et al., 2015]. Results show better heat distribution than contiguous cluster mapping and better communication delay than a regular torus network using decentralized mapping.

### 2.1.2 Quad-Core Cluster Architecture - Lei Yang et al.

This work, [Yang et al., 2017a], proposes a quad-core cluster architecture, with four cores types, with different energy efficiencies, where active cores are isolated on-chip to avoid thermal hotspots. This work also proposes a hierarchical management strategy where tasks are classified according to their computational demand and then mapped to the most efficient processor that meets its requirement to save energy and avoid thermal hotspots.

Figure 2.2 shows the quad-core cluster architecture. Each cluster in this architecture has four processors, arranged as a grid on the physical platform, interconnect through a 2D-mesh. Each cluster has a High Performance (HP), a General Purpose (GO), a Power Saving (PA), and a Low Energy (LE) processor. HP processor takes the highest power consumption for compute-intensive workloads, while LE is the most energy-efficient for tasks with lower performance requirements. The clusters communicate through a shared cache implemented in each cluster. Communication among clusters is done through routers by message-passing using XY-routing.

Task mapping is based in an application model that considers a deadline-constrained application referred as a task graph with a set of computing tasks and a set of weighted communication edges. Mapping is done in two phases, inter-cluster mapping and intra-cluster

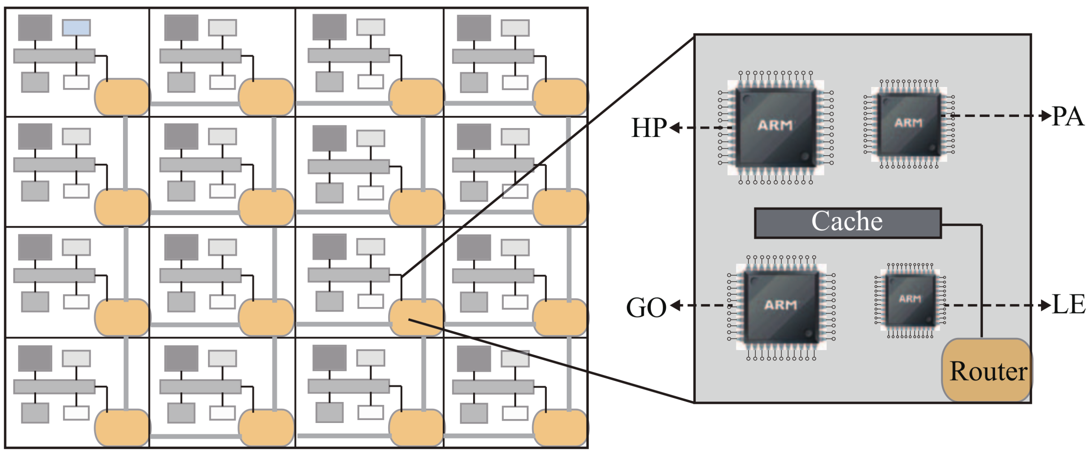


Figure 2.2 – Heterogeneous Quad-core cluster architecture [Yang et al., 2017a].

task assignment to cores. Inter-cluster mapping is done using an external algorithm. Within a cluster, a core with the most power-efficient that meets each task performance requirement is selected with the most power saving voltage-frequency pair.

Separate voltage-frequency islands is implemented in each core. At runtime, under safe temperature, DVFS is implemented to use time slacks to save energy consumption of task execution by decreasing the performance of non-critical tasks without effects on application performance. Migration is also modeled within a cluster to boost a task performance in a faster core or to save power by migrating to a more power-efficient core when the performance is over the one supplied on the current core. Task migration within a cluster is considered to have negligible overhead, since it is done through the shared cache. Authors also consider migrating tasks to a new cluster due to regional high temperature, but claim that an inter-cluster migration will take a relatively long time among additional overheads.

This works uses power models from the Alpha 21264 processor in 22-nm technology extracted from McPAT [Li et al., 2009], and defines the simulation as a Python simulator for deadline-constrained applications. Temperature data is extracted from *MatEx* [Pagani et al., 2015], but many details on simulation process are missing in this paper. Authors claim a reduction in energy consumption, communication latency, average temperature, and peak temperature when comparing with the prior work [Yang et al., 2017b].

## 2.2 Weichen Liu et al.

Liu et al. [Liu et al., 2018] proposes a homogeneous many-core platform on which the routers can be reconfigured to enable multi-hop bypass on the NoC to reduce communication latency between non-adjacent cores. Based on this platform, the authors proposed an Integer-Linear Programming (ILP) model to explore the network reconfiguration architecture to apply task mapping with minimal network contention. Based on the ILP model, a heuristic was proposed, *TopoMap*, to enable the execution in polynomial time with minor penalties in communication and application performance.

The Authors propose a thermal-aware task mapping based on the euclidean distance of cores. In this strategy, tasks will be restrictively mapped to cores that are at least 2-hop away from active cores as long as there are still available cores that meet this restriction, as shown in Figure 2.3(a). Otherwise, the selection is made among candidates that will not cause a safe temperature violation. Authors show that by mapping applications according to this pattern, the thermal reliability can be maintained in a better heat condition, benefiting from the interval arrangement of active/dark cores.

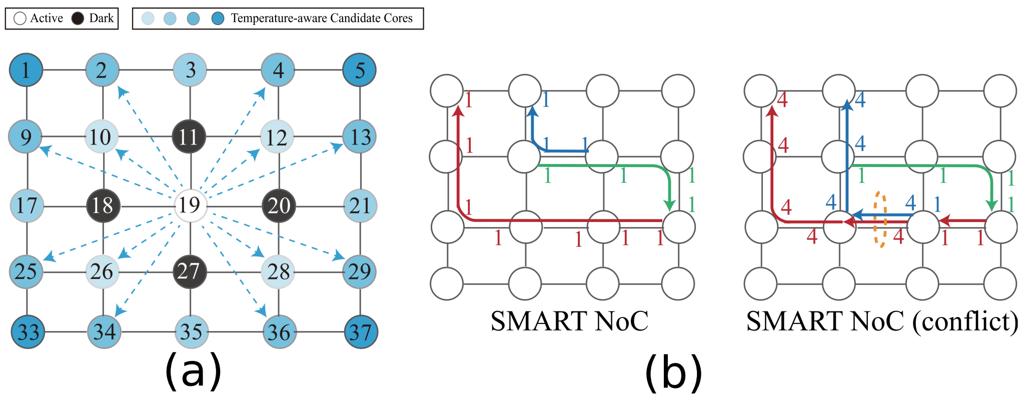


Figure 2.3 – (a) Thermal-aware mapping strategy.(b) Contention in SMART NoC [Liu et al., 2018].

The main drawback of mapping applications in physically decentralized cores is the increased communication distance. Usually, the latency for a packet transmission is linear with the distance of the path it travels and the routing stages in the routers. To overcome the latency problem, the authors propose a mapping algorithm based on SMART NoC. SMART is a reconfigurable network that enables single-cycle multi-hop bypass, where flits can travel multiple routers in one clock cycle, up to a $HPC_{max}$, which is the maximum hop count that a flit can jump. However, SMART is sensitive to conflicts, where flows using the same link should be stopped and buffered (Figure 2.3(b)). Based on these characteristics, a communication contention-aware task mapping is proposed to maximize the utilization of bypass and reduce communication contention, improving communication efficiency. Initially, an ILP algorithm is proposed to explore contention-free mapping on top of SMART NoC, but since the mapping problem is NP-hard, the ILP execution rises rapidly with the applications' sizes. Based on this strategy, a heuristic algorithm is proposed, called *Cont_Map*, where the datapath is analyzed to avoid transmission overlap among tasks.

Finally, the authors propose a thermal-aware mapping algorithm with contention prevention for SMART NoC, using both proposed mapping strategies together, restricting the tasks hop count from 2 to $HPC_{max}$ to explore all candidate cores while reducing the search space. The mapping algorithm uses *MatEx* [Pagani et al., 2015] to obtain transient temperatures of all cores and calculate chip peak temperature at runtime. Simulations are executed using McPAT [Li et al., 2009] to obtain power data, and GEM5 [Binkert et al., 2011] to simulate communication and microarchitecture. Results show an increase in performance of up to 33.5 percent when comparing SMART NoC and a regular mesh NoC due to reduced latency and contention. The Authors also observed a reduction in energy consumption, comparing the proposed architecture with a regular mesh NoC and FoToNoC [Yang et al., 2017b], primarily due to faster execution and communication efficiency.

## 2.3 Pagani et al.

Pagani et al. published two works considering many-core systems in the Dark Silicon era. In the first work [Pagani et al., 2014], the term TSP (Thermal Safe Power) is introduced, in contrast with TDP (Thermal Design Power), to provide new power constraints for many-core systems depending on the number of active cores. The second work [Pagani et al., 2015] focuses on proposing a boosting technique to satisfy the performance needs of specific running applications without sacrificing the performance of other applications unnecessarily.

### 2.3.1 TSP: Thermal Safe Power - Pagani et al.

This work, [Pagani et al., 2014], proposes a new power budget technique for many-core systems to deal with many-core systems in the dark silicon era named Thermal Safe Power (TSP). In this thermal-aware power budget, the power constraint of each core is a function of the number of active cores and the peak temperatures for a particular mapping. Thus, the use of TSP on task mapping can provide better performance than using a single power budget, like TDP (Thermal Design Power), without violating safe temperature.

This paper presents two strategies to compute TSP. The first strategy is to compute a solution for a particular core mapping and ambient temperature. The second is to compute TSP for the worst-case core mappings for $m$ active cores. The authors define a specific core mapping as a column vector $\mathbf{Q}$, where $q_i = 1$ means that node $i$ corresponds to an active core and $q_i = 0$ means that node $i$ corresponds to an inactive core.

The first algorithm computes a uniform value of TSP per-core, for all active cores in $\mathbf{Q}$. That is, one power constraint value for each core in the specified mapping, that results in maximum temperature in the steady-state. The maximum temperature can also be received as an input of the algorithm. This power constraint is defined as $P_{TSP}(\mathbf{Q})$. Figure 2.4(a) presents a thermal map of an example of uniform TSP calculated for two different core mappings. The top numbers are the power consumption, in Watts, of each active core. Bottom numbers in parenthesis are the temperatures in the center of the core, in $^\circ$C. It is possible to observe that when the mapping is decentralized, the TSP number is higher, which means that each core can consume more power without reaching the maximum specified temperature, 80 °C in this example.

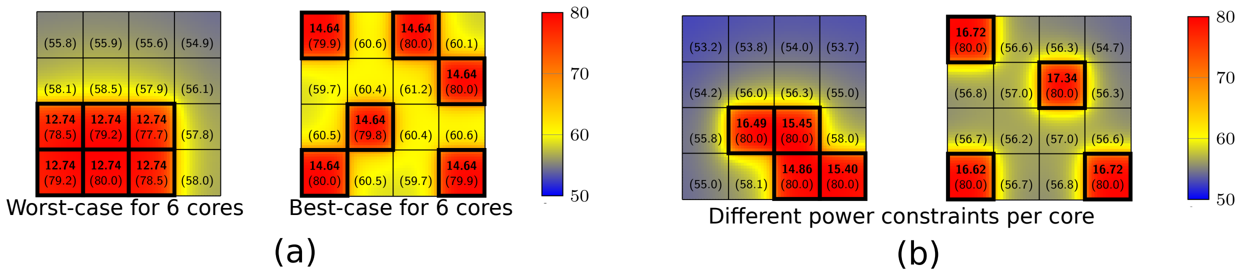


Figure 2.4 – (a) Uniform TSP illustration for 6 active cores.(b) TSP illustration with different power constraints [Pagani et al., 2014].

In Figure 2.4(a) we can also observe that not all cores are at the maximum temperature, so there is room for improvement, meaning that some cores may consume more power. Figure 2.4(b) shows an example of cores consuming different values of power that results in the maximum temperature. For this scenario, the authors present a formulation based in ILP but did not solve the problem in this work.

The second algorithm presents a solution to compute the TSP worst-case value for $m$ active cores. The power constraint value for each active core in any possible core mapping, with $m$ simultaneously active cores, results in a maximum temperature in the steady-state. Figure 2.4(a) shows, on the left, the worst-case mapping when $m = 6$. Using this strategy, mapping decision can be abstracted since this power value is valid for any possible mapping, as opposed to the TSP computed with the first strategy.

The evaluations presented in this work consider a simulated platform using GEM5 [Binkert et al., 2011], McPAT [Li et al., 2009] and HotSpot [Huang et al., 2006]. The Authors compared the TSP values computed for particular mappings with constant power budgets for the whole system. Results show that using TSP as power constraint results in higher total performance for almost all cases. In some cases, using a single power budget for the system can provide better results, once the cores might not have an exact speed step to consume the TSP value.

## 2.3.2 seBoost - Pagani et al.

In this paper, Pagani et al. [Pagani et al., 2015], propose a selective boosting technique for heterogeneous many-core systems called seBoost. The proposal's main objective is to provide an efficient runtime boosting technique that guarantees the runtime requirements of applications, with minimum performance losses for the applications running on the non-boosted cores. The technique is based on prior knowledge of power profiles for every thread and every application for all available voltage and frequency pairs. With this information, the algorithm tries to find the best combination of VF pairs for the boosted and the non boosted cores that will keep the system below the critical temperature – $T_{crit}$. This technique requires runtime temperature monitoring since the initial temperature is needed, and the algorithm constantly calculates steady and transient temperatures to validate VF configuration among cores.

Three scenarios are considered in this work:

- When a required VF level and a maximum boosting time is known: in this scenario, the algorithm finds the throttle-down levels for the non-boosted cores, such that the boosted cores execute at the required VF levels during the entire maximum expected boosting time without violating $T_{crit}$, and the non-boosted cores do not suffer unnecessary performance losses.
- The maximum boosting time is given, but the required VF levels are unknown: for this case, the algorithm tries to maximize the VF levels of the cores requiring boosting for the expected boosting time, minimizing the performance losses for the non-boosted cores. The higher priority is the maximization of the VF levels of the boosted cores.
- The maximum boosting time is unknown: in this scenario, the goal is to find VF levels that can be sustained indefinitely. Instead of finding VF levels where the critical temperature is reached at the end of the boosting time, the algorithm considers that the steady temperature of the system should not exceed $T_{crit}$.

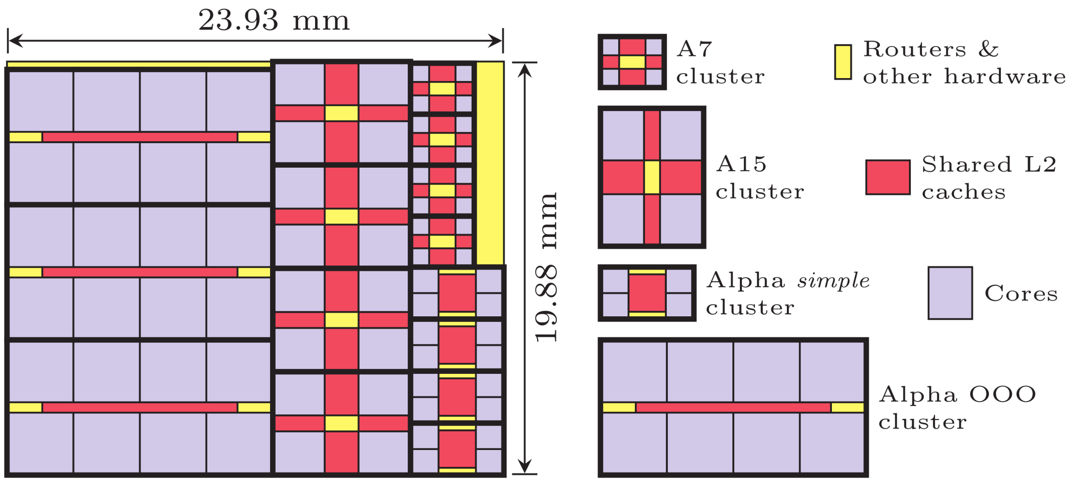


Figure 2.5 – Heterogeneous many-core setup used to evaluate seBoost proposal [Pagani et al., 2015].

This work uses a heterogeneous architecture with 72 cores divided into 24 high-performance Alpha cores, with out-of-order (OOO) execution, 16 simple Alpha cores, 16 ARM A15, with OOO execution support, and 16 ARM A7 as an energy-efficient core. The architecture is evaluated using GEM5, McPAT, HotSpot [Huang et al., 2006] and measures of traces on a commercial platform that uses ARM's big.LITTLE architecture. The proposed strategy is compared with Intel's Turbo Boost technology and with a simple boosting method that throttles down the non-boosted cores to the lowest frequencies.

Results show that if the performance boosting requirement is feasible, given an initial temperature vector, the proposed technique and the simple boosting method can satisfy the requirements 100% of the time, but with a simple boosting technique the non-boosted application will suffer from unnecessary losses of performance. Using Turbo Boost, the performance of the non-boosted cores achieved higher average performance, but in most cases, Turbo Boost failed to satisfy the runtime requirements during the entire boosting interval for the boosted cores. As Turbo Boost is not aware of the performance requirement of the applications, the boosting mechanism can set VF levels higher than necessary, causing the critical temperature to be reached before the end of the boosting interval, triggering the control mechanism that will degrade the performance.

## 2.4 Shi Sha et al.

This work, [Sha et al., 2018], proposes a frequency oscillation-based technique to maximize the throughput performance of multi-core platforms under a maximum allowed temperature constraint. The proposed technique is based in two concepts: *step-up schedule* and *m-Oscillating schedule*. The proposed technique focuses on periodic schedules that can deliver steady and sustainable performance.

The paper discusses the importance of this technique by assuming that each processor features discrete running modes. Previous works focused on solving the throughput maximization problem by assuming that the speed of each core can be continuously and instantaneously varied. Based on previous work, a simple method to maintain the peak temperature constraint is to round down the speed to the available discrete one, named in the paper as lower neighboring method (LNS). However, the results of this technique might be overly pessimistic when the available speed levels are limited. To improve this method, the work proposes an algorithm that searches all speed combinations to find one that can maximize the throughput without exceeding the temperature threshold, called exhaustive search (EXS). The main limitation of LNS and EXS is that each core can only execute one single speed, and the slack in temperature cannot be filled by raising the speed of any core due to the possible violation of the maximum allowed temperature.

Based on the limitations of LNS and EXS, this work aims to maximize the throughput of a multi-core platform, where the processors have discrete frequencies available, using a periodic schedule of two or more frequencies to improve performance. To define this schedule, the author first defines the step-up schedule, which is basically a periodic schedule with multiple state intervals where the processing speed (VF levels) increases at each interval. Based on a step-up schedule the authors define an m-Oscilating schedule, which is a method to determine the length of the period for the periodic schedule.

The m-Oscilating schedule is derived from a step-up schedule by scaling down each interval length by *m*. The authors prove that the smaller the period is (higher *m*), the higher the throughput can achieve. Considering different values of *m*, the total dynamic energy consumption and the average temperature remain constant, but the peak temperature tends to reduce with higher values of *m*. The main drawback of using higher values of *m* is the transition overhead to change VF settings on a processor, so the authors take this overhead in consideration when defining the schedule.

This work uses its own temperature estimation model to define the step-up schedule, ensuring that the temperature will not exceed the critical temperature. The algorithm builds possible schedules and determines the highest temperatures to verify if the peak temperature constraint is guaranteed. The experimental results use different configurations on the number of cores, up to 9, and possible voltage levels, up to 5. The proposed technique is simulated on hypothetical configurations of Alpha 21264 cores using power parameters from the McPAT simulator and temperatures collected from HotSpot. Results show an average improvement of 11 percent in performance when comparing the proposed technique with the EXS, which only allows using a single speed for each core.

## 2.5 Mengquan Li et al.

Mengquan Li et al. [Li et al., 2015, Li et al., 2018] propose a simplified thermal model to predict steady temperatures in a many-core system assuming a task-to-core mapping is given, and tasks' power consumption is known. Integrated with the thermal model, this work presents a mixed integer linear programming (MILP) model to obtain the optimal task assignment with the minimum chip peak temperature. If the optimal task-to-core assignment is still beyond the safe temperature limit, a temperature-constrained task selection (TCTS) algorithm is also proposed to maximize performance within safe temperature limit.

The thermal model proposed is a simplification of the HotSpot model. While HotSpot models the thermal conduction considering four layers, the proposed model uses only two layers, one for the circuit itself, which consumes power and acts as a heat source, and one passive layer that acts as a heatsink element. The temperature of each element is expressed as a function of its power consumption, ambient temperature, and temperature of neighboring thermal elements (heatsink element and neighbor cores), while the temperature of each element in HotSpot is calculated based on the power consumption of all active elements of the circuit, the temperature of all thermal elements and the ambient temperature. To overcome the loss in precision generated by the simplified model, the work considers two scaling factors, one for the ambient temperature and one for the vertical dissipation, which need to be calibrated by repeated executions of HotSpot with random power samples to minimize the error induced by the model. The result is a fast model that can predict steady state temperatures with the precision required to make right task mapping decisions.

The mapping algorithm takes as inputs the computation demand of tasks to be mapped, the voltage and frequency levels available on the system, the power consumption of the cores running at each frequency and the thermal model of the system. The algorithm uses the proposed thermal model to minimize the peak temperature of the system, searching the best task-to-core assignment by solving the MILP formulation.

It is important to note that the actual power consumption of a processor depends not only on its voltage and frequency level but also on the characteristics of the task that is executed on the core. However, this work considers that each task has a preassigned running frequency that is related to its performance requirement, and that the tasks have a constant power consumption that is used by the algorithm to estimate the steady temperatures of the system.

Finally, this work proposes the TCTS algorithm to choose the best set of tasks that can be executed in the system without violating the safe temperature when even the optimal mapping found by the MILP solver has its peak temperature above the safe constraint. This method is based on a greedy selection that discards tasks with higher computing demand first until a key point is reached, when all remaining tasks can be executed within thermal constraint. After this point, the algorithm tries to discard tasks with lower performance demand to find the set of tasks that will maximize the system throughput within the safe temperature limit. The paper presents a proof that this greedy algorithm also finds the optimal solution to define which tasks must be delayed to use the system at full capacity.

The results presented in this paper focus on showing the accuracy of the thermal model and in comparing the MILP based mapping with random and continuous mapping, measuring the peak temperature achieved by each mapping for each computation demand. The proposed thermal model results are compared with results obtained from HotSpot [Huang et al., 2006] simulations, showing a maximum error of 0.31 °C in temperature prediction. The mapping results consider a system with 16 Alpha 21364 cores organized in a 4x4 grid using McPAT [Li et al., 2009] to estimate the power consumption of tasks. Results show an average temperature reduction of 2.98 °C when comparing the proposed technique with random mapping, and 4.59 °C when comparing with continuous mapping, enabling more tasks to be executed within the safe temperature limit.

## 2.6 Castilhos et al.

Guilherme Castilhos et al. [Castilhos et al., 2016] propose a lightweight software-based runtime temperature model for many-core systems. The proposed model simplifies the HotSpot model to enable its execution in an embedded processor, targeting a small performance overhead. Furthermore, it allows dynamic temperature management in many-core systems at runtime, focusing on execution time, accepting some loss of precision. The model was implemented and tested in an NoC-based system with a cycle-accurate RTL description with integrated power monitoring [Martins et al., 2014].

The thermal characterization of the target platform is extracted from the HotSpot RC model. After the characterization, this work defines the temperature calibration process as a series of assumptions made to formulate the simplified model. The first step of the temperature calibration is observing the heat flow when applying the maximum power in a PE. Figure 2.6 show the observed effect of temperature over time in a PE and its neighbors.

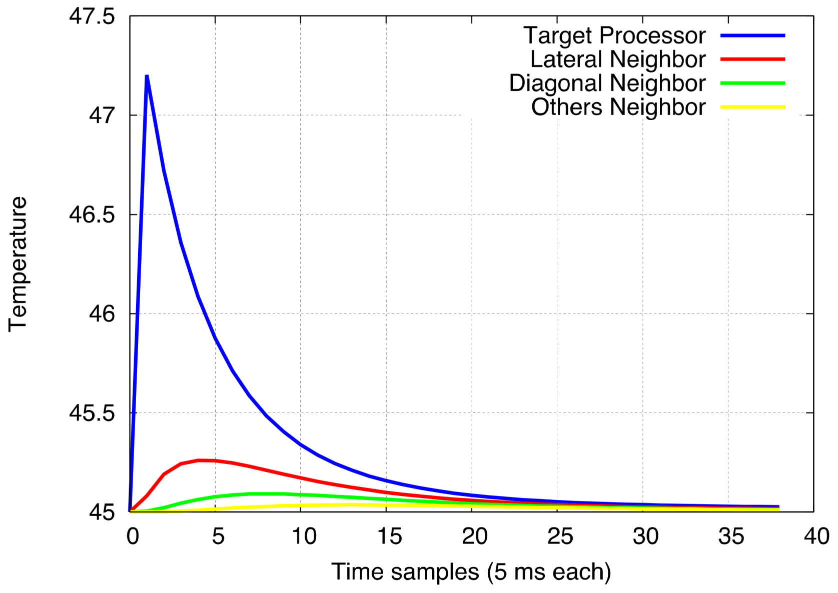


Figure 2.6 – Effect of temperature (°C) over the time in a PE and its neighbors [Castilhos et al., 2016].

Based on the observation of the heat flow, the following assumptions were made to create a model with reduced complexity:

- The thermal influence of a processor affects only its direct neighbors, lateral and diagonal;
- The transient effect of the temperature, after an application of power, is limited to 100 ms, corresponding to 20 sampling windows of 5 ms on Figure 2.6;
- The power values of the processors was discretized in intervals, assigning a temperature behavior for each power interval;
- Only integers are used instead of floating-point numbers.

The transient temperature behavior for each application of power is discretized and stored in a LUT for the target PE and the direct neighbors. The temperature of a PE is estimated by storing all the power values consumed during a transient effect window (20 samples), and querying the effect of each sample in the thermal behavior LUTs. The final estimated temperature for each PE is the sum of the transient effect of the target processor, regarding its power consumption, and the effect of all its neighbors, regarding their power consumption.

The result was a simple model with a moderate memory requirement that can run fast enough, allowing runtime temperature monitoring in a many-core system. The time required to estimate the temperatures of a system with 36 PEs is 0.35 ms at 100MHz. The accuracy of the proposed model was compared with HotSpot, and the observed average error was 3.52%, while the maximum error can reach up to 10%.

## 2.7 Martins et al.

André Martins et al. [Martins et al., 2019a] proposes a resource management strategy targeting large many-core systems running dynamic workloads under a restricted power budget named Multi-Objective Resource Management (MORM). The proposed method adopts a hierarchical approach. The system is divided into clusters that can execute applications in different modes to address conflicting resources, like power and performance, concomitantly. MORM adopts a proactive approach to map tasks while keeping the system running below the power budget. The method requires prior knowledge of tasks' profiles and constant monitoring of system state. Figure 2.7 presents an overview of the MORM hierarchical monitoring and actuation strategy. Gray and white boxes correspond to decision algorithms and virtual sensors, respectively. The monitoring mechanism takes advantage of the hierarchical organization of the system, where PEs send their monitoring data to its cluster manager (CM), and the CM collects data from its cluster and sends the cluster power to the global manager (GM).

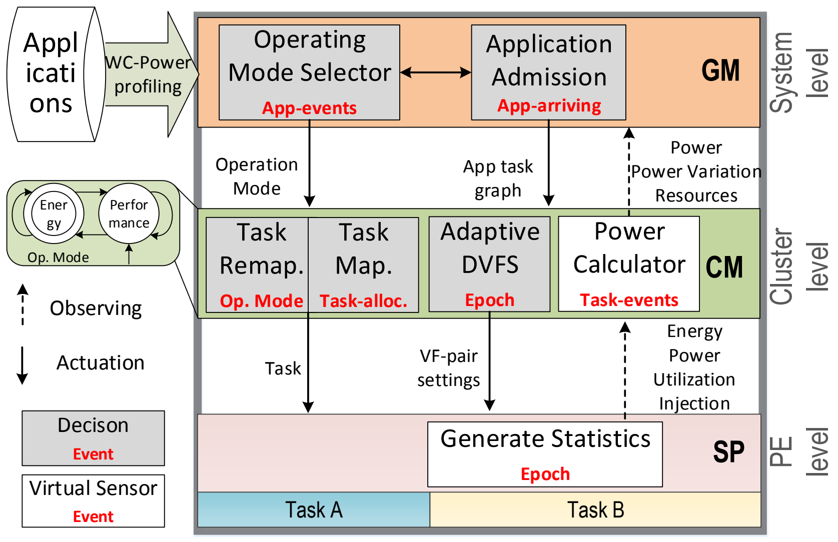


Figure 2.7 – MORM overview [Martins et al., 2019a].

The actuation is based on the operation modes allowed for clusters. MORM defines two operation modes: (*i*) *performance mode* - CM optimizes resources to minimize execution time of the running applications; (*ii*) *energy mode* - CM optimizes the resources to improve the energy efficiency. MORM allows a new application to execute if the application will not cause the system to exceed the power budget, and the system has available resources. If these conditions are not met, the mapping algorithm can modify the operation mode of the clusters to find room for the incoming application.

At the system level, the GM is responsible for deciding the operation mode of each cluster, and the cluster must execute all its applications in this mode. However, at the cluster level, the CM may apply low VF settings for tasks with low processor utilization or tasks with high network injection rates, regardless of the operation mode. *Performance mode* admits only one task per PE, which optimizes the application parallelism, and most of the tasks will be running at high VF settings to optimize the execution time, while *Energy mode* admits that communicating tasks may be mapped in the same processor and that most of the tasks will be running at the most energy efficient VF settings.

This strategy was evaluated using a clock-accurate RTL SystemC model of the reference many-core system and compared with another state-of-art technique to control the system power budget [Rahmani et al., 2015], named PF-only. Results show that in either dynamic and high workload scenarios, MORM can save energy while having better performance than the PF-only approach. The improvements obtained by MORM were mainly due to the hierarchical monitoring, the acting approach and the multitask mapping support.

## 2.8 Anup Das et al.

Das et al. [Das et al., 2016] propose a reinforcement-learning-based runtime manager (RTM) for thermal management in embedded systems, simultaneously addressing energy and temperature, considering the three aspects of temperature optimization: peak temperature average temperature, and thermal cycling. However, the problem of finding task-to-core mappings increases exponentially with the number of threads and the number of cores. To reduce the complexity of this problem, the authors propose a hierarchical management with two stages. The first stage controls the thread allocation to control thermal cycling, and the second stage controls a chip-wide DVFS to control the temperature and energy consumption.

The main focus of this work is to provide improvements in the lifetime reliability of the circuit by reducing the damage caused by thermal cycles. The Authors argue that the average and peak temperature plays an important role in lifetime reliability, and numerous works in the literature address those aspects. However, the emerging lifetime concern with scaled transistor geometry is thermal cycling, which is defined as wear-out caused by thermal stress due to a mismatched coefficient of thermal expansion of the adjacent material layers. The lifetime reliability is measured in terms of the mean time to failure (MTTF), and the thermal-cycling related MTTF is computed by calculating thermal cycles using Downing's simple rainflow counting algorithm and the number of thermal cycles to failure using Coffin-Manson's rule. The article presents details of the equations used to measure MTTF caused by thermal cycles.

