

# Thermal aware task assignment for multicore processors using genetic algorithm

Mohammed Parwez, Diary R. Sulaiman

Department of Electrical Engineering, College of Engineering, Salahaddin University, Erbil, Iraq

---

## Article Info

### Article history:

Received Sep 16, 2022

Revised Jan 31, 2023

Accepted Feb 4, 2023

---

### Keywords:

Gem5

Genetic algorithm

McPAT

Multicore processor

Thermal management

---

## ABSTRACT

Microprocessor power and thermal density are increasing exponentially. The reliability of the processor declined, cooling costs rose, and the processor's lifespan was shortened due to an overheated processor and poor thermal management like thermally unbalanced processors. Thus, the thermal management and balancing of multi-core processors are extremely crucial. This work mostly focuses on a compact temperature model of multicore processors. In this paper, a novel task assignment is proposed using a genetic algorithm to maintain the thermal balance of the cores, by considering the energy expended by each task that the core performs. And expecting the cores' temperature using the hotspot simulator. The algorithm assigns tasks to the processors depending on the task parameters and current cores' temperature in such a way that none of the tasks' deadlines are lost for the earliest deadline first (EDF) scheduling algorithm. The mathematical model was derived, and the simulation results showed that the highest temperature difference between the cores is 8 °C for approximately 14 seconds of simulation. These results validate the effectiveness of the proposed algorithm in managing the hotspot and reducing both temperature and energy consumption in multicore processors.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



---

## Corresponding Author:

Diary R. Sulaiman

Department of Electrical Engineering, College of Engineering, Salahaddin University

Kirkuk Road, Erbil, Iraq

Email: diary.sulaiman@su.edu.krd

---

## 1. INTRODUCTION

Nowadays, the difficulties with thermal management in modern multi-core central processing units (CPUs) become increasingly significant and essential hence, one of the difficulties faced by electrical and computer designers is the thermal management of integrated circuits (ICs), as IC performance and reliability are impacted by temperature. According to studies in [1] the IC lifetime can be reduced by 50% for every 10 °C to 15 °C increase in the IC peak temperature hence, many research and articles are focusing on this issue. One of the methods of controlling chip temperature is consisting of active cooling integration (e.g., fan cooling, water circulation, and oil cooling and heat pipe). This technique is not always suitable for embedded systems especially those with limited size and battery. Another method of chip thermal management is a program management approach in which the temperature of the central processing unit (CPU) can be balanced by assignment of tasks to the CPU cores according to the cores' thermal spot. This method doesn't have a limitation of the method as mentioned above.

Many polished articles studied the reduction of the peak/overall core temperatures by program management such as Rubio-Anguiano *et al.* [2] proposed a scheduler consisting of two stages: offline stage and online stage. In the offline stage, a minimum clock frequency is calculated that fulfills the deadline and partitioning scheme. And in the online stage, a fixed-priority zero laxity policy is applied as a global task

allocation. The online stage scheduler in which accepts or rejects soft real-time aperiodic tasks selecting the upper lowest available frequency to minimization on power consumption while meeting time and thermal constraints. Also, Rodriguez and Yomsi [3] proposed a framework that captures both the temporal and thermal behavior of the system after scheduling tasks by one of the fixed priority algorithms like Deadline Monotonic orate Monotonic.

Many works tried to control CPU temperature reduction by using bio-inspired algorithms as such, Rupanetti and Salamy [4] proposed a novel strategy for task migration in multiprocessors by combining modified ant colony algorithm (ACO) and first-fit task allocation heuristic. The ACO algorithm tries to split tasks and migrate them to processors with low task utilization to minimize the overall power consumption of the multiprocessor system on a chip (MPSoC). Then tasks are scheduled by the earliest deadline first algorithm. In the same way, many other works use bio-inspired algorithms for controlling CPU thermal as it is done in [5], [6]. Regarding CPU frequency, Singh and Thangaraju [7] in their article labeled running processes or jobs as cold and hot processes. And they introduced a hybrid frequency scaling governor to reduce the overall power of the platform. This was caused by temperature reduction of the cores from 85 °C to 47 °C according to their experimental results. Li *et al.* [8] tried to minimize both the temperature and energy consumption of heterogeneous MPSoC by proposing two phases of task scheduling. First phase tasks are assigned to the processor while thermal and power dissipation factors are taken into account. And the second phase deduces the thermal/energy optimal speed assignment for tasks by considering the heterogeneity of both processors and tasks. For controlling peak temperatures, Jayaseelan and Mitra [9] proposed two techniques to control the peak temperature of the chip. First, they analyzed the peak temperature of the repeating task sequence and develop an optimal sequence of the tasks to minimize the peak temperature. The second proposed technique is the iterative algorithm that combines task sequencing with voltage scaling to further lower the peak temperature while satisfying the timing constraints. Some researchers are attempting of reducing the temperature by utilizing the dynamic voltage frequency (DVFS) technique frequency as [10]–[13] used DVFS technique to reduce power consumption by lowering chip voltage and frequency accordingly as a result the CPU temperature is reduced. And Durand and Lesecq [14] analyzed the nonlinearity between power and temperature. And they used a technique that implements a chopped scheme on top of a robust DVFS approach in order to prevent increasing temperature. Unfortunately, the DVFS technique has a limited impact on temperature and causes CPU performance reduction, as the execution time increases by lowering frequency [15]. Another way of reduction of CPU temperature is also can be performed by using adaptive supply and body voltage control technique as Sulaiman *et al.* [16] that they used particle swarm algorithm (PSO) combined with the pareto front (PF) to determine the optimal solution of threshold and supply voltage ( $V_{th} - V_{dd}$ ) that caused by thermal reduction rages from 8 °C to 12 °C for each body bias strep voltage. Also, they used adaptive supply and body voltage control in [17] to compensate the threshold voltage and clock frequency for ultra-low power design by supplying optimal  $V_{dd}$  and NMOS-PMOS body bias voltages ( $V_{BB} - N$  &  $V_{BB} - P$ ) to the microprocessor unit and results in a power saving up to 20% and thermal reduction in a range of 8 °C for each body bias step voltage. CPU temperature also can be reduced by CPU floor planning as it is done by Xie and Hung [18] utilized a floor planning technique for peak temperature reduction and thermal balancing on the CPU.

