INDIAN INSTITUTE OF TECHNOLOGY INDORE

UNDERGRADUATE THESIS

Thermal Management Methodology for Heterogeneous System

Author:
ANIKET TIWARI
ID No. 150002006

Supervisors:
Dr. KAPIL AHUJA
Dr. AKASH KUMAR

Thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Technology

in the

Department of Computer Science and Engineering

December 4, 2018

Declaration of Authorship

I, ANIKET TIWARI declare that this thesis titled, "Thermal Management Methodology for Heterogeneous System" and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for the BTP project at IIT Indore.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

Certificate

This is to certify that the thesis entitled, "*Thermal Management Methodology for Heterogeneous System*" and submitted by Aniket Tiwari ID No 150002006 in partial fulfillment of the requirements of EE 493 B.Tech Project embodies the work done by him under my supervision.

Supervisor

Dr.KAPIL AHUJA
Associate Professor,
Indian Institute of Technology Indore
Date:

Supervisor

Dr.Akash Kumar
Chair of Processor Design,
Technische Universität Dresden
Date:

"Keep your face always toward the sunshine - and shadows will fall behind you. "

Walt Whitman

INDIAN INSTITUTE OF TECHNOLOGY INDORE

Abstract

Department of Computer Science and Engineering

Bachelor of Technology

Thermal Management Methodology for Heterogeneous System

As hardware platforms are evolving, the transistor density in the system is increasing. This has led to rise in the power density of the system.The power density of the microprocessors have already exceeded the power density of hot plates and some have even reached the power density of a nuclear reactor. This has increased the demand for an effective temperature management system. In this project, I have proposed a methodology to reduce this increase in power density and formation of high temperature zones, also called hotspots. The project has shown two approaches to do so: Proportional-Integral-Derivative Controller (PID) and Model Predictive Controller (MPC). I have also shown a comparison between the two controllers. The model of each of the cores involved in the system are made for determining the temperature of the system considering each core present in the microprocessor as a different entity. This model is made using Extra Randomized Tree Regression. Proposed method also uses workload predictor which uses Support Vector Regression (SVR). The proposed method then uses optimization function (LQR) available in a library of Python to find the operational frequency of the cores of the system.

Acknowledgements

I would like to thank my B.Tech Project supervisors **Dr. Kapil Ahuja** and **Dr. Akash Kumar** for their guidance and constant support in structuring the project and their valuable feedback throughout the course of this project. Their overseeing the project meant there was a lot that I learned while working on it. I thank them for their time and efforts.

I am grateful to **Mr. Tuan Nguyen** without whom this project would have been very difficult. He provided valuable guidance with the ways of implementing the ideas that I had. He also helped me with presentation of the results for academic purposes.

I owe a sincere thanks to the Processor Design lab of cfaed- TU Dresden for providing the necessary hardware utilities to complete the project as well as to IIT Indore for providing me with this opportunity to be involved in systematic research.

Lastly, I offer my sincere thanks to everyone who helped me complete this project, whose name I might I have forgotten to mention.

Contents

List of Tables

List of Abbreviations

DVFS	**D**ynamic **V**oltage and **F**requency **S**caling
PID	**P**ropotional-**I**ntegral-**D**erivative Controller
IPC	**I**nstructions **P**er **C**ycle
IPS	**I**nstructions **P**er **S**econd
Perf	**Perf**ormance Counter
MPC	**M**odel **P**redictive **C**ontroller
SVM	**S**upport **V**ector **M**achine
ETR	**E**xtra **T**ree **R**egression
MKL	**M**ultiple **K**ernel **L**earning
MGmean	**M**ean **G**eometric **M**ean
MSE	**M**ean **S**quare **E**rror
LQR	**L**inear **Q**uadratic **R**egulator
MTTF	**M**ean **T**ime **T**o **F**ailure
NPB	**N**ASA **P**arallel **B**enchmarks

Dedicated to all the learners whose eternal thirst for knowledge keeps them motivated to achieve success.

Chapter 1

Introduction

1.1 Background

Moore's law states that with the development of a new node in semiconductor technology every two years the area occupied by an electronic circuit is reduced by half due to the shrinking size of transistors. The semiconductor industry has followed this trend in their development plans over many decades creating products with ever higher circuit densities and increased functionality.From the circuit design perspective, the industry focus shifted with the rise of mobile and wearable computing from traditional Central Processing Units (CPUs) to System-on-Chips (SoCs). SoCs feature increased functionality by integrating on the same piece of silicon, additionally to Multi-Core CPUs, components such as memory, wireless connectivity (Bluetooth, WiFi, 3G, 4G, LTE), dedicated graphics processors, power management circuits, global positioning system circuits, etc.

The growth and advancement in technology has lead to rise of a new era in the field of microprocessors called the More than Moore era. In this era the transistor size has already decreased more than that predicted by Moore's Law. The advent of 22μm and more recent 5μm technology has lead to much more heat dissipation and an increase in both functional density and power density of the over all system. The chip temperature and cooling techniques available today are already facing major consequences and drawbacks because of the same.

As can be seen in the figure shown , the power density curve for PentiumIII is much higher that hot plate's power density. The current model of intel i7(Ivy Bridge) has been found out to be almost 0.481W/mm2. The newer versions on Intel Skylake and Intel E bridge processors also have similar power densities. I have taken into consideration the high power density of the processors and have implemented it using the already available softwares like HotSpot v6.0.

Increased power density is an important topic for modern planar ICs. Figure 1 shows the power density curve for the planar IC. Power density in Intel CPUs increased nearly exponentially from about 2 W/cm2 for the i386 fabricated in 1.5 μm technology introduced in 1985 to about 100 W/cm2 for the Pentium IV fabricated in 0.13 μm technology introduced in 2000. Today's CPUs and SoCs with transistor counts in the billions reach even higher values.

This causes occurrence of hotspots(Hotspots are regions of especially high computational density that feature heat fluxes that can exceed several kW/cm2) with elevated temperature on the chip.The elevated circuit temperatures degrade gate oxide reliability[1] and cause reduced life-time as well as reduced performance and degraded functionality of the whole chip. In addition since silicon is not a good heat conductor.This makes efficient heat and power management essential.

In this project, I tried to solve this problem by providing a specific methodology which can be easily implemented on a real-life system like the hardware of Odroid-XU4. The management technique was also replicated on the Intel- i5 6500T processor and on Odroid XU3. This ensured the flexibility of the system and also helped me determine that the methodology is independent of the system we are working on.

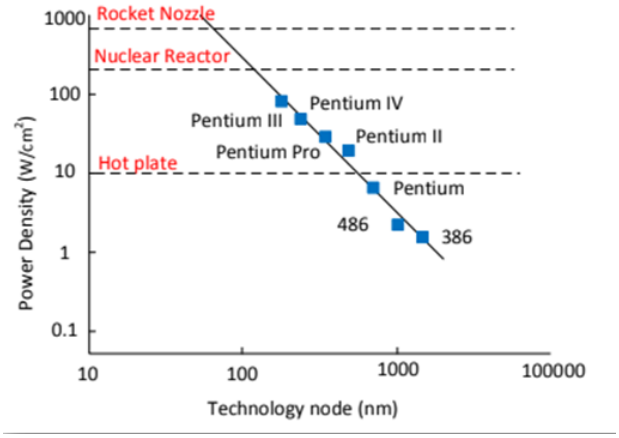

FIGURE 1.1: Trend in Power Density by Intel

The present literature emphasizes the use of DVFS for effective temperature control[2]. This method is more preferable to many other methods as it prevents unnecessary losses in performance. The methodology involves the following step:1)Model making 2)Controller assembling 3)Optimizer configuration 4)Encapsulation

I used the Extra Tree Regression Randomized Tree algorithm to get a model which will help us predict the temperature for the next time sample by taking the input as current temperature and current operational frequency.The workload predictor is made using Support Vector Machine[3]. The optimization used scipy.optimize library of Python[4].