Experiments use an NVIDIA Tegra K1 SoC with quad-core ARM Cortex-A15 running Linux and supporting chip-wide DVFS. A set of multithreaded benchmarks are considered. Each application is transformed to a periodic structure, where each iteration is accompanied by a deadline, which serves as the performance requirement. The proposed RTM is compared to native Linux governors and an energy-only approach. Results show Linux Governors ignore the latency required by periodic applications leading to underperformance or overperformance. The proposed RTM reduces the average temperature by 5.8°C, the peak temperature by 9°C, and thermal-cycling related damage by twice compared to the energy-only approach. In comparison to the Linux default governor, RTM reduces the average temperature by 12.5°C, the peak temperature by 11°C, and thermal-cycling related damage by 2.4x.

## 2.9 Mohammed et al.

Mohammed et al. [Mohammed et al., 2020] proposed a DTM technique named dynamic thermal-aware performance optimization of dark silicon many-core systems (DTaPO). The proposed technique combines task migration and dynamic voltage and frequency scaling (DVFS) to optimize the performance of a many-core system while keeping the temperature in a safe operating limit.

Figure 2.8 shows an overview of the proposed DTaPO technique. The active and dark cores are arranged in the form of a chessboard pattern for better heat dissipation. Tasks are mapped to the actives cores, surrounded by dark cores, and migrated in a cluster-based scheme. In this work, a cluster is arranged as the cores that share the last level cache (LLC), so when a task is migrated within the same cluster, the number of memory access due to cache misses reduces since the data is available to cores that share the LLC.

DTaPO monitors the number of active cores, the number of dark cores, the current setting of frequency, and the transient temperature. The decision algorithm may move a task from an active core to a dark core if the transient temperature exceeds a predefined threshold and the temperature of the dark cores is below the threshold temperature by a safe margin. If the temperature of the active cores is above the threshold and the dark cores are just below the threshold temperature, the algorithm may choose to lower the DVFS settings to keep the system in a safe operating temperature.

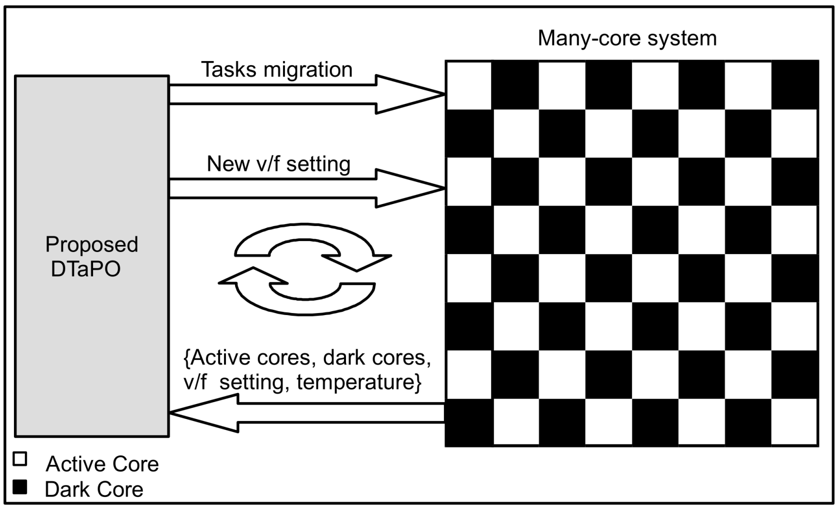


Figure 2.8 – DTaPO overview [Mohammed et al., 2020].

The experimental setup uses LifeSim simulation tool [Rohith et al., 2018]. LifeSim is a tool that contains sniper [Carlson et al., 2011], McPAT [Li et al., 2009] and HotSpot [Huang et al., 2006]. Sniper is a parallel x86-64 many-core simulator. It offers simulation with an average performance error of 25% compared to real hardware. Results are extracted using a 64-core NoC-based many-core system with shared memory divided into 8 clusters containing eight cores with 8MB of L3 (LLC) cache each. The simulations were executed using several threshold temperatures, ranging from 60 to 85℃. Results show that the technique achieved an execution time penalty 18% lower than using only DVFS for all thermal thresholds. Also, when comparing with the optimal sprinting technique (OSP) [Wang et al., 2021], DTaPO can reduce the execution time by up to 80% and reduce the peak temperature up to 13.6℃.

## 2.10 Rathore et al.

Rathore et al. [Rathore et al., 2019] proposes the LifeGuard approach, a performance-centric reinforcement learning-based task mapping that evaluates the impact of applications on aging for improving system health. The proposed strategy assumes that the cores of a many-core system have different safe operating frequencies, process variations (PV), and aging, prioritizing cores that are mapping candidates for a larger number of applications, based on applications performance requirements and the cores safe operating frequency.

This work considers negative-bias temperature instability (NBTI) as the major aging mechanism since it affects mainly the threshold voltages. The authors define core health as its frequency, as it is a function of both PV and aging due to NBTI, which may reduce the operating frequency. The main objective of the reinforcement learning technique is to optimize the weighted sum of core health while meeting the applications' performance requirements. The only tool used by the algorithm to meet its objective is task-to-core mapping.

Figure 2.9 shows the overview of the LifeGuard approach. The applications enter the system randomly at fixed scheduling ticks. If an application enters the system for the first time, the application is mapped in a contiguous region for characterization. The characterization process returns the average power (P) and temperature (T) of the executed application. During the learning phase, the algorithm maps the tasks into bins according to the application's performance requirements and measures each mapping's score. After the learning phase, the algorithm chooses the mappings that generated the best scores to map the applications.

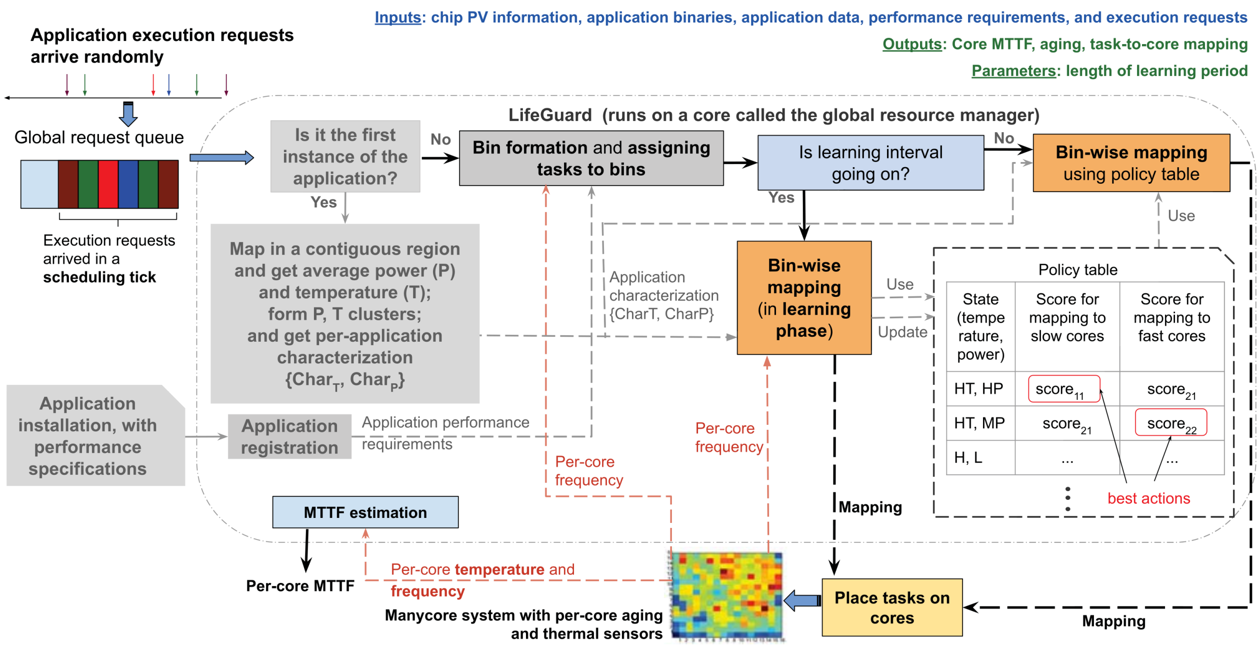

Inputs: chip PV information, application binaries, application data, performance requirements, and execution requests
Outputs: Core MTTF, aging, task-to-core mapping
Parameters: length of learning period
Application execution requests arrive randomly
Global request queue
Execution requests arrived in a scheduling tick
LifeGuard (runs on a core called the global resource manager)
Is it the first instance of the application?
No
Yes
Bin formation and assigning tasks to bins
Is learning interval going on?
No
Yes
Bin-wise mapping using policy table
Use
Map in a contiguous region and get average power (P) and temperature (T); form P, T clusters; and get per-application characterization {Char_T, Char_P}
Application characterization {CharT, CharP}
Bin-wise mapping (in learning phase)
Use
Update
Policy table
State (temperature, power)
Score for mapping to slow cores
Score for mapping to fast cores
HT, HP
score11
score21
HT, MP
score21
score22
H, L
...
...
best actions
Application installation, with performance specifications
Application registration
Application performance requirements
Per-core frequency
Per-core frequency
Mapping
MTTF estimation
Per-core MTTF
Per-core temperature and frequency
Manycore system with per-core aging and thermal sensors
Place tasks on cores
Mapping

Figure 2.9 – LifeGuard overview [Rathore et al., 2019].

The experimental setup used in this work uses snipersim [Carlson et al., 2011] as the ISA simulation tool, McPAT [Li et al., 2009] and HotSpot [Huang et al., 2006] for thermal simulation. The setup consists of a 256-core system in 22nm technology. The workload contains nine applications from the SPLASH-2 benchmark. Results show that the LifeGuard approach obtained a higher value of the maximum change in threshold voltage. However, it achieved an improved frequency/health for 191 cores compared with their previous work, HiMap [Rathore et al., 2018].

## 2.11 Haghbayan et al.

Haghbayan et al. [Haghbayan et al., 2020] proposed a thermal-cycling-aware dynamic reliability management (DRM) approach for shared memory many-core system running multi-threaded applications. The novelty of this approach is the incorporation of thermal-cycling awareness, while past approaches focused only on respecting the power budget to avoid aging effects.

Figure 2.10 presents a general overview of the resource manager proposed in this work. The manager contains 4 main components: (*i*) reliability analysis unit; (*ii*) reliability-aware mapping unit; (*iii*) reliability-aware thread scheduling unit; (*iv*) reliability-aware DPM (Dynamic Power Management). The reliability analysis is measured the same way as Das et al. [Das et al., 2016], by using the rainflow counting algorithm. The reliability analysis unit can detect cores aged due to thermal cycling and use power gating to slow down the aging trend. The DPM unit uses a PID control system to keep the system under the TDP by dynamically setting the DVFS of individual cores. The reliability-aware mapping unit uses the metrics defined in Das et al. [Das et al., 2013] for memory affinity, including aging and a temperature metric to improve system lifetime. Finally, the reliability-aware thread scheduling unit schedules the threads in an allocated region of the system, aiming to reduce short-term temperature variations.

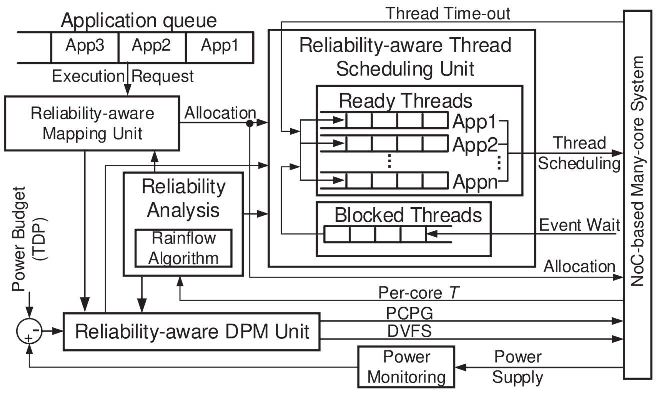


Figure 2.10 – Thermal-cycling-aware resource manager overview [Haghbayan et al., 2020].

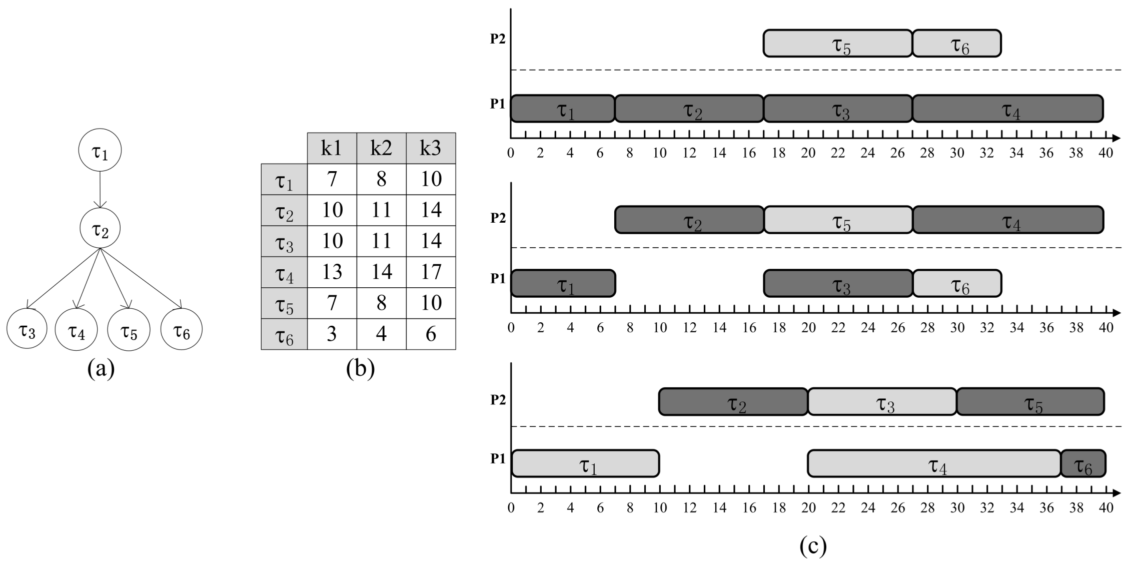
The experimental setup adopts an 8x8 NoC-based many-core system, using a simulation environment provided by Das et al. [Das et al., 2013], with 60 processing cores and four memory controllers in the corners of the mesh network. Power and temperature modeling are handled by McPAT [Li et al., 2009] and HotSpot [Huang et al., 2006], respectively. Experimental results show that this approach obtained a 39% longer lifetime when compared with a similar reliability-aware-mapping defined by Das et al. [Das et al., 2013].

## 2.12 Weichen Liu et al.

Weichen Liu et al. [Liu et al., 2019] propose a mixed linear programming (MILP) model to determine the mapping and scheduling for real-time applications in MPSoC platforms. The objective of the approach is to minimize energy consumption and satisfy lifetime reliability and a temperature limit constraint. Authors argue that chip temperature and system lifetime reliability should both be considered explicitly to design energy-efficient application mapping and scheduling.

This work focus on two critical failure factors: (*i*) electromigration (EM); (*ii*) time dependent dielectric breakdown (TDDB). The lifetime reliability model used is based on several works from integrated circuits aging literature but mostly follows the model proposed by Srinivasan et al. [Srinivasan et al., 2004]. The work presents a custom thermal model to estimate temperatures, used as input in the MILP model. The thermal model is based on HotSpot [Huang et al., 2006] and uses two scaling factors to simplify the HotSpot computation. The scaling factors are empirically trained with respect to HotSpot results at design time.

Figure 2.11 shows an overview of the scheduling and mapping problem that the MILP formulation solves. This example consists of an application with six tasks that runs in a two core system. The MILP solver explores all the application mapping and scheduling alternatives to find a globally optimal solution with minimal energy consumption. The first scheduling in Figure 2.11(c) can complete the application in the required deadline of 40 time units. However, it is the one that uses more energy to be completed and does not satisfy the lifetime requirement. On the other hand, the last scheduling also meets all the performance requirements while using less energy to be completed, fulfilling the expected system lifetime.



| | k1 | k2 | k3 |
|---|---|---|---|
| $\tau_1$ | 7 | 8 | 10 |
| $\tau_2$ | 10 | 11 | 14 |
| $\tau_3$ | 10 | 11 | 14 |
| $\tau_4$ | 13 | 14 | 17 |
| $\tau_5$ | 7 | 8 | 10 |
| $\tau_6$ | 3 | 4 | 6 |

Figure 2.11 – (a) Task graph. (b) Execution time. (c) Possible schedules for the example task graph [Liu et al., 2019].

The experimental setup consists of multicore platforms sized from 4 (2x2) up to 16 (4x4) Alpha 21364 cores, using area and power models from McPat [Li et al., 2009]. The main drawback of this work is that the MILP computation must be executed at design time to obtain the optimal application mapping and scheduling. In addition, this procedure requires previous knowledge of the workload that the system will execute, and the resulting strategy may not be adapted for dynamic workloads.

## 2.13 Related Work Analysis

Most of the reviewed works present strategies to reduce the power density issues on many-core systems. Four of the reviewed works [Yang et al., 2017b, Yang et al., 2017a, Liu et al., 2018, Mohammed et al., 2020] present architectures based on dark silicon patterning, which is the mapping of tasks in a chessboard pattern to increase heat transfer between dark and active cores, together with a communication strategy that can reduce the impact of decentralized mapping on network latency. These works rely on complex temperature predictions executed by MatEx to take online mapping decisions, which may reduce the performance of the scheduler by up to dozens of milliseconds. The patterning approach leads to system underutilization, and results (Section 5.4.5) demonstrate that proposals made in this Thesis reduce the peak average temperature with regard to the patterning approach.

Pagani et al. proposed a power budget for many-cores, based on the safe temperature constraint that the system can hold to ensure reliability and lifetime [Pagani et al., 2017], called Thermal Safe Power (TSP). A proactive approach using TSP as the thermal budget can estimate the amount of power that cores can consume according to each core mapping scenario. However, PEs might not have all available speed steps to consume the exact amount of power estimated by this algorithm, and tasks behavior is not considered in this mapping strategy. The worst-case estimated by the TSP approach might be very conservative, and the algorithm is very time demanding to be executed for every task mapping decision.

Boosting techniques are also present in the works related to thermal management [Pagani et al., 2015], and the limitation of available speed steps to control the power consumption of tasks, which is a problem for TSP, is mitigated in [Sha et al., 2016] proposal. Our work does not use boosting techniques since our method already considers the highest VF combination to increase performance while maintaining a uniform thermal distribution. We adopt thermal-aware mapping, task migration, and DVFS actuation to avoid hotspots and keep tasks running at the best performance while keeping the system in the safe temperature range.

Some of the related works propose dynamic reliability management (DRM) [Das et al., 2016, Rathore et al., 2019, Haghbayan et al., 2020, Liu et al., 2019]. In DRM approaches, the main metric used to manage the system is the expected lifetime, extracted from a theoretical model. While some of the DRM reviewed works consider thermal cycling as a metric to be managed [Das et al., 2016, Haghbayan et al., 2020], others focus on electromigration and time dependent dielectric breakdown [Liu et al., 2019], or negative bias temperature instability [Rathore et al., 2019]. We also observed that two works proposed learning techniques to manage the system [Das et al., 2016, Rathore et al., 2019], while [Liu et al., 2019] proposed an offline MILP approach and [Haghbayan et al., 2020] proposed a more traditional heuristic to deal with thermal cycling.

Other related works do not focus on DRM or DTM proposals [Castilhos et al., 2016, Martins et al., 2019a]. Martins et al. [Martins et al., 2019a] proposed a lightweight proactive mapping strategy to keep a many-core system under a power budget. While power consumption and temperature are related, using only energy as a constraint may lead to underutilization of the system and still create hotspots. Castilhos et al. [Castilhos et al., 2016] proposed a lightweight temperature estimation model, also used to guide applications' mapping.

Table 2.1 summarizes the reviewed works according to the classification chosen for power and thermal management. The first column contains the Author and reference. The second column shows the design goals of each work. The third column presents the architecture (homogeneous or heterogeneous) and the core types and counting given by the number of processor elements or the NoC size (if adopted). The fourth column lists the management techniques used to reach the goals, focusing on classifying if the technique is proactive or reactive according to the actuation and listing the technique's name according to the Authors definitions. Finally, the fifth column presents the experimental setup used by the Authors to model the architecture for the experiments.

Most works adopt high-level architectural models, using tools like GEM5, to emulate the instruction set and the communication structure of the target architecture, and McPAT, to evaluate the power consumption of the processors. While these high-level models are extensively used in computer architecture research, they make assumptions that may reduce their applicability in power management strategies. These high-level models do not consider that each task may present different power consumption behavior, even when running at the same voltage and frequency, leading to a significant impact on power prediction error [Xi et al., 2015]. Another issue related to these models is the accuracy of the power consumption related to the processors and NoC. Our work models the many-core system at two different levels: clock cycle-accurate level and instruction set level. The first one enables accurate power and performance evaluation, and the second one allows longer simulations to observe the effect of the proposed dynamic thermal management heuristics.

We also classified related works in homogeneous and heterogeneous architectures. Heterogeneous architectures in the reviewed works contain cores with different energy efficiencies but the same instruction set (ISA). We agree that using cores with better energy efficiency for low demanding tasks is a path to save energy. The many-core architecture is orthogonal to our proposal because our concern is to propose a dynamic thermal management strategy. Therefore, the proposed methods may be applied to such heterogeneous architectures, requiring the PE characterization.

Finally, we noticed that many works adopt proactive approaches for temperature management, relying on peak temperature prediction since this is a feasible task to be exe-

Table 2.1 – State-of-art summary.

| Author | Design goals | Architecture | Management strategy | Modeling |
|---|---|---|---|---|
| Lei Yang 1 [Yang et al., 2017b] | Peak Temperature Communication Latency | Heterogeneous Hierarchical FoToNoC 8x8 Alpha 21264 | Proactive Patterning | McPAT(Power) GEM5(architecture) MatEx(Temperature) |
| Lei Yang 2 [Yang et al., 2017a] | Peak Temperature Communication Latency | Heterogeneous Hierarchical Quad-Core Cluster 8x8 Alpha 21264 | Proactive/Reactive Patterning | McPAT(Power) GEM5(architecture) MatEx(Temperature) |
| Weichen Liu 1 [Liu et al., 2018] | Peak Temperature Communication Latency Communication Contention | Homogeneous Reconfigurable NoC up to 8x8 Alpha 21264 | Proactive Patterning | McPAT(Power) GEM5(architecture) MatEx(Temperature) |
| Santiago Pagani 1 [Pagani et al., 2014] [Pagani et al., 2017] | Thermal Safe Power Power Bugdet | Homogeneous 8x8 Alpha 21264 | Proactive TSP | Custom Thermal Model McPAT(Power) GEM5(Architecture) HotSpot(Evaluation) |
| Santiago Pagani 2 [Pagani et al., 2015] | Peak Temperature Performance | Heterogeneous 72 cores Alpha 21264 ARM A7, A15 | Proactive Boosting | Custom Thermal Model McPAT(Power) GEM5(Architecture) HotSpot(Evaluation) |
| Shi Sha [Sha et al., 2018] | Temperature Constraint Performance | Homogeneous 3x3 Alpha 21264 | Proactive m-Oscilating | Custom Thermal Model McPAT(Power) HotSpot(Evaluation) |
| Mengquan Li [Li et al., 2015] [Li et al., 2018] | Peak Temperature | Homogeneous 4x4 Alpha 21364 | Proactive MILP | Custom Thermal Model McPAT(Power) HotSpot(Evaluation) |
| Guilherme Castilhos [Castilhos et al., 2016] | Lightweight Software Temperature Estimation | Homogeneous 6x6 MIPS | Proactive | Custom Thermal Model Cycle-accurate simulation Low level power analysis HotSpot(Evaluation) |
| André Martins [Martins et al., 2019a] | Power Budget Performance | Homogeneous 6x6 MIPS | Proactive MORM | Cycle-accurate simulation Low level power analysis |
| Anup Das [Das et al., 2016] | Three thermal aspects Lifetime Reliability | Homogeneous quad-core | Proactive Q-learning | Real system nVidia Tegra(4xARM A15) |
| Mohammed Sultan M. [Mohammed et al., 2020] | Temperature Constraint | Homogeneous 8x8 x86 | Proactive Patterning DTaPO | Lifesim McPAT(Power) HotSpot(Temperature) |
| Vijeta Rathore [Rathore et al., 2019] | Lifetime Reliability NBTI | Homogeneous 8x8 Alpha 21364 | Reinforcement Learning LifeGuard | Snipersim(Architecture) McPAT(Power) HotSpot(Temperature) |
| Mohammad Haghbayan [Haghbayan et al., 2020] | Lifetime Reliability Thermal-Cycling | Homogeneous 8x8 Niagara 2 | Proactive | Intel PIN(Architecture) McPAT(Power) HotSpot(Temperature) |
| Weichen Liu 2 [Liu et al., 2018] | Lifetime Reliability EM/TDDB | Homogeneous 2x2/3x3/4x4 Alpha 21364 | Proactive MILP | Custom Thermal Model McPAT(Power) |
| **Proposal** | Temperature Constraint Power Budget Performance Reliability Analysis | Homogeneous 6x6/8x8 MIPS | Proactive/Reactive Lightweight Thermal Aware Mapping | Cycle-accurate simulation Low level power analysis Accelerator (temperature) MatEx (Reference Model) Instruction-level (OVP) |

cuted at runtime in software. However, to predict temperatures in a real scenario, the system must be aware of the temperature of the cores and the different layers of the cooling solution [Sharifi et al., 2013], requiring constant monitoring of the system thermal status. The monitoring strategy plays an essential role in system management, and it is often omitted from the related works. Proactive approaches also require task characterization to predict power consumption and temperatures accurately, and this might not be feasible for heavy dynamic workloads.

### 2.13.1 Final Remarks

After reviewing the state-of-the-art for DTMs and DRMs we found the following gaps in the literature:

- Most works focus on high-level architectural simulations with imprecise power and temperature estimations.
- None of the reviewed works presents a feasible runtime temperature monitoring, usually relying on thermal models that only run as a design time estimation.
- Some works propose a patterning approach, which may limit the system utilization, while others rely on complex heuristics designed to be executed at design time, requiring previous knowledge of the executed workload. None of the reviewed works present a simple heuristic that can execute at runtime, allowing all cores to be active at the same time.
- Reviewed works focus on thermal management or reliability management. None of the works propose thermal management with a lifetime reliability analysis, showing that the proposed thermal management also improves lifetime reliability.

Each of the following chapters of this Thesis presents our approach to fill the gaps mentioned above:

- Chapter 3 presents the architecture used to develop the thermal monitoring and the DTM solution proposed in this Thesis. The architecture has an RTL implementation with low-level power analysis and estimation.
- Chapter 4 presents the temperature monitoring strategy adopted to enable runtime thermal estimation, allowing the development of runtime DTM approaches.
- Chapter 5 presents our heuristics used to manage system temperature distribution. The proposed heuristics are modular and scalable, allowing its execution at runtime with minimal overhead.
- Chapter 6 presents an analysis of the proposed heuristics related to the obtained lifetime reliability improvements.

# 3. REFERENCE ARCHITECTURE AND BASIC CONCEPTS

The goal of this Chapter is to provide the reader with concepts used throughout this Thesis. The organization of this Chapter comprises five sections, distributed as follows:

- Section 3.1 details the reference architecture, which will receive the hardware and software mechanisms to manage temperature and lifetime.
- Section 3.2 details the application model adopted by the reference architecture, which corresponds to applications partitioned into tasks, communicating through message-passing.
- Section 3.3 briefly describes how the reference architecture is modeled: (*i*) RTL level, VHDL, used for characterization (area, power, performance); (*ii*) RTL level, System C, used to validate the methods herein proposed; (*iii*) instruction-level model, using OVP [Imperas, 2019], to enable the observation of the proposals through the simulation for long periods.
- Section 3.4 reviews the power characterization method. The power samples are the basis for temperature estimation.
- Section 3.5 details the hierarchical monitoring scheme necessary to reduce the traffic related to monitored data, and hence its disturbance on the applications' traffic.

These Sections do not correspond to the Thesis's original contributions but allow the construction of the framework of concepts and methods necessary for the development of the proposals presented in the subsequent Chapters.

## 3.1 Reference Architecture

The reference architecture used to design and validate the proposals presented in this Thesis is a NoC-based many-core system based in the Memphis framework [Ruaro et al., 2019].

Figure 3.1(a) presents the main components of the many-core system. The system contains two regions [Ruaro et al., 2019]: (*i*) a homogeneous set of processing elements (PEs) - General Purpose Processing Cores (*GPPC*) region; (*ii*) peripherals attached to the *GPPC* borders. Peripherals may be dedicated hardware to inject new applications into the *GPPC* (as the Application Injector in the Figure 3.1(a)), or hardware accelerators. Figure 3.1(b) presents the PE internal modules: a processor (CPU), a network interface (DMNI - Direct Memory access Network Interface [Ruaro et al., 2016]), local memory, and the NoC router (PS).

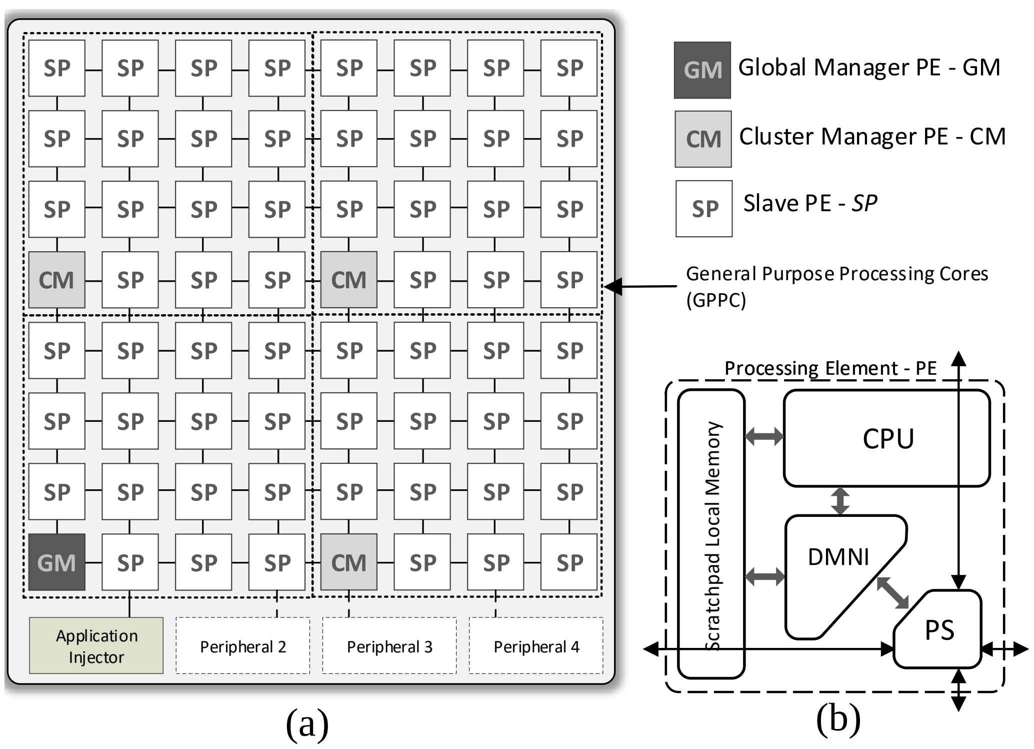


Figure 3.1 – NoC-based many-core system with peripherals (adapted from [Ruaro et al., 2019]).

## 3.1.1 Cluster-based management

The many-core management may adopt different approaches: (*i*) centralized; (*ii*) per-application management (PAM); (*iii*) cluster-based management (CBM). With one manager Processing Element (PE), centralized approaches are not suitable for large-scale systems since the manager PE easily becomes a bottleneck.