This article presents the analysis of the proposed task assignment algorithm using a natural inspiration genetic algorithm for thermal balancing in multi-core processors by taking the temperature of each core and then assigning tasks depending on the current cores' temperature and the expected energy consumption of the tasks using the proposed assignment algorithm. The algorithm analyzes the behavior of different core temperature states while considering the energy expended by each task that the core performs including parallel application tasks. The results of the performance achieved with its application in different simulation environments are analyzed and compared to thermal unbalanced approaches. The proposed approach collects information about the real hotspot of all cores besides the task mapped and executed by each core on each processor to decide the next task mapped and assigned for each core. This approach decides to dynamically move tasks between light and heavy types so that the temperature difference between the cores reduces. The results validate the effectiveness of the proposed algorithm in managing the hotspot and reducing both temperature and energy consumption in multicore processors in high-performance computer systems.

The rest of the paper is organized as: section 2 explains the theoretical background and relation between power and temperature by algebraic equations. In section 3 system models are explained which consist of power and task modeling. Section 4 talks about the assignment algorithm in detail by expressing the pseudo-code at the end of the section. the simulation and results of two cases (equal cores' initial temperature and different cores' initial temperature) are illustrated and discussed in section 5. Finally, the conclusions are expressed in section 6.

## 2. THEORETICAL BACKGROUND

Overall, this paper introduces the use of thermal RC-modeling as an accurate efficient modeling system for temperature estimation in multi-core processors considering  $P_i$  is the average power of a task with an execution time  $t_i$ , the temperature rise on each core caused by task ( $i$ ) is  $T_i$  that can be expressed as (1) [15],

$$T_i = P_i * R + T_A - (P * R + T_A - T_{initial})e^{-t_i/RC} \quad (1)$$

The temperature rise according to the next task can be stated as (2) [15],

$$T_{i+1} = P_{i+1} * R + T_A - (P * R + T_A - T_i)e^{-t_{i+1}/RC} \quad (2)$$

where:

$P_i$  is the average power of Task ( $i$ ).

$t_i$  is the execution time of Task( $i$ )

$T_A$  is ambient temperature.

$T_{initial}$  is the initial temperature.

$R$  is thermal resistance.

$C$  is thermal capacitance.

From the above equations, we can conclude that four factors have an impact on the core's transient temperature: average power of the executed task on the core, initial temperature, execution time, and the number of assigned tasks on the core. Thus, if we execute more tasks on a core, the temperature rise of the core will be the sum of the power and execution time of each task.

If the periodic tasks are mapped as ( $Task_1, Task_2, Task_3, \dots, Task_n$ ) to be executed by a multicore CPU ( $C_1, C_2, C_3, \dots, C_n$ ). the problem is how to assign Tasks in such a way that keeps the temperature of the cores balanced. This can be done by performing an assignment algorithm which takes the cores' temperature and task parameters as input, and implementation of the assignment algorithm goal is the core with the highest temperature take over a set of tasks which their energy summation is less, compared to the energy sum of other cores' assigned task set by a factor which's covered in the next sections. And the same way for the core with the second-highest temperature. The algorithm will continue until the assignment of the task-set for the coldest core.

## 3. THE PROPOSED SYSTEM MODEL

### 3.1. Power model

In complementary metal-oxide-semiconductor (CMOS) ICs which are building block of modern processors, power dissipation can be classified generally into two aspects; dynamic power and static power dissipation. The dynamic power dissipation is caused by the charging and discharging of the transistor's junction capacitors and the short interval short circuit during toggling between P-MOS and N-MOS. However, the static power is dissipated because of leakage current through reverse biased junctions of the transistor [19]. The dynamic power dissipation ( $P_d$ ) can be expressed (3) [20],

$$P_d = C_{eff} \cdot V_{dd}^2 \cdot f \quad (3)$$

where:

$V_{dd}$  is the supply voltage,

$C_{ef}$  is the effective capacitance and

$f$  is the clock frequency.

And the frequency can be represented by (4) [21],

$$f = k \cdot (V_{dd} - V_t)^2 / V_{dd} \quad (4)$$

where  $V_t$  is threshold voltage and  $K$  is hardware constant.