1.2 Related Works

The following section discusses literature pertaining to previously known methods of thermal management for real life application. The techniques were based on DVFS algorithms for the same

1.2.1 Distributed and Self calibrating Thermal Management System

Conventional thermal management algorithms aim to maintain the temperature of the core in the safe region(upto 385K) by changing the frequency of the system. In this technique, Model Predictive controller(MPC) has been used. The schematic of the same has been shown in Figure 2. The author used Linear Model to predict the temperature for the next time instance while taking the temperature and the frequency for the system as input. Each system has its own model for the prediction.

The algorithm calibrates itself at the start of the system. The control is online and had been tested on a software called HotSpot v6.0.[5]. The self calibration routine is a heuristic and uses pessimistic approach for calibration.

1.2.2 Reinforcement Learning

This method used Q-Table algorithm for predicting the next course of action under any situation.The variables that were considered the state of the system were:*Stress* and *Aging*. The action was the allocation of applications and frequency and voltage changing.

Thermal Stress is defined in the terms of thermal cycles. The thermal aging is defined in terms of MTTF. The main focus of the work was to differentiate the case of intra-process switching and inter-process switching. The difference between these processes has been shown in the

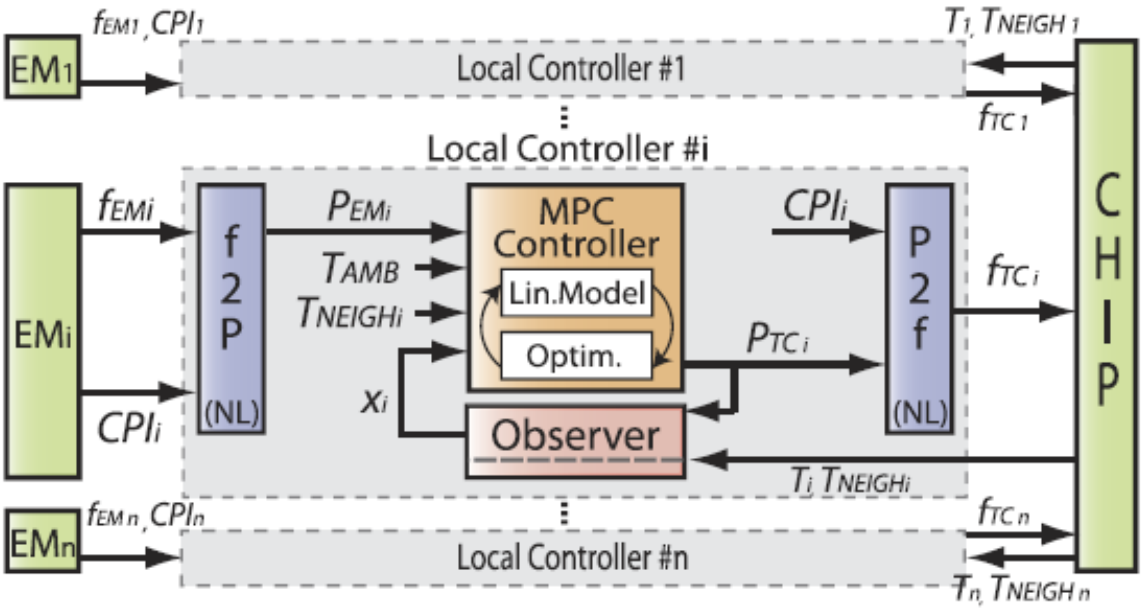

FIGURE 1.2: Model Predictive control by Bartolini et al.

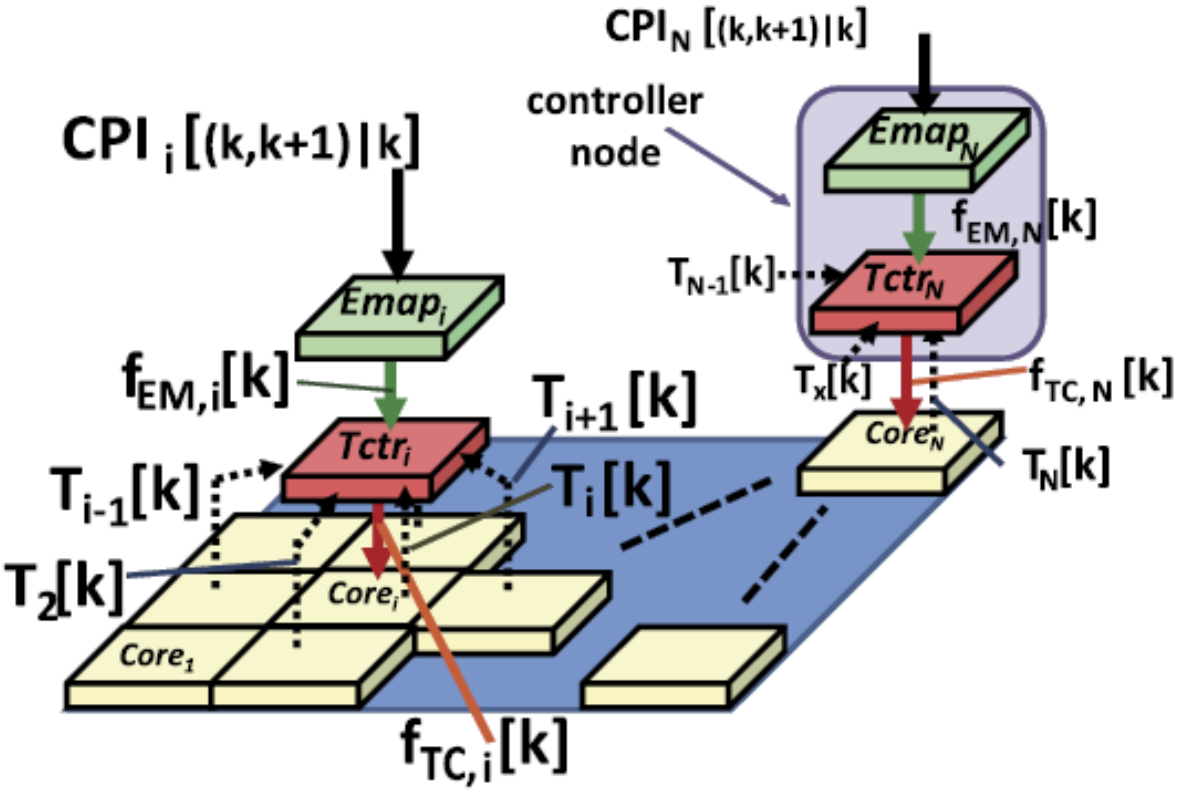

FIGURE 1.3: Tile structure by Bartolini et al.

following graph. The first half shows inter-process scheduling and the second half shows intra-process scheduling.