PAM [Tsoutsouras et al., 2018] assigns one manager PE per application. PAM also assumes a cluster manager PE for managing the resources of a given set of PEs (as available PEs for executing new applications, power consumption). PAM's drawback is the number of resources reserved for management and the decision heuristics' complexity due to the interaction between applications and cluster managers.

CBM [Al Faruque et al., 2010] divides the system into clusters, each with a manager PE and a set of slave PEs (the ones executing user's applications). CBM does not require a manager PE per application, being each manager responsible for the applications running on its cluster. According to our evaluations [Castilhos et al., 2013], a cluster size with 16-18 PEs is a good trade-off between total execution time and PEs dedicated to the management.

The reference architecture used in this work adopts a CBM approach. The system management adopts a hierarchical organization, where each cluster has its manager PE. All PEs have the same hardware, being the differentiation made in software at the operating system (OS) level. Each PE may assume the following roles:

- Slave PE – *SP*: execute applications' tasks.
- Manager PE – *MP*: manage the SPs of a given cluster, executing functions such as application admission, mapping, remapping, DVFS control. Manager PEs only execute management functions. Manager PEs may be local to a given cluster (CM) or execute global actions besides the cluster management (GM).

Such hierarchical CBM organization brings several advantages compared to other approaches concerning management [Rahmani et al., 2018], such as scalability by mitigating the impact on the applications' traffic, allowing the distribution of management traffic, and enabling hierarchical management, which is detailed in Section 3.5. The cost of using this organization is the overhead of having some PEs reserved for cluster management.

### 3.1.2 GPPC Connection with Peripherals

As shown in Figure 3.1(a), peripherals are connected to the boundaries of the GPPC. Standard mesh topologies do not enable to send/receive packets to/from the NoC borders. Thus, to allow communication between PEs and peripherals, it is necessary to add hardware and kernel (OS) support. At the kernel level, a dedicated communication API creates packets with a flag in the header flit notifying the NoC that the packet should go to a peripheral (IO packet). The target of an IO packet is the router address, not the peripheral itself. At the hardware level, when the IO packet reaches the target router, it goes to a border port and not to the local port.

## 3.2 Application Model

Applications are modeled as directed acyclic task graphs, $A = (T, E)$, where the vertex $t_i \in T$ is a task and the directed edge $e_{ij} \in E$ is the communication between tasks $t_i$ and $t_j$. The adopted communication model is message passing. Figure 3.2 presents two examples of applications modeled as task graphs. In the current Thesis, applications are described in C language and use MPI-like communication primitives.

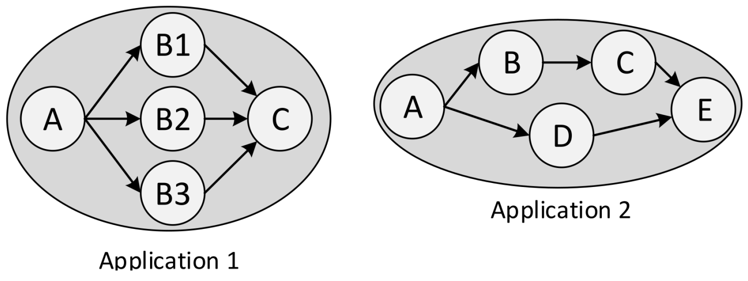


Figure 3.2 – Application graph model examples.

Figure 3.3 presents the flow to send and receive a packet between two different PEs. The *Send()* primitive generates a system call, *send_packet()*, that programs the DMNI to send the packet, copying the data from memory and transmitting it to the NoC. At the consumer side, the DMNI interrupts the processor when receiving a packet. The interruption handler calls the *read_packet()* procedure, which programs the DMNI to read the packet by copying it from NoC to memory. Once the packet is completely received, the kernel executes functions related to the contents of the packet. For example, if the packet has data to a user task *t*, the packet is written in the *t* memory space, the *Receive()* call is unlocked, and *t* is scheduled to execute.

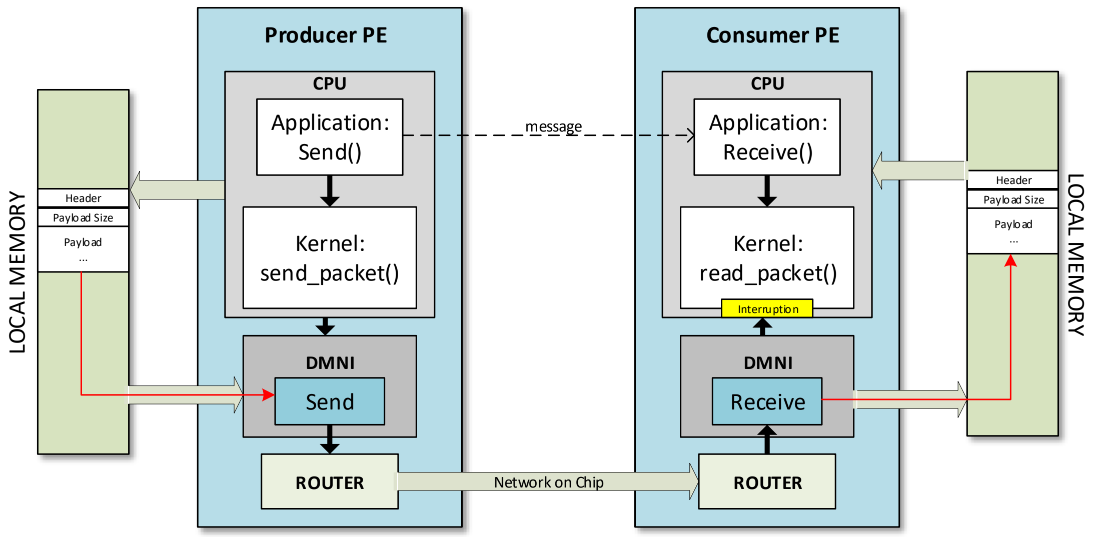


Figure 3.3 – Inter task communication interface (adapted from [Ruaro et al., 2019]).

## 3.3 Architecture Models

This Thesis adopts three models for the reference architecture presented in Section 3.1: (*i*) two clock cycle-accurate models; (*ii*) an instruction-level model. The clock cycle-accurate models have two equivalent descriptions: synthesizable VHDL and SystemC. The instruction-level model uses OVP [Imperas, 2019] as the simulation framework, with peripherals and other PE components, like network interface and routers, described in C.

### 3.3.1 Clock Cycle-accurate Models

The reference platform has two different register transfer level (RTL) descriptions modeling the platform in a clock cycle-accurate behavior: synthesizable VHDL and SystemC.

An RTL description defines a circuit in terms of memory elements (registers) and combinational logic. It can be interpreted by a synthesis tool to generate a gate-level description of the system. A gate-level description provides enough details to estimate the system's power consumption, enabling the characterization presented in Section 3.4. The power characterization allows us to estimate the temperature (Chapter 4), which provides the background to design and evaluate dynamic thermal management techniques (Chapter 5).

The VHDL description of the reference architecture is used to synthesize the system components and generate the power characterization. The SystemC description uses an instruction set simulator (ISS) for the CPU to speed up the simulation while representing the same timing behavior of the VHDL model, with the other components modeled in RTL level systemC. The power data collected from the VHDL synthesis is annotated in the systemC model to estimate the system's power consumption in a faster simulation scenario.

References "HeMPS-S: A homogeneous NoC-based MPSoCs Framework Prototyped in FPGAs" [Wächter et al., 2011]" and "Hardware and Software Infrastructure to Implement Many-core Systems in Modern FPGAs" [Bortolon and Moraes, 2017] are examples of works that successfully prototyped the reference platform in FPGA devices.

### 3.3.2 Instruction-level Model

OVP adopts a quantum-based simulation paradigm, which divides the time into discrete time steps, as depicted in Figure 3.4. The number of instructions that each PE executes in a step is called *quantum*. The quantum size is parameterizable, and its size affects the simulation performance and the synchronization between PEs. Each PE simulates sequentially by one quantum. Once every processor executes a quantum, the simulation advances to the next quantum. Summarizing, this paradigm simulates parallelism by alternating each processor's execution. Note that there is a trade-off between the quantum size and the communication synchronism. A large quantum delays the communication between PEs, and a small quantum increases the simulation time. Experimentally, we tuned the quantum to 250 instructions, a value that brought the behavior of this platform closer to the RTL behavior.

OVP does not support NoCs but provides an API to create peripherals that can be connected to processors. Thus, according to the RTL behavior, we modeled the router and a NI (Network Interface) as a peripheral. As OVP peripherals do not have their execution associated with the quantum size, they are triggered by callbacks executed atomically. The virtual platform received a peripheral called *iterator* to synchronize the communication between processors at the end of each quantum execution. The *iterator* evaluates all routers sequentially with flits to transmit, sending one flit to the next router or local port. This process stops when there are no more flits to transmit. After this condition, a new quantum starts.

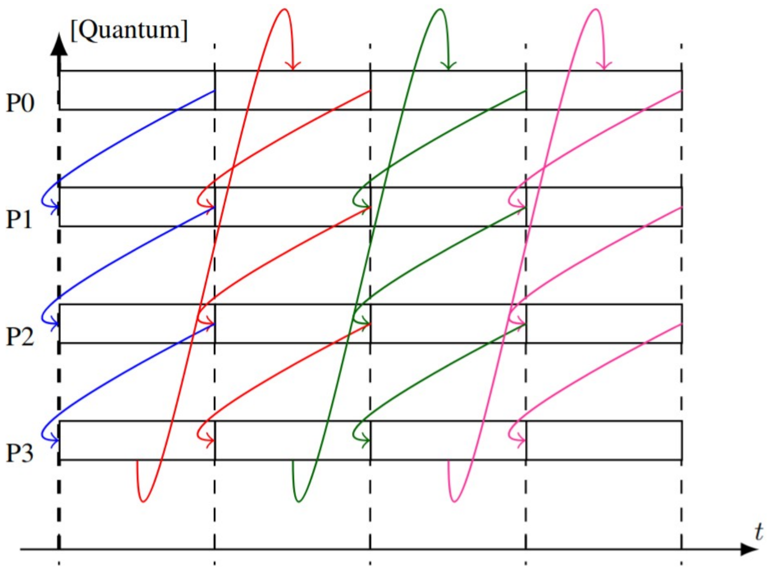


Figure 3.4 – Quantum-based simulation paradigm.

Details related to this instruction-level platform are available in [Lopes et al., 2021].

### 3.3.3 Clock Cycle-Accurate and Instruction-level Model Comparison

The clock cycle-accurate model simulation provides enough details to model the system's power consumption using synthesis tools and netlist simulations while providing a more accurate simulation than the instruction-level model. However, the SystemC RTL model enables the simulation of only a few milliseconds per hour. The OVP model allows more than 1 second of simulation per hour, providing longer simulations to evaluate the system's thermal behavior.

The clock cycle-accurate model's main advantage is the NoC simulation, which provides the actual latency and congestion of the system. Network timing cannot be modeled with accuracy in OVP since the simulation engine is based in quanta. Network simulation is the main source of error introduced in the OVP model.

## 3.4 Power Estimation

This Section presents a general method to characterize the energy and power parameters for the main PE modules: CPU, router and memory. The power due to the DMNI module is not considered because it is a small module compared to the processor, router, and memory. The method to estimate power and energy is generic because it is based on a calibration process to define the energy/power values. The characterization flow employs the synthesizable VHDL description of the reference platform.

### 3.4.1 Processor Characterization

The process of characterizing the processor for energy and power relies on calibration. Initially, the instruction set is divided into classes. The goal is to obtain measures of energy per instruction and power per instruction for all classes, as well as presenting these values in parcels of leakage and dynamic power. The processor power characterization comprises five steps [Martins et al., 2014]:

1. Group the instruction set into classes. For each instruction class, an assembly program is written with the class's instructions in such a way to use all processor registers. The goal of the assembly programs is to maximize the switching activity of the processor modules (e.g., ALU, registers) by generating a significant Hamming distance between results (Figure 3.5).

```
int main(){
   int i;
   for(i=0;i<600;i++)
   {
     asm volatile
        ("   addi $8,  $0,   0xffff\n"
         "\t addi $9,  $0,   0x0001\n"
         "\t lui  $1,  0xA5A5\n"
         "\t addi $10,$1,  0xA5A5\n"
         "\t addi $1,  0x5A5A\n"
         "\t addi $11,$1,  0x5A5A\n"
         "\t lui  $12,  0xF0F0\n"
         "\t addi $12,$1,  0xF0F0\n"
         "\t lui  $13,  0x0F0F\n"
         ....
         "\t addu $17,$26,$17\n"
         "\t sub $17,$26,$17\n"
         "\t subu $17,$26,$17\n"
         "\t addiu  $24,$25,0x0810\n"
       );
    }
}
```

Figure 3.5 – Assembly code snippet used for processor characterization [Martins, 2018].

2. RTL simulations for each assembly program are executed to count the number of executed instructions and the number of clock cycles to execute the programs. Table 3.1 shows the instruction classification and the data obtained from the RTL simulations. The generated code for branches and jumps presents a smaller percentage of instructions belonging to these classes due to NOP instructions' insertion. The effect of the NOP instructions is accounted to compute the energy for these two classes.

3. Logic synthesis of the processor for a given technology generates a gate level description and an SDF file (Standard Delay Format - annotated delay data and timing checks

Table 3.1 – RTL simulation results obtained from instruction classes (adapted from [Martins et al., 2014]).

| class | # of executed instructions | # of CLASS instructions | % of executed instructions |
|---|---:|---:|---:|
| arithmetic | 110,457 | 109,223 | 98.88% |
| logical | 108,657 | 108,001 | 99.40% |
| shift | 46,431 | 46,200 | 99.50% |
| move | 77,031 | 75,000 | 97.36% |
| load_store | 58,287 | 53,407 | 91.63% |
| nop | 25,471 | 25,392 | 99.69% |
| branches | 220,031 | 105,017 | 47.73% |
| jumps | 220,031 | 100,002 | 45.45% |

file obtained after logic synthesis). The timing constraints (the frequency of PE is supposed to run) are defined at this step. We did not considered a physical synthesis, however we used the PLE (Physical Layout Estimation) option during the logic synthesis. PLE is a physical aware synthesis that estimates the wire lengths and their capacitances to generate the timing and power estimation results.

4. Switching activity annotation from netlist simulation. The netlist obtained in step 3 is simulated using the same assembly programs of step 1. This simulation generates the switching activity at the gate level, in a VCD (Value Change Dump) or TCF (Toggle Count Format) file.

5. Power analysis of the switching activity files provides accurate and reliable measurement of dynamic and static power since the results come from a netlist synthesized for a given technology. Table 3.2 shows the power and energy obtained from the netlist simulation.

A set of codes with minimal switching activity was written to better evaluate the effect of the Hamming distance in the assembly codes. It was observed that the switching activity could induce up to 30% of average power concerning the results obtained in Table 3.2. Thus, assembly codes with maximum Hamming distance are employed because the characterization must be general (i.e., used for any benchmark), and it does not capture the Hamming distance between operations belonging to different instruction classes. As the assembly programs consider instructions belonging to the same class, a set of bits (6 for the MIPS processor) does not switch.

Table 3.2 presents the characterization results for each instruction class, considering the Plasma processor (MIPS architecture), in a 28-nm SOI technology, 1.0V@1GHz. As the frequency of operation achieved by the processor after the synthesis was 1GHz, each instruction's energy and the average power obtained in the characterization are strongly correlated.

Table 3.2 – Power characterization results and energy estimation for each instruction class of the processor. Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC (T=1ns) (adapted from [Martins et al., 2014]).

| Class | Avg. Power (mW) | | Energy per inst. (pJ) | |
|---|---|---|---|---|
| | Leakage | Dynamic | Leakage | Dynamic |
| arithmetic | 0.012 | 13.692 | 0.012 | 13.692 |
| logical | | 11.846 | | 11.846 |
| shift | | 10.505 | | 10.505 |
| move | | 11.442 | | 11.442 |
| load_store | | 12.790 | | 24.915 |
| nop | | 7.181 | | 7.181 |
| branches | | 14.340 | | 21.575 |
| jumps | | 10.048 | | 12.119 |

From the energy per instruction, the total energy consumption and, consequently, the power dissipation can be estimated for the processor as follows:

$$E_{processor} = \sum_{i=1}^{n_{class}} n_{instructions_i} \cdot E_{class_i} \tag{3.1}$$

$$P_{processor} = \frac{E_{processor}}{T} \tag{3.2}$$

where: $n_{class}$ is the number of instruction classes, $n_{instructions_i}$ is the number of executed instructions for $class_i$, $E_{class_i}$ is the energy per instruction for $class_i$ and $T$ is the period (*ns*) of the clock cycle.

It is important to mention that if the processor has no task to execute or is waiting for data from another task in another processor, the processor enters in a hold state, consuming only static power.

Finally, the simulation of different benchmarks enables the validation of the calibration process. Table 3.3 summarizes the validation of the processor calibration for seven benchmarks. The energy and power estimated results come from applying equations 3.1 and 3.2 in a trace file generated by an RTL simulation. The measured results are extracted from power reports of netlist simulation. The errors presented on latest columns are mainly due to switching activity (Hamming distance), and pipeline stalls due to dependence between operators.

The data presented in this Section were obtained at room temperature, 25℃. It is worth mentioning that the temperature mainly affects the static consumption due to the leakage current, which increases exponentially with the temperature (up to 9 times at 125℃). The effect of the temperature on the dynamic power is smaller than the leakage power (+3% at 125℃) because it corresponds to capacitors' charge and discharge. Using the method herein described, it is possible to characterize the components for other temperatures.

Table 3.3 – Processor energy estimation error (adapted from [Martins et al., 2014]).

| Benchmark | Estimated Results | | Measured Results | | Error (%) | |
|---|---|---|---|---|---|---|
| | Energy (nJ) | Power (mW) | Energy (nJ) | Power (mW) | Energy | Power |
| binarySearch | 962.40 | 13.23 | 967.319 | 13.244 | -1% | 0% |
| bubble | 984.98 | 14.08 | 1008.466 | 14.199 | -2% | -1% |
| compress | 2470.30 | 13.11 | 2383.534 | 12.412 | 4% | 6% |
| crc | 1621.09 | 13.68 | 1834.327 | 15.454 | -12% | -12% |
| fft | 451.17 | 10.72 | 449.857 | 12.058 | 0% | -11% |
| switchCase | 3094.33 | 12.46 | 3093.3190 | 12.411 | 0% | 0% |
| usqrt | 1093.47 | 13.06 | 1092.600 | 13.044 | 0% | 0% |

### 3.4.2 Router Characterization

The main router internal components include input buffers, crossbar, and control logic (responsible for arbitration and routing). Therefore, the router characterization requires exciting all internal components and providing a payload with an important Hamming distance between flits to induce a high switching activity in the router logic gates. The characterization process of the router is similar to the processor characterization. The router power/energy characterization comprises four steps [Martins et al., 2014]:

1. Traffic generation for maximizing the switching activity of all input buffers. The power dissipation of a router is a function of the reception rate in the input buffers [Ost et al., 2009]. The reception rate is the relationship between the traffic rate and the available bandwidth of the physical link. Figure 3.6 presents the traffic flows to characterize the 5-port central router in a 3x3 NoC. Each traffic flow source injects 1,000 32-flit packets, with a Hamming distance between flits superior to 80%. The injection rates vary from 0% (idle) to 50% of the link bandwidth.

2. Logic synthesis of the instance of a 5-port router to generate a netlist and an SDF file. To account the energy consumed in the links (wires), the output load settings in the timing constraints file consider the wire capacitance between two routers (1 mm long, metal 5, 200fF).

3. Simulation of a 3x3 NoC with the replacement of the RTL description of the central router by the netlist obtained in step 2 (Figure 3.6). The simulations of each traffic scenario from step 1 produce the switching activity at the gate level (VCD or TCF files) for a given injection rate.

4. Power analysis of the switching activity file. A particular feature of the PE is the injection/reception of the packets in burst mode. The DMNI makes the bridge between the router and the memory. Thus, packets are transmitted with an injection rate equal to

100% without blocking the processor. Despite several injection rates used in the characterization method, the power estimation adopts two rates: 100% - active mode, and 0% - idle mode. The 100% rate derives from the extrapolation of the available injection rates.

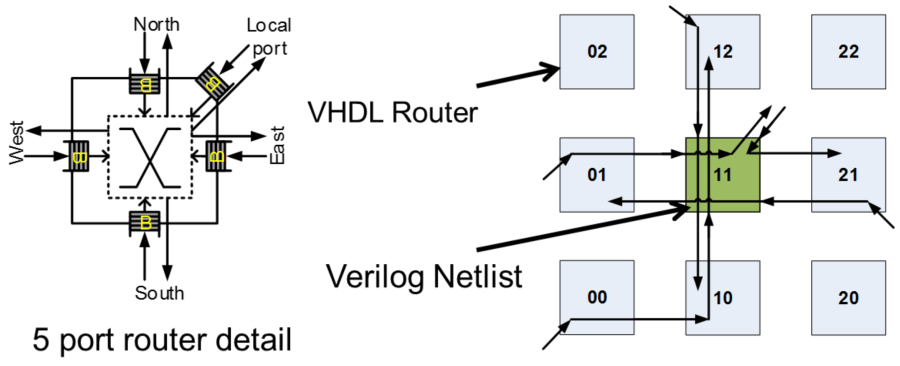


Figure 3.6 – 3x3 NoC traffic scenario for the 5-port router characterization [Martins, 2018].

Table 3.4 presents the power characterization of the router for the two traffic rates. The second column presents the dynamic average power consumption for one buffer. The third column, combinational logic, corresponds to the remaining parts of the 5-port router. The last Table column presents the router leakage power.

Table 3.4 – Router average power. Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC (T=1ns) (adapted from [Martins, 2018]).

| Trafic Rate | One buffer | Combinational Logic | $P_{leak_{router}}$ |
|---|---|---|---|
| 0% - idle | 799.72µW | 525.09µW | 1.646µW |
| 100% - active | 1881.12µW | 1896.38µW | |

Let $E_{active}$ be the dynamic active energy to receive one flit (with one active buffer) and $E_{idle}$ be the spent dynamic energy in idle mode, which are derived from Table 3.4 as follows:

$$E_{active} = [(n_{ports} - 1) \times P^{idle}_{buffer} + P^{active}_{buffer} + P^{active}_{comb}] \times T \tag{3.3}$$

$$E_{idle} = [n_{ports} \times P^{idle}_{buffer} + P^{idle}_{comb}] \times T \tag{3.4}$$

where: $n_{ports}$ is the number of ports of the router, $P^{idle}_{component}$ is the average idle power of a given component, $P^{active}_{component}$ component is the average active power, and T is the period used to characterize the router.

### 3.4.3 Memory Characterization

In general, a memory generator tool provides memories as black boxes in the technology design kit without an RTL model. The CACTI-P [Li et al., 2011] tool models distinct memory types and generate estimations like access time, silicon area, and power. This tool also supports Dynamic Voltage Scaling (DVS). CACTI-P allows the characterization of the PE local memory's energy consumption, configured as a 64KB scratchpad memory, with two ports. Table 3.5 presents the characterization data produced by CACTI-P. CACTI-P tool allows some technology options (like cell and peripheral circuits) different from the industrial libraries of standard cells previously deployed. The technology settings are calibrated to generate consistent results when comparing with processor and memory. In Table 3.5, the access time corresponds to the period used to characterize the processor, and the router (1 ns), $P_{leak_{memory}}$ is the leakage power, $E_{load}$ is the dynamic read energy per access, and $E_{store}$ is the dynamic write energy per access.

Table 3.5 – CACTI-P Report for a scratchpad Memory (28nm, 1.0V, 25℃)

| Access time | $P_{leak_{memory}}$ | $E_{load}$ | $E_{store}$ |
|---|---|---|---|
| 0.369 ns | 0.32 mW* | 93.96pJ | 134.97 pJ |

The leakage value estimated by CACTI-P does not consider silicon on insulator (SOI) technology. SOI leakage power tends to be negligible compared to bulk technologies. For this reason, we did not considered the leakage value provided by CACTI-P in our estimations.

## 3.5 Hierarchical Monitoring

Figure 3.7 displays the many-core hierarchical organization by highlighting the hierarchy levels and the communication pattern. The exchanged messages related to the system management may be intra- or inter-cluster. SPs belonging to a cluster communicates with the manager PE of its cluster, characterizing an intra-cluster communication. Similarly, the intercluster communication occurs when CMs communicate with the GM. Note that, intra-cluster communication also occurs in the cluster managed by the GM because the GM acts as a CM in this case.

Figure 3.8 displays the 3-level hierarchical power monitoring scheme. Each SP monitors its power consumption and sends the sample per PE periodically to the CM of its cluster. At the cluster level, the CM receives the observing data of its corresponding SPs and updates look-up tables with the received samples to compute the cluster data. The intra-cluster look-up tables size at the cluster level corresponds to the number of SPs per cluster. In particular, the SPs of the GM cluster send the sample per PE straight to the GM. Finally, the CMs send to the GM the sample per cluster when the CM received all samples

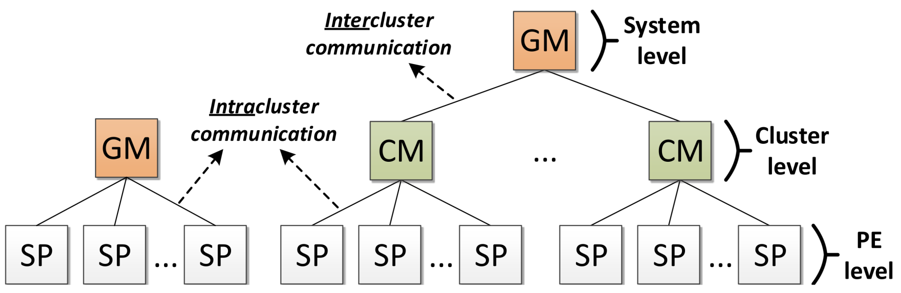


Figure 3.7 – Hierarchical organization of the PEs [Martins, 2018].

from the SPs. Similarly to the sample per cluster, as the GM receives energy data from all CMs and SPs of its cluster, the GM observes the system by updating inter-cluster look-up tables. Accordingly, the inter-cluster look-up tables size at the system level corresponds to the number of clusters.

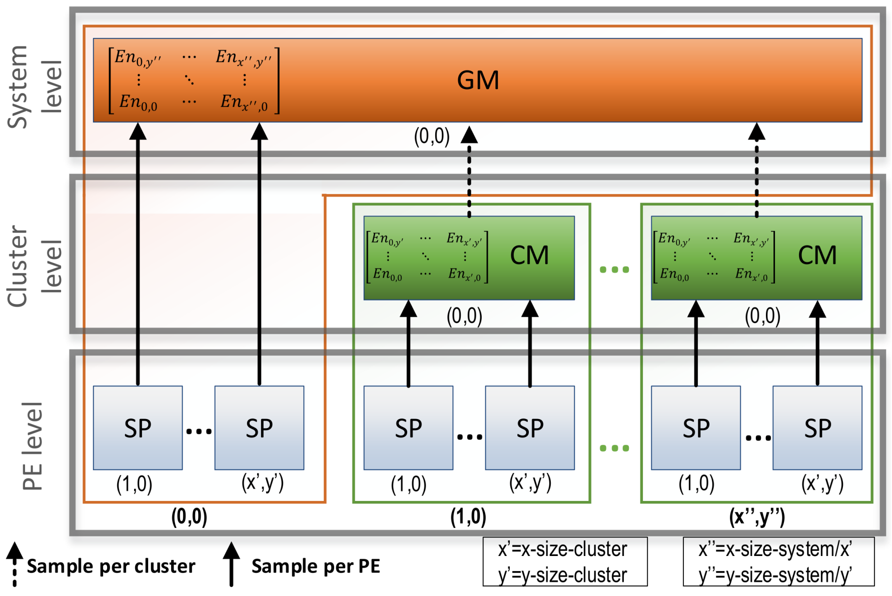


Figure 3.8 – Hierarchical energy monitoring of the reference architecture [Martins et al., 2015].

This hierarchical monitoring method allows each cluster manager to know their SPs' consumption, allowing local decisions to be made at the cluster level with precision. At the system level, the GM knows each cluster's consumption, allowing systemic decisions to be made based on each cluster's state.

# 4. TEMPERATURE ESTIMATION

This Chapter presents the studies of thermal modeling that led to developing a thermal estimation accelerator (TEA), which is the first contribution of this Thesis. The Chapter is organized as follows:

- Section 4.1 shows the studies of temperature models considered to be adopted in this work to provide temperature monitoring.
- Section 4.2 details MatEx and how it was adapted to provide runtime temperature estimation.
- Section 4.4 describes the hierarchical temperature monitoring scheme and how TEA communicates with the system.
- Section 4.3 presents our proposed accelerator, TEA, to enable temperature monitoring, with performance, area, and accuracy results.
- Section 4.6 concludes this Chapter.

## 4.1 Temperature Models

The study of temperature models is an important step towards designing and evaluating dynamic thermal management techniques at the design phase of a many-core system. Section 4.1.1 analyses the HotSpot model since it is widely adopted as a tool for estimating the temperature at early stages of system design, and it was also the base for a model proposed by Castilhos et al. [Castilhos et al., 2016]. Castilhos' model is analyzed according to its precision and applicability in a runtime thermal management solution (Section 2.6) because it was a candidate to provide temperature estimation thermal management. Next, MatEx model [Pagani et al., 2015] is briefly described (Section 4.1.3), since it is the model adopted to provide runtime temperature estimation in this work, and it is explained in more details in Section 4.2, where we show the simplifications made to enable runtime temperature monitoring.

### 4.1.1 HotSpot

HotSpot is a modeling methodology to develop compact thermal models based on stacked-layer packaging of modern very large-scale integration (VLSI) systems [Huang et al., 2006]. HotSpot compact thermal model is based on the duality between thermal and electrical phenomena, where the heat flow passing through a thermal resistance is analogous to the electrical current, and the temperature difference is analogous to voltage. Thermal capacitance defines the heat absorbing capability, defining the transient behavior of temperature over time, being analogous to electrical capacitance, which defines the absorption of electrical charge.

HotSpot thermal model includes the RC modeling of four stacked layers made of different materials, including the silicon substrate, the thermal interface material, the heat spreader, and the heat sink. The silicon substrate layer and the interface material are divided according to architecture-level units, Figure 4.1(c). Other layers, usually larger in size than the die, are divided according to Figure 4.1(a), with the center part divided like the die, with additional resistances to model the heat transfer between the border of the die to the material and the border of the material to the air. In addition to these subdivisions, which model the lateral resistances, each block one vertical resistance connected to the next layer, Figure 4.1(b). The thermal resistances and capacitances are calculated based on the block's geometry, such as the area and the layer's thickness and some material properties like the specific heat and thermal conductance.

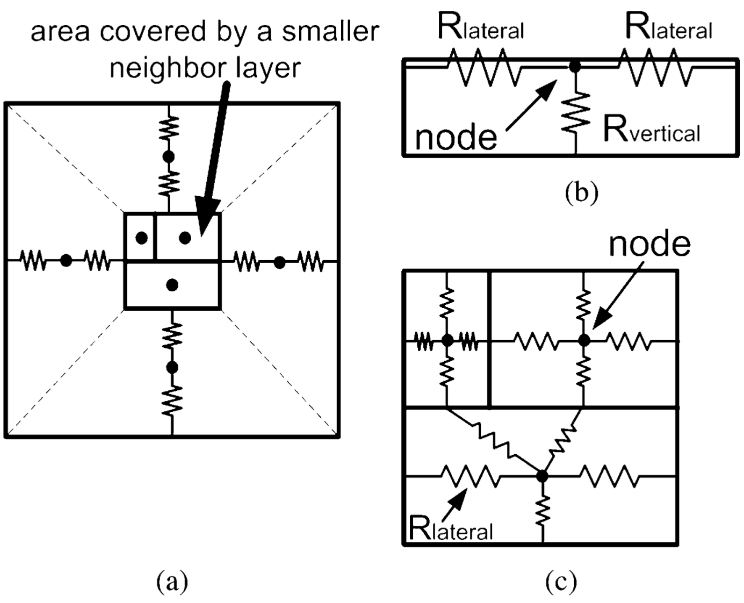


Figure 4.1 – Architectural partitioning of the system. (a) Top view with heat sink resistances. (b) One block with its lateral and vertical thermal resistances. (c) Silicon die layer divided into three blocks [Huang et al., 2006].