The leakage power dissipation is expressed (5) [21],

$$P_l = V_{dd} \cdot I_{leak} \quad (5)$$

where  $I_{leak}$  is the leakage current which consists of gate-oxide leakage current  $I_g$  and sub-threshold leakage current  $I_s$  Gate-oxide leakage current and they expressed as (6) [22],

$$I_g = MW(V_{dd}/W_{ox})^2 e^{-aW_{ox}/V_{dd}} \quad (6)$$

where  $M$  and  $\alpha$  are hardware parameters,  $W$  is the gate width and  $W_{ox}$  is the oxide thickness. Sub-threshold leakage current is calculated by (7) [22],

$$I_s = KW e^{-V_{th}/nV_h} (1 - e^{-V_{dd}/V_h}) \quad (7)$$

where  $K$  and  $n$  are hardware parameters.  $V_h$  is the voltage related to the current chip temperature.

### 3.2. Task classification and modeling

According to [23] tasks can be classified into three classes. Depending on the application, we decide to use the class of the task. For instance, if we need to monitor a process by a specified sensor, then we should use a periodic task, however, if monitoring of that process is obliged to be within a specified time, then this task should be a real-time class. Task classes are described stated in the followings:

- Independent vs dependent tasks: When one activity's completion depends on that of another work or task, that task is said to be dependent. The majority of general-purpose programs operate on the dependent task approach. Directed acyclic graph (DAG) is used to depict the dependency between tasks (GAG). Tasks are represented in a DAG by nodes, while interdependence between tasks is represented by edges.
- Real time vs non-real time tasks: When the CUP is required by the operating system to complete a task before the deadline expires, the task is said to be in real time. Real time tasks have two subclasses, the first of which is hard real time tasks, wherein the deadline for completing the task must be met. The processor is permitted to finish the task execution by some intervals that depend on the application and the level of application seriousness in the second subclass of real time tasks, known as soft real time tasks.
- Periodic vs aperiodic tasks: A task that arrives in fixed time intervals is called a periodic task. The instant of the first activation is called phase  $\phi$ . For a periodic task  $\tau$ , its activation time for  $K^{th}$  instant can easily be expressed by  $\phi + (k - 1)T$  where  $T$  is period. And in real time systems the task period is assigned as the task's deadline in most of the cases. Aperiodic tasks, on the other hand, have an indefinite series of activity and arise erratically, making it difficult and inaccurate to estimate when they will appear.

This study considers independent non-real time periodic tasks which are modeled as follows:

$Task_i(t_i, P_i, En_i)$  where:

$t_i$  is the execution time of  $Task_i$ .

$P_i$  is the average power consumption of  $Task_i$ .

$En_i$  energy dissipated by  $Task_i$ .

The task period depends on the application of the task. However, the other parameters of the tasks depend on the CPU configuration and they will be determined through the simulation setup. One of the parameters is  $t_i$  and could be accurately measured in selected CPU configuration by GEM5 [24],  $P_i$  is could be measured using multicore power, area, and timing (McPAT) [25] with the same CPU configuration that was used for GEM5, and finally  $En_i$  could be achieved by multiplying  $P_i$  and  $t_i$  together to determine the energy of each Task as explained by the block diagram in Figure 1.

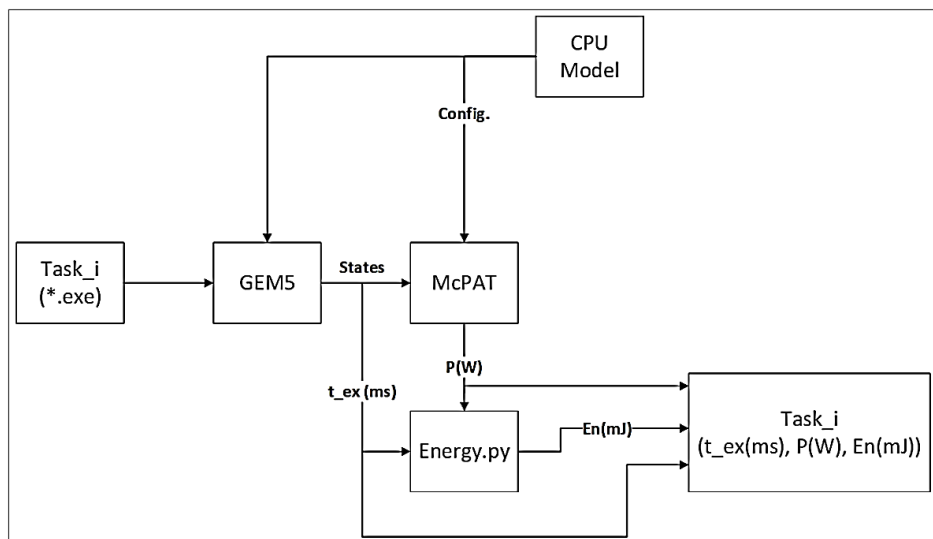


Figure 1. The signal flow diagram of the task model

**4. THE PROPOSED TASK ASSIGNMENT ALGORITHM**

Let's consider a set of independent, heterogeneous (i.e., tasks with different parameters), and periodic tasks. The algorithm of the task assignment will assign the tasks for each hyper-period of the tasks based on the decision factors ( $a_1, a_2, \dots, a_m$ ) which depend on the cores' temperature ( $T_1, T_2, T_3, \dots, T_4$ ) and the tasks' energy as illustrated in Figure 2. The factor ( $a_j$ ) assigns the percent of the total energy that should be consumed by each core. Hence, the energy that should be consumed by each core to keep the temperature balance of the CPU is computed by multiplying  $a_j$  factor by the total energy of the tasks as expressed in (8).