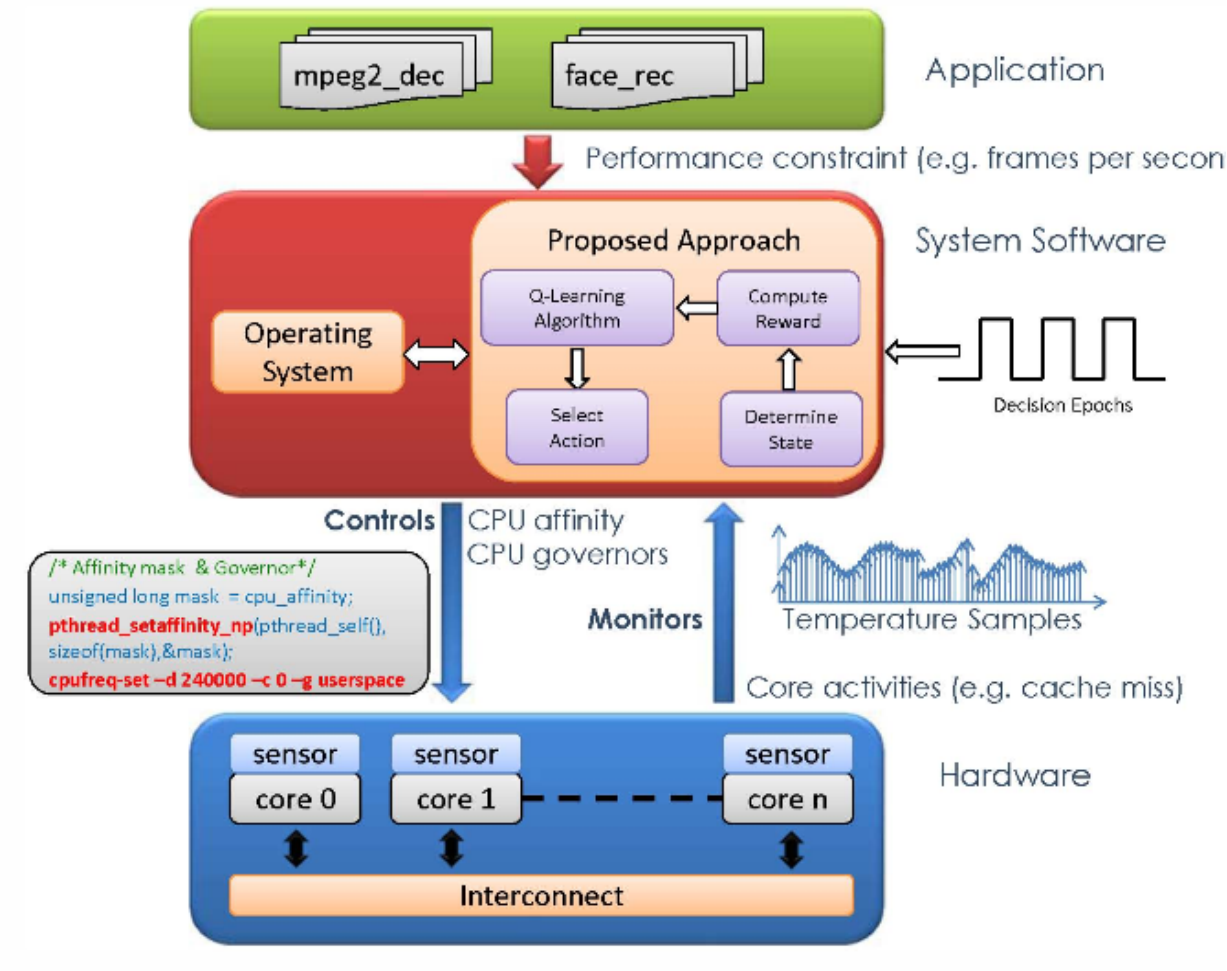

FIGURE 1.4: Model Predictive control by Anup et al.

The most important drawback of this method is that it takes a lot of time to calibrate at each boot time. The system also has to store the value of each action to each state(Q-Table). This increases the memory requirement for the system.

Past Work	Machine Learning	Temperature Measurements	Platform
[1]	Yes	HotSpot	Simulation
[6]	No	HotSpot	Simulation
[7]	Yes	Sensors	Multicore
[8]	Yes	HotSpot	Simulation
[9]	Yes	Thermal Model	Simulation
Proposed	Yes	Sensors	Multicore

1.3 Overall Objective

- Device a methodology to be implemented on a real system. The current literature has most of the techniques been tested and implemented from a simulation software.(As shown in the Table)

- The control should be online control to tackle the case of applications whose profiling have not been done yet. This will also add to the universal nature of the system and methodology.

Chapter 2

Prelimnary Concepts

2.1 Dynamic Voltage and Frequency Scaling

Many studies have focused on power and thermal management of multi-core processors. The most popular approach for processor power optimization is dynamic voltage and frequency scaling (DVFS). In the DVFS technique, the operational voltage and the operating frequency of the IC are changed based on threshold values for circuit temperature or other variables in order to reduce heat and lower the power consumption. In electronic devices, running on battery such as laptops or mobile devices, DVFS is used to lower power consumption and extend battery-life while in high-performance CPUs it is employed to avoid overheating. DVFS has limitations as there is a trade-off between the performance and the power consumption of an IC. Lowering the operational voltage or frequency of cores of an SoC reduces its performance. This can cause delays for time- critical applications such as multimedia, control or gaming. Recently an optimized DVFS technique has been proposed that is supposed to work better under performance constraints.

In this technique, the execution time for each computational task is estimated and taken into account for dynamic voltage scaling in order to better handle time-sensitive applications. While this is an important improvement on the existing DVFS technique, it has to be combined with a smart and fast control system that is scalable in order to be useful for 2D ICs. In addition to the power versus performance trade-off mentioned above, the current DVFS technique faces more challenges. Abrupt changes from low power-mode to the regular-power mode of operation or vice versa increases delay and consumed energy. Furthermore, abrupt changes of the mode of operation of the cores worsen thermal cycles and degrade reliability and life-time of the IC[1]. In order to reduce thermal cycles, frequency and voltage changes in the IC's functional blocks should be smooth and continuous in combination with strategies that allow predicting temperature and workload.

2.2 Model Predictive Controller(MPC)

In order to overcome the issues of threshold-based techniques, the use of a controller is reasonable. Classic proportional-integral-derivative (PID) and linear-quadratic regulator (LQR) controllers do not have predictive abilities and are also not flexible enough for controlling the dynamic behavior of complex multi-core processor architectures [WMW11]. Model Predictive Controllers employ a control algorithm that optimizes the current time-slot while taking future time-slots into account. A model of the system that is controlled is used to optimize a finite time-horizon while implementing the current time-slot. By taking future time-slots into account when making the next control decision, smooth transitions between states of operation can be achieved and sudden changes are avoided. Because they rely on a model of a system they control, MPCs allow making precise control decisions even for complex systems such as multi-core processors. Recent studies have shown that MPC-based thermal controllers for multi-core processors

perform better compared with threshold-based DVFS and LQR control techniques [Zan+11]. For the same temperature capping, MPC-controlled CPUs could handle a higher computational workload, performing 25better compared to threshold-based DVFS systems. For the power and heat management of 3D multi-core architectures, MPCs are therefore the premier choice [10].

The main challenge for the implementation of an MPC is identifying an accurate model of the system to be controlled. As the model is an integral part of the controller feedback, the more accurate the model, the more precise the actions of the controller can be. Unfortunately, a model of the system is not always readily available, especially for multi-core processors. In previous works, the authors used different methods to identify or estimate the model of the chip to be controlled. In [7] the authors used a state-space model which changes based on the chip power consumption depending on the dynamic thermal behavior of the chip. They calibrated their model on an Intel Core i7 940 CPU fabricated in 45 nm technology. Relationships between temperature, performance counters and operating frequency were extracted first. These pre-captured thermal models were stored in a lookup table and k-means clustering techniques were used to relate each new workload during CPU operation to the appropriate thermal model. Based on the selected model the circuit temperature was controlled using the DVFS technique. In this approach, it is necessary to identify the thermal model in the design phase of the controller because the characterization of the thermal model during run-time is time-consuming and uses computational resources. Additionally, this approach is also not easily scalable.