For transient temperature computations, HotSpot solves the system of differential equations using a fourth-order *Runge-Kutta* numerical method, with an adaptive number of iterations [Pagani et al., 2015]. This approach provides faster results than traditional thermal simulations, and it is suitable for preliminary VLSI design estimations.

HotSpot algorithm cannot provide real-time temperature estimation with enough time resolution to be used in DTM. However, it is still used by a number of works to provide design time estimations to enable DTM proposals [Pagani et al., 2017] [Chakraborty and Kapoor, 2018] [Sha et al., 2018].

### 4.1.2 Castilhos' Model

Castilhos et al. [Castilhos et al., 2016] proposed a software-based temperature monitoring model for many-core systems. Castilhos' Model was based on a series of simplifications of the HotSpot model to allow the temperature estimation in an embedded processor with a low performance overhead. Castilhos' model was initially considered for the development of thermal management techniques, given the possibility of its runtime execution in software, in a given PE.

The maximum error of Castilhos' Model is 10%, when compared to HotSpot, which may affect the accuracy of DTM techniques that uses this model to collect temperature information. To evaluate the feasibility of using Castilhos Model to employ a runtime temperature management strategy, we implement the algorithm and collect power information while running a dynamic worload scenario on the reference platform. Figure 4.2 presents temperature results for selected PEs, comparing Castilhos' Model to HotSpot in a scenario where tasks may start and finish its execution in short periods of time, and new tasks may be mapped on different PEs.

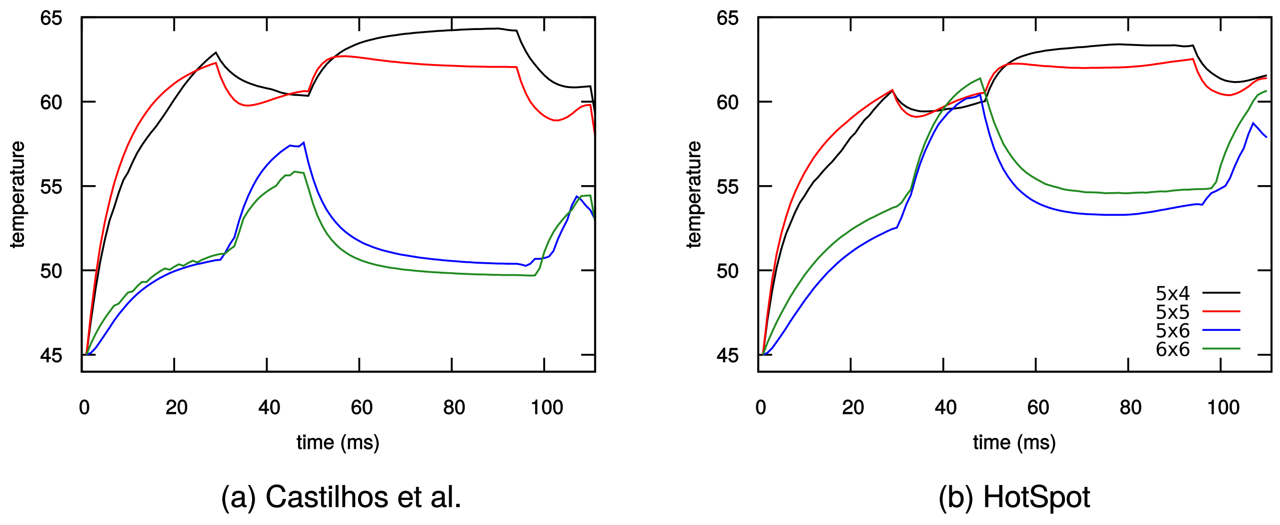


Figure 4.2 – Comparison of temperature estimation models in a 6x6 system.

As expected, Castilhos Model presents a reasonable approximation of the system's thermal behavior. However, the error introduced, especially in border PEs (5x6, 6x6), can be a limiting factor for its use in thermal management techniques. Besides the peak temperature on nodes 5x6 and 6x6 being inaccurate on Figure 4.2, it is possible to observe an inversion in the temperature of these two nodes, which may lead to wrong thermal management decisions.

The major limitations of Castilhos Model are that the estimation of a PE temperature considers only the power consumed by the PE itself and its direct neighbors, lateral and diagonal, and that the temperature contribution of a given power sample is considered in a limited number of samples. According to how HotSpot models temperature in a node of the system, all nodes power consumption contributes to the temperature in a given node, and that is not negligible. The simplifications employed in Castilhos Model reduce the algorithm's complexity, but they can induce an accumulation of the error over the time and lead to wrong thermal decisions.

### 4.1.3 MatEx

MatEx [Pagani et al., 2015] is the reference algorithm used in this Thesis to build an accelerator that enables temperature estimation at runtime. MatEx uses a thermal model based on RC thermal networks, similar to HotSpot, relating temperatures in different areas of the chip with their power consumption. MatEx differentiates itself from HotSpot in the method used to solve first-order differential equations from RC networks. While HotSpot uses conventional numerical methods, MatEx solves the differential equations by using matrix exponentials and linear algebra. This method results in a polynomial time algorithm based on matrix multiplications that, similarly to graphics rendering, is executed faster and more efficiently with dedicated structures.

MatEx uses the same structure as HotSpot to extract RC constants of an input circuit based on its floorplan and thermal dissipation characteristics. This technique results in a generic tool that can be used to estimate the thermal behavior of any circuit based on its power consumption. This work considers a previously characterized system to build a peripheral capable of evaluating its temperature.

MatEx is the model of choice to provide runtime temperature estimation for this Thesis because it presents a negligible error when compared to HotSpot, while running significantly faster. Castilhos' model runs faster, when compared to MatEx, enabling the software execution of the estimation algorithm, but do not provide enough accuracy to be used as a reference for thermal management of a many-core system.

## 4.2 Runtime Temperature Estimation

MatEx open-source tool can estimate transient and peak temperatures after a power state change of the system, and the duration of power states can be variable. To evaluate the adaptation of the MatEx algorithm to estimate all transient temperatures of the system at runtime, we first present how the algorithm works for a generic floorplan and power state changes as follows:

$$AT' + BT = P + T_{amb}G \tag{4.1}$$

where:

- $A = [a_{i,j}]_{N\times N}$ contains the thermal capacitances between nodes $i$ and $j$,
- $B = [b_{i,j}]_{N\times N}$ contains the thermal conductances between nodes $i$ and $j$,
- $T = [T_i(t)]_{N\times 1}$ vector containing the temperature on every thermal node,
- $T' = [T_i'(t)]_{N\times 1}$ vector containing the first order derivative of the temperature at time $t$,
- $P = [p_i]_{N\times 1}$ vector containing the power consumption on every thermal node,
- $G = [g_i]_{N\times 1}$ vector containing the thermal conductance between every thermal node and the ambient temperature ($T_{\text{amb}}$).

The RC model used to estimate the thermal behavior of the system contains $N$ thermal nodes interconnected through thermal conductances ($B$). Each thermal node has also a thermal capacitance ($A$) to consider the behavior of transient temperatures of each node. $T_{\text{amb}}$ is considered to be constant, and the power consumption ($P$) of the active nodes are considered as heat sources.

The characterization process required by MatEx to extract A and B matrices and compute the temperatures based in a power vector ($P$) involves two main steps executed at design time: floorplanning and power consumption evaluation. The floorplanning of the system, described in Section 3.1, is a function of the technology node used to synthesize the PEs. The complete PE was synthesized using a 28nm technology, resulting in a 0.27 mm$^2$ area. The power consumption evaluation method (Section 3.4) considers the CPU, memory and router consumption, main modules of the PE.

It is worth to note that the number of thermal nodes ($N$) is greater than the number of PEs, as the heat dissipation structure is also modeled. Each PE results in 4 thermal nodes: the first is the circuit itself that considers its power consumption as a heat source, the second is the silicon substrate and interface material, the third is the heat sink, and the fourth is the heat spreader [Huang et al., 2006]. The model also considers that the ambient temperature interacts with the three top layers of the system in their four sides, adding 12 more thermal nodes to the model. Therefore, a system modeled with 16 PEs results in a model with 76 thermal nodes, a system with 36 PEs generates 156 thermal nodes and so on, increasing significantly the amount of memory required to hold the matrices used by the algorithm.

The first step performed by the model is the computation of steady state temperatures ($T_{\text{steady}}$) for a given power vector of the active thermal nodes. $T_{\text{steady}}$ depend only on the thermal conductances, once the capacitances are used to model the transient behavior of temperatures. Thus, $T_{\text{steady}}$ can be computed by modifying the Equation 4.1 as follows:

$$BT_{\text{steady}} = P + T_{\text{amb}}G \quad \text{or} \quad T_{\text{steady}} = B^{-1}P + T_{\text{amb}}B^{-1}G \tag{4.2}$$

Once $T_{\text{steady}}$ is found, it is possible to calculate the transient temperatures using Equation 4.1. Once initial temperature ($T_{\text{init}}$) and $T_{\text{steady}}$ are calculated, the temperature estimation ($T$) after $t$ seconds can be found according to the following equation:

$$T = T_{\text{steady}} + e^{Ct}(T_{\text{init}} - T_{\text{steady}}) \tag{4.3}$$

where $e^{Ct}$ is the exponential of matrix $C$ at time interval $t$, and the matrix $C$ is a derived from Equation (4.1) as follows:

$$C = -A^{-1}B \tag{4.4}$$

MatEx tool solves matrix exponentials using eigenvalues and eigenvectors, applying linear algebra. This procedure has high complexity ($O(n^3)$) but only needs to be executed once for a given chip. The matrix exponential $e^{Ct}$ is constant for a fixed interval of $t$, and matrix C is the relationship between the capacitances and conductances of the system (Equation 4.4). In MatEx standard algorithm, the time interval for temperature estimation can be variable, thus an auxiliary matrix H is used to compute transient temperatures. However, we compute transient temperatures in a fixed interval of time, named *monitoring window* to enable runtime temperature estimation (Sections 4.3 and 4.4). Thus, in our implementation, the temperature monitoring is done at a fixed time interval, so the same matrix $e^{Ct}$ can be used to compute all transient temperatures for each predefined time interval. The resultant equations for temperature estimation in our approach are the following:

$$T_{\text{steady}k} = \sum_{j=1}^{N} b^{-1}_{k,j} \cdot p_j + T_{\text{amb}} \tag{4.5}$$

$$T_k(t) = T_{\text{steady}k} + \sum_{j=1}^{N} cexp_{k,j} \cdot (T_{\text{init}j} - T_{\text{steady}j}) \tag{4.6}$$

where: $j$ and $k$ are the indexes of matrices and vectors, and $cexp_{i,j}$ represents the matrix exponential $e^{Ct}$ for a fixed time interval $t$. To apply this model it is necessary to extract matrices $B^{-1}$ and $e^{Ct} = [cexp_{i,j}]_{N\times N}$ from MatEx.

An important concern about the implementation of temperature estimation is the amount of memory required to store the matrices that are used in the computation. To reduce the amount of memory necessary to run the algorithm we first observed that the contribution of $T_{\text{amb}}$ in $T_{\text{steady}}$ is proportional to the multiplication of vector $G$ and matrix $B^{-1}$, which are both constants for a given chip. This way, the contribution of the ambient temperature in each thermal node will always be the same in this model, so that only the columns related with active power nodes of matrix $B^{-1}$ need to be stored in memory. In our tests, the contribution of $T_{\text{amb}}$ in $T_{\text{steady}}$ was equal to $T_{\text{amb}}$, so we just sum the $T_{\text{amb}}$ to the power contribution to compute $T_{\text{steady}}$, as shown in equation (4.5). As the matrix $e^{Ct} = [cexp_{i,j}]_{N\times N}$ is used by equation (4.6), the total memory requirement to run this model is $PN + N^2$, where $P$ is the numbers of PEs, that is, active thermal nodes, and $N$ is the total number of thermal nodes.

The amount of the registers and the matrices' memory increases quadratically with N. For example, in a 9x9 system, P=81 and N=336, and matrices should store 140,112 values. This work adopts two additional strategies to reduce the amount of required memory: 32-bit representation and discretization. The model discretizes the matrices' values into 128 ranges, representing the values with 7 bits. A lookup table translates the 7-bit value to a 32-bit floating point value. With these two approaches, the total amount of memory for a 12x12 system corresponds to 367.79 KB, instead of 1.7 MB, without discretization. We tested discretizations using 6, 7 and 8 bits. when using a 6 bits discretization, the error was above 1%. Using 7 and 8 bits discretization resulted in an average error lower than 1%, so we decided to use 7 bits, for greater memory savings. Table 4.1 presents the memory requirements according to the system size.

Table 4.1 – *TEA* memory and register sizes.

| System size | 3x3 | 6x6 | 9x9 | 12x12 |
|---|---|---|---|---|
| Number of PEs ($P$) | 9 | 36 | 81 | 144 |
| Thermal Elements ($N$) | 48 | 156 | 336 | 588 |
| Power registers ($P$) | 9 | 36 | 81 | 144 |
| Temperature registers ($2N$) | 96 | 312 | 672 | 1,176 |
| Total registers (32 bits) | 105 | 348 | 753 | 1,320 |
| Registers' size (bytes) | 420 | 1,392 | 3,012 | 5,280 |
| Matrices values ($N^2 + NP$) | 2,736 | 29,952 | 140,112 | 430,416 |
| Mem. size 32-bit words (KB) | 10.69 | 117.00 | 547.31 | 1,681.31 |
| Mem. size 7-bit words (KB) | 2.00 | 25.59 | 119.72 | 367.79 |

## 4.3 Thermal Estimation Accelerator - TEA

TEA creation was important in the context of this Thesis for two main reasons: (*i*) it enables runtime temperature monitoring without disturbing software execution, (*ii*) the computation of temperature requires knowledge of the power consumed by all PEs, thus a centralized approach is preferred.

TEA [da Silva et al., 2021b] executes the following actions: (*i*) receives the power samples from the manager PEs periodically; (*ii*) computes the current temperature of each PE using equations 4.5 and 4.6 at the end of a monitoring window; (*iii*) sends a packet with the temperature values to the Global Manager ($GM_{PE}$) after updating the PEs' temperature.

Figure 4.3 presents the TEA block diagram. TEA uses three single port memory blocks, responsible for storing: (*i*) $B_{inv}$ matrix ($B^{-1}$); (*ii*) $C_{exp}$ matrix ($e^{Ct}$); (*iii*) Temp memory.

Temp memory stores the steady temperatures ($T_{steady}(t)$ - Equation 4.5), intermediate values of Equation 4.6 ($T_{init(j)} - T_{steady(j)}$) and transient temperatures ($T_k(t)$ - Equation 4.6). A register bank stores the power values of each PE (Power registers in Figure 4.3).

TEA has a single 32-bit multiplier-accumulator (MAC) instance because estimating the system temperature consists of vector and matrices multiplication. Although it is simple to parallelize vectors and matrices' multiplication, a single MAC provides enough performance to estimate the temperatures of a system with up to 220 PEs using a monitoring window of 1ms@1GHz. For larger systems, it is possible to compute more than one temperature value simultaneously using parallelization. Thus, it is necessary to add MACs and modifications in the output FSM (Figure 4.3) to address different power registers and matrix positions.

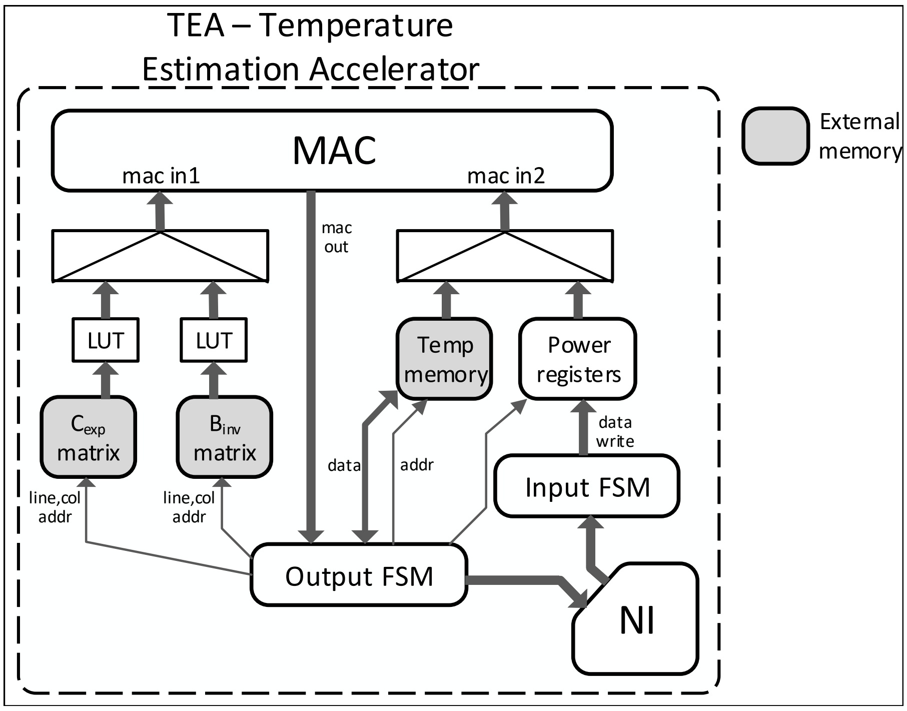


Figure 4.3 – TEA Architecture Overview.

Two finite state machines (FSMs) communicate with the network interface (NI), generate addresses for the memories and registers, and control the MatEx algorithm's execution. According to the cluster that sent the packet, the input FSM handles incoming packets, identifying the sender and storing the received power values in the correct positions of the register bank. The output FSM waits for the end of a monitoring window and starts executing the MatEx algorithm. At the end of the execution, the output FSM sends the packet with the estimated temperatures to the $GM_{PE}$.

The input FSM has three states:

- *WAIT_st*: waits for the reception of a new packet;

- *HEADER_st*: reads the packet header and identifies the cluster that sent the power data;
- *DATA_st*: receives the packet's payload and stores the power data in the correct locations of the power register bank.

The output FSM has five states:

- *WAIT_st*: waits for the end of the monitoring window;
- *STEADY_st*: computes the steady-state temperatures (Equation 4.5) by reading the $B_{inv}$ memory and the power registers, storing the steady temperature values and the temperature delta between the current and steady temperatures in the Temp memory;
- *TRANSIENT_st*: computes the transient temperatures (Equation 4.6) by reading the $C_{exp}$ memory and the delta temperature in the Temp memory, storing the results back to the Temp memory;
- *SEND_HEADER_st*: assembles and sends the packet header to the $\text{GM}_{\text{PE}}$;
- *SEND_PAYLOAD_st*: sends the payload data with estimated transient temperatures for the current monitoring window.

## 4.4 Runtime Temperature Monitoring

The temperature in each PE affects the temperature on neighboring PEs due to the thermal conductance effect. For this reason, to estimate the temperature of a given PE, it is necessary to know the power consumption and the previous temperature of all other PEs. Due to this data dependency, adopting a centralized temperature estimation accelerator is preferable to adopting multiple accelerators. The estimation's distribution would imply several communication overheads, compromising the performance and interfering with NoC traffic.

Figure 4.4 overviews the hierarchical temperature monitoring scheme [da Silva et al., 2019] by extending the hierarchical monitoring presented at previous Chapter (Figure 3.7). At the lower level, SPs monitor their power consumption. The PE is instrumentalized to count the number of executed instructions, the number of flits traversing the router, and the memory accesses at each monitoring window (Section 3.4). The power data comes from PE characterization at the gate level, considering the voltage regulators' power overhead needed to apply DVFS at the PE level. Periodically, the SPs send the observed data to their manager, and each manager sends its cluster power data directly to the accelerator.

At the cluster level, managers receive the power samples of their SPs and update lookup tables. Each manager creates a packet with the power information related to the

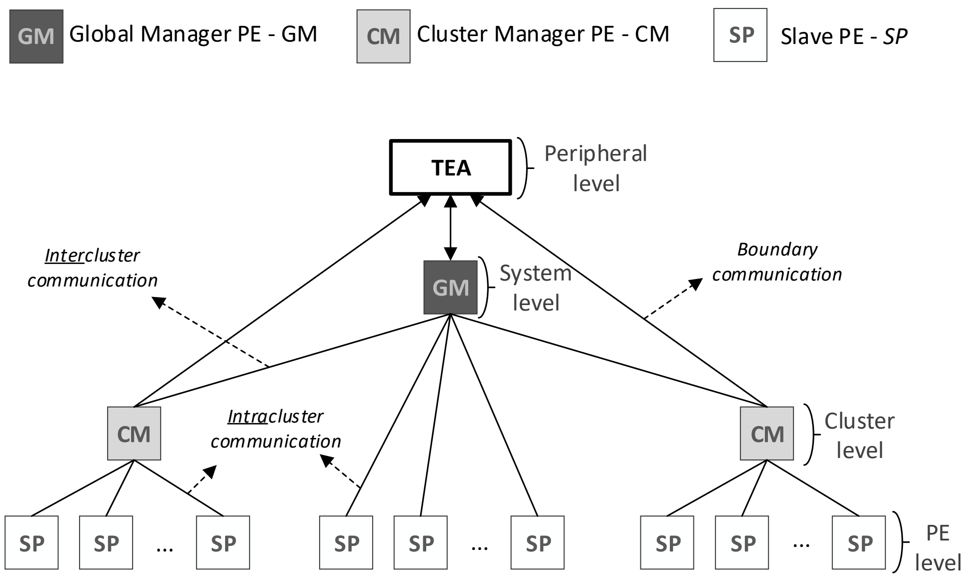


Figure 4.4 – Hierarchical temperature monitoring strategy.

cluster and transmits it to TEA. Even for large systems, the number of packets sent to TEA is small, corresponding to the clusters' number. The new level added, peripheral level, enables the temperature monitoring based on power monitoring, presented in Figure 3.7.

TEA executes the upper level of the fine-grain temperature monitoring scheme. TEA receives the power consumed by all PEs and executes the temperature estimation procedure. Next, TEA sends a packet with the temperatures of all PEs to the $GM_{PE}$ after the estimating process. Finally, the $GM_{PE}$ receives this temperature message, storing the current temperature data, and transmits to the $CM_{PEs}$ the temperature of the SPs belonging to each cluster.

Although the monitoring approach gives precise power data, TEA operation is independent of the measurement method. Other complementary power/temperature estimation methods may be used, as from physical sensors, together with TEA to improve the temperature estimation accuracy.

The software implementation used to compare the performance with TEA uses a similar monitoring method. The difference is that the $GM_{PE}$ executes the temperature estimation instead of TEA.

## 4.5 Results of Thermal Monitoring using TEA

Two hardware descriptions model the TEA accelerator: (*i*) clock cycle-accurate SystemC model, used to speed up simulations; (*ii*) synthesizable RTL model (VHDL), used to extract area and power results.

Results evaluate the proposed hardware accelerator in terms of performance and power (compared to software) – Section 4.5.1, area consumption – Section 4.5.2, and temperature estimation accuracy (compared to HotSpot) – Section 4.5.3.

### 4.5.1 Software versus Hardware Approaches

Table 4.2 presents the time required to compute the temperatures (ms@1 GHz) and the power consumption (mW) by TEA compared with a software implementation of the same algorithm, considering three system sizes. We estimated the power consumption by simulating the RTL design of TEA with Cadence Xcelium and extracting the switching activity. The estimation of the power consumption of the CPU running the algorithm uses the same process.

Table 4.2 – Power and Performance results for TEA and software implementation of the temperature model. TEA implementation: Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC (T=1ns).

| | **System size** | **6x6** | **8x8** | **10x10** |
|---|---|---|---|---|
| | Thermal Nodes | 156 | 268 | 412 |
| **TEA** | Execution time (ms) | 0.0301 | 0.0892 | 0.2112 |
| | Power (mW) | 10.078 | 18.433 | 31.821 |
| **Software** | Execution time (ms) | 0.6618 | 2.007 | 4.805 |
| | Power (mW) | 23.93 | 76.716 | 218.762 |

For 6x6 systems, the software implementation consumes 137.45% more power than TEA, with the computation time still shorter than the monitoring window. A single core running at 1 GHz cannot deliver the temperature results in a 1 ms monitoring window for larger systems (8x8 and 10x10 systems). Results show that TEA consumes 0.2112 ms for a 10x10 system to estimate the temperatures, using a single MAC. Thus, TEA has enough performance to complete its task, considering the 1 ms monitoring window.

The computation time for the software implementation, in a 10x10 system size, requires 4.8 ms. As the monitoring window is 1 ms, it is impossible to use a centralized software implementation for this system size. As the algorithm can be parallelized, each processor can compute its temperature. However, to distribute the temperature computation, the communication overhead may be the limiting factor since each PE would need to transmit its energy consumption to all other PEs. Another issue related to the distributed implementation is that all processors of the system would spend some time running the algorithm, impacting the applications' performance. Considering a 10x10 system, each processor would spend about 10% of the execution time running the temperature estimation algorithm, considering a 1ms monitoring window and a variable amount of time to receive the power samples from all other PEs. We argue that the dedicated IP is the best choice to perform a runtime temperature estimation for those reasons.

### 4.5.2 Area Overhead

Figure 4.5 compares TEA to the PE area (logic blocks, without considering the memory blocks) using Cadence Genus and a 28 nm SOI library. TEA logic area increases with the system size due to the power registers needed to store the power consumption of each PE (Table 4.1).

The TEA-PE area ratio for a 10x10 system is lower than 1 (85.2%), while for a 6x6 system, TEA consumes only a fraction of the PE area (43.2%). Memory consumption plays an important role in the overall area consumption of both TEA and PEs. Using the techniques described in section 4.2 (32-bit representation and discretization) to reduce the amount of memory required by TEA, the memory requires to estimate the temperature for a 10x10 system is lower than the local PE memory (128 KB).

Thus, the total TEA area is smaller than that of a PE, even in larger systems. Hence, its silicon area consumption is not an obstacle to its adoption.

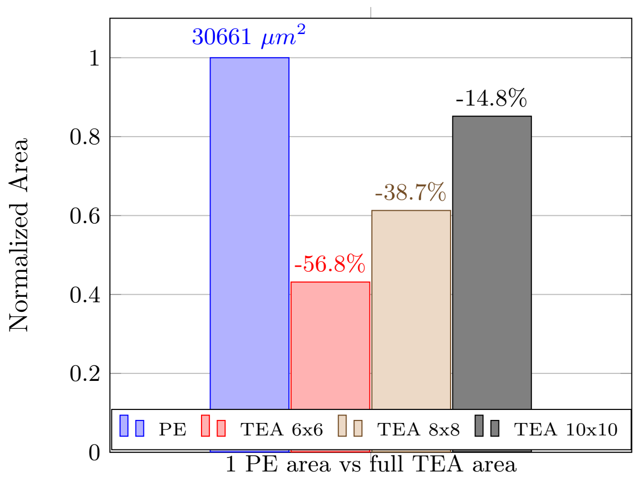


Figure 4.5 – PE and TEA area comparison. Library C28SOI_SC_12 (28nm).

### 4.5.3 Temperature Estimation Accuracy

The analysis of TEA temperature estimation precision compares its temperature results with HotSpot. Each testcase was executed in two different simulations, one considering only the integer matrices implementation and the other considering the matrix discretization technique using floating point values. The top rows in Table 4.3 present the error values for the integer implementation, while the bottom rows present the error when using the matrix discretization technique. The table shows the average, maximum, and standard deviation of the error in absolute values (left) and percentages(right). The Table shows that the simulations' error was never higher than 1%, and the average error was below 0.5%, which is considered a good accuracy to employ runtime management decisions. The results also show that using the matrix discretization technique does not imply an increase in the estimation error, allowing memory reduction, adding scalability to the TEA.

Table 4.3 – Temperature error results for different scenarios. Reference: HotSpot.

| Testcase | | Absolute Error | | | Percentage | | |
|---|---|---|---|---|---|---|---|
| | | Average | Std. dev | Max | Average | Std dev | Max |
| Integer Matrices | 1 | 0.2379°C | 0.1223°C | 0.4843°C | 0.4234% | 0.2102% | 0.8291% |
| | 2 | 0.2723°C | 0.1313°C | 0.5193°C | 0.4291% | 0.1917% | 0.795% |
| | 3 | 0.2386°C | 0.1193°C | 0.4859°C | 0.4044% | 0.1855% | 0.7605% |
| Discrete Matrices | 1 | 0.1846°C | 0.1051°C | 0.4151°C | 0.3281% | 0.1821% | 0.6991% |
| | 2 | 0.1891°C | 0.1048°C | 0.4061°C | 0.297% | 0.157% | 0.6162% |
| | 3 | 0.1741°C | 0.0992°C | 0.3956°C | 0.294% | 0.1573% | 0.657% |

We observed that integer matrices resulted in a higher error than using floating point matrices with discretization. The optimal solution in terms of hardware usage would be the use integer matrices with discretization, because it would result in a simpler MAC unit and lower memory usage. However, testing this solution we found that the error tends to accumulate in time, increasing the average error in longer simulations. For this reason we adopted floating point discrete matrices.

Figure 4.6 shows the simulation results for an 8x8 system, where each square is the temperature for 1 PE. The simulation consists of a typical utilization scenario, with tasks having a long execution time mixed with tasks that start and end in several moments, characterizing a dynamic workload. We used a mix of computation and communication-bound applications, such as Dijkstra, Sort, MPEG, audio and video decoding, DTW, AES, and a synthetic application. Five temperature snapshots were taken during the simulation, represented in the z-axis in the Figure. The simulation provides the power vectors, which feed the HotSpot tool to produce Figure 4.6(a). Figure 4.6(b) presents the results computed by TEA.

Analyzing the results graphically, it is not possible to observe the error introduced by the proposed method. The average error is about 0.5$^o$C (standard deviation equal to 0.2$^o$C degrees), and the maximum error observed in this simulation was 0.9$^o$C degrees.

## 4.6 Final Remarks

The Temperature Estimation Accelerator (TEA) presented in this Chapter computes temperatures of PEs accurately at runtime, based on the power consumption and the physical characteristics of the system. TEA is positioned at the top of the hierarchical monitoring scheme to avoid network traffic issues. TEA is suitable for Dynamic Thermal Management (DTM) techniques that require fine-grain temperature monitoring to provide control decisions to ensure safe operation and good performance.