$$C_{En_j} \approx a_j * En_{total} \tag{8}$$

Where  $C_{En_j}$  is the total energy that should be executed by *core<sub>j</sub>* during a hyper-period.

$$En_{total} = \sum_{n=1}^n En_i \tag{9}$$

$n$  is the total number of the tasks  $En_i$  is the energy of Task <sub>$i$</sub>

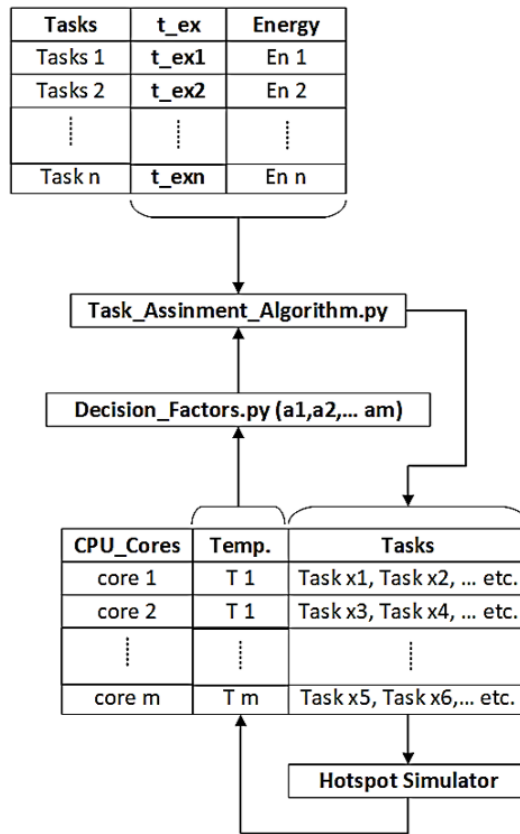


Figure 2. Task assignment signal flow

The factor  $a_j$  will be computed by the principle of load assignment according to the core's temperature. Such that if the cores' temperatures are equal,  $a_j$  of all the cores will be equal and if there's any difference between them, the factor  $a_j$  will be highest for the lowest core temperature to execute the lowest energy and vice versa for the heist core temperature. This load distribution can be performed utilization of a square curve as shown below in Figure 3. The factor  $a_j$  can be found by substitution the temperature of each core into (12) and determining their corresponding Y-axis value as shown in Figure 3.

$$y_j = (T_j - 2 * T_{max})^2 \tag{10}$$

where:

$$T_{\max} = \text{Max}\{T_1, T_2, T_3, \dots, T_m\}. \tag{11}$$

Now,  $a_j$  can easily be found by the following expression:

$$a_x = \frac{y_x}{\sum_{j=1}^m y_j}; x \in \{1, 2, 3, \dots, m\}. \tag{12}$$

After determination of the decision factors, we can perform a task assignment by genetic algorithm (Geno-type) [26] such that each core ( $C_j$ ) executes ( $a_j$ ) of total tasks energy given that ( $0 \leq a_j \leq 1$ ). This can be done by treating each gene in a chromosome as a task as shown in Figure 4. Gene 1 represents Task 1, Gene 2 represents task 2, and so on.

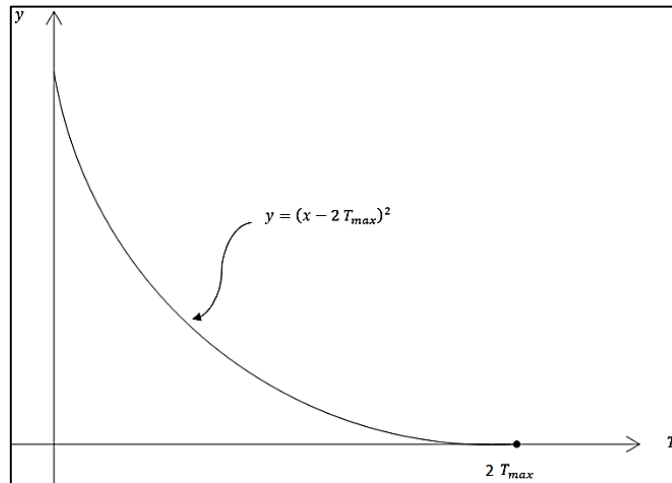


Figure 3. Decision factor curve to find  $\alpha_x$  factor of decision

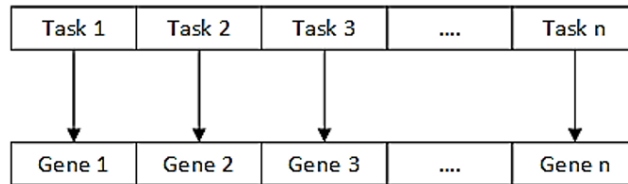


Figure 4. Mapping tasks to genes, in which each task is represented by a gene

At the beginning of the algorithm, a number ( $N_p$ ) of chromosome vectors with a length equal to the tasks vector length ( $n$ ) will be generated randomly which's called initial population matrix with a size equal to ( $N_p * n$ ). Then the generated genes of the chromosomes are changed with each iteration according to the genetic algorithm (expressed in algorithm-1) such that we achieve the best fitness which's expressed in (13). The chromosome vector of the best fitness is mapped to the tasks vector in such a way that the tasks in which corresponding genes are equal to "1" are assigned to the core during a hyper-period. This assures that the energy consumed by the core is equal to  $a_j * En_{total}$ . See Figure 5.