In [6] a model of the CPU was estimated during run-time using the linear relationship between power and DVFS level. The estimated model was used to predict the temperature of a multi-core CPU using an MPC. The model parameters of the MPC were dynamically updated based on the measured power data. On the one hand, the accurate model increases the precision of the controller but on the other hand updating the model at run-time increases the computational load on the CPU. It is estimated that the computational load for this approach increases exponentially with the number of CPU cores. In the proof of concept study , the authors characterized a temperature model for a multi-core system using power input traces. A Linear Time Invariant (LTI) state space model, using a subspace system identification called Numerical Algorithm for Subspace State Space System Identification, was derived. In order to avoid the problem of over-fitting they reduced the order of their model. The authors of [2] reduced the computational complexity of MPC by using the explicit approximation method. Their implementation on an 8-core Niagara-1 system showed a significant reduction in computational load while performance loss related to simplification was negligible. In the studies advantages of decentralized and distributed MPC over the centralized ones are described. The main feature of distributed MPC is that the control system is divided into different local controllers, whichhave some level of communication established between them.

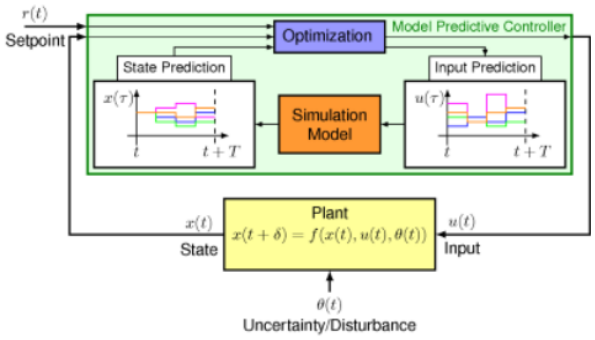

FIGURE 2.1: Generic Model Predictive control

For this project, we plan to combine the advantages from different approaches. We plan to follow a decentralized or distributed MPC approach to reduce model complexity and ensure fast

run-time. Each subsystem of the multi-core SoC will be modeled by the power-series expansion of its power in terms of DVFS value and computational load. The complete model will be tested by running dynamic 3D thermal simulations of the multi-core SoC that we use as a proof of concept platform. The proof of concept SoC is based on ARMs big.LITTLE architecture and features eight cores. By using thermal simulations of the multi-core SoC, we can adjust the parameters for each subsystem to reflect heat distribution in the system under load well. Interaction terms will be included to reflect the interdependence of subsystems. Once the simulations have confirmed the model, we will use empirical data measured on the multi-core SoC by running test applications to fine tune the model parameters. A sample for MPC is given in the figure below:

2.3 Extra Randomized Tree Regression

This is a machine learning algorithm which is a type of ensemble algorithm. An ensemble algorithm is an algorithm which is mixture of more than one algorithms. The method is similar to decision tree regression, where data is divided into subsets in order to maximize the prediction accuracy. In extra tree regression, the decision of tree formation is completely randomized and the best tree is chosen for higher accuracy. The code was a python code and the ExtratreeRegression function of sklearn library in python was used for making this model.

The mathematical equations for the same are as shown: For node , representing a region with observations, common criteria to minimise as for determining locations for future splits are Mean Squared Error, which minimizes the L2 error using mean values at terminal nodes, and Mean Absolute Error, which minimizes the L1 error using median values at terminal nodes.

Mean Square Error:

$$c_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - c_m)^2$$

Mean Absolute Error:

$$\bar{y_m} = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y_m}|$$

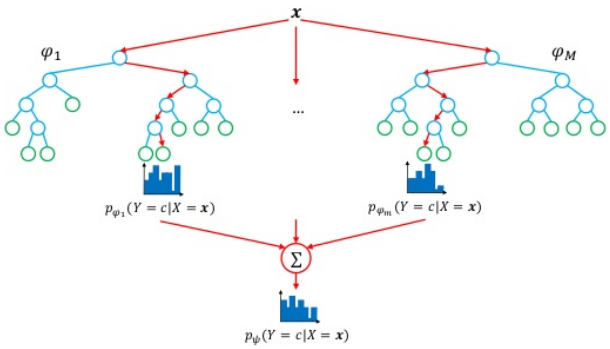

FIGURE 2.2: Extra Randomized Tree Regression

In extra randomized tree regression, the sample data set is distributed randomly into many subset with each of these subsets being used for training the model. These models are of tree type as has been explained previously. The final result of the algorithm is the arithmetic mean of the resulting models. This property has many advantages and disadvantages.

Advantages:

- It gives better results for high dimensional sample space
- Takes into account the property of randomness in the input data

Disadvantages:

- It has a chance of over-fitting of a large number of subtrees is chosen
- The error for deterministic data is quite high

Chapter 3

PID Implementation

The basic idea of the controller is as shown in the diagram below:

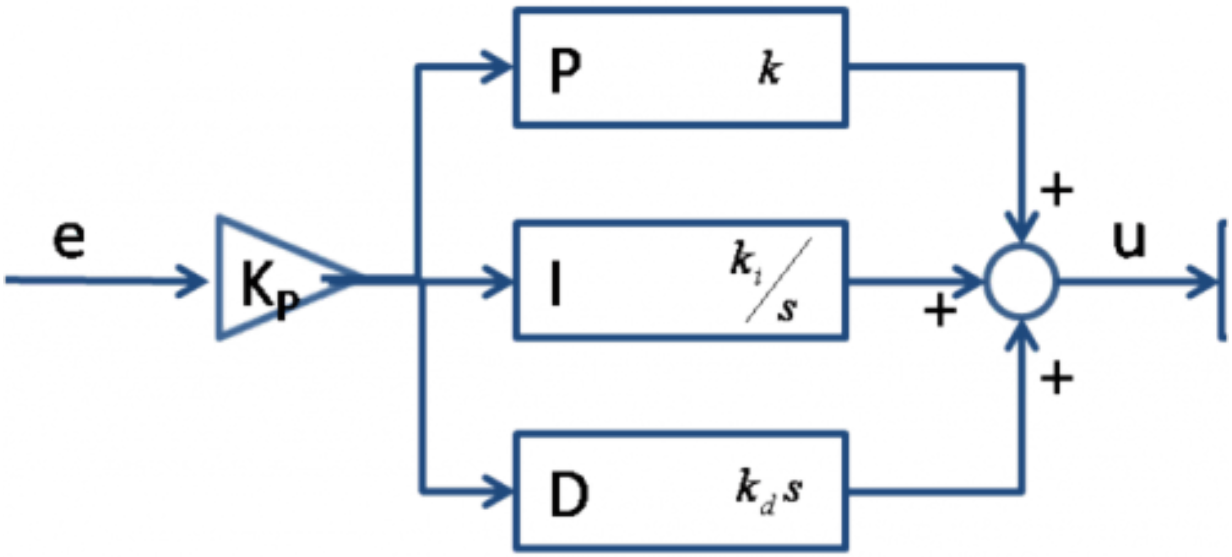

FIGURE 3.1: Basic design of PID controller

Before trying the idea of PID , it was essential to try it on HotSpot Simluation Software. PID has been said as not powerful enough to control the temperature of the system in previous works[2][10]. So, we tried it on simulation.