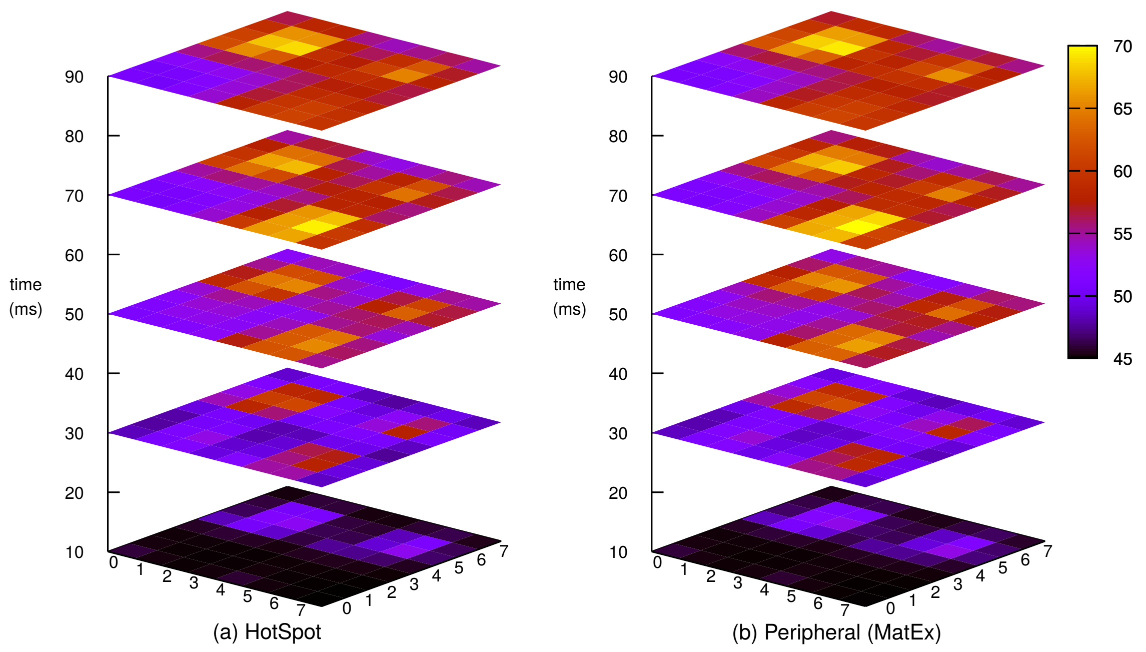


Figure 4.6 – Typical scenario temperature results (8x8 system) for (a) HotSpot and (b) TEA.

The proposed approach is generic in terms of adoption in other many-core systems. TEA sends PE temperatures to the global manager after receiving power samples from each PE. To adopt our proposal in other systems, it is necessary the system characterization, considering its floorplanning and the monitoring window, to produce the thermal capacitance and resistance matrices used by MatEx.

Results show that a software approach cannot provide temperature estimation for systems with many PEs due to the execution time to estimate the temperature. The software approach is also less efficient in terms of power than TEA, making our proposal a candidate solution to provide runtime temperature estimation. Finally, we observed that even adopting simplifications in the temperature model, the temperature estimation accuracy is not sacrificed compared to the thermal reference model.

# 5. DYNAMIC THERMAL MANAGEMENT

Dynamic Thermal Management requires multiple layers of sensors and actuators to provide system self-awareness. Self-aware systems are capable of adapting their behavior and resources to automatically find the best way to accomplish a given goal despite changing environmental conditions and demands [Dutt et al., 2015]. We adopt the Observe-Decide-Act (ODA) paradigm to connect the monitoring to the actuation infrastructure. As illustrated in Figure 5.1(a), the system monitors key features, applies a control and decision algorithm, and deploys appropriate actions to adapt to changes in its state.

The PE senses its power consumption and sends this information to its cluster manager. The cluster-level observation flows in two directions, the power observation is done from the CM to the GM, and the temperature observation flows in the opposite direction. The system-level observation is the power dissipation information that the GM sends to TEA. The peripheral-level observation is the temperature of all PEs that TEA sends to the GM.

The right side of Figure 5.1(a) shows that actuation occurs in the top-down direction. The peripheral-level actuation may occur when the application injector requests the GM to execute a new application. The system-level actuation is decided at the GM, which chooses a cluster to run the application. The cluster-level actuation may be an application mapping or a task migration, which is decided at the CM and sent to the SPs. The PE level actuation, as the DVFS, is also decided at the CM and sent to the destination PE.

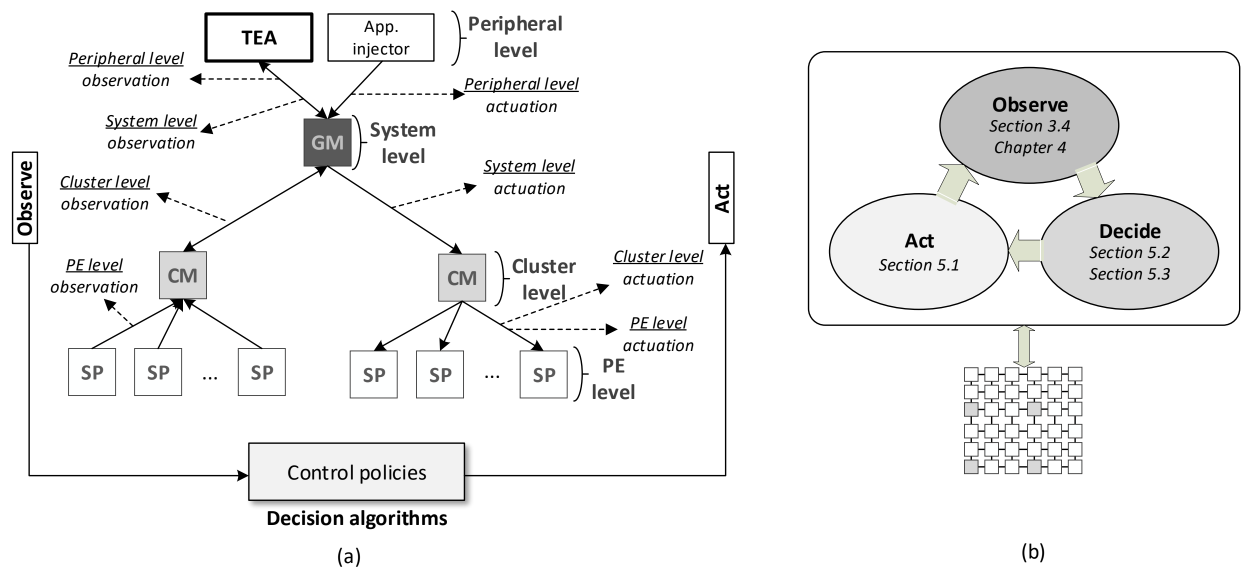


Figure 5.1 – Observe-Decide-Act paradigm is the methodology for the development of Dynamic Thermal Management applied to the reference platform.

Figure 5.1(b) presents the relationship between the ODA phases and the Thesis' sections. Note that the Observe phase was previously presented in Section 3.4 (power estimation) and Chapter 4 (temperature estimation). Section 5.2 and Section 5.3 corresponds to the second original contribution of the Thesis.

This Chapter is organized as follows:

- Section 5.1 details the actuation knobs adopted in this work, including application admission, task mapping, task migration, DVFS, and the power states that PEs can work.
- Section 5.2 presents the algorithms responsible for managing each actuation knob.
- Section 5.3 contains a set of algorithms that select a given cluster or PE according to the computation done by the algorithms detailed in Section 5.2.
- Section 5.4 evaluates the algorithms previously presented.
- Section 5.5 concludes this Chapter, summarizing the contributions of the proposed DTM heuristics.

## 5.1 Actuation Knobs

The actuation knobs are the tools the decision algorithms have available to manage the system. Therefore, the details of actuation operations are essential to develop and understand the decision algorithms. Due to the hierarchical organization of the reference platform, the actuation set is distributed in hierarchy levels according to the ones presented in Figure 5.1(a).

The actuation mechanisms are described through protocols among the system actors. The presentation in the form of protocols, using sequence diagrams, presents in a didactic way how the decision algorithms (Section 5.2) communicate with the PEs.

### 5.1.1 Application Admission

The Application Admission protocol selects the cluster with free resources to receive an incoming application. Concerning the hierarchy, Application Admission is a system-level actuation knob/tool being described in software. Thus, the Global Manager (GM) is in charge of running the Application Admission.

The Application Admission is fired by a request sent by the Application Injector (Figure 3.1(a)). The GM runs the decision algorithms based on the system temperature because it receives from TEA the temperatures of all PEs. Figure 5.2 shows the sequence diagram with the messages exchanged in the reference platform to enable temperature-aware application admission and task mapping.

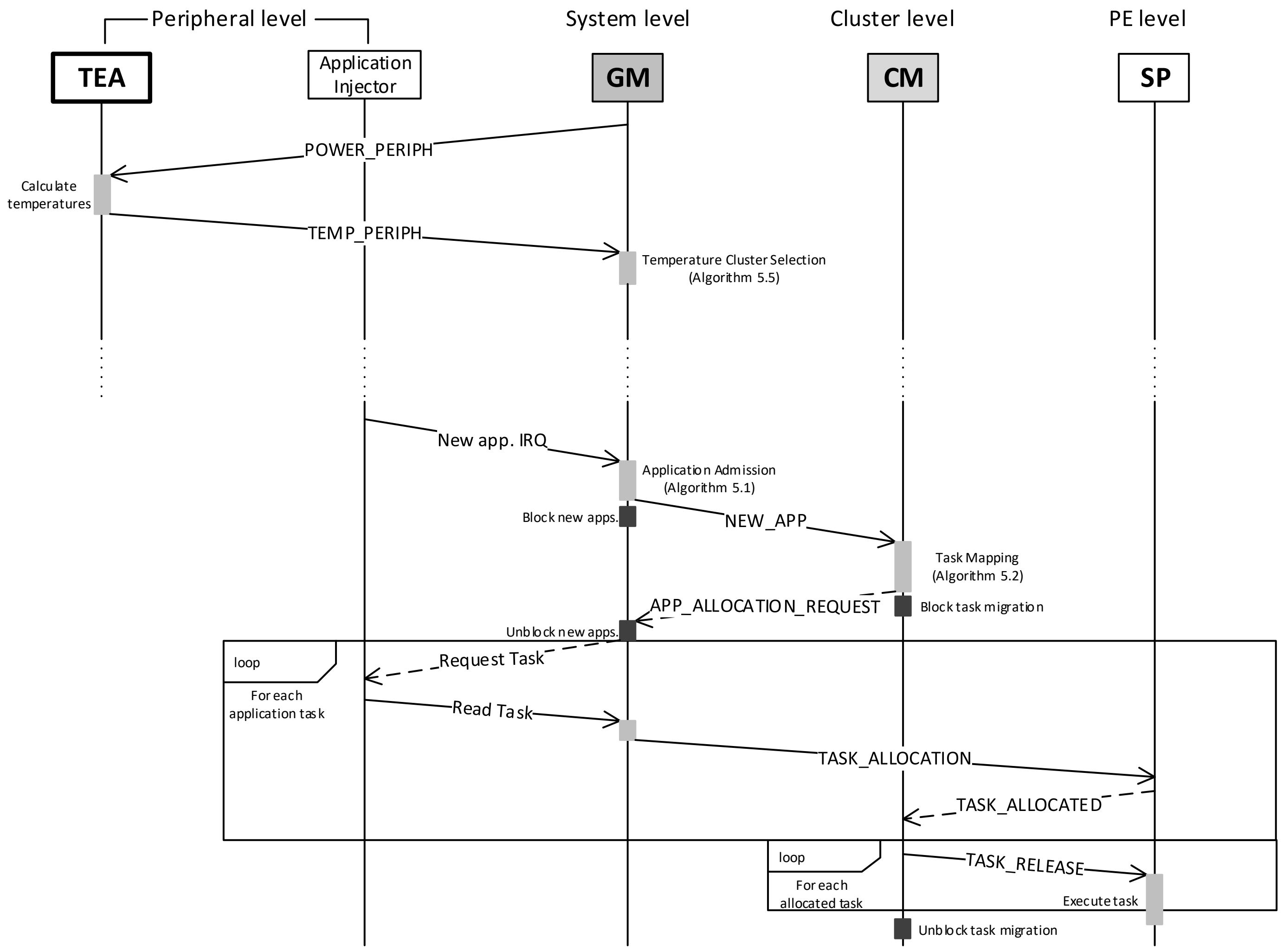


Figure 5.2 – Application Admission and Task Mapping protocol.

The first set of events in Figure 5.2 corresponds to the periodic packet exchange between GM and TEA. GM sends a `POWER_PERIPH` packet with the current power dissipation of all PEs, and TEA answers with a `TEMP_PERIPH` packet with the current temperature of all PEs. The GM may use the temperatures to give clusters priority to execute new applications (Algorithm 5.5).

Differently from the periodic GM↔TEA communication, the Application Injector can request the execution of a new application at any moment, transmitting a `New app` interruption to the GM.

Once the GM accepts the interruption, it executes the application admission procedure (Algorithm 5.1), deciding if the system can execute the incoming application and selecting a cluster to execute it. Next, the GM sends a `NEW_APP` packet to the selected cluster manager (CM) with the application task graph. The CM runs the task mapping algorithm (Algorithm 5.2), deciding where to map the application tasks, and sends an APP_ALLOCATION_REQUEST packet to the GM to start the task mapping procedure.

### 5.1.2 Task Mapping

Task Mapping corresponds to the allocation of the application tasks to the selected PEs in a cluster. Task mapping algorithm runs at the CMs, so it is considered a cluster level actuation since it primarily affects the cluster that receives the application. However, as depicted in Figure 5.2, the SP receives the task to be allocated directly from the GM to reduce the number of packets traversing the network. The GM receives the task object code, and transmits it to the correct PE.

The task mapping protocol starts when the decision algorithm (task mapping algorithm) finishes. The CM sends an APP_ALLOCATION_REQUEST packet to the GM, with a set of tuples $\{t_i, add_i\}$, where $t_i$ is the task identifier, and $add_i$ the task address computed by the mapping algorithm. For each $t_i$, the GM request to the application injector the object code (Request_Task), transmitting it to the PE through a TASK_ALLOCATION packet. Each PE that receives a task sends a TASK_ALLOCATED packet to its CM, signalizing that it is ready to start the task execution. After mapping all application tasks, the CM sends a TASK_RELEASE packet to all PEs with the application tasks to start execution.

### 5.1.3 Migration

The task migration is a cluster level actuation where the goal is to move a task from a source SP ($SP_{src}$) to a target SP ($SP_{tgt}$). The task migration employs a low latency protocol for many-cores with distributed memory [Ruaro and Moraes, 2017]. The task migration protocol requires neither checkpoints nor task code replication and allows task migrations in parallel.

The CM triggers the task migration algorithm after the reception of a temperature violation packet from the GM (TEMP_CLUSTER packet). The migrated tasks can only migrate again after a period (currently set to 20 monitoring windows). Such restriction has two goals: (*i*) avoid a ping-pong effect with unnecessary migrations; (*ii*) amortize the migration cost.

Figure 5.3 presents the task migration protocol. When the task migration algorithm decides to migrate a task from $SP_{src}$ to $SP_{tgt}$, it sends a TASK_MIGRATION packet to $SP_{src}$ to start the migration process. First, $SP_{src}$ sends the static part of the task code $SP_{tgt}$ (MIGRATE_TEXT packet). Next, $SP_{src}$ save the context of the task and sends the dynamic parts of the task in two packets: (*i*) MIGRATE_STACK, which contains the values of the task stack; (*ii*) MIGRATE_BSS_DATA, which contains the necessary values for the $SP_{tgt}$ to restore the task context. Finally $SP_{tgt}$ sends a `TASK_MIGRATED` packet to the CM, signalizing the end of the task migration protocol, and continues the execution of the received task.

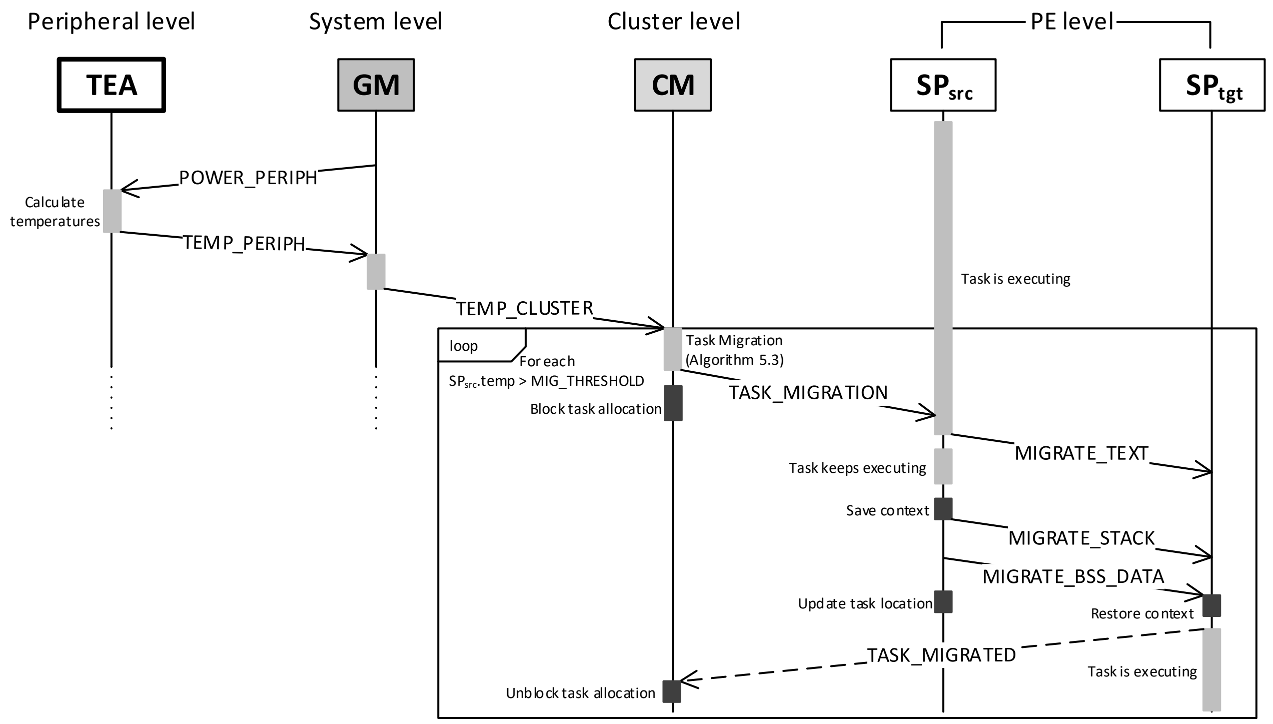


Figure 5.3 – Task migration protocol.

### 5.1.4 Dynamic Voltage and Frequency Scaling

Dynamic voltage and frequency scaling (DVFS) consists of changing the clock frequency and the supply voltage levels according to the PEs temperature to guarantee that the temperatures are not higher than the critical core temperature. DVFS is the most commonly used DTM mechanism to manage the temperature in many-core systems [Mohammed et al., 2020].

The main overhead to enable DVFS in a per-core grain is the voltage regulators and the PLLs to control the supply voltages and the generated frequencies. To evaluate the impact of using DVFS in the reference platform, we generated the power estimation described in Section 3.4 for the three voltage levels available in the adopted 28nm standard cells library: 1.0V, 0.9V, and 0.8V. Table 5.1 shows the maximum frequency reached at the logical synthesis for each voltage in the reference platform and the three VF levels used in our simulations.

The frequency scaling occurs in the processor, memory and in part of the DMNI. The router and the portion of the DMNI that communicates with the router runs always at

Table 5.1 – DVFS levels used in the reference platform. Library C28SOI_SC_12 (28nm), 1.0V@1GHz, 25ºC.

| DVFS level | Voltage | Max. Frequency |
|---|---|---|
| 1 | 1.0V | 1,000 MHz |
| 2 | 0.9V | 667 MHz |
| 3 | 0.8V | 500 MHz |

nominal frequency, even when the voltage is changed [Martins et al., 2016]. These components were synthesized in lower voltages with the frequency of 1GHz and the system still met the timing constraints. The reason to keep the router working at the nominal frequency is to avoid increased network latency between PEs running at nominal voltage and frequency.

### 5.1.5 Power states

The Advanced Configuration and Power Interface (ACPI) Specification [UEFI, 2021] defines the power state of system processors while in the working state as being either active (executing) or sleeping (not executing). Processor power states include C0, C1, C2, C3, . . . Cn. The C0 power state is an active power state where the CPU executes instructions. The C1 through Cn power states are processor sleeping states where the processor consumes less power and dissipates less heat than leaving the processor in the C0 state. While in a sleeping state, the processor does not execute any instructions. Each processor in sleeping state has a latency associated with entering and exiting that corresponds to the power savings. In general, the longer the entry/exit latency, the greater the power savings when in the state.

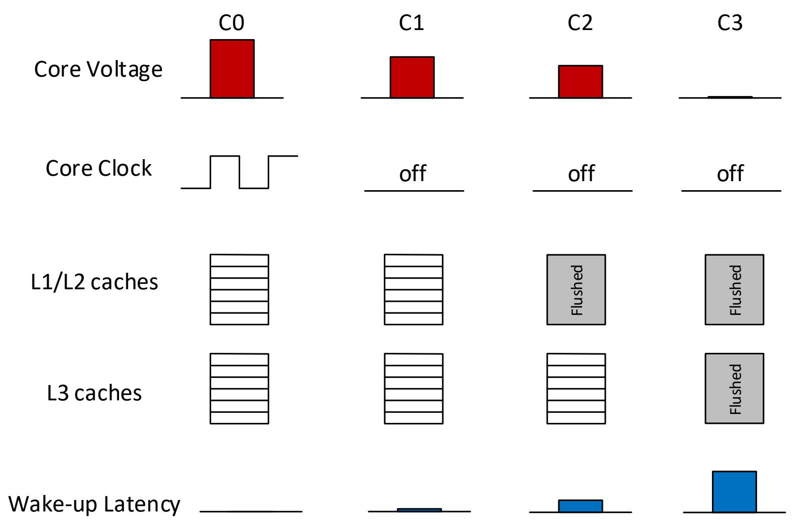


Figure 5.4 – CPU Power States Illustration. Adapted from [Mohammed et al., 2020].

In the reference architecture, we adopt C0 and C1, mainly to reduce the migration cost and due to the memory organization. The PE memory needs to stay powered since the DMNI uses it to receive packets from the network. If a packet is received during a C1 state, the DMNI interrupts the processor setting its state to C0.

## 5.2 Decision Policies

This section presents the decision policies showing how the actuation tools presented in the previous section are applied in the reference platform. While this section present general algorithms for admission, mapping and migration, next section (Section 5.3) presents the heuristics to define the priorities used in this section.

### 5.2.1 Admission

When the Application Injector (Figure 3.1(a)) interrupts the GM requesting the admission of a new application into the system, the first action is to select a cluster to execute the application, using a procedure called *application admission*. The GM selects a cluster based on a previously generated priority (*cluster_prio*), and the availability of resources in the cluster. Algorithm 5.1 details the application admission procedure.

**Algorithm 5.1** Application Admission (executed at system-level – GM)

```
 1: Inputs: app, cluster_prio
 2: Outputs: $cl_{output}$
 3: $cl_{output} \leftarrow \emptyset$
 4: $admitted \leftarrow false$
 5: for each $cl_{candidate} \in cluster\_prio$ do
 6:     $cl_{candidate} \leftarrow cluster\_prio.top()$
 7:     if $cl_{candidate}.\#free\_resources \geq app.\#tasks$ then
 8:         $cl_{output} \leftarrow cl_{candidate}$
 9:         $admitted \leftarrow true$
10:     end if
11:     if admitted then
12:         return $cl_{output}$
13:     end if
14:     cluster_prio.pop()                              ▷ next Cluster in cluster_prio
15: end for
16: return $cl_{output}$
```

Algorithm 5.1 receives as inputs: (*i*) the application task graph description – *app*; (*ii*) the set of clusters ordered by priority – *cluster_prio*. The algorithm starts by initializing the selected cluster as empty, $cl_{output}$, and sets the *admitted* flag as false (line 3-4). Next, all clusters are verified to check the availability of *resources* to receive the application (line 7-10). A *resource* is a memory page assuming a paged memory system. If there are no clusters with enough resources, the algorithm returns an empty value (line 16). Otherwise, the cluster with the highest priority with enough resources is assigned as the output of the algorithm (lines 6-13).

If it is not possible to execute the application with the available resources in the clusters ($cl_{output} = \emptyset$), the MP may borrow resources from another cluster (a process named reclustering [Castilhos et al., 2013]), increasing the cluster size, re-executing the application admission. In the last case, the application is enqueued for later execution.

### 5.2.2 Mapping

The next step after the application admission is to map the application tasks in the selected cluster. Algorithm 5.2 receives as inputs: (*i*) *app*; and (*ii*) *mapping_prio* – a list of SPs belonging to a given cluster sorted according to one of the prioritizing algorithms (Section 5.3).

At line 2, Algorithm 5.2 sorts the application's tasks according to the communication dependence, i.e., traversing the CTG (communicating task graph) from initial tasks up to final ones. The main loop (lines 3-13) iterates on every task of the application. Line 6 selects the SP with the highest mapping priority. If the SP has resources to receive the task (line 7), the task is mapped into the SP (line 8). Otherwise, the search for resource goes to the next PE in *mapping_prio* (line 11). Note that Algorithm 5.2 always finds an SP for task mapping, because the Application Admission (Algorithm 5.1) ensured resources availability.

**Algorithm 5.2** Task Mapping (executed at cluster-level – GM or CM)

```
 1: Inputs: app, mapping_prio
 2: sortTasks(app)                          ▷ sort app tasks according to the comm. dependence
 3: for each task ∈ app do
 4:     mapped ← false
 5:     while task is not mapped do
 6:         SP ← mapping_prio.top()
 7:         if SP.#free_resources > 0 then
 8:             map_task(task, SP)
 9:             mapped ← true
10:         end if
11:         mapping_prio.pop()                                    ▷ next PE in mapping_prio
12:     end while
13: end for
```

### 5.2.3 Migration

When the MP receives a packet with temperature data from thermal monitoring, Algorithm 5.3 evaluates the need to migrate tasks. In general, a task migration occurs when a given SP is above a temperature threshold. If the cluster presented a recent task event (as task mapping or task migration), migration is temporarily blocked, due to the reasons pointed-out in Section 5.1.3.

**Algorithm 5.3** Task Migration (executed at cluster-level – GM or CMs)

```
1: Inputs: $mapping\_prio$, $cl$
2: for each $SP_{src} \in cl$ do
3:     if $SP_{src}.temp > MIG\_threshold$ then
4:         for each $task \in SP_{src}.tasks$ do
5:             $migrated \leftarrow false$
6:             while $task$ is not $migrated$ do
7:                 $SP_{tgt} \leftarrow mapping\_prio.top()$
8:                 if $SP_{tgt}.\#free\_resources > 0$ then
9:                     $migrate\_task(task, SP_{src}, SP_{tgt})$
10:                    $migrated \leftarrow true$
11:                end if
12:                $mapping\_prio.pop()$                    ▷ next PE in $mapping\_prio$
13:            end while
14:        end for
15:    end if
16: end for
```

The inputs of Task Migration, Algorithm 5.3, are: (*i*) mapping priority – *mapping_prio*; (*ii*) the monitoring data of SPs belonging to the cluster – *cl*, obtained from TEA. Algorithm 5.3 starts a search in all SPs of the cluster (line 2). If there are SPs with temperature above the *MIG_threshold* (line 3), all tasks running on the SPs, denoted as $SP_{src}$, must migrate to another SP, denoted as $SP_{tgt}$ (lines 4-15). The search of $SP_{tgt}$ to receive the migrated task from $SP_{src}$ (lines 6-13) follows *mapping_priority* order, similarly to the mapping algorithm. If $SP_{tgt}$ has resources to receive a new task (line 8), the task is migrated from $SP_{src}$ to $SP_{tgt}$ (line 9). Otherwise, the search for a $SP_{tgt}$ goes to the next PE in *mapping_prio* (line 12).

It is possible to have SPs with the temperature above the threshold not executing tasks. This happens due to the thermal conductance, or due to a previous state of tasks running on it. Satisfying the conditions defined at line 4, all tasks running on this SP migrate (line 5). For each task, an SP is picked from the *mapping_priority* vector (line 8), and if possible, migrated to it (line 10).

### 5.2.4 Dynamic Voltage and Frequency Scaling

The DVFS algorithm, as the migration, is triggered at the reception of a temperature packet from TEA. However, unlike the migration decision, the DVFS may occur at any time, and is not blocked after one actuation. Basically, the decision to be made is to lower the VF level, according to Table 5.1, when the temperature of the PE is above a predefined threshold, or set back to nominal VF if the PE is not already operating at nominal VF and it is below the threshold temperature. Algorithm 5.4 shows how the DVFS decision is implemented in the reference architecture.

The algorithm receives as inputs the DVFS levels of each PE and generates as outputs the new DVFS levels based on the received temperatures. If the temperature of a PE is above the threshold (line 5), the algorithm certifies that the PE is not already at the lowest VF level (line 6), and reduce the VF level of the PE (line 7). At line 8 the CM calls the function that actually sends the `DVFS_CHANGE` message to the target SP with its new DVFS level. When the temperature is below the threshold, the algorithm verifies if the PE is not running at nominal VF (line 10), and sets the DVFS to 1 (lines 11-12).

**Algorithm 5.4** DVFS (executed at cluster level – GM or CMs)

```
 1: Inputs: DVFS_levels
 2: Outputs: DVFS_levels
 3: cluster_temps ← receive_packet(TEMP_PERIF)
 4: for each $SP_i$ ∈ cluster_temps do
 5:     if $SP_i$.temp > DVFS_threshold then
 6:         if $DVFS\_levels_i < 3$ then
 7:             $DVFS\_levels_i = DVFS\_levels_i + 1$
 8:             setCurrentVF($SP_i$, $DVFS\_levels_i$)
 9:         end if
        else
10:         if $DVFS\_levels_i != 1$ then
11:             $DVFS\_levels_i = 1$
12:             setCurrentVF($SP_i$, $DVFS\_levels_i$)
13:         end if
14:     end if
15: end for
```

### 5.2.5 Power States

In our implementation, we consider only the C0 and C1 power states for the reference platform. C0 for the PEs that are executing tasks and C1 for the PEs with idle processors. The idle PEs must be able to quickly receive new or migrate tasks, so the local memory needs to stay powered up, and the wake-up latency needs to be short. For this reason, when estimating the power of an idle PE, we consider the CPU leakage power and the router power consumption since the router is always on.

## 5.3 Decision Heuristics for Cluster and PE Selection

Previous Section described algorithms to map and migrate tasks relying on inputs called *cluster_prio* and *mapping_prio*. We define the strategies to prioritize clusters and PEs (*cluster_prio*, *mapping_prio*) described in the following Subsections.

Figure 5.5 shows the relationship between the algorithms presented in the previous section and the heuristics algorithms presented in this section. There are two events that trigger the execution of algorithms: (*i*) `New app` event; (*ii*) `Monitoring` event.