$$Fitness = Abs\{C_{En_j} - a_j * En_{total}\}. \tag{13}$$

After that, the assigned tasks are extracted from the tasks vector and the remaining tasks will be a candidate to be assigned to the next core. This process will continue until all the tasks are assigned to all the CPU cores. But it is critical to mention that the assignment algorithm won't be applied on the last core even if we do not achieve exact ( $C_{en_{last-core}} \approx a_{last-core} * En_{total}$ ) to guarantee the implementation of all tasks. The task assignment algorithm is expressed in algorithm 1.

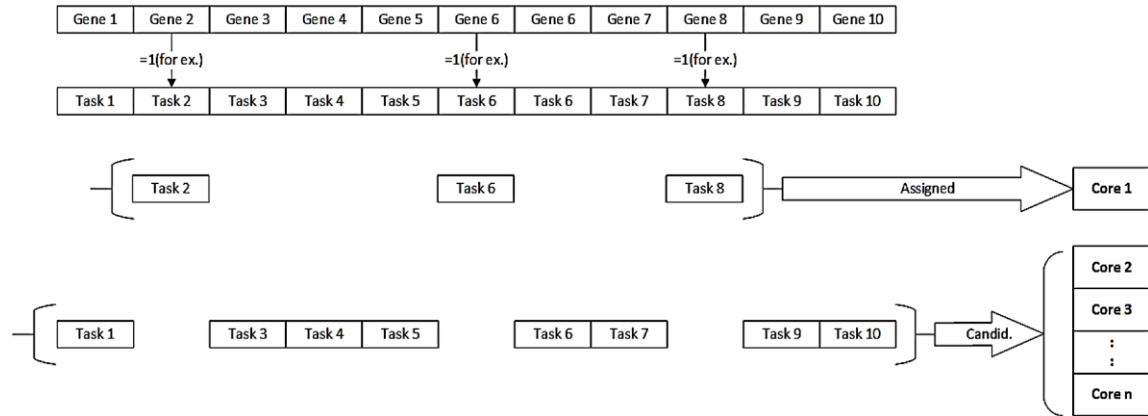


Figure 5. Illustration of using genetic algorithm in task assignment

#### Algorithm-1

- 1: Input: Tasks'  $(p_1, p_2, p_3, \dots, p_n)$  and  $(En_1, En_2, En_3, \dots, En_n)$  parameters.
- 2: Input: Cores' Temperature  $(T_1, T_2, T_3, \dots, T_m)$ .
- 3: Compute Total Energy.
- 4: Input Energy assignment decision factors  $(a_1, a_2, a_3, \dots, a_m)$  where:  $0 \leq a_j \leq 1$ .
- 5: Perform Genetic Algorithm for each core such that  $C_{en_j} \approx a_j * \sum_1^n En_i$ .
  - Input: *Fitness function*,  $N_p$ , *iter*,  $n$ ,  $k$ ,  $\rho_c$ ,  $\rho_m$
  - Initialize a random Population of binary string (size =  $N_p * n$ ).
  - For  $j=1$  to  $(m-1)$
  - Evaluate Fitness:  $Fitness = Abs(\sum(CH \odot En) - a_j * En_{total})$
  - For  $g=1$  to *iter*
    - Perform Tournament selection of tournament size  $k$
    - For  $h=1$  to  $N_p/2$ 
      - Randomly choose two parents
      - Generate a random number ( $r$ )
      - If  $r < \rho_c$ 
        - Select a random crossover site
        - Generate two offspring using single-point-crossover
      - else
        - Copy the selected parents as offspring
    - End
  - For  $g=1$  to  $N_p$ 
    - Generate  $n$  random number between 0 and 1
    - Perform bitwise mutation of  $g^{th}$  offspring if the number is less than  $\rho_m$
    - Evaluate the Fitness of offspring
  - End
  - Combine Population and Offspring to perform  $(\mu + \lambda)$
- End
- Map the chromosome of the best fitness to the task set so that each "1" in the chromosome is an assigned task to Core ( $j$ )
- Extract the assigned tasks form the task set
- $n=n$ -number of extracted tasks.
- End
- Assign the remaining tasks to the last core ( $m$ ).

Once all the tasks are assigned to the cores according to their temperature, they will be simulated by a hotspot simulator to get the temperature response of the processor. And the steady-state value yield in the output of the hotspot simulator will be an input to the assignment algorithm for the next hyper-period. Again, the algorithm performs task assignment depending on the latest cores' temperature of the previous period.

## 5. SIMULATION AND RESULTS

### 5.1. Simulation setup

The Architecture used in this paper is a Quad-Core O3 (out of order) Processor with an I-Cache and D-Cache capacity of 32KB. And a dedicated L2-Cache memory with 1MB the capacity of the RAM used in this architecture is 64 MB integrated with DDRx1 memory controller type and the clock frequency is set to 2 GHz. The complete architecture is shown in Figure 6.