3.1 Implementation on Simulation

We tried to take a 3x3 core floorplan for the experiment and tried reducing the temperature from the given example in HotSpot example suite. The figures shown below shows heatmaps of the core system before and after implementation of PID controller on the system.

The standard way of tuning a PID system is that the values for Derivative constant and Integral constant are left as zero and the value of Proportionality constant is increased gradually till an oscillating output is obtained. We the call this value of Proportionality constant as **Kp** and the time period of oscillation as **Ti**.

There are two methods of obtaining the other parameters of the PID controller and they are as follows:

The graphs obtained from both are shown below:

As we can see from the graphs that the overshoot value for Tyreus-Luyben method is less compared to its counterpart and hence we choose to use it.

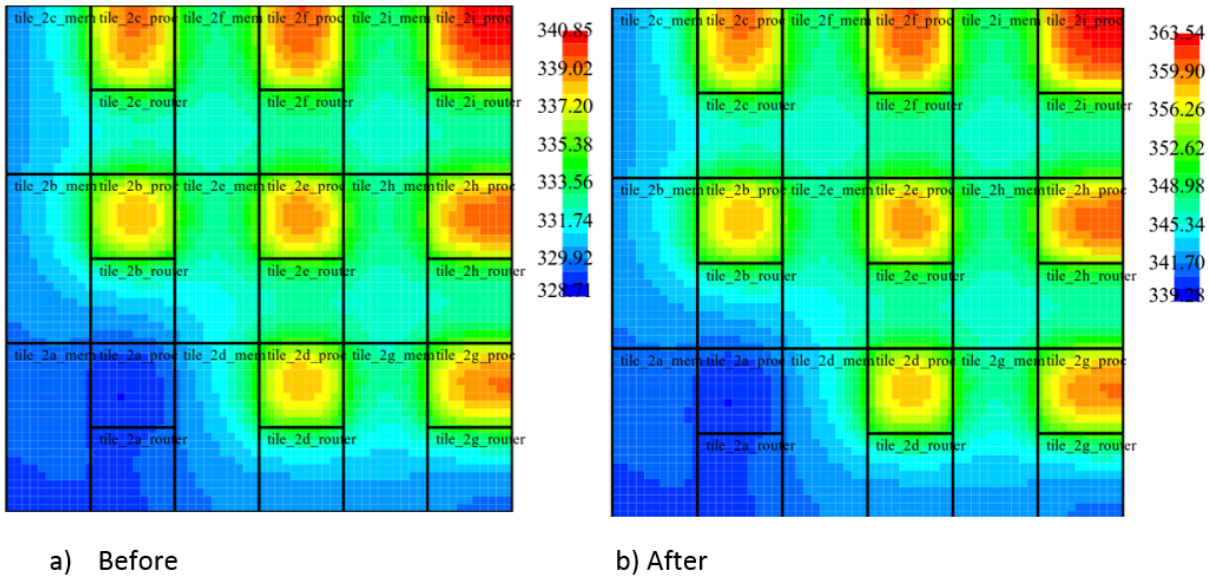

FIGURE 3.2: Effect of PID controller

Method	Kp	Ti	Td
Tyreus-Luyben Method	Ku/2.2	Tu*2.2	Tu/6.3
Zeigler Nichols Method	Ku*0.2	Tu/2	Tu/3

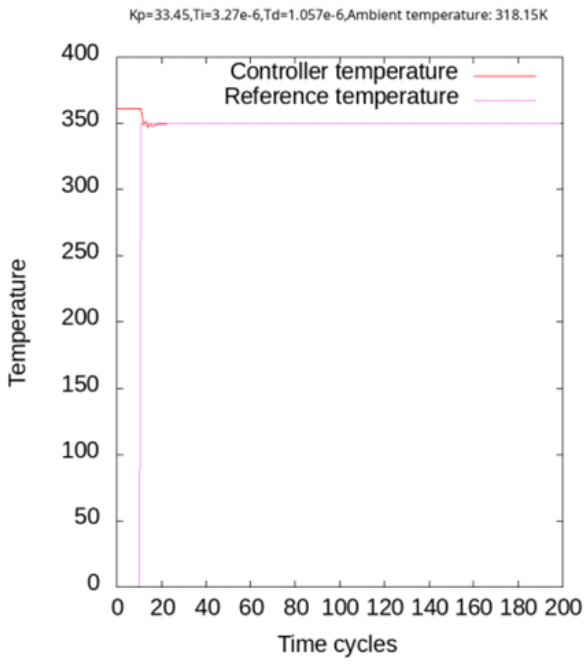

FIGURE 3.3: PID with Tyreeus-Luyen tuning

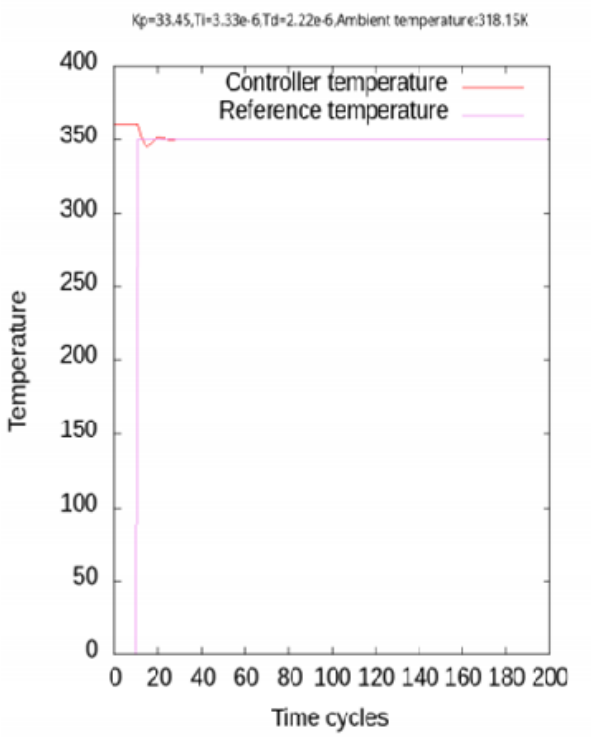

FIGURE 3.4: PID with Zeigler-Nichols tuning

3.2 Implementation on Odroid XU4

The next step was implementing the PID on Odroid XU4 board as it proved to be working on the system. The tuning of the system was done as shown in the last section. The tuned PID was then tested on Nasa Parallel Benchmarks(NPB)[11]. These benchmarks are standardized benchmarks for the purpose of application profiling. We have the results as shown in the figure.

As can be seen in the graph obtained for the application the temperature of the board goes to a high of 95^{o}C without any thermal management policy and remains above 85^{o}C for atleast 10 seconds. It can also be clearly seen that due to the effect of the thermal management policies the temperature of the system goes down considerably. This justifies the effective working of the system.

The PID method does have some drawbacks. To mention a few:

- The system conditions like moisture content, ambient temperature ,etc. do play an important role in determining the temperature curve of the system. This may require repeated tuning.
- The PID requires some time to stabilize which may cause performance degradation and overhead in some cases.
- Due to the causal nature of the control system the possibility of an upcoming thermal emergency can not be discovered hence increasing the safety risks involved.