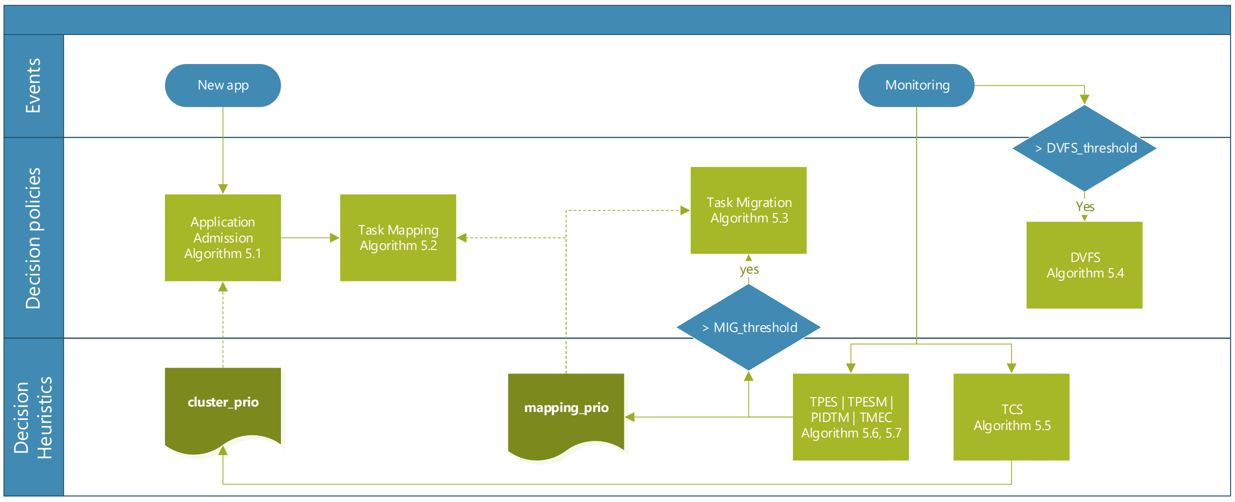


Figure 5.5 – Relationship between the algorithms. Continuous lines: event that triggers an action. Dashed lines: input for a given algorithm.

The `Monitoring` event is periodic and carries the PEs temperatures. A `Monitoring` event may trigger the DVFS algorithm if a temperature is above the *DVFS_threshold*. The `Monitoring` event also trigger the temperature cluster selection algorithm (TCS), which generates the *cluster_prio* queue to be used by the application admission algorithm. This event also triggers one of the PE selection heuristics, that generates the *mapping_prio* queue, used by the task mapping and task migration algorithms. If one of the temperatures is above *MIG_threshold*, the heuristics may trigger the execution of the task migration algorithm, which uses the *mapping_prio* previously generated by the heuristic being executed.

A `New app` event can happen at any time, being triggered by the application injector when a new application requests execution. This event triggers the execution of the application admission and the task mapping algorithm. The application admission algorithm uses the *cluster_prio* queue, and the task mapping algorithm uses the *mapping_prio* queue, both generated after a `Monitoring` event.

### 5.3.1 Temperature Cluster Selection

The reference algorithm for cluster selection [Martins et al., 2019a] chooses the cluster with the largest number of free resources, prioritizing clusters further away from the GM to avoid communication congestion.

The cluster selection occurs in the GM, which receives the temperatures of all PEs. In our temperature cluster selection (TCS) algorithm, the GM computes the average temperature of the clusters and prioritizes the cluster with the lowest temperature, with enough resources for the application. Algorithm 5.5 details the TCS algorithm. Algorithm 5.5 receives as input the cluster set ($cl_{set}$) with the average temperature computed for each cluster, and generate as output the cluster priority queue for admitting new applications (*cluser_prio*), used in Algorithm 5.1. The algorithm starts by creating an empty set for the cluster priority queue (line 3) and creating an auxiliary cluster set ($cl_{aux}$) (line 4). For each cluster in $cl_{set}$ (line 5), the algorithm finds the coolest cluster in $cl_{aux}$ (lines 6-12) and then inserts the selected cluster in *cluser_prio* (line 13) and remove the cluster from $cl_{aux}$ (line 14).

**Algorithm 5.5** Temperature Cluster Selection (executed at system level – GM)

```
1: Inputs: $cl_{set}$
2: Outputs: $cluser\_prio$
3: $cluser\_prio \leftarrow \emptyset$,
4: $cl_{aux} \leftarrow cl_{set}$
5: for each $cl_i \in cl_{set}$ do
6:     $coolestTemp \leftarrow \infty$
7:     for each $cl_j \in cl_{aux}$ do
8:         if $cl_j.temp < coolestTemp$ then
9:             $coolestTemp \leftarrow cl_j.temp$
10:            $cl_{coolest} \leftarrow cl_j$
11:        end if
12:    end for
13:    $cluster\_prio.push(cl_{coolest})$
14:    $cl_{aux}.remove(cl_{coolest})$
15: end for
16: return $cluster\_prio$
```

### 5.3.2 Temperature PE Selection and Temperature PE Selection with Migration

The temperature PE selection (TPES) is an extension of the TCS strategy. After selecting the coldest cluster, the priority for tasks mapping within the cluster is given to the coldest PEs. Temperature PE Selection with Migration (TPESM) algorithm is a straightforward strategy that prioritizes the coldest PEs for mapping and migrates tasks from time to time to provide better thermal distribution.

Algorithm 5.6 presents the main steps executed in TPESM heuristic. Each time an MP receives a packet with temperature data (line 1), the procedure *generateMapPrio*() creates a vector that sorts PEs by its temperatures (line 2). The procedure *generateMapPrio*() is a generic function that receives scores and generates a vector of PE indexes ordered by the received scores, used by all proposed heuristics.

**Algorithm 5.6** TPESM (executed at cluster-level – GM or CMs)

```
1: $cluster\_temps \leftarrow$ receive_packet($TEMP\_CLUSTER$)
2: $mapping\_prio \leftarrow$ generateMapPrio($cluster\_temps$)
3: for each $cluster\_temps_i \in cluster\_temps$ do
4:     if $cluster\_temps_i > MIG\_threshold$ then
5:         migration($mapping\_prio, cluster\_temps$)          ▷ Algorithm 5.3
6:         break
7:     end if
8: end for
```

The execution of algorithm 5.6 is triggered by the reception of a temperature packet by a manager PE and the migration procedure runs every time a new temperature data arrives and at least one of the temperatures is above *MIG_threshold* (lines 3-8). However, the application admission and the task mapping procedures (Algorithms 5.1 and 5.2) are triggered by the injection of a new application in the system. In this case, the priority used in algorithm 5.2 is the last one generated when receiving a temperature packet.

### 5.3.3 Proportional, Integral and Derivative Temperature Management

TPESM is a strategy to balance the temperature distribution in a system and has been used by multi-core architectures with available temperature sensors [Rao and Vrudhula, 2008, Coskun et al., 2008]. However, some mapping decisions using only the instant temperature value can impact the thermal distribution of the system. For example, if a task is mapped in a PE with the lowest temperature but it is increasing, the resulting peak temperature can be higher than if mapped in other PE with decreasing temperature. For this reason, we proposed the Proportional, Integral, and Derivative Temperature Management (PIDTM). The Proportional, Integral and Derivative temperature management should not be confused with PID from control theory, where the objective is to set the value of a signal equal to a reference value. In our algorithm, the objective is to avoid hotspots.

The *Proportional* value is related to the instantaneous temperature of a cluster or a PE, the *Integral* value is the average temperature value of a predefined number of monitoring windows, and the *Derivative* value is the tendency of the temperature value, and means how fast the temperature is changing.

Algorithm 5.7 shows the PIDTM score computation to generate the mapping priority vector. The *time_idx* entry corresponds to the number of times the function was called, increased after its execution. For each SP, there is a circular buffer, named *integral_buf*, which stores the last INT_WINDOW temperature samples. Every time a new temperature sample arrives, the algorithm computes the PID score for each SP in the cluster. Lines 4 and 5 replace the last temperature value in the circular buffer by the current SP temperature, while line 6 computes the average SP temperature in the last INT_WINDOW intervals. The derivative value is the subtraction of the previous temperature from the new temperature value (line 7). The final PIDTM score is the sum of the proportional value, which is the current temperature, the integral value and the derivative value, each one multiplied by a constant that defines the weight of each component of the sum in the final score (line 8). *KP*, *KI*, and *KD* are proportional, integral, and derivative constants, respectively. It is possible to tune those constants according to the control objectives. Line 11 generates the mapping priority, used in task mapping and task migration procedures, as shown in algorithms 5.2 and 5.3. Finally, if there is any temperature above *MIG_threshold*, the migration algorithm is executed (lines 12-17).

**Algorithm 5.7** PIDTM (executed at cluster-level – GM or CMs)

```
 1: Inputs: $time\_idx$
 2: $cluster\_temps \leftarrow$ receive_packet($TEMP\_CLUSTER$)
 3: for each $SP_i \in cluster\_temps$ do
 4:     $last \leftarrow time\_idx$ mod $INT\_WINDOW$
 5:     $integral\_buf_i(last) \leftarrow SP_i.temp$
 6:     $integral_i \leftarrow \frac{\sum_{n=1}^{INT\_WINDOW} integral\_buf_i(n)}{INT\_WINDOW}$
 7:     $derivative_i \leftarrow SP_i.temp - temperature\_prev_i$
 8:     $pid\_score_i \leftarrow KP * SP_i.temp + KI * integral_i + KD * derivative_i$
 9:     $temperature\_prev_i \leftarrow SP_i.temp$
10: end for
11: $mapping\_prio \leftarrow$ generateMapPrio($pid\_score$)
12: for each $cluster\_temps_i \in cluster\_temps$ do
13:     if $cluster\_temps_i > MIG\_threshold$ then
14:         migration($mapping\_prio$, $cluster\_temps$)        ▷ Algorithm 5.3
15:         break
16:     end if
17: end for
```

### 5.3.4 Temperature Management With Energy Constraint

The main drawback of TPESM and PIDTM strategies is the risk of always selecting the same PEs to provide a balanced temperature distribution in the system. Thus, a balanced temperature distribution may not always generate a balanced workload across all PEs in a cluster, which may stress some PEs while others are left unused. For this reason, we proposed the Temperature Management With Energy Constraint (TMEC) strategy, which takes the energy consumption of each PE to generate the score of mapping priority.

In this strategy, we add the energy consumption to the PIDTM score (Algorithm 5.7, line 8), since PIDTM tends to generate a better temperature distribution when compared with TPESM. The same procedures of task mapping and migration are used (Algorithms 5.2 and 5.3), based on the calculated TMEC score, to keep a uniform thermal distribution during the execution of applications while balancing the workload between all PEs.

## 5.4 Results

This section presents the results of the proposed heuristics using the reference architecture described in RTL (Subsections 5.4.2 – 5.4.4) and OVP (Subsection 5.4.5). Before presenting the results, Subsection 5.4.1 presents the experimental setup used in the evaluations.

### 5.4.1 Experimental Setup

A set of benchmarks, six available in the Memphis platform [Ruaro et al., 2019] and one developed during this work, were used to evaluate the heuristics:

- Adaptation of three applications to receive new inputs and execute different number of iterations: (*i*) Dijkstra (DIJ), with 7 tasks; (*ii*) audio and video application (AV), with 7 tasks, that implements a video and audio decoding pipeline in parallel; (*iii*) AES encoder, with 5 tasks.
- Three applications without modifications: (*i*) MPEG decoder (5 tasks); (*ii*) DTW - *Digital Time Warping* (6 tasks); communication intensive synthetic application (SYN) (6 tasks).
- New *sort* application, with 4 tasks, that implements a parallel quick sort algorithm.

Five scenarios were created to evaluate different thermal management characteristics. Table 5.2 presents the applications used in each scenario. In the first three scenarios (HW, AW, and LW) the workload remains constant throughout the simulation and the difference between scenarios is the percentage of system utilization. The high workload (HW) scenario execute the application set that occupy 75% of system resources. Similarly, the low workload (LW) scenario maintains 33% of resource utilization, and the average workload (AW) scenario keeps 50% of resource utilization through the entire simulations. The last two scenarios (DW1, DW2) consist of typical dynamic workload scenarios, which present periods of high workload and low workload. Although the larger number of tasks in dynamic workloads than the remaining ones, the number of running tasks varies in dynamic workloads scenarios to reproduce peaks and valleys of system utilization while in HW, LW, and AW scenarios, the number of running tasks is constant during the simulation.

Table 5.2 – Thermal evaluation scenarios.

| Scenario | | Applications | | | | | | | #Tasks |
|---|---|---|---|---|---|---|---|---|---|
| | | MPEG | DTW | AES | SYN | DIJ | Sort | AV | |
| **HW** | High Workload | 1 | 2 | 0 | 1 | 1 | 2 | 1 | 47 |
| **LW** | Low Workload | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 18 |
| **AW** | Average Workload | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 36 |
| **DW1** | Dynamic workload 1 | 1 | 2 | 10 | 4 | 4 | 2 | 2 | 142 |
| **DW2** | Dynamic workload 2 | 2 | 6 | 10 | 4 | 4 | 6 | 2 | 192 |

These scenarios were executed in a 8x8/6x6 systems, partitioned in 4 4x4/3x3 clusters using the reference architecture (Section 3.1), described in the clock cycle-accurate and instruction-level models (Section 3.3). We compare our thermal management strategies with a patterning mapping and with a Multi-Objective Resource Management (MORM) approach

[Martins et al., 2019a]. Recent works on DTM use the patterning approach [Yang et al., 2017b, Liu et al., 2018] in a known set of applications, without the support of remapping by task migration. MORM manages applications without considering temperature (it uses power dissipation and performance as cost functions), using task migration dynamically.

## 5.4.2 Temperature Cluster Selection – TCS

To evaluate the TCS benefits, we executed scenario DW1 from Table 5.2. Figure 5.6 shows the simulation results for a 6x6 system, where each square represents the temperature of one PE. This is the only result using a 6x6 system due to the gains observed when running TCS in a system with this size. Five temperature snapshots were taken during the simulation, represented in the *z* axis. Figure 5.6(a) presents the temperature results from MORM and Figure 5.6(b) present the results using TCS. All temperatures were extracted using TEA.

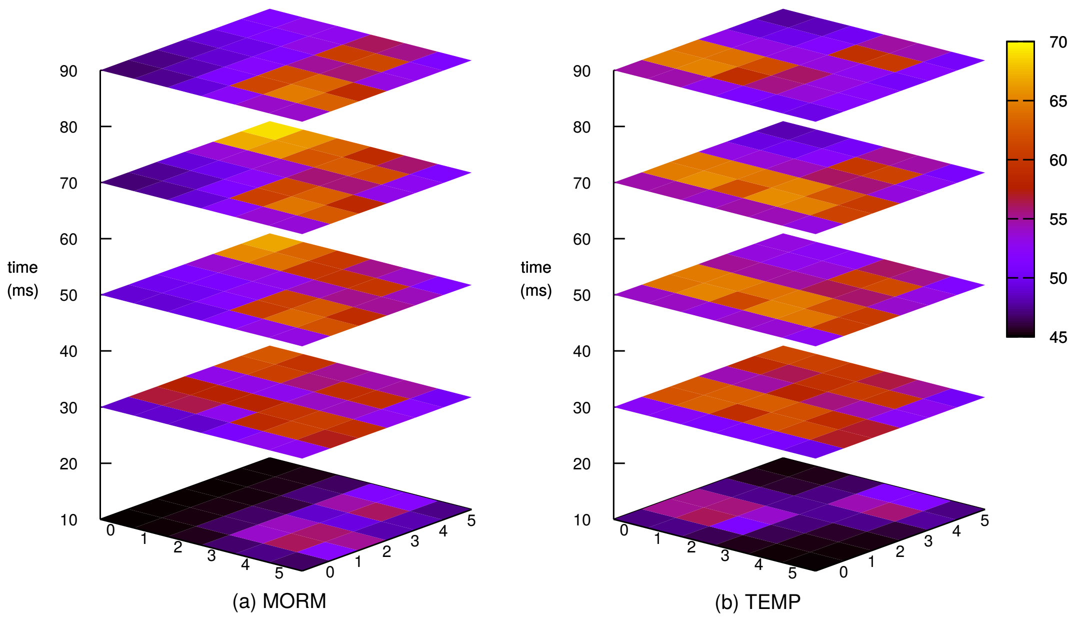
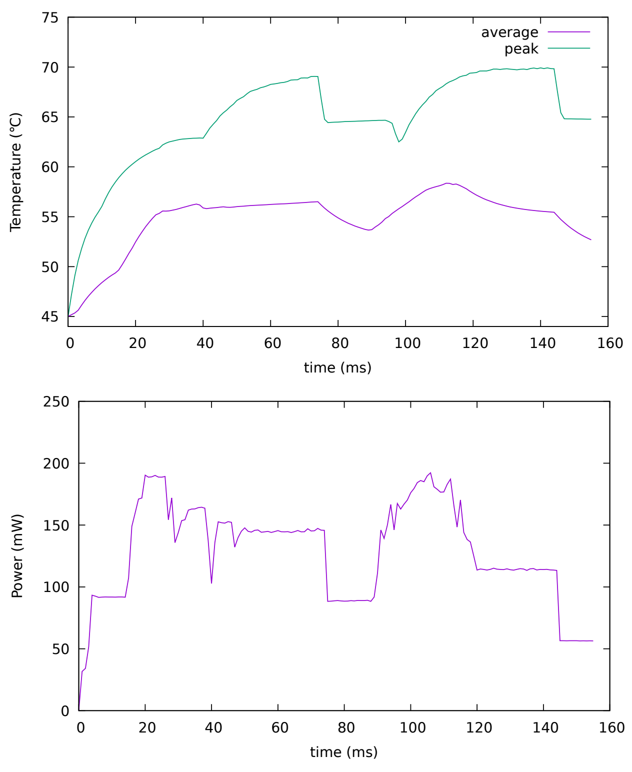


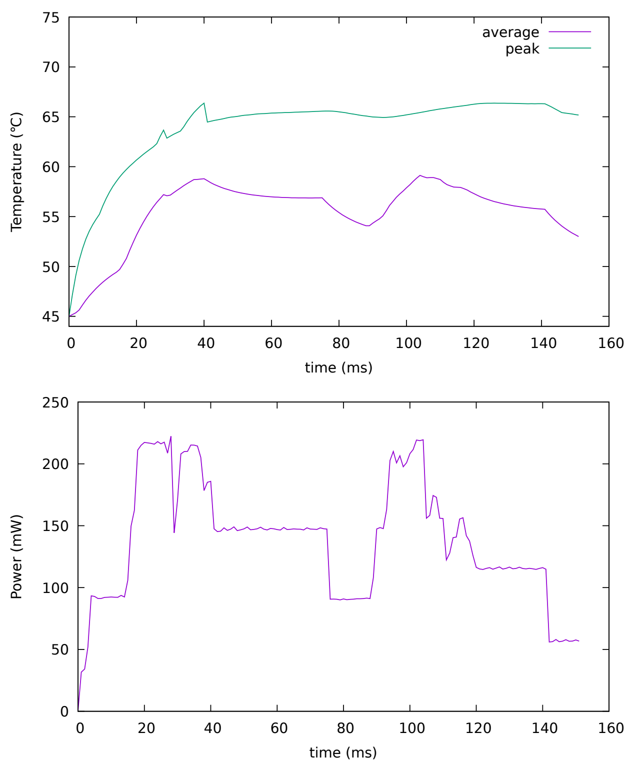
Figure 5.6 – Heatmaps extracted from scenario 1. (a) Original MORM. (b) Temperature cluster selection.

Analyzing the results, we observed that the resulting mapping of each proposal is different, resulting in distinct thermal distributions for the same scenario. MORM simulation presents a hotspot in the fourth snapshot, avoided using TCS. Using TCS, the scenario is executed 2.8% faster than MORM when using a 200mW power budget.

Figure 5.7 presents the peak and average temperature among all PEs during the simulation in the top row, and the power consumption in the bottom row. The first column shows MORM results, and the second column shows the results using the temperature cluster selection strategy. These graphs shows that using TCS, the power consumption is higher during the peaks of workload, but the peak temperature stays lower than MORM most of the time. The Figure also shows that peak temperatures among all PEs in MORM present more oscillation, resulting in thermal stress and potentially affecting the system lifetime. The total energy consumption of TCS is 4.5% higher than MORM, but the overall peak temperature is 5.3% lower. The overall peak temperature of this simulation in MORM was 69.91°C, versus 66.36°C in TCS. The average peak temperature during the simulation using MORM was 64.81°C, while in TCS, the average was 63.85°C.



(a) MORM



(b) Temperature Cluster Selection

Figure 5.7 – Peak and average temperatures (top) and power consumption (bottom): (a) MORM (200mW power budget), (b) Temeprature cluster selection.

These results show that a simple strategy, like TCS, can reduce the peak temperature of the system by providing a better thermal distribution of the running applications in dynamic workloads.

### 5.4.3 PE selection

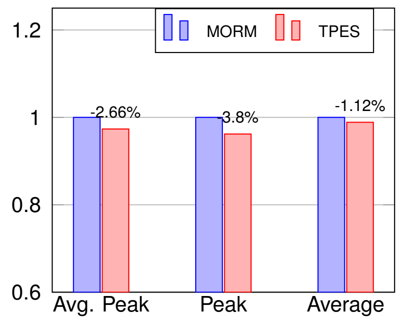
This Section evaluates the four heuristics to select a given PE for mapping and migration:

- TPES - Temperature PE selection;
- TPESM - Temperature PE selection with Migration;

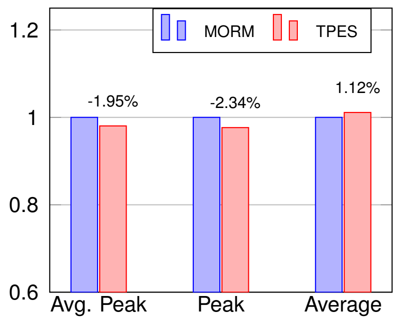
- PIDTM - Proportional, Integral and Derivative Temperature Management;
- TMEC - Temperature Management with Energy Constraint.

All heuristics were executed using scenarios DW1, HW, and LW (Table 5.2), comparing them to MORM.

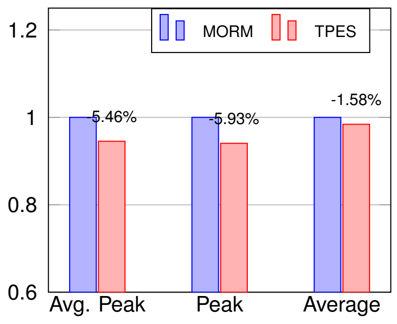
Figure 5.8 shows the results of temperature comparison and Figure 5.9 shows the results of energy consumption and execution time for each scenario, using the **TPES** heuristic.



(a) DW1

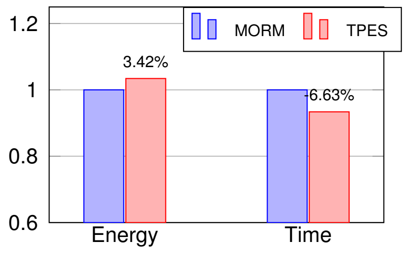


(b) HW

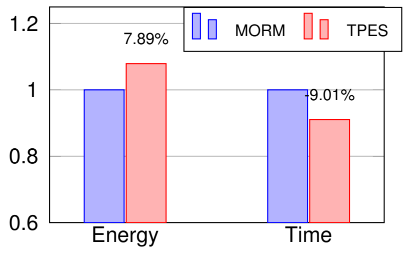


(c) LW

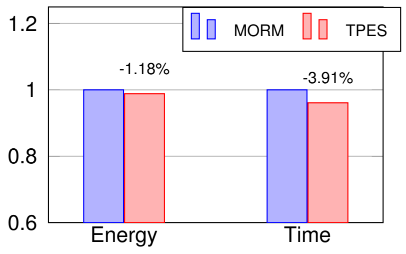
Figure 5.8 – MORMxTPES temperature results.



(a) DW1



(b) HW



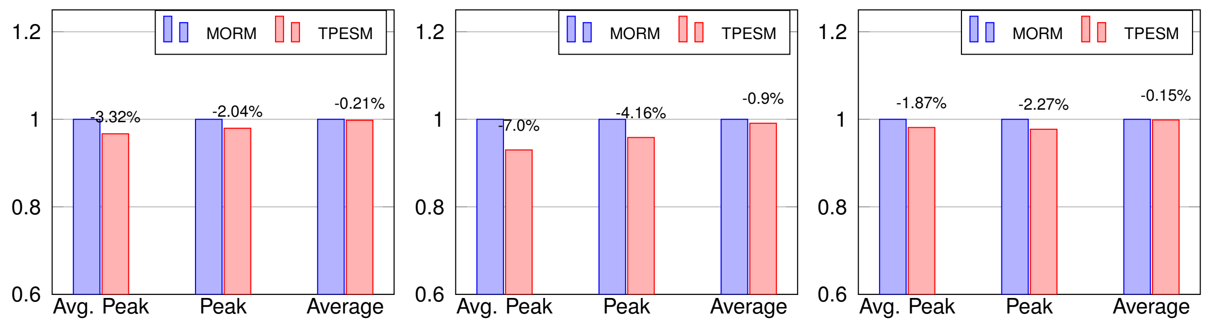
(c) LW

Figure 5.9 – MORMxTPES energy consumption and execution time.

Results show that TPES has a smaller peak temperature and average peak temperature than MORM in all simulated scenarios. TPES also achieved better performance by exchanging time for energy consumption in DW1 and HW scenarios. This happens because MORM sets VF levels to improve energy efficiency when a cluster enters in energy mode, and a cluster may enter in energy mode to enable multitasking and allow more applications to run. In TPES, the VF actuation is disabled. There is no actuation in both approaches in the LW scenario, being the difference in the results given only by the different mappings. In this case, TPES had a lower energy consumption. Thus, using TPES, it is possible to obtain better performance with lower peak temperatures due to a better heat distribution provided by the mapping algorithm.

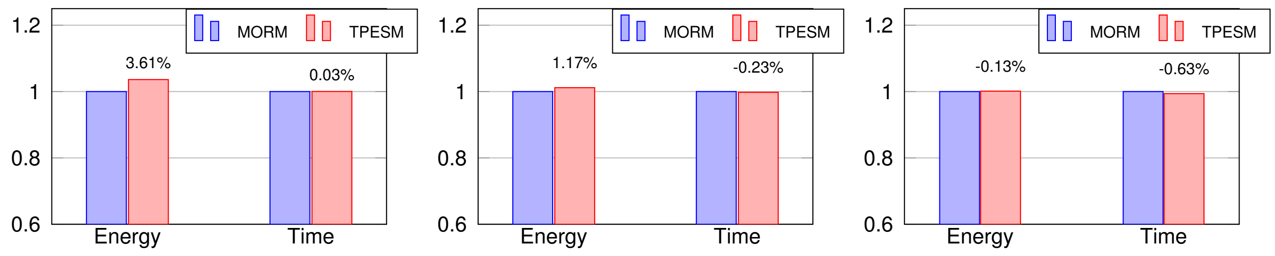
Finally, we conclude that a simple and fast heuristic, like TPES, may result in a good thermal distribution without the need to run complex algorithms such as ILP (Integer Linear Programming [Li et al., 2018]) or TSP (Thermal Safe Power [Pagani et al., 2015]) at runtime. Once an initial application is mapped, the system react and the resultant thermal behavior can give hints about the best places to map new tasks.

Figure 5.10 shows the results of temperature comparison and Figure 5.11 shows the results of energy consumption and execution time for each scenario, using the **TPESM** heuristic.



(a) DW1 (b) HW (c) LW

Figure 5.10 – MORMxTPESM temperature results.

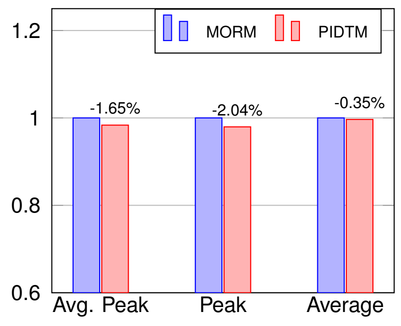


(a) DW1 (b) HW (c) LW

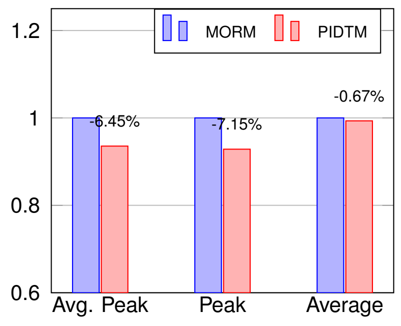
Figure 5.11 – MORMxTPESM energy consumption and execution time.

We observe that the migration introduced in TPESM reduces the average peak temperature and the overall peak temperature in the executed scenarios compared to TPES. However, the execution time benefits obtained with TPESM are negligible compared with TPES because the migration procedure introduces a time overhead. Overall, the migration is a useful tool for better thermal distribution in many-cores, especially in high workload scenarios, where temperatures are higher and can potentially reach an unsafe state.

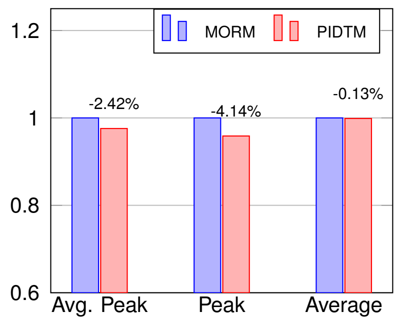
Figure 5.12 shows the results of temperature comparison and Figure 5.13 shows the results of energy consumption and execution time for each scenario, using the **PIDTM** heuristic. We observed that in DW1 the benefits of this heuristic are negligible when comparing with TPESM. However, in the HW scenario, this heuristic achieved a 7.15% reduction in peak temperature, which is significant considering it is almost 6℃ lower peak temperature when compared to MORM. This heuristic also presented a better performance in the LW scenario than TPESM, achieving 4.14% lower overall peak temperature. The execution time reduction, as observed in TPESM, is negligible due to the migrations occurred to balance the temperature distribution. However, the overhead introduced by migrating tasks was not enough to make the execution time longer than MORM, and the peak temperature reduction is significant.



(a) DW1

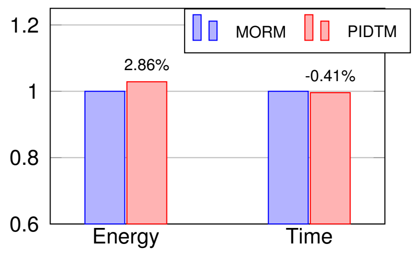


(b) HW

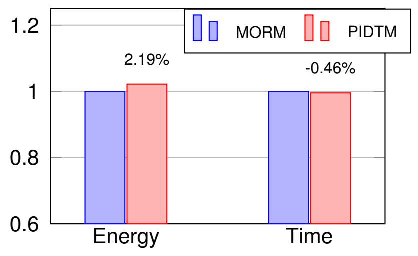


(c) LW

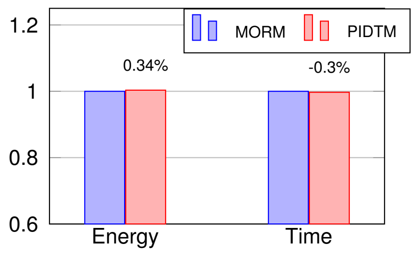
Figure 5.12 – MORMxPIDTM temperature results.



(a) DW1



(b) HW

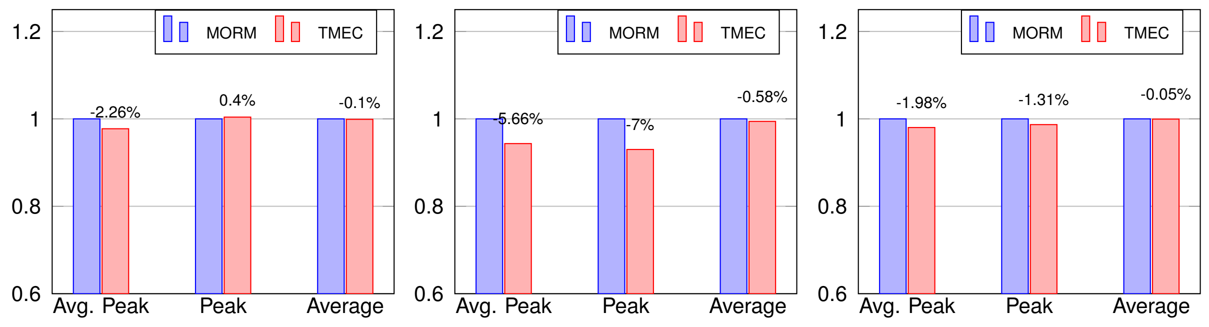


(c) LW

Figure 5.13 – MORMxPIDTM energy consumption and execution time.