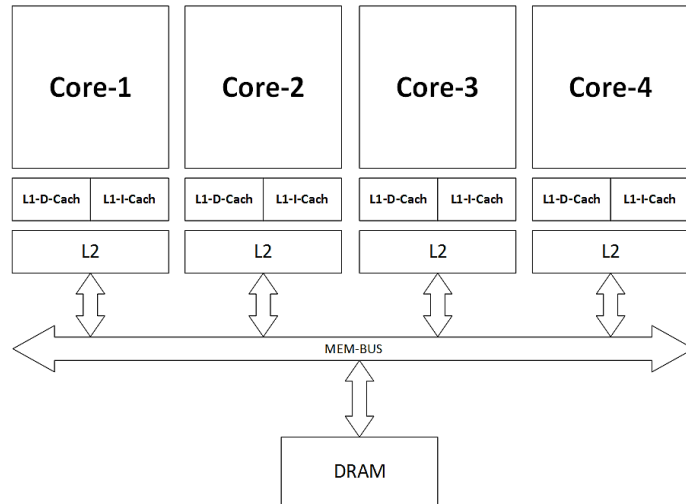


Figure 6. Simulation architecture consisting of a 4 cores CPU with dedicated L1 and L2 of each core

**5.2. Task benchmarks and results**

For testing the efficiency of our algorithm, we have to use some standard test benchmarks. The benchmarks which are used for the simulation of our algorithm consist of Spec2006 [27], Mibench [28], and Mediabench [29]. The detail of each benchmark is expressed in Table 1.

Table 1. Real benchmarks

Group	Tasks#	Benchmark Name	Ex. Time (ms)	Leakage Power (W)	Dynamic Power (W)	Energy (mJ)	Description
Spec2006	0	401.bzip2	83.654	1.15327	0.318345	123.106	File compression program
	1	456.hmmer	105.657	0.72241	0.105194	87.4425	Used in computational biology to search for patterns in DNA sequences
	2	470.lbm	70.205	1.15327	0.364266	106.539	Implements the so-called "Lattice Boltzmann Method" to simulate incompressible fluids in 3D
	3	429.mcf	75.89	0.72241	0.144257	65.7716	The program is designed for the solution of single-depot vehicle scheduling (sub-) problems occurring in the planning process of public transportation companies.
	4	458.sjeng	174.763	1.15327	0.178489	232.742	It attempts to find the best move via a combination of alpha-beta or priority proof number tree searches
	5	basicmath	490.239	0.66866	0.041022	347.912	Performs simple mathematical calculations
	6	bitcount	62.553	1.15327	0.340039	93.411	Tests the bit manipulation abilities of a processor by counting the number of bits in an array of integers
Mi Bench	7	qsort	71.714	0.84359	0.187796	73.9645	Sorts a large array of strings into ascending order using the well-known quick sort algorithm
	8	susan	513.823	1.15327	0.129601	659.169	An image recognition package
	9	patricia	340.339	1.1158	0.279612	474.913	A data structure used in place of full trees with very sparse leaf nodes
	10	sha	882.386	0.84359	0.058882	796.325	The secure hash algorithm that produces a 160-bit message digest for a given input
	11	blowfish	67.093	1.15062	0.331208	99.4203	A symmetric block cipher with a variable length key
	12	pgp	490.239	0.66866	0.041022	347.912	A public key encryption algorithm developed by Phil Zimmerman
	13	sign/verify tiff2bw	280.534	0.84359	0.077	258.256	Converts a color TIFF image to black and white image
Media Bench	14	typeset	70.75	0.74111	0.135449	62.0162	General typesetting tool, that has a front-end processor for HTML
	15	FFT/IFFT	99.812	2.48721	0.868513	334.941	Performs a fast fourier transform
	16	PEGWIT	82.678	2.48721	1.19988	304.841	A program for public key encryption and authentication
	17	EPIC	174.763	0.95743	0.179687	198.725	An experimental image compression utility
	18	RASTA	84.629	0.72241	0.127025	71.8871	A program for speech recognition
	19	Ghostscript	72.412	1.15065	0.357371	109.199	An interpreter for the PostScript language



**5.3. Simulation results**

To evaluate the assignment algorithm the simulation was executed for 10 hyper-periods. For each hyper-period, all the tasks are assigned and executed by the processor, and the final temperature of each hyper-period will be the initial temperature of the next hyper-period. This will be repeated for 10 hyper-periods. The simulation is performed in two parts, first part is setting the initial temperature of cores to equal values to know the ability of the assignment algorithm to maintain the balance of the temperature between the cores. And the second part is performed by setting each core’s initial temperature with different values to inspect if the assignment algorithm is capable of re-balance the temperature of the processor cores.

**5.3.1. Equal initial temperature**

Figure 7 shows the temperature trend of each core and the executed task by each core on the same time domain, and the initial temperature of the cores is set at 45 °C and the ambient temperature is set at 45 °C. From the results which are shown in the following Figure 8. It can be seen that the temperature of each core is slightly equal to the temperature of the other cores. This shows that the algorithm is successful in maintaining the temperature balance of the processor cores in such a way that the maximum difference in temperature between cores is nearly 9 °C for each hyper period. See Figure 8.