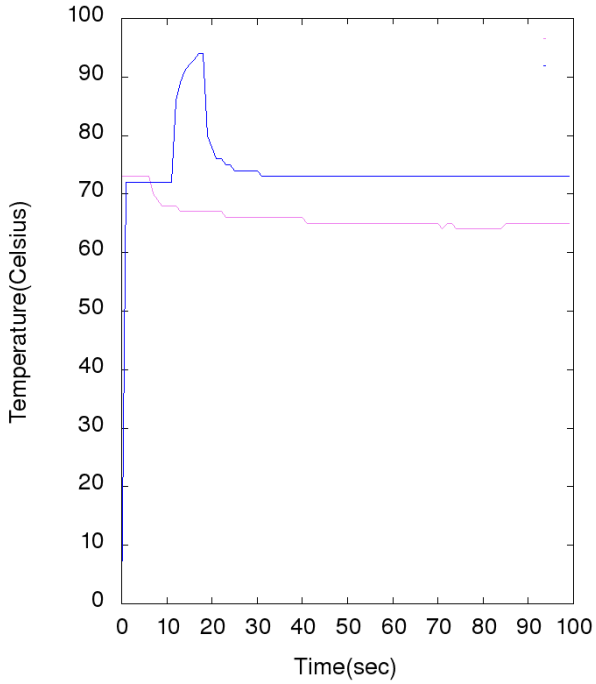

FIGURE 3.5: Temperatures for Application UA of Class W in NPB

Chapter 4

MPC Implementation

The MPC or Model Predictive Controller have been around since a few decades with their uses varying from the oil field industries to the day to day electronics working and functioning. The recent advancement of these controllers in industry have caused a significant increase in their application on different fields. The poosibility of these controllers being used has been explored already in works like [2].

The basic steps would be:

- Formulation of the system model of the system
- Making a workload predictor for the system
- Making an optimizer for the system
- Encapsulating all the above to form the final controller

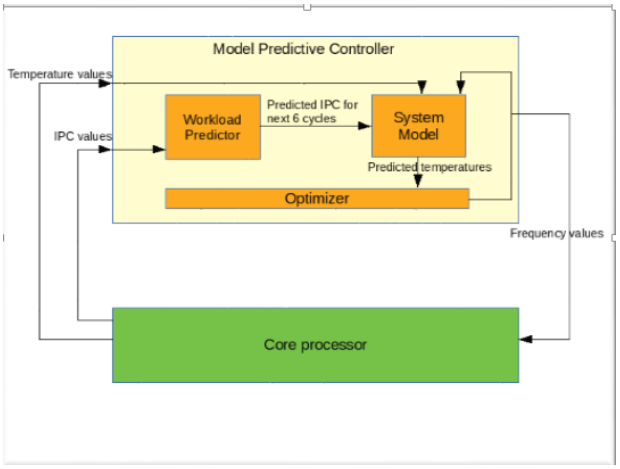

FIGURE 4.1: Proposed MPC Model

4.1 Formulation of System Model

We know that the temperature of a system is directly proportional to the power dissipated in the system[9]. The relation between power and other physical properties of the system and the properties of the application is [6].

$$P = k_A freq \cdot V_{DD}^2 + k_B + (k_C + k_D freq) \cdot CPI^{k_E}$$

Where k_A,k_B,k_C and k_D are dependent only on the physical properties of the system. The value V_{eff} is the input voltage of the transistors and CPI is clocks per instructions. In today's age and of multithreading applications, the term CPI has been replaced by Instructions per Cycle(IPC). We will be using the Perf tool to measure the value of IPC and we shall be using the temperature sensors present onboard for measuring the temperature.

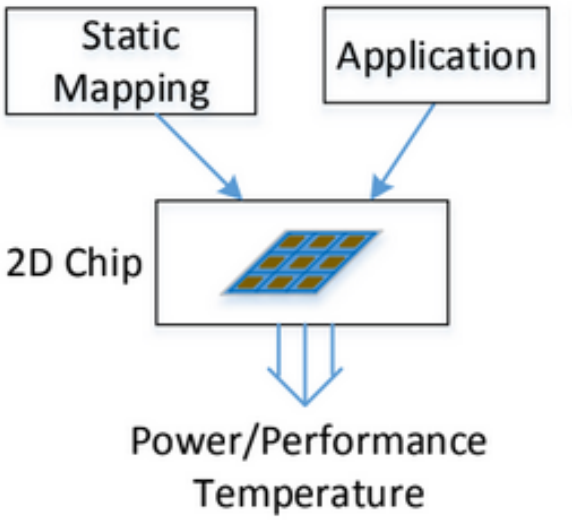

FIGURE 4.2: Proposed System Model

The following were the steps involved in making the model:

- Selecting applications from the NSB Benchmarks which represent tha maximum range of applications possible.
- Collecting values of IPC , Frequency of the cores and temperature of the core for a fixed interval of time for all possible permutations of active core and frequency values
- Making the model using ML algorithms and other curve fitting tools

I tried to make the model using the sklearn library of Python. The following algorithms gave results with more than 85accuracy:

- Extra Randomized Decision Tree Regression (accuracy:90.4
- Decision Tree Regression (accuracy:87
- Linear Regression (accuracy:86

We choose Extra Randomized Tree Regression for the Model Making Task.

An Active core is that core of the board on which the application is runnig while an inactive core is the one on which no application is ruuning.The graphs shown below give an estimate of how well does the System Model perform.

As can be easily inferred from the graphs above, the prediction error is very less in both the cases of active and inactive core (never exceeds 4^oC in any case)

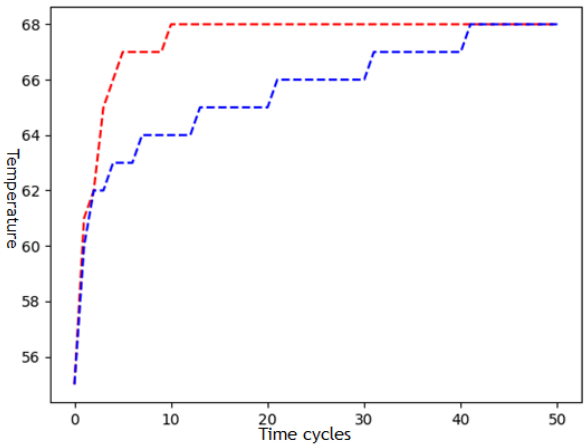

FIGURE 4.3: Prediction for Inactive Core

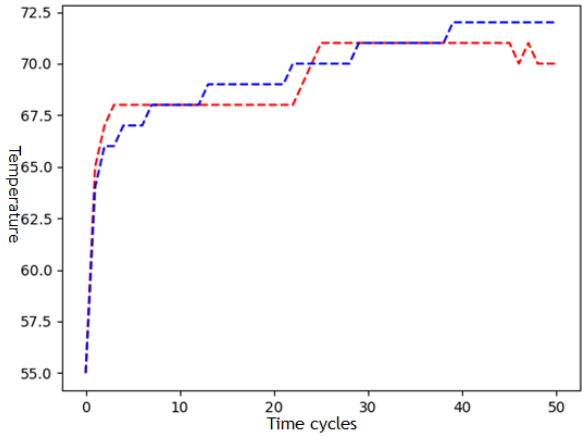

FIGURE 4.4: Prediction for Active core

4.2 Workload Predictor

The workload predictor is made using Support Vector Regression as has been proven effect in [3]. The techniques used in Vapnik's ϵ sensitive method.A tube with the width of ϵ is considered about the predicted function. The values outside of the tube are penalized for the error and the values inside the tube are desirable. The value of the output(y_i) is a linear function of input (x_i). As we have to penalize any value outside of the tube, the loss function will be :

$$L_{\varepsilon SVR}(t,y) = \begin{cases} 0 & if\ |t-y| \leq \varepsilon \\ |t-y| - \varepsilon & otherwise \end{cases}$$

Hence by optimizing the above constraints, we will get the SVR.