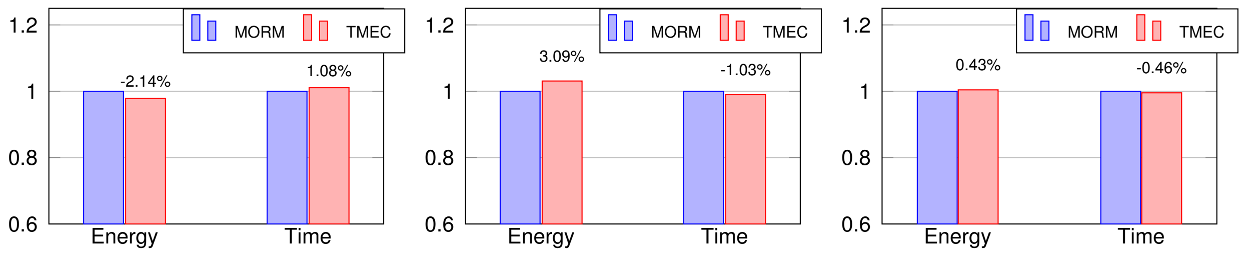
Figure 5.14 shows the results of temperature comparison and Figure 5.15 shows the results of energy consumption and execution time for each scenario, using the **TMEC** heuristic. The main goal of TMEC is to provide a balanced energy consumption between all PEs in a cluster while still looking at the temperature to provide a good temperature distribution. However, providing the optimal thermal balance is not the main objective of this heuristic.

In the DW1 scenario, we observed that TMEC produced a slightly higher peak temperature than MORM but still provided a better average peak temperature. In the HW scenario, we observed expressive improvements in the peak temperature and average peak temperature, almost as good as the PIDTM results, while providing a better energy consumption balance between PEs. We observed a lower peak and average peak temperature



(a) DW1 (b) HW (c) LW

Figure 5.14 – MORMxTMEC temperature results.



(a) DW1 (b) HW (c) LW

Figure 5.15 – MORMxTMEC energy consumption and execution time.

in the LW scenario, but not significantly as the PIDTM results. The energy and execution time results show the same trend observed in TPESM and PIDTM, except for the DW1 scenario, where the TMEC heuristic used less energy but got a slightly higher execution time. This is possible if the resulting mapping and migrations cause a greater spread of tasks belonging to the same application, which can happen when trying to balance the energy consumption of the PEs in a cluster.

### 5.4.4 Summary of the RTL DTM Results

Previous sections presented results for each heuristic individually. This Section compares the heuristics against two methods, MORM and Patterning [da Silva et al., 2020]. Results presented in this section use DW1, HW, AW, and LW scenarios (Table 5.2) and they are not the same results of the previous sections due to differences in the applications execution time and the actuation thresholds.

Figure 5.16 compares the peak temperature (obtained with TEA) for the four scenarios. The peak temperature is the highest temperature that one core achieved during execution. MORM algorithm is the reference to compare other heuristics because, in MORM, the temperature is not an input of the mapping algorithm. In the patterning approach, the temperature is also not an input of the algorithm, but this approach is usually a reference to produce a better thermal distribution.

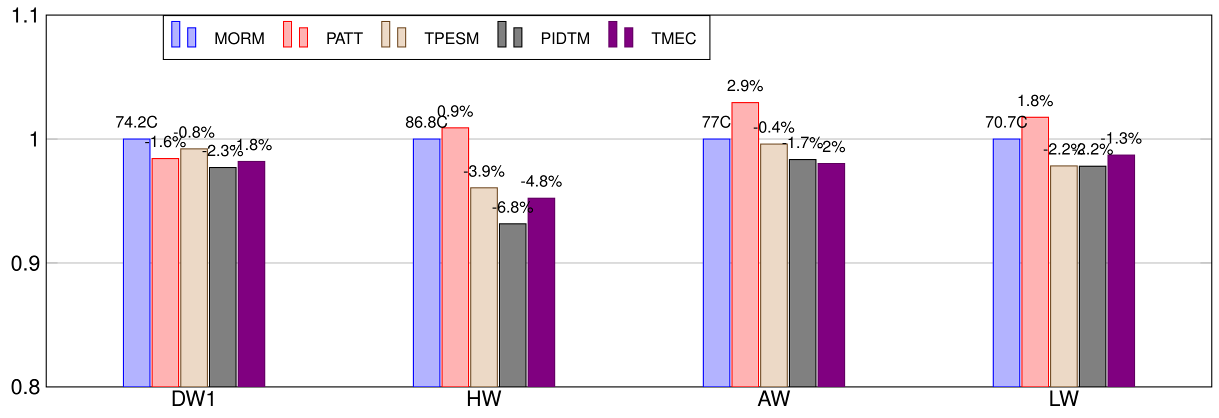


Figure 5.16 – MORM, Patterning, TPESM, PIDTM, TMEC peak temperature results [da Silva et al., 2020].

The patterning approach produces peak temperatures higher than MORM while achieving negligible improvements in overall average temperature. These results are mainly due to the hierarchical management used in MORM, which forces some physical separation between different applications among PEs.

TPESM, a simple heuristic that always chooses the coldest PE, and the proposed PIDTM and TMEC, presented improvements in the thermal distribution compared to MORM and patterning approaches. The PIDTM approach presented the best results for peak temperature, losing to TMEC only on the AW scenario. In the HW scenario, PIDTM improved 6.8% in the peak temperature relative to the MORM algorithm, while TMEC got 4.8%. PIDTM gets better results than TPESM because this heuristic considers the change rate of the temperature, i.e., if the temperature is rising or decreasing during mapping or migration, and the history of the temperature, which tends to balance the temperature.

Although PIDTM tends to present better results than TMEC, we observed that in clusters with a low workload, some PEs were left unused during the execution when using this algorithm. Then, to increase the workload distribution between all PEs, we argue that it is also important to use energy consumption as an input of the algorithm, not only the temperature. For this reason, we consider that the TMEC algorithm is a good candidate when the focus is to reduce the probability of failures caused by the temperature stress in the same PE.

Figure 5.17 presents a visual comparison between the mapping heuristics at different time snapshots, using the AW scenario. Each slice represents one snapshot of temperatures in five distinct moments of the simulation, and each colored square represents one PE. We observed that the different spacing obtained with MORM and patterning mapping approaches still produces hotspots. The proposed heuristics can react dynamically to deal with the produced hotspots, producing a better thermal distribution, as observed in the fifth slice of the figure. Figure 5.17 also shows that TMEC produced a better result for this scenario.

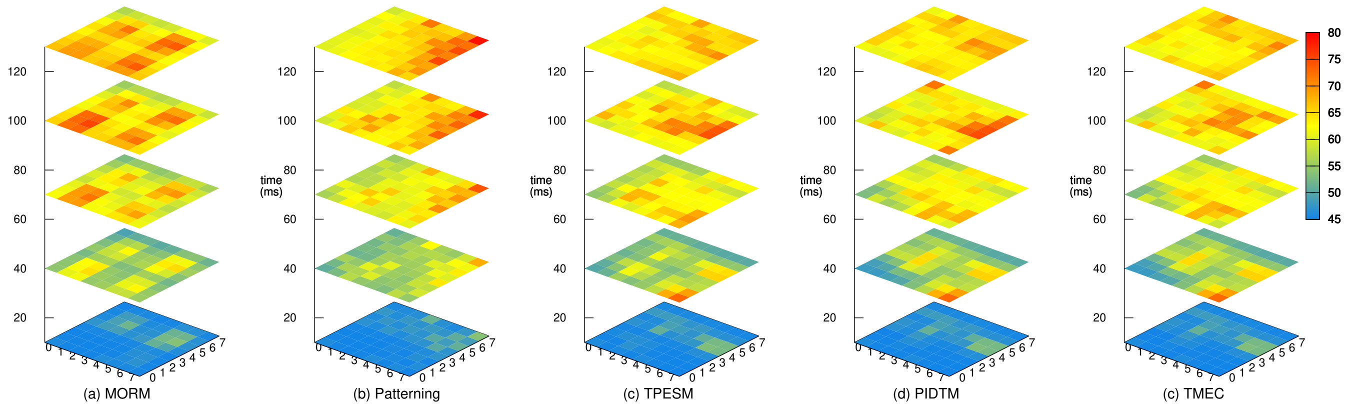


Figure 5.17 – Heatmaps comparison between different mapping and migration strategies [da Silva et al., 2020].

### 5.4.5 OVP DTM Results

This Section presents results obtained by applying the PIDTM heuristic using the instruction-level model of the reference architecture (Section 3.3.2) [da Silva et al., 2021a]. The instruction-level model allows longer simulations than the RTL model. Therefore, this experiment is important to validate our heuristic in a realistic steady-state scenario. We evaluated scenarios AW, HW, DW1, and DW2.

Figure 5.18 compares the peak temperature between the initial mapping used in MORM, patterning mapping and PIDTM. The initial MORM mapping algorithm is used because the MORM algorithm was not fully implemented in the instruction-level model. This mapping consists in a spiral line in each cluster to prioritize the PEs to receive new tasks. The initial MORM mapping (*spiral*), is the reference heuristic because the temperature is not the primary cost-function.

We observed that the patterning mapping, in some scenarios, can produce a peak temperature higher than the spiral mapping but still producing a better thermal distribution with a lower average temperature. The higher peak temperature happens because applications may have a dominant thread, which consumes more power than the other tasks. The spiral mapping usually maps applications far away from each other, while the patterning mapping may map two dominant tasks near to each other, increasing peak temperature. In contrast, the PIDTM heuristic avoids the mapping of dominant tasks near to each other. If this happens, the DTM identifies the temperature increase and migrate one of the tasks.

Using PIDTM heuristic we didn't observed any issues related with mapping two dominant tasks next to each other. This is due to the nature of the algorithm that take into account the tendency of the temperature to map and migrate tasks. And even if mapping

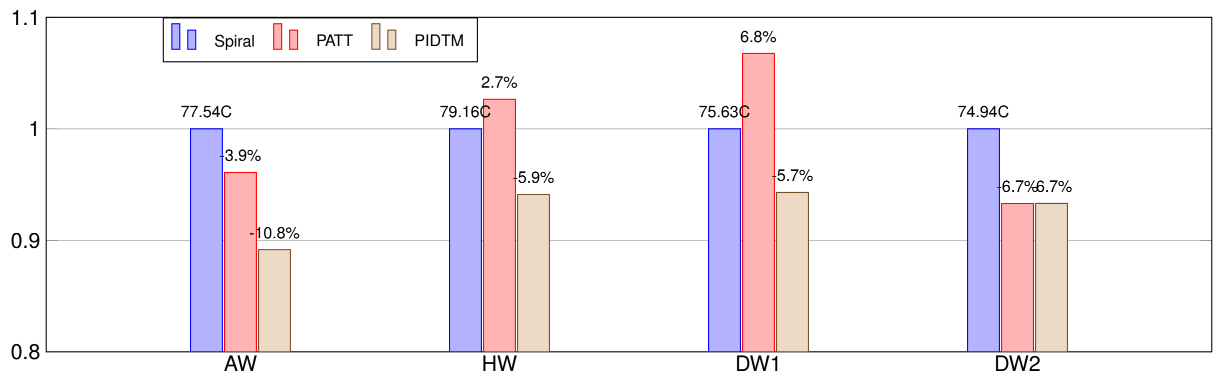


Figure 5.18 – Spiral, Patterning and PIDTM peak temperature results [da Silva et al., 2021a].

of two dominant tasks next to each other may still happen, the algorithm can quickly react, migrating the tasks further apart again.

PIDTM effectively reduced the peak temperature in all scenarios. In AW, we observed a 10.8% peak temperature reduction related to the spiral mapping algorithm, which represents more than 8℃ of peak temperature reduction. The HW scenario penalized the patterning mapping, while PIDTM presented a peak temperature reduction. In DW1, we observed a 12,7% temperature reduction in PIDTM related to the patterning approach, representing almost 10℃ of peak temperature reduction. DW2 is the worst-case scenario for PIDTM, with a result equal to the patterning approach. This situation may happen because two dominant tasks have not been placed next to each other in the patterning mapping. Even though PIDTM just matched the patterning algorithm in DW2, its execution time and the average temperature was lower.

This evaluation shows that the *patterning approach might not be a good option for temperature aware-mapping*. On the other side, an approach using monitoring and actuation, as PIDTM, is a path to follow to reduce peak temperature, improving the system reliability and lifetime.

Figure 5.19 compares the peak temperature between the PIDTM heuristic and the patterning mapping for HW and DW1 scenarios. We observe that the peak temperature is significantly lower with the proposed heuristic (10℃). The better thermal distribution obtained by using PIDTM causes a lower peak temperature most of the time. This behavior obtained with the PIDTM heuristic can potentially increase the chip's lifetime and reliability and open temperature room for more applications to execute.

## 5.5 Final Remarks

This Chapter presented mapping and migration strategies to provide a better thermal distribution in a many-core system supported by a runtime thermal estimation, considering different workload scenarios. The proposed heuristics produced an improvement of

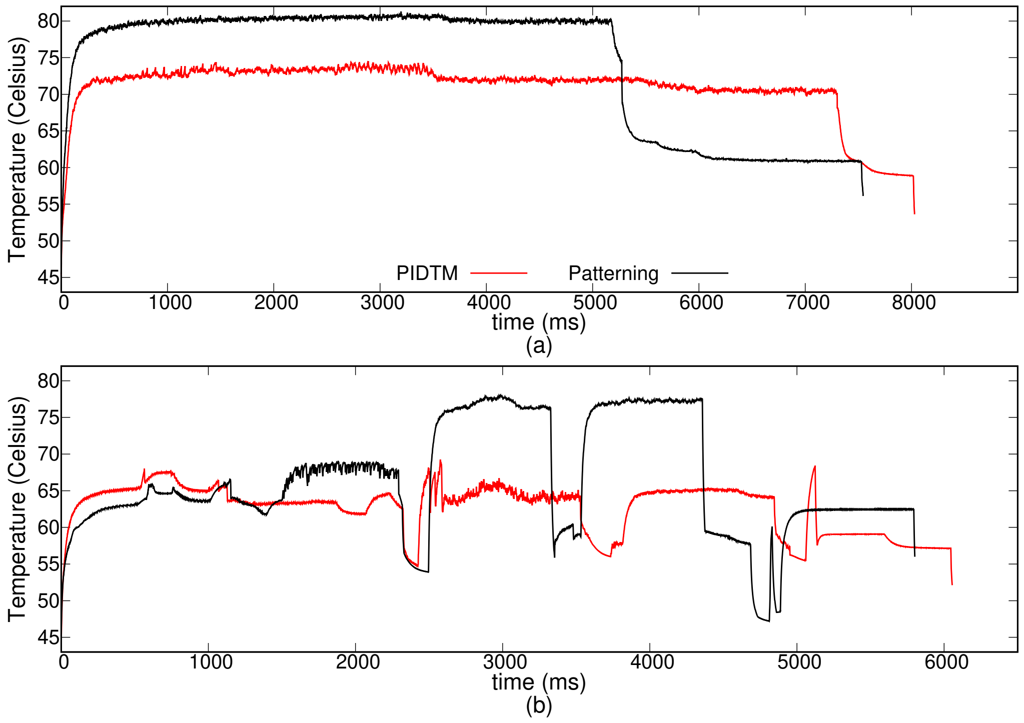


Figure 5.19 – Comparison between peak temperatures in 2 scenarios: (a) High Workload (HW); (b) Dynamic Workload (DW1) [da Silva et al., 2021a].

up to 7.15% in the peak temperature of the system in a high workload scenario, which is a worst-case scenario for temperature management. The results show that a better thermal distribution provided by the proposed strategies also reduces hotspots' occurrence compared with state-of-the-art approaches.

The proposed thermal management strategies use a hierarchical structure and a hardware accelerator for temperature estimation. Manager processors execute the proposed algorithms, which present a low computational complexity, do not affecting the overall system performance.

# 6. LIFETIME RELIABILITY

The main goal of this Chapter is to corroborate the hypothesis of this Thesis, “an efficient dynamic thermal management can improve the lifetime of a many-core system”. This Chapter is organized as follows:

- Section 6.1 discusses the main aging mechanisms affecting integrated circuits’ lifetime.
- Section 6.2 presents lifetime simulation tools publicly available for collecting results in experiments and lists the previously discussed aging mechanisms each tool supports.
- Section 6.3 evaluates lifetime with the selected tool, using results from DTMs and comparing existing heuristics against the proposed ones.
- Section 6.4 finishes this Chapter by discussing the proposed DTM heuristics regarding lifetime reliability results.

## 6.1 Concepts

This section presents fundamentals related to aging mechanisms. All aging effects reviwed in this Section are modeled by at least one simulation tool (Section 6.2). This Section reviews five aging effects: electromigration (EM), negative bias temperature instability (NBTI), time-dependent dielectric breakdown (TDDB), thermal cycling (TC), and stress migration.

### 6.1.1 Electromigration

Electromigration (EM) is a physical phenomenon of the migration of metal atoms along a direction of the applied electrical field. EM occurs in aluminum and copper interconnects due to the momentum transfer between the electrons and the metal atoms in the direction of the electron flow. This oriented atomic flow results in metal density depletion at the cathode and a corresponding metal accumulation at the anode ends of the metal wire. The depletion areas may observe an increased resistance or even open circuits, while the accumulation areas may cause short circuits. Over time, the lasting unidirectional electrical load increases these stresses, as well as the stress gradient along the metal line [Huang et al., 2014].

Electromigration has exponential temperature dependence, as observed in Black’s equation [Black, 1969]. Black’s equation is the most used model for mean time to failure (MTTF) due to EM [Srinivasan et al., 2004]:

$$MTTF_{EM} \propto (J - J_{crit})^{-s} e^{\frac{E_{aEM}}{kT}} \tag{6.1}$$

where: $J$ is the current density, $j_{crit}$ is the critical current density required for EM, $E_{aEM}$ is the activation energy for EM, $k$ is the Boltzmann's constant, and $T$ is the absolute temperature in Kelvin. $s$ and $E_{aEM}$ are constants that depend on the interconnect metal used (1.1 and 0.9 for copper interconnects as modeled in RAMP). As $J$ is usually much higher than $J_{crit}$, $(J - J_{crit}) \sim J$.

$J$ can be modeled as [Dasgupta and Karri, 1996]:

$$J = \frac{CV_{dd}}{WH} \times f \times p \tag{6.2}$$

where: $C$, $W$ and $H$ are the capacitance, width and thickness of the line, $f$ is the clock frequency and $p$ is the switching probability.

Finally, the model shows that the EM is mainly affected by the temperature, the supply voltage and the activity of the circuit, which is the $f \times p$ product.

Figure 6.1 illustrates the electromigration effect, inducing an open circuit in an interconnection.

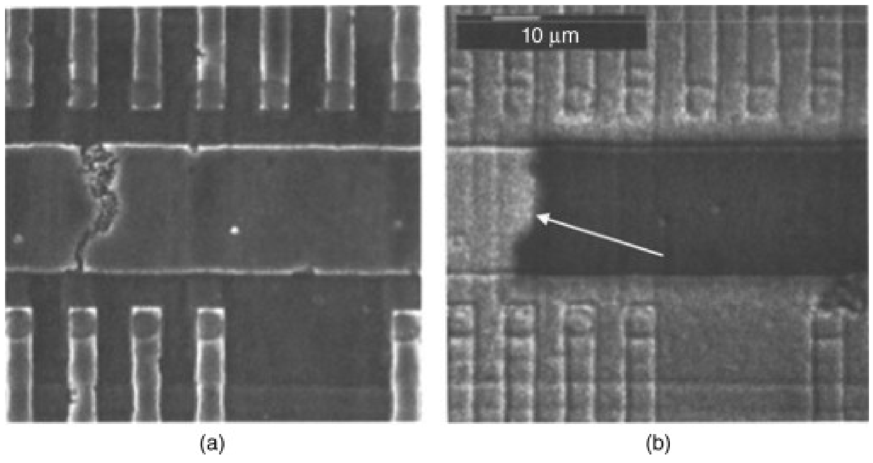


Figure 6.1 – Scanning electron micrographs of the open circuit induced by EM: (a) normal topographic image; (b) voltage contrast image [Read et al., 2011].

### 6.1.2 Time Dependent Dielectric Breakdown

Time-Dependent Dielectric Breakdown (TDDB) in silica($SiO_2$)-based dielectrics is one of the most important failure mechanisms in integrated circuits [McPherson, 2012]. The formation of a conducting path causes TDDB through the gate oxide to substrate due to the application of a constant low electric field. While applying a high enough electric field can cause an immediate breakdown of the device, the TDDB is a slow process that occurs when the device is affected by an electric field. Once a conducting path is formed between source and drain, the transistor cannot control the current flow, which is considered a failure. TDDB has become increasingly severe as the thickness of the gate oxide decreased and also because the supply voltage is not scaling down effectively [Srinivasan et al., 2004].

The accepted model for TDDB is presented on Equation (6.3) and shows that the TDDB effect has an inversely proportional dependence on voltage and also an exponential degradation due to the temperature:

$$MTTF_{TDDB} \propto \left(\frac{1}{V}\right)^{a-bT} \times e^{\frac{X+\frac{Y}{T}+ZT}{kT}} \tag{6.3}$$

where: $k$ is the Boltzman's constant and $a$, $b$, $X$, $Y$ and $Z$ are model fitting parameters and are determined from experimental data. In this work, we use RAMP as a reference, and the values used are $a$ = 78, $b$ = -0.081, $X$ = 0.759$ev$, $Y$ = -66.8$evK$, and $Z$ = -8.37e$^{-4}$$ev/K$.

### 6.1.3 Negative Bias Temperature Instability

Negative Bias Temperature Instability (NBTI) affects PMOS transistors. NBTI occurs when positive charges build up in the gate oxide due to the application of a negative gate voltage at high temperatures. NBTI manifests as an increase in the threshold voltage and consequent decrease in drain current and transconductance of a PMOS. The increase in the threshold voltage can slow down the device and cause the electronic circuit to fail due to timing constraints [Srinivasan, 2006]. NBTI has become more severe with the technology downscaling and with the reduction in supply voltages.

Equation (6.4) models the MTTF due to NBTI, $MTTF_{NBTI}$. The model shows that NBTI is also related on temperature.

$$MTTF_{NBTI} \propto \left[\left(ln\left(\frac{A}{1+2e^{\frac{B}{kT}}}\right) - ln\left(\frac{A}{1+2e^{\frac{B}{kT}}} - C\right)\right) \times \frac{T}{e^{\frac{D}{kT}}}\right]^{\frac{1}{\beta}} \tag{6.4}$$

where: $A$, $B$, $C$, $D$ and $\beta$ are model fitting parameters. In this work we use values from RAMP [Srinivasan, 2006]: $A$ = 1.6328, $B$=0.07377, $C$=0.01, $D$=-0.06852, and $\beta$=0.3.

### 6.1.4 Stress Migration

Stress migration (SM) refers to the movement of metal atoms in the interconnects due to the mechanical stress caused by different thermal expansion rates of the materials. If the stress is generated by differing thermal expansion rates, then the stress is called "thermomechanical stress" and is proportional to the change in temperature, $\delta T = T_0 - T$, where $T$ is the operating temperature and $T_0$ is the metal deposition temperature. Therefore, the equation that describes the MTTF due to stress migration is given by [JEDEC, 2006]:

$$MTTF_{SM} \propto |T_0 - T|^{-S} e^{\frac{E_{a_{SM}}}{kT}} \tag{6.5}$$

where: the temperatures are in Kelvin, and the values used in RAMP are: $S = 2.5$, $E_a$=0.9$eV$ and $T_0$=500K [Srinivasan, 2006].

The equation shows that temperatures close to $T_0$ may increase the MTTF. However, the metal deposition temperatures tend to be much higher than the operating temperatures. The exponential portion of the equation usually reduces the MTTF when the operating temperatures are high.

### 6.1.5 Thermal Cycling

Thermal cycling (TC) or thermal shock can cause damage to integrated circuit devices. Every time there is a temperature cycle, permanent damage may occur in the device, eventually leading to failure. Temperature oscillations caused by powering up and powering down the device also causes damage. All parts of the device may suffer damage due to thermal cycling. However, the effect occurs mainly in the package and die interface (wire bonds, solder joints).

Two types of cycles may cause damage on the device: (*i*) low frequency and (*ii*)high frequency. Low-frequency cycles typically occur a few times a day and may cause damage after thousands of cycles. High-frequency cycles occur few times a second, mainly due to changes in workload, and may require millions of cycles to cause permanent damage. Low-frequency cycles are better understood and modeled using the Coffin-Manson equation [JEDEC, 2006] :

$$N_f = C_0(\Delta T)^{-q} \tag{6.6}$$

where: $N_f$ is the number of thermal cycles to failure, $C_0$ is an empirically determined constant, $\Delta T$ is the thermal cycle temperature range, and $q$ is the Coffin-Manson exponent, also empirically determined.

RAMP only models the thermal cycling in the package, which occurs in low frequency and is more pronounced. The equation used to model the MTTF due to thermal cycling in RAMP is the following:

$$MTTF_{TC} \propto \left(\frac{1}{T - T_{ambient}}\right)^q \tag{6.7}$$

where: $T - T_{ambient}$ represents the amplitude of the thermal cycle and the Coffin-Manson exponent $q$, for package, is 2.35.

## 6.2 Lifetime Simulation Tools

Previous section showed that temperature has a strong correlation with aging effects. This Section describes lifetime simulation tools modelling the aging effects where temperature is one of the input parameters.

### 6.2.1 LifeSim

LifeSim [Rohith et al., 2018] is a simulation tool that integrates a many-core architecture simulator, a thermal simulator, and a lifetime reliability analyzer. The simulation tool is modular and interacts with user-defined mapping, migration, and DVFS algorithms. LifeSim comes with failure models for EM, NBTI, TDDB, and TC.

LifeSim uses the architectural simulator *Snipersim* [Carlson et al., 2011], which integrates McPAT [Li et al., 2009] for power estimation. LifeSim also integrates HotSpot [Huang et al., 2006] for thermal simulation, offering the possibility for the user to use a custom thermal model. Last, LifeSim provides its reliability management unit (RMU), which calculates the aging and mean time to failure (MTTF) of each core. The models used for aging and lifetime reliability for TDDB, TC, and NBTI come from RAMP [Srinivasan, 2006], and EM uses the model from [Huang et al., 2014].

LifeSim expresses the lifetime in MTTF measured in years for each PE, considering only one aging effect at a time. The user can change the aging effect that the tool will evaluate by changing the configuration file. The MTTF is calculated without any statistical analysis, considering that the Failure in Time (FIT) value returned by the model is equivalent to one failure in $10^9$ hours of operation.

### 6.2.2 REST

Yamamoto and Ababei [Yamamoto and Ababei, 2014] proposed a unified reliability evaluation methodology for NoC-based chip multiprocessors (CMPs). REST stands for Reliability ESTimation, being an integration of simulation tools that provides the reliability estimation of the CMP considering the computational and communication components in a unified manner, using a Monte Carlo algorithm.

REST comes with failure models for NBTI and TDDB. Both models use RAMP [Srinivasan, 2006] as a reference. While the tool focuses on these two aging mechanisms for simplicity, the proposed framework is general. It can be extended to include other aging mechanisms such as electromigration, stress migration, and thermal cycling.

While the authors present a reliability estimation tool, the work aims to propose a new dynamic reliability management (DRM) scheme for CMPs. The proposed DRM scheme uses a neural network (NN) based reliability estimation module to reallocate tasks based on the user reliability target.

The inputs to REST are a floorplan and a steady temperature file. With this information, the tool cannot consider a complete simulation to compute the expected MTTF based on a predefined workload. For each provided input, the tool generates one FIT value for the whole system after a Monte Carlo simulation using the distribution generated with the NBTI and TDDB models.

### 6.2.3 RAMP

Jayanth Srinivasan et al. [Srinivasan et al., 2004, Srinivasan, 2006] proposed a methodology for evaluating processor lifetime reliability called RAMP. RAMP stands for Reliability Aware Micro-Processors and provides a simulation tool to be used together with timing, power, and thermal simulators. Authors claim that RAMP could also be implemented in hardware if the required sensors are provided to estimate in real-time the expected lifetime of each core in a system.

The first version of RAMP [Srinivasan et al., 2004] included models for EM, SM, TDDB, and TC, while the later work [Srinivasan, 2006] included all of the previous models plus a model for NBTI. Another difference is that the first work uses a sum-of-failure-rates (SOFR) model to obtain the processor MTTF, while the last discusses the limitations of using a SOFR model and proposes a lognormal distribution model, using a Monte Carlo simulation to determine the final MTTF of the system, being named RAMP 2.0.

In this Thesis, we use RAMP 2.0 since it is the most comprehensive simulation tool for many-core systems, modeling the five aging mechanisms presented in Section 6.1. We also have our own architectural, power, and temperature simulation methodologies, described in Chapter 3 and Chapter 4, and RAMP is a tool that could be easily adapted to estimate the MTTF in our environment, unlike LifeSim, which has its architectural simulator. Finally, RAMP is the tool of choice because even the newer works use the RAMP models, with one exception being the EM model in LifeSim [Huang et al., 2014], which presents a newer but less conservative model for MTTF estimation due to EM.

RAMP 2.0 expresses the lifetime of the system considering all five aging effects. First, the tool generates a FIT file containing the FIT for each model in each PE of the system. Then, the FIT file is analyzed by the Monte Carlo tool, which is provided with RAMP

2.0. The output of the Monte Carlo tool is the MTTF measured in years for the whole system, considering all aging effects in all PEs after the statistical analysis.

RAMP does not provide a tool to analyze the lifetime of systems. Instead, it provides code snippets to be integrated into a simulator to analyze the lifetime at runtime. As we do not use any DRM technique that requires lifetime estimation at runtime, we analyze the temperature and power logs of the simulations to estimate the lifetime for each test case using the code provided by RAMP. Figure 6.2 shows the RAMP usage flow used to generate the results presented in Section 6.3.

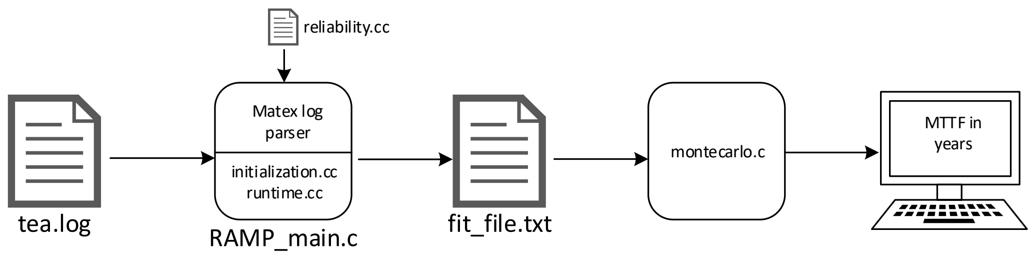


Figure 6.2 – RAMP 2.0 usage flow.