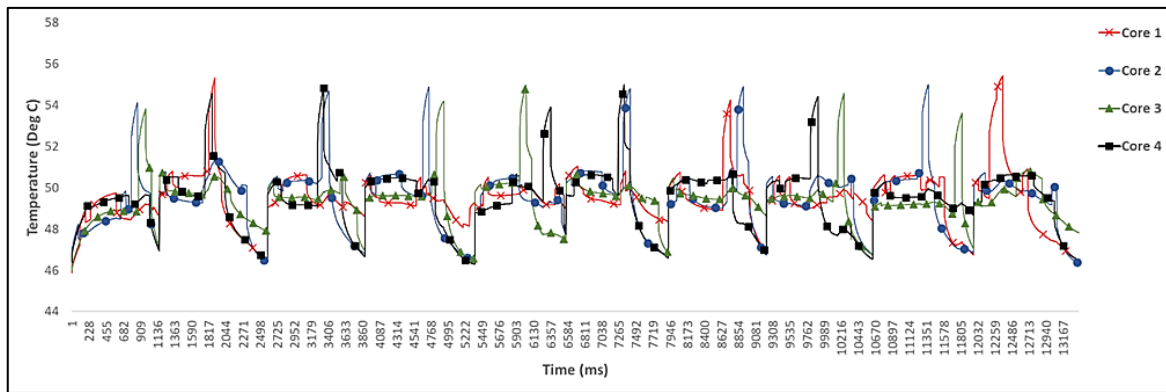


Figure 7. Cores temperature trend for the case of equal initial temperature

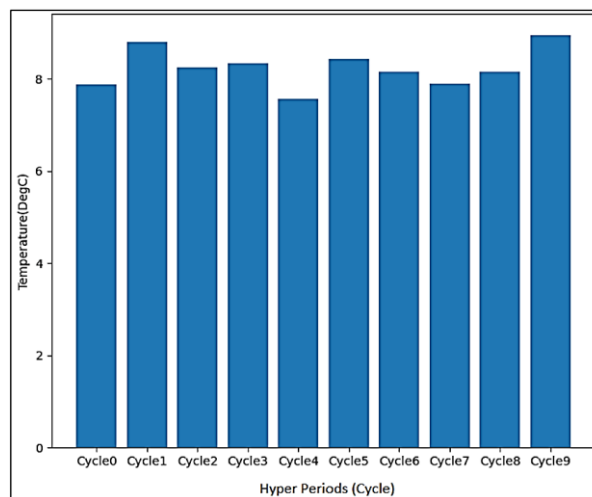


Figure 8. Maximum temperature different between each core for 10 hyper-periods for equal initial temperature case

**5.3.2. Different initial temperature**

In the same previous way, we performed the simulation of our architecture with the same algorithm and test benchmarks. The only thing that has been changed, is the processor core’s initial temperature. Which sat with different initial temperatures as follows:

- Core1 Temp: 65.48 °C
- Core2 Temp: 49.77 °C
- Core3 Temp: 45.82 °C
- Core4 Temp: 51.86 °C
- Ambient Temperature: 45 °C

According to the results shown in Figure 9. The temperature gets balanced after a few seconds of task execution. This proves that the algorithm can balance processor cores' temperatures even if their temperatures are different as it can be seen in Figure 10 shows the maximum temperature difference between each core for each hyper period is slightly 8 °C except for the first hyper-period as the cores' initial temperatures are different.

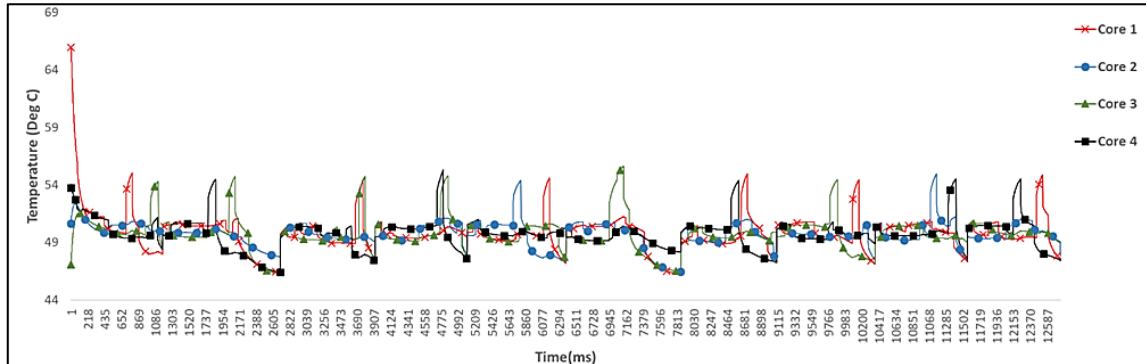


Figure 9. Cores temperature trend for non-equal initial temperature

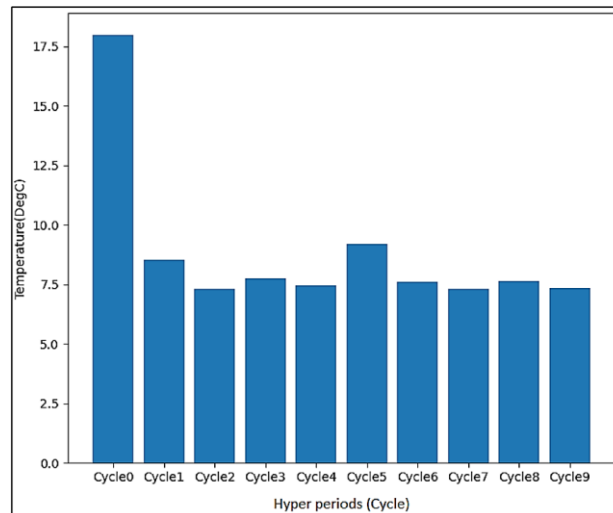


Figure 10. Maximum temperature difference between each core for 10 hyper-periods for the case of non-equal initial temperature

### 6. CONCLUSION

In this paper, an assignment algorithm has been proposed to keep the balance of temperature between the cores. The simulation is performed on a Quad-Core platform with two levels of cache and L2-cache dedicated. The mentioned task benchmarks are simulated using GEM5 to measure the task execution time on the platform. The McPAT simulator is used to measure the power of each task by exploiting the GEM5 statistics outputs. The energy of each task is measured through both GEM5 and McPAT outputs. Once the parameters of each task are achieved, they are employed and assigned to the cores according to the cores' current Temperature and energy of each task. The algorithm assigns the most energy to the lowest temperature core and the least energy to the core with the highest temperature. The algorithm uses genetic optimization for assignment of the tasks. The simulation results showed that the highest temperature difference between the

cores is 8 °C for approximately 14 seconds. These results validate the effectiveness of the proposed task assignment algorithm in managing the hotspot and reducing both temperature and energy consumption in multicore processors. From Figures 7 and 9 it is noticeable from the results that there's a large peak takes place during each hyper-period of the cores this can be eliminated by using a proper task portioning algorithm after assignment of the tasks. Hence our future is to integrate task partitioning algorithm with our proposed assignment algorithm, to reduce the peak temperature of the cores.





## REFERENCES

- [1] W. H. Gerling, A. Preussger, and F. W. Wulfert, "Reliability qualification of semiconductor devices based on physics-of-failure and risk and opportunity assessment," *Quality and Reliability Engineering International*, vol. 18, no. 2, pp. 81–98, Mar. 2002, doi: 10.1002/qre.468.
- [2] L. Rubio-Anguiano, G. Desirena-López, A. Ramírez-Treviño, and J. L. Briz, "Energy-efficient thermal-aware multiprocessor scheduling for real-time tasks using TCPN," *Discrete Event Dynamic Systems*, vol. 29, no. 3, pp. 237–264, Sep. 2019, doi: 10.1007/s10626-019-00285-x.
- [3] J. P. Rodríguez and P. M. Yomsi, "Thermal-aware schedulability analysis for fixed-priority non-preemptive real-time systems," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2019, pp. 154–166. doi: 10.1109/RTSS46320.2019.00024.
- [4] D. Rupanetti and H. Salamy, "Energy-aware task migration through ant-colony optimization for multiprocessors," in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, Dec. 2021, pp. 0901–0907. doi: 10.1109/UEMCON53757.2021.9666584.
- [5] M. S. Babadi, M. E. Shiri, M. R. M. Goudarzi, and H. H. S. Javadi, "Multi-objective tasks scheduling using artificial bee colony algorithm based on cellular automata in cloud computing environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 5, pp. 5657–5666, Sep. 2022, doi: 10.11591/ijece.v12i5.pp5657-5666.
- [6] T. P. Thanh, L. N. The, S. Elnaffar, C. N. Doan, and H. D. Quoc, "An effective PSO-inspired algorithm for workflow scheduling," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 5, pp. 3852–3859, Oct. 2018, doi: 10.11591/ijece.v8i5.pp3852-3859.
- [7] B. P. Singh and B. Thangaraju, "Thermal aware power save policy for hot and cold jobs," in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Jul. 2021, pp. 1–7. doi: 10.1109/CONECCT52877.2021.9622715.
- [8] T. Li, G. Yu, and J. Song, "Minimizing energy by thermal-aware task assignment and speed scaling in heterogeneous MPSoC systems," *Journal of Systems Architecture*, vol. 89, pp. 118–130, Sep. 2018, doi: 10.1016/j.sysarc.2018.08.003.
- [9] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2008, pp. 618–623. doi: 10.1109/ICCAD.2008.4681641.
- [10] M. Kadin and S. Reda, "Frequency planning for multi-core processors under thermal constraints," in *Proceeding of the thirteenth international symposium on Low power electronics and design - ISLPED '08*, 2008, pp. 213–216. doi: 10.1145/1393921.1393977.
- [11] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, 2001, pp. 171–182. doi: 10.1109/HPCA.2001.903261.
- [12] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang, "Thermal vs energy optimization for DVFS-enabled processors in embedded systems," in *8th International Symposium on Quality Electronic Design (ISQED'07)*, Mar. 2007, pp. 204–209. doi: 10.1109/ISQED.2007.158.
- [13] S. Eyerman and L. Eeckhout, "Fine-grained DVFS using on-chip regulators," *ACM Transactions on Architecture and Code Optimization*, vol. 8, no. 1, pp. 1–24, Apr. 2011, doi: 10.1145/1952998.1952999.
- [14] S. Durand and S. Lesecq, "Nonlinear and asymmetric thermal-aware DVFS control," in *2013 European Control Conference (ECC)*, 2013, pp. 3240–3245.
- [15] Z. Wang, S. Ranka, and P. Mishra, "Efficient task partitioning and scheduling for thermal management in multicore