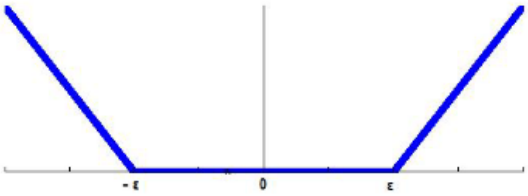

FIGURE 4.5: Epsilon Loss Function

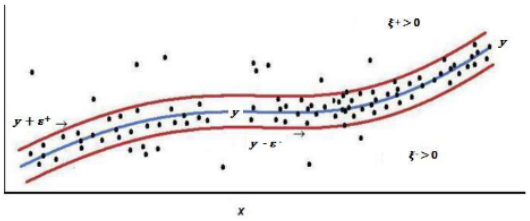

FIGURE 4.6: SVR Tube and prediction error

The workload prediction done using this method has been tested in SPEC 2006 benchmark. The following figure shows the available counters being utilized.

The analysis on SPEC 2006 Benchmark showed the accuracy to be 86The analysis was also carried on by me for NSB benchmarks(UA class W, EP class A and BT class A). The average accuracy was of 80The main focus of workload predictor is to prevent or warn about any upcoming thermal emergencies and as can be seen later, this goal is successfully achieved.

4.3 Optimization

The literature has cases of optimization being chosen for the MPC implementation on Embedded Systems platform. The popular techniques are :

- Linear Quadratic Regulator
- Look up Table
- Gradient descent method

1	CPU_CLK_UNHALTED.REF_TSC
2	CPU_CLK_UNHALTED.THREAD
3	CYCLE_ACTIVITY.CYCLES_L1D_PENDING
4	CYCLE_ACTIVITY.CYCLES_L2_PENDING
5	CYCLE_ACTIVITY.CYCLES_MLC_PENDING
6	CYCLE_ACTIVITY.CYCLES_NO_DISPATCH
7	CYCLE_ACTIVITY.L1D_PENDING
8	CYCLE_ACTIVITY.L2_PENDING
9	CYCLE_ACTIVITY.MLC_PENDING
10	CYCLE_ACTIVITY.NO_DISPATCH
11	CYCLE_ACTIVITY.STALL_CYCLES_L1D_PENDING
12	CYCLE_ACTIVITY.STALL_CYCLES_L2_PENDING
13	FP_COMP_OPS_EXE.X87
14	FREERUN_CORE_C3_RESIDENCY
15	FREERUN_CORE_C6_RESIDENCY
16	FREERUN_CORE_C7_RESIDENCY
17	INST_RETIRED.ANY
18	INST_RETIRED.X87
19	LONGEST_LAT_CACHE.MISS
20	LONGEST_LAT_CACHE.REFERENCE
21	MEMORY_INSTR.ALL_LOADS_AND_STORES
22	MEMORY_INSTR.CACHEABLE_LOADS_AND_STORES
23	OFFCORE_REQUESTS.ALL_DATA_RD
24	OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.REMOTE_DRAM_0
25	OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.REMOTE_DRAM_1
26	UNC_C_CLOCKTICKS
27	UNC_C_LLC_LOOKUP.DATA_READ
28	UNC_C_LLC_LOOKUP.WRITE
29	UNC_M_CLOCKTICKS
30	UOPS_DISPATCHED.STALL_CYCLES
31	UOPS_ISSUED.CORE_STALL_CYCLES
32	UOPS_ISSUED.STALL_CYCLES
33	UOPS_RETIRED.STALL_CYCLES
34	UOPS_RETIRED.TOTAL_CYCLES

FIGURE 4.7: Counter Types

It has been found out that Look up Table method is the fastest method in application . The drawback being the we will have an offline control in case of Look up table method and I wanted to get an online Thermal Management System. Thus exploring the next best option, I tried implementing the LQR optimization with the help of Scipy library[4] The optimization equation was of the form:

$$\text{Objective function: } J = \Sigma^{4}_{i=0}(w1(T_i - T_{thres}) + w2(freq_i - freq_{MAX})^2)$$

$$\text{constraints: } \quad 0.2 \leq freq \leq 2$$
$$T_i < T_{threshold}$$

where:

- $freq_i$ is the operational frequency of the cores.
- $freq_{MAX}$ is the maximum operational frequency for the cores.
- T_i is the temperature at the time instances.
- $T_{Threshold}$ is the threshold temperature to avoid thermal emergencies.
- $w1$ and $w2$ are constants whose value is independent of the application being run

The scipy library was not able to respect the constraints as they are non-linear. Due to this reason, I set the threshold temperature lower than the actual threshold temperature for the performance to degrade. The penalizing in the equation helps in avoiding thermal emergencies.

The second limitation of Scipy library is that it takes time to solve which can be easily overcome by decreasing the complexity of system model.

4.4 Encapsulating the code

The final code is divided into two parts. The interface to collect data from the system and reading and writing to the system registers. The second half is implementation of the whole MPC.The codes are in Python3. The libraries used in the project are:

- numpy
- pandas
- pickle
- scipy
- sklearn

Chapter 5

Experiments and Limitations

This chapter deals with the experimentation and actual implementation on the Odroid XU4 board. The following are the graphs for the thermal management system obtained at the end of execution of the methodology. The Odroid XU4 board also have some limitations:

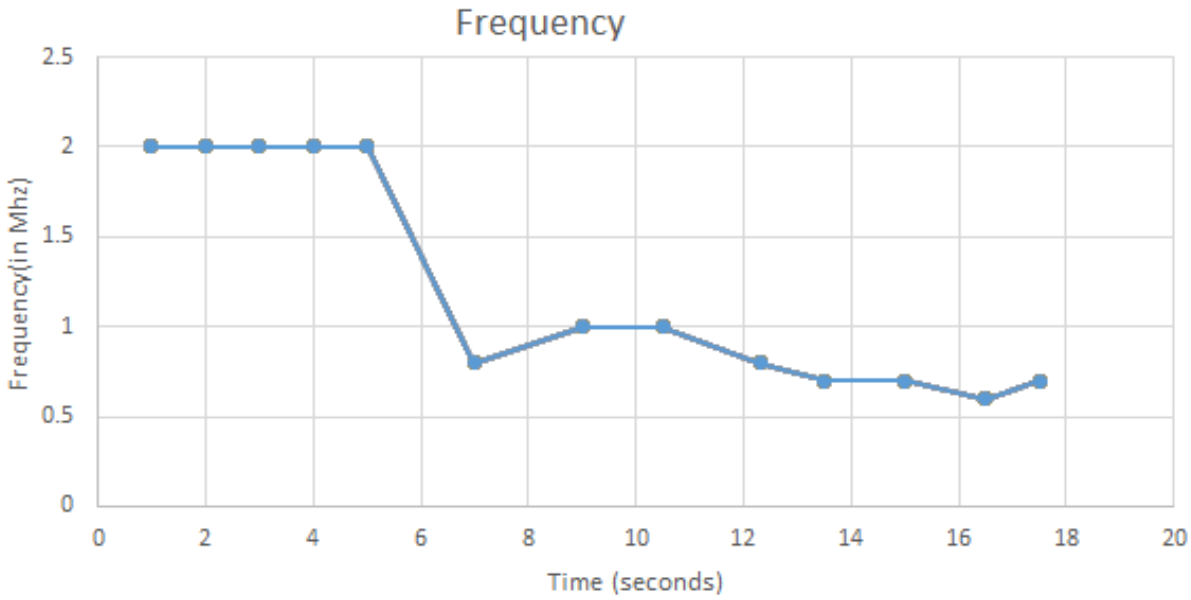

FIGURE 5.1: Frequency Variation with time

- The board has Exynos5422 chip which has big.LITTLE architecture. The big cores are ARM Cortex A15 cores which are performance intensive while the LIITLE cores are ARM Cortex A7 cores which are power efficient. As there is complex calculations involved in the MPC, and float point functionality is only present in A15, the controller has to be mapped in A15, or big , cores only
- The board does not have individual hardware knob for each controlling the frequency of each core. This means that the system can no longer be treated as purely heterogeneous system. However, for a heterogeneous system, the steps shall not change much.
- The temperature sensors are present only on the big cores and not on the LITTLE cores. This limits the ability to get an accurate model for the LITTLE cores