The tool *RAMP_main.c* was created to analyze the log provided by TEA during the simulation (*tea.log*) with the power and temperature samples for each simulation. *RAMP_main.c* uses the code snippets *initialization.cc* and *runtime.cc* provided with RAMP to generate the FIT file (fit_file.txt) to be analyzed by the Monte Carlo simulation. The Monte Carlo simulation (*montecarlo.c*) was used without changes and it outputs on the terminal the expected MTTF in years based on the analysis of the FIT file.

## 6.3 Results

This Section presents the expected lifetime results for the reference platform using the heuristics proposed in Chapter 5. The testcases used for evaluation are be the same presented in Section 5.4.1.

### 6.3.1 RTL platform

The evaluation of lifetime reliability requires long simulations to capture the steady-state temperature behavior of the selected test case. Long simulations using the RTL description of the reference platform are time-consuming, usually about two days of simulation for each second of results. Thus, we used only the high workload (HW) test case to evaluate the MTTF in the RTL platform. The test case was executed in an 8x8 system, and Figure 6.3 presents the expected MTTF results for this scenario.

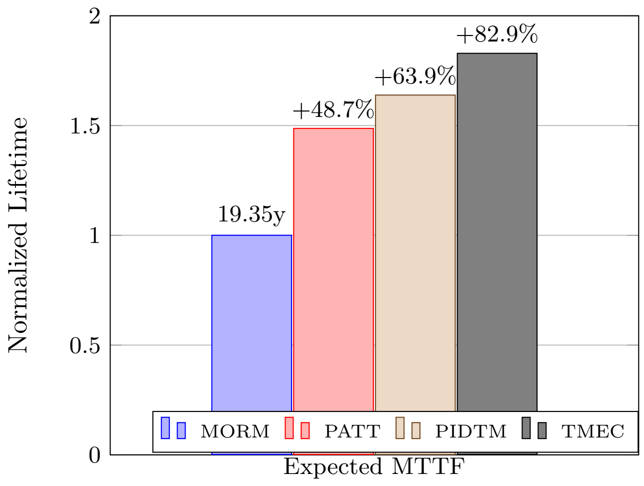


Figure 6.3 – Expected MTTF comparison between MORM, patterning, PIDTM and TMEC for the HW scenario.

Results show that the expected lifetime when using MORM in the high workload scenario is about 19 years of operation, while the patterning approach may increase the lifetime by 48.7%, even achieving a slightly higher overall peak temperature. This observation shows that reducing only the peak temperature is not enough to provide a higher MTTF. The average temperature is also important to consider when the objective is to increase lifetime.

PIDTM and TMEC heuristics both get a lifetime improvement when comparing with MORM and patterning approaches. TMEC got the best MTTF result, even getting a higher peak temperature than PIDTM. This is due to the workload distribution provided by the TMEC heuristic, reinforcing that the overall peak temperature is not the most important metric to improve the MTTF.

## 6.3.2 OVP platform

The OVP platform enables faster simulations than the RTL platform. Therefore, we executed four scenarios to evaluate the expected lifetime in the OVP platform: average workload (AW), high workload (HW), dynamic workload 1 (DW1), and dynamic workload 2 (DW2). All scenarios were executed using: (*i*) spiral mapping, which is the initial mapping used in MORM; (*ii*) patterning mapping; (*iii*) PIDTM; (*iv*) TMEC.

Figure 6.4 shows the results of the expected MTTF for the four executed scenarios. Considering the spiral mapping as a baseline, it is possible to observe that the higher the workload, the lower the expected MTTF. In high workload, the spiral mapping got an average MTTF of 25.2 years, while in dynamic workload 1, which is the lightest workload, the expected lifetime was about 51.7 years. The improvements obtained with the PIDTM and TMEC heuristics were also proportional to the workload. Although the AW shows the greater improvements when using the proposed heuristics (25%), the DW1 shows that in

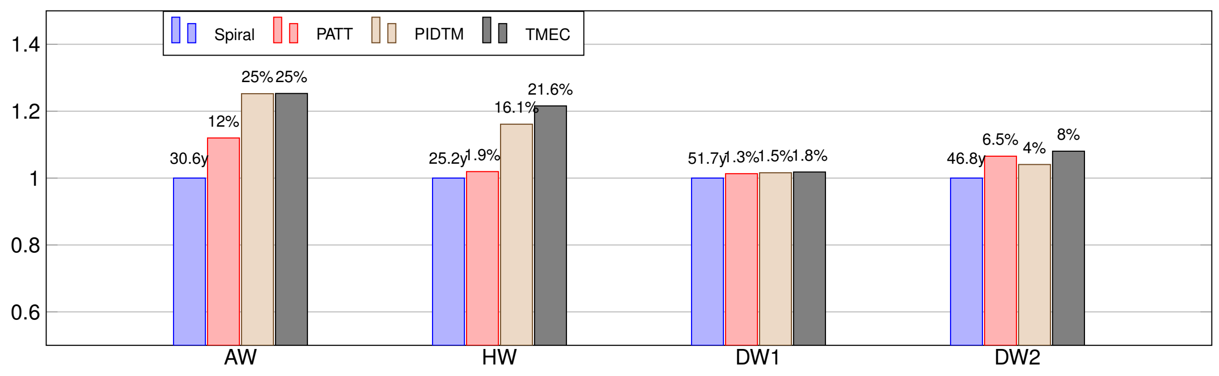


Figure 6.4 – Spiral, Patterning, PIDTM and TMEC normalized MTTF results.

a lighter workload, the improvements are negligible. In DW2, which has a slightly higher workload than DW1, we got a maximum improvement of 8% in the expected lifetime when using TMEC. For the HW, which is the worst-case scenario for lifetime reliability, TMEC got an improvement of 21.6% in the lifetime, while PIDTM got 16.1%.

DW1 is the lightest workload of all executed scenarios. There are peaks and valleys of processor usage in this scenario. Thus there is no average temperature advantage for the proposed heuristics since there is a cooldown period during the valleys of workload. As the results provided by RAMP consider the temperature during the whole simulation, and the average temperatures are similar in all simulations, there are no significant differences in lifetime reliability for this scenario.

Comparing the HW results of Figure 6.4 to the results obtained with the RTL simulations from Figure 6.3, it is possible to observe that the MTTF baseline for MORM (19.35 years), compared to the spiral mapping in the OVP simulation (25.3 years), is significantly smaller. This happens for two main reasons: (*i*) the architectures are different; (*ii*) tasks in the quantum-based simulation wait for longer periods for data to compute. While in the RTL model we model a MIPS-like CPU, with a subset of the MIPS instructions, the OVP CPU is an OR1K instance, which uses Open RISC instructions, which may cause different computation times for each architecture. Furthermore, the communication latency induced by the quantum-based simulation in OVP makes computation-intensive tasks wait more time for the required data. So, these tasks might generate less heat in the OVP model, leading to a longer lifetime, especially with the spiral mapping, where the computation-intensive tasks may be mapped close to each other.

## 6.4 Final Remarks

This Chapter presented a study related to the aging effects on CMOS circuits and the lifetime reliability results for the heuristics proposed in this Thesis.

After studying the aging effects and the tools that provide MTTF estimations, we choose RAMP 2.0 to estimate the expected MTTF using the proposed heuristics. The results confirmed our hypothesis that managing the system temperature is important to improve the lifetime reliability of the system. We observed that the peak temperature reduction is important, but the average temperature also plays an important role in improving the expected system lifetime. We also concluded that choosing the best PE to reduce the temperature is also not ideal for improving the system lifetime. Balancing the workload between all PEs is important to distribute the aging accumulation in each PE.

# 7. CONCLUSIONS

This work adopted the following paragraph as the Thesis statement in the Introduction: *Thermal and energy management are required in recent technology nodes to keep the operation within the limits of power density, increasing the circuit lifetime reliability. Management techniques employed in many-core designs must be aware of the power consumed and the temperature of each processing element. Thermal-aware task mapping allied with migration and DVFS must deal with the conflicts between thermal constraints and performance demand. The hypothesis to be fulfilled by this Thesis is the demonstration that DTM techniques, applied in high and dynamic workloads, keep the temperature within safe limits and increases the system lifetime reliability.*

To demonstrate the Thesis statement, we first presented the reference many-core architecture used to develop the management proposals. The reference architecture adopted was a NoC based many-core system designed in two equivalent descriptions: a cycle-accurate RTL description and an instruction-level model using a quantum-based simulation. The reference architecture already supported energy and power consumption monitoring, presented in previous works for a 65nm technology node. In the context of this Thesis, we updated the energy and power monitoring characterization for a 28nm technology node and proposed a new method to enable temperature monitoring in the reference architecture.

The temperature monitoring development was essential to provide precise temperature data, enabling the DTM development for the reference architecture. Most of the state-of-art works in DTM do not provide a thermal monitoring solution, relying on thermal models such as HotSpot or MatEx, without clarifying how these models would be executed. As our reference platform description is much closer to the actual hardware implementation than most of the works, we had to find a feasible way to provide temperature estimation for the platform. The result was the thermal estimation accelerator (TEA), a scalable hardware accelerator that enables fine-grain temperature monitoring for many-core systems. TEA provides the required data for the development of DTM proposals for the reference architecture.

The DTM proposals of this Thesis follow the ODA (Observe-Decide-Act) paradigm. The ODA model was divided into levels to fit the reference architecture. The levels available in the reference architecture considering the ODA model are peripheral level, system level, cluster level, and PE level. At the peripheral level, there are temperature observation and application injection. At the system level, there is the application admission procedure. At the cluster level, there are application mapping and migration, and DFVS decision. At the PE level there are the DVFS actuation and the power states.

We proposed the temperature cluster selection algorithm (TCS) for the admission phase, where the cluster with the lowest average temperature with enough resources is selected to execute a new application. This simple strategy alone was enough to provide a better temperature distribution in the executed test cases. For application mapping and task migration, we defined three heuristics for prioritizing PEs: TPESM, PIDTM, and TMEC. TPESM was the first attempt to provide a better temperature distribution within the cluster, prioritizing PEs only by instantaneous temperature. PIDTM and TMEC were the main DTM proposals of this Thesis. PIDTM also uses the temperature derivative to avoid PEs already in a rising temperature and the integral to balance the workload between the PEs. TMEC also considers the energy consumption of each PE to further improve the workload balancing.

It was essential to evaluate the proposed heuristics concerning lifetime to corroborate the Thesis hypothesis. Therefore, we studied the main aging effects on CMOS circuits and the tools that provide lifetime estimation. Lifetime estimations are more accurate with longer simulations. For this reason, we relied on the instruction-level model (OVP) to execute most of the test cases. Nonetheless, a very long simulation was executed on the clock cycle-accurate model (RTL) to verify the differences in the results. Lifetime simulations highlighted the differences between the OVP and the RTL model. The main differences are related to the NoC timings, which are still not as precise in OVP as in the RTL model. The realistic NoC latency from the RTL model simulation makes the results accurate, highlighting the expected lifetime difference between the proposed heuristics and other state-of-art proposals.

Results show that a simple heuristic like PIDTM can significantly improve the system's thermal distribution, reducing the peak and the average temperature in high workload scenarios. PIDTM reduced up to 7.15% in the overall peak temperature and presented an increase of 63.9% in expected lifetime compared with a state-of-art power management proposal in the RTL model. However, TMEC got better overall lifetime reliability results since it tracks the energy consumed by each PE, balancing the system aging. TMEC improved up to 82.9% in the system's expected lifetime, demonstrating the importance of this proposal.

Back to the Thesis statement, we concluded that the proposed DTM heuristics provided lifetime improvements in high workload scenarios, while in dynamic workloads the gains were less pronounced. There are opportunities for the system to cool down in dynamic workloads, so the base expected lifetime for the compared state-of-art techniques is already high. The conclusion we draw from these observations is that in the worst case, where lifetime reliability is a real concern, the proposed heuristics significantly improved the expected MTTF of the system. Therefore, we claim that the Thesis statement is valid.

## 7.1 Future works

As a guideline for future work, this Thesis has room for improvements as follows:

- Energy characterization of the reference architecture in a newer library. Before this Thesis, the reference platform had an energy characterization in a 65nm library. However, in the context of DTM proposals, it was essential to create a new characterization in 28nm to increase the power density of the system. The smaller the technology node is, the higher the power density, and the larger is the need to apply DTM techniques. Ideally, future works should include a new characterization of the reference platform in a 14nm library or smaller to create more challenging DTM scenarios.

- Add peak temperature calculation in TEA to enable proactive mapping. TEA calculates the instant transient temperatures based on the power samples received. However, the original MatEx algorithm also provides the peak temperature calculation after a change in the system's power state. This feature could enable the prediction of the future temperature of the system before an application mapping. The required change to add this feature on TEA is mainly the addition of a new matrix memory since the logic is the same as calculating the transient temperatures.

- Improve OVP model NoC timings to get results closer to RTL. OVP model enables much longer simulations in less time than the RTL model. However, the differences in NoC timings prevent the results from being closer to the RTL model, which tends to be more accurate. The quantum-based simulation might prevent the NoC from having a lower latency in the OVP model. However, if the NoC timings could be modeled better in OVP, it would be possible to execute much longer simulations with the required accuracy to evaluate DTM techniques.

- Include expected lifetime in the heuristics. In this Thesis, we proposed and analyzed only dynamic thermal management (DTM) techniques. However, state-of-art presents some proposals of dynamic reliability management (DRM), including the results of the lifetime estimation in the heuristics to provide better MTTF in the systems. The proposed TMEC heuristic provided significant lifetime improvements, but we still wonder if we can get better lifetime results by including the expected lifetime in the heuristics.

- Apply the proposed heuristics in FPGA or in a real platform for real-time evaluation. RTL simulation enables evaluating the proposed heuristics of just hundreds of milliseconds, while the OVP simulation increased the evaluation time to a few seconds. A real implementation of the reference architecture in FPGA would enable hours of execution time, which could create new opportunities to evaluate and improve the DTM proposals.

## 7.2 Summary of the publications produced during the Thesis

Table 7.1 presents the summary of the publications produced during the Thesis period, linking each one with the related Thesis Chapter.

Table 7.1 – Summary of Publications.

| Publication Reference | Relationship with the Thesis |
|---|---|
| Da Silva, Alzemiro Lucas; Weber, Iaçanã; Martins, André Luís Del Mestre; Moraes, Fernando Gehm<br>**Dynamic Thermal Management in Many-Core Systems Leveraged by Abstract Modeling**<br>In: ISCAS, 2021 | Chapter 5 - Evaluation of PIDTM heuristic in the OVP platform |
| Da Silva, Alzemiro Lucas; Weber, Iacaça Ianiski; Martins, André Luís Del Mestre; Moraes, Fernando Gehm<br>**Hardware Accelerator for Runtime Temperature Estimation in Many-cores**<br>IEEE Design & Test, March, 2021. | Chapter 4 - Presents detailed results of TEA proposal |
| Da Silva, Alzemiro Lucas; Martins, André Luís Del Mestre; Moraes, Fernando Gehm<br>**Mapping and Migration Strategies for Thermal Management in Many-Core Systems**<br>In: SBCCI, 2020 | Chapter 5 - TPESM, PIDTM and TMEC heuristics proposal |
| Da Silva, Alzemiro Lucas; Martins, André Luís Del Mestre; Moraes, Fernando Gehm<br>**Fine-grain Temperature Monitoring for Many-Core Systems**<br>In: SBCCI, 2019 | Chapter 4 - Presents TEA proposal |
| Martins, André Luís Del Mestre; Da Silva, Alzemiro Lucas; Rahmani, Amir M.; Dutt, Nikil; Moraes, Fernando Gehm<br>**Hierarchical Adaptive Multi-objective Resource Management for Many-core Systems**<br>Journal of Systems Architecture, vol. 97, August 2019, Pages 416-427. | Participation as a coauthor in MORM proposal. Created benchmarks to evaluate MORM. MORM is used as a reference to compare the results in Section 5.4. |

# REFERENCES

[Al Faruque et al., 2010] Al Faruque, M., Jahn, J., Ebi, T., and Henkel, J. (2010). Runtime Thermal Management Using Software Agents for Multi- and Many-Core Architectures. *IEEE Design Test of Computers*, 27(6):58–68.

[Binkert et al., 2011] Binkert, N. et al. (2011). The Gem5 Simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7.

[Black, 1969] Black, J. (1969). Electromigration — A brief survey and some recent results. *IEEE Transactions on Electron Devices*, 16(4):338–347.

[Borkar, 2007] Borkar, S. (2007). Thousand Core Chips: A Technology Perspective. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 746–749.

[Bortolon and Moraes, 2017] Bortolon, F. T. and Moraes, F. G. (2017). Hardware and software infrastructure to implement many-core systems in modern FPGAs. In *Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 79–83.

[Carlson et al., 2011] Carlson, T. E., Heirman, W., and Eeckhout, L. (2011). Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC)*, pages 1–12.

[Castilhos et al., 2013] Castilhos, G., Mandelli, M., Madalozzo, G., and Moraes, F. G. (2013). Distributed resource management in NoC-based MPSoCs with dynamic cluster sizes. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 153–158.

[Castilhos et al., 2016] Castilhos, G., Moraes, F. G., and Ost, L. (2016). A Lightweight Software-based Runtime Temperature Monitoring Model for Multiprocessor Embedded Systems. In *Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6.

[Chakraborty and Kapoor, 2018] Chakraborty, S. and Kapoor, H. K. (2018). Analysing the Role of Last Level Caches in Controlling Chip Temperature. *IEEE Transactions on Sustainable Computing*, 3(4):289–305.

[Coskun et al., 2008] Coskun, A. K., Rosing, T. T., Whisnant, K., and Gross, K. C. (2008). Static and Dynamic Temperature-aware Scheduling for Multiprocessor SoCs. *IEEE Transactions on VLSI Systems*, 16(9):1127–1140.

[da Silva et al., 2019] da Silva, A. H. L., Martins, A. L. d. M., and Moraes, F. G. (2019). Fine-grain Temperature Monitoring for Many-Core Systems. In *Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6.

[da Silva et al., 2020] da Silva, A. H. L., Martins, A. L. d. M., and Moraes, F. G. (2020). Mapping and Migration Strategies for Thermal Management in Many-Core Systems. In *Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6.

[da Silva et al., 2021a] da Silva, A. H. L., Weber, I., Martins, A. L. d. M., and Moraes, F. G. (2021a). Dynamic Thermal Management in Many-Core Systems Leveraged by Abstract Modeling. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5.

[da Silva et al., 2021b] da Silva, A. H. L., Weber, I. I., Martins, A. L. d. M., and Moraes, F. G. (2021b). Hardware Accelerator for Runtime Temperature Estimation in Many-cores. *IEEE Design Test of Computers*, 38(4):62–69.

[Das et al., 2016] Das, A., Al-Hashimi, B. M., and Merrett, G. V. (2016). Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems. *ACM Transactions on Embedded Computing Systems*, 15(2):24:1–24:25.

[Das et al., 2013] Das, R., Ausavarungnirun, R., Mutlu, O., Kumar, A., and Azimi, M. (2013). Application-to-core mapping policies to reduce memory system interference in multi-core systems. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 107–118.

[Dasgupta and Karri, 1996] Dasgupta, A. and Karri, R. (1996). Electromigration Reliability Enhancement via Bus Activity Distribution. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 353–356.

[Dutt et al., 2015] Dutt, N., Jantsch, A., and Sarma, S. (2015). Self-Aware Cyber-Physical Systems-on-Chip. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 46–50.

[El Ahmad et al., 2018] El Ahmad, M., Najem, M., Benoit, P., Sassatelli, G., and Torres, L. (2018). PoETE: A Method to Design Temperature-Aware Integrated Systems. *Journal of Low Power Electronics*, 14(1):1–7.

[Esmaeilzadeh et al., 2011] Esmaeilzadeh, H., Blem, E., Amant, R. S., Sankaralingam, K., and Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Proceedings of the International International Symposium on Computer Architecture (ISCA)*, pages 365–376.

[Haghbayan et al., 2020] Haghbayan, M.-H., Miele, A., Zou, Z., Tenhunen, H., and Plosila, J. (2020). Thermal-Cycling-aware Dynamic Reliability Management in Many-Core System-on-Chip. In *Proceedings of the Design, Automation Test in Europe Conference (DATE)*, pages 1229–1234.

[Huang et al., 2006] Huang, W., Ghosh, S., Velusamy, S., Sankaranarayanan, K., Skadron, K., and Stan, M. R. (2006). HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513.

[Huang et al., 2014] Huang, X., Yu, T., Sukharev, V., and Tan, S. X.-D. (2014). Physics-based electromigration assessment for power grid networks. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 1–6.

[Iizuka, 2015] Iizuka, T. (2015). CMOS technology scaling and its implications. In *Digitally-Assisted Analog and Analog-Assisted Digital IC Design*, pages 1–20. Cambridge University Press.

[Imperas, 2019] Imperas (2019). Open Virtual Platforms - OVP. Source: http://www.ovpworld.org/.

[JEDEC, 2006] JEDEC (2006). Failure Mechanisms and Models for Semiconductor Devices. Technical report, Solid State Technology Association. Source: https://www.jedec.org.

[Li et al., 2018] Li, M., Liu, W., Yang, L., Chen, P., and Chen, C. (2018). Chip temperature optimization for dark silicon many-core systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(5):941–953.

[Li et al., 2015] Li, M., Yi, J., Liu, W., Zhang, W., Yang, L., and Sha, E. H.-M. (2015). An efficient technique for chip temperature optimization of multiprocessor systems in the dark silicon era. In *Proceedings of the International Conferences on High Performance Computing and Communications (HPCC)*, pages 688–693.

[Li et al., 2009] Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., and Jouppi, N. P. (2009). McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the ACM/IEEE International Symposium on Microarchitecture (MICRO)*, pages 469–480.

[Li et al., 2011] Li, S., Chen, K., Ahn, J. H., Brockman, J. B., and Jouppi, N. P. (2011). CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 694–701.

[Liu et al., 2018] Liu, W., Yang, L., Jiang, W., Feng, L., Guan, N., Zhang, W., and Dutt, N. D. (2018). Thermal-aware Task Mapping on Dynamically Reconfigurable Network-on-Chip based Multiprocessor System-on-Chip. *IEEE Transactions on Computers*, 67(12):1818 – 1834.

[Liu et al., 2019] Liu, W., Yi, J., Li, M., Chen, P., and Yang, L. (2019). Energy-Efficient Application Mapping and Scheduling for Lifetime Guaranteed MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(1):1–14.

[Lopes et al., 2021] Lopes, G., Weber, I., Marcon, C., and Moraes, F. G. (2021). Chronos: an Abstract NoC-based Manycore with Preserved Temporal and Spatial Traffic Distribution. In *Proceedings of the IEEE Latin American Symposium on Circuits and Systems (LASCAS)*, pages 1–4.

[Martins, 2018] Martins, A. L. d. M. (2018). *Multi-Objective Resource Management for Many-core Systems*. PhD thesis, PUCRS, Porto Alegre, RS, Brasil, 149 pages.

[Martins et al., 2019a] Martins, A. L. d. M., da Silva, A. H. L., Rahmani, A. M., Dutt, N., and Moraes, F. G. (2019a). Hierarchical adaptive Multi-objective resource management for many-core systems. *Journal of Systems Architecture*, 97:416–427.

[Martins et al., 2019b] Martins, A. L. d. M., Garibotti, R., Dutt, N., and Moraes, F. G. (2019b). The Power Impact of Hardware and Software Actuators on Self-Adaptable Many-core Systems. *Journal of Systems Architecture*, 97:42–53.

[Martins et al., 2015] Martins, A. L. d. M., Ruaro, M., and Moraes, F. G. (2015). Hierarchical energy monitoring for many-core systems. In *Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 657–660.

[Martins et al., 2016] Martins, A. L. d. M., Sant'Ana, A. C., and Moraes, F. G. (2016). Run-time energy management for many-core systems. In *Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 380–383.

[Martins et al., 2014] Martins, A. L. d. M., Silva, D. R. G., Castilhos, G. M., Monteiro, T. M., and Moraes, F. G. (2014). A method for NoC-based MPSoC energy consumption estimation. In *Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 427–430.

[McPherson, 2012] McPherson, J. (2012). Time dependent dielectric breakdown physics – Models revisited. *Microelectronics Reliability*, 52(9):1753–1760.

[Mohammed et al., 2020] Mohammed, M. S., Al-Kubati, A. A. M., Paraman, N., Ab Rahman, A. A.-H., and Marsono, M. N. (2020). DTaPO: Dynamic Thermal-Aware Performance Optimization for Dark Silicon Many-Core Systems. *Electronics*, 9(11):1–18.

[Ost et al., 2009] Ost, L., Guindani, G. M., Indrusiak, L. S., Reinbrecht, C., da Rosa, T. R., and Moraes, F. G. (2009). A high abstraction, high accuracy power estimation model for networks-on-chip. In *Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6.

[Pagani et al., 2015] Pagani, S., Chen, J.-J., Shafique, M., and Henkel, J. (2015). Thermal-aware power budgeting for dark silicon chips. In *Proceedings of the International Green and Sustainable Computing Conference (IGSC)*, pages 1–6.

[Pagani et al., 2017] Pagani, S., Khdr, H., Chen, J., Shafique, M., Li, M., and Henkel, J. (2017). Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon. *IEEE Transactions on Computers*, 66(1):147–162.

[Pagani et al., 2014] Pagani, S., Khdr, H., Munawar, W., Chen, J., Shafique, M., Li, M., and Henkel, J. (2014). TSP: Thermal Safe Power - Efficient power budgeting for many-core systems in dark silicon. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10.

[Pagani et al., 2015] Pagani, S., Khdr, H., Munawar, W., Chen, J., Shafique, M., Li, M., and Henkel, J. (2015). MatEx: Efficient transient and peak temperature computation for compact thermal models. In *Proceedings of the Design, Automation Test in Europe Conference (DATE)*, pages 1515–1520.

[Pagani et al., 2015] Pagani, S., Shafique, M., Khdr, H., Chen, J.-J., and Henkel, J. (2015). seBoost: selective boosting for heterogeneous manycores. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 104–113.

[Rahmani et al., 2015] Rahmani, A., Haghbayan, M., Kanduri, A., Weldezion, A. Y., Liljeberg, P., Plosila, J., Jantsch, A., and Tenhunen, H. (2015). Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 219–224.

[Rahmani et al., 2018] Rahmani, A. M., Donyanavard, B., Mück, T., Moazzemi, K., Jantsch, A., Mutlu, O., and Dutt, N. D. (2018). SPECTR: Formal Supervisory Control and Coordination for Many-core Systems Resource Management. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 169–183.

[Rao and Vrudhula, 2008] Rao, R. and Vrudhula, S. (2008). Efficient Online Computation of Core Speeds to Maximize the Throughput of Thermally Constrained Multi-core Processors. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 537–542.

[Rathore et al., 2018] Rathore, V., Chaturvedi, V., Singh, A. K., Srikanthan, T., Rohith, R., Lam, S.-K., and Shaflque, M. (2018). HiMap: A hierarchical mapping approach for enhancing lifetime reliability of dark silicon manycore systems. In *Proceedings of the Design, Automation Test in Europe Conference (DATE)*, pages 991–996.

[Rathore et al., 2019] Rathore, V., Chaturvedi, V., Singh, A. K., Srikanthan, T., and Shafique, M. (2019). LifeGuard: A Reinforcement Learning-Based Task Mapping Strategy for Performance-Centric Aging Management. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 1–6.

[Read et al., 2011] Read, D., Tewary, V., and Gerstle, W. (2011). 2 - Modeling electromigration using the peridynamics approach. In Kim, C.-U., editor, *Electromigration in Thin Films and Electronic Devices*, pages 45–69. Woodhead Publishing.

[Rohith et al., 2018] Rohith, R., Rathore, V., Chaturvedi, V., Singh, A. K., Thambipillai, S., and Lam, S.-K. (2018). LifeSim: A lifetime reliability simulator for manycore systems. In *Proceedings of the IEEE Computing and Communication Workshop and Conference (CCWC)*, pages 375–381.

[Ruaro et al., 2019] Ruaro, M., Caimi, L., Fochi, V., and Moraes, F. G. (2019). Memphis: a Framework for Heterogeneous Many-core SoCs Generation and Validation. *Design Automation for Embedded Systems*, 23(3-4):113–122.

[Ruaro et al., 2016] Ruaro, M., Lazzarotto, F. B., Marcon, C. A., and Moraes, F. G. (2016). DMNI: A specialized network interface for NoC-based MPSoCs. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1202–1205.

[Ruaro and Moraes, 2017] Ruaro, M. and Moraes, F. G. (2017). Demystifying the cost of task migration in distributed memory many-core systems. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 148–151.

[Sha et al., 2016] Sha, S., Wen, W., Fan, M., Ren, S., and Quan, G. (2016). Performance maximization via frequency oscillation on temperature constrained multi-core processors. In *Proceedings of the International Conference on Parallel Processing (ICPP)*, pages 526–535.

[Sha et al., 2018] Sha, S., Wen, W., Ren, S., and Quan, G. (2018). M-Oscillating: Performance Maximization on Temperature-Constrained Multi-Core Processors. *IEEE Transactions on Parallel and Distributed Systems*, 29(11):2528–2539.

[Sharifi et al., 2013] Sharifi, S., Krishnaswamy, D., and Rosing, T. S. (2013). PROMETHEUS: A Proactive Method for Thermal Management of Heterogeneous MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7):1110–1123.

[Srinivasan, 2006] Srinivasan, J. (2006). *Lifetime Reliability Aware Microprocessors*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 104 pages.

[Srinivasan et al., 2004] Srinivasan, J., Adve, S. V., Bose, P., and Rivers, J. A. (2004). The Case for Lifetime Reliability-Aware Microprocessors. In *Proceedings of the International International Symposium on Computer Architecture (ISCA)*, pages 276–287.

[Tsoutsouras et al., 2018] Tsoutsouras, V., Anagnostopoulos, I., Masouros, D., and Soudris, D. (2018). A Hierarchical Distributed Runtime Resource Management Scheme for NoC-Based Many-Cores. *ACM Transactions on Embedded Computing Systems*, 17(3).

[UEFI, 2021] UEFI (2021). Advanced Configuration and Power Interface (ACPI) Specification. Source: https://uefi.org/specifications.

[Wang et al., 2021] Wang, J., Chen, Z., Guo, S., Li, Y., and Lu, Z. (2021). Optimal Sprinting Pattern in Thermal Constrained CMPs. *IEEE Transactions on Emerging Topics in Computing*, 9(1):484–495.

[Wächter et al., 2011] Wächter, E. W., Biazi, A., and Moraes, F. G. (2011). HeMPS-S: A homogeneous NoC-based MPSoCs framework prototyped in FPGAs. In *Proceedings of the Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–8.

[Xi et al., 2015] Xi, S. L., Jacobson, H., Bose, P., Wei, G., and Brooks, D. (2015). Quantifying sources of error in McPAT and potential impacts on architectural studies. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 577–589.

[Yamamoto and Ababei, 2014] Yamamoto, A. Y. and Ababei, C. (2014). Unified reliability estimation and management of NoC based chip multiprocessors. *Microprocessors and Microsystems*, 38(1):53–63.

[Yang et al., 2017a] Yang, L., Liu, W., Guan, N., Li, M., Chen, P., and Sha, E. H. M. (2017a). Dark silicon-aware hardware-software collaborated design for heterogeneous many-core systems. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 494–499.

[Yang et al., 2017b] Yang, L., Liu, W., Jiang, W., Li, M., Chen, P., and Sha, E. H. M. (2017b). Fotonoc: A folded torus-like network-on-chip based many-core systems-on-chip in the dark silicon era. *IEEE Transactions on Parallel and Distributed Systems*, 28(7):1905–1918.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br