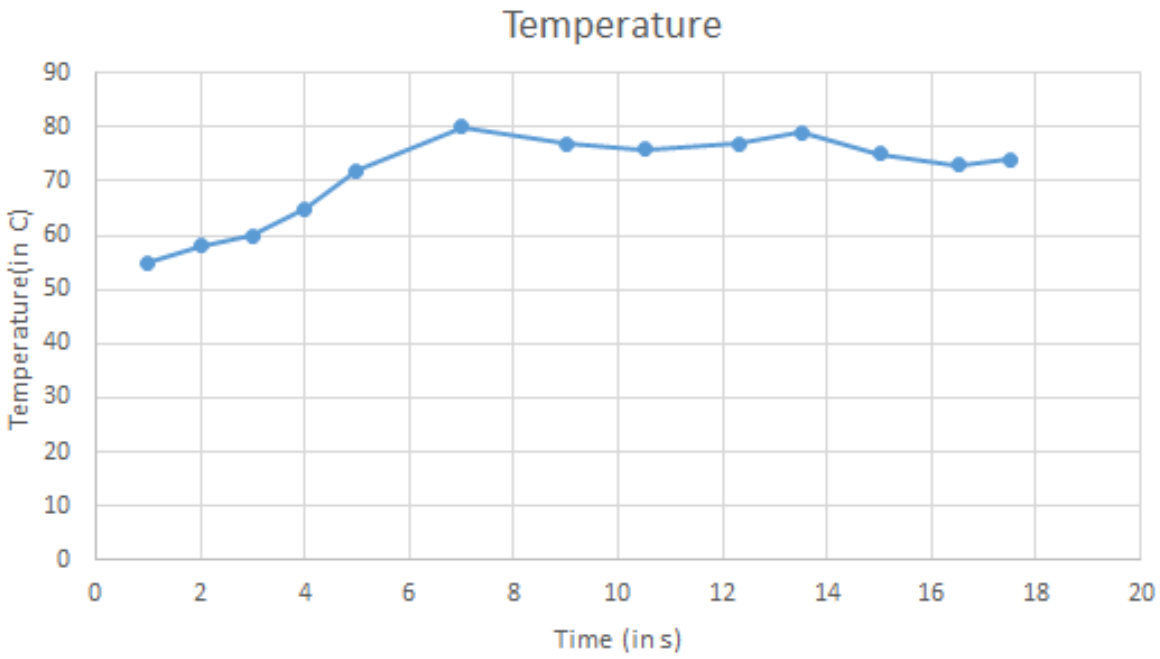

FIGURE 5.2: Temperature Variation with time

Chapter 6

Conclusion and Future Work

The objectives of this project were to

- Introduce a new and novel thermal management system for heterogeneous systems
- Provide a step by step methodology to implement it
- Evaluate the working of the thermal management system

In this project I have proposed a new methodology and a new thermal management system for controlling the high temperature. The previous methods were either implemented on a software or they were offline methods which were not adaptable to any new set of applications which should be processed.

I also prepared an alternative methodology of using PID in place of MPC which will ease the load on the processors and make sure that the temperature rise in the cores is not due to the workload of the system. The methodology has been tried on Intel i5 6500T which gives similar results and helps to prove the universal nature of the methodology.

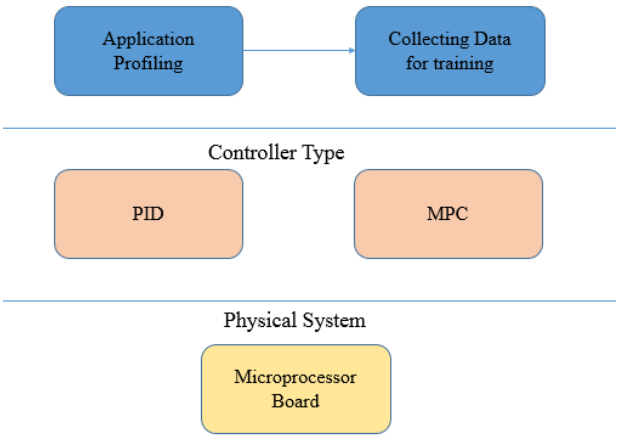

FIGURE 6.1: Final Methodology

The future work in this domain would be to explore the flexibility of the methodology on different scenarios like those of Mixed Criticality Systems and Mixed Priority Systems. It will help us evaluate as to how good is the methodology performing in presence of extra factors.

Bibliography

[1] A. Afzali-Kusha M. Pedram. M. Kamal A. Iranfar. ""A thermal stress-aware algorithm for power and temperature management of MPSoCs"." In: *Date'15* 99 (2015), pp. 954–959.

[2] Francesco Zanini. "Design of Thermal Management Control Policies for Multiprocessors Systems on Chip". In: *PhD Thesis* (2011).

[3] Mel Stockman ; Mariette Awad ; Haitham Akkary ; Rahul Khanna. "Thermal status and workload prediction using support vector regression". In: *Energy Aware Computing* (2012), pp. 1–5.

[4] *Scipy Library*. 2018 (accessed October 12,2018). URL: https://docs.scipy.org/doc/scipy/reference/index.html.

[5] *HotSpot v6.0*. University of Virginia. 2015 (accessed May 24,2018). URL: http://lava.cs.virginia.edu/HotSpot/index.htm.

[6] Kenny C. Gross Ayse Kivilcim Coskun Tajana Simunic Rosing. "Temperature management in multiprocessor SoCs using online learning". In: *DAC* (2008), pp. 890–893.

[7] Geoff V. Merrett Bashir M. Al-Hashimi Akash Kumar Bharadwaj Veeravalli Anup Das Rishad A. Shafik. "Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems". In: *DAC* (2014), pp. 1191–1199.

[8] Matteo Cacciari Andrea Tilli Bartolini Andrea and Luca Benini. "Thermal and Energy Management of High-Performance Multicores: Distributed and Self-Calibrating Model-Predictive Controller". In: *Transactions on Parallel and Distributed Systems* 24 (2013), pp. 170–183.

[9] Hamayun Khan ; Qaisar Bashir ; M. Usman Hashmi. "Scheduling based energy optimization technique in multiprocessor embedded systems". In: *ICEET* (2008), pp. 1–8.

[10] Qinru Qiu Yang Ge. "Dynamic Thermal Management for Multimedia Applications Using Machine Learning". In: *DAC* (2011), pp. 95–100.

[11] NASA. *Nasa Parallel Benchmarking*. 2009 (accessed October 12,2018). URL: https://www.nas.nasa.gov/publications/npb.html#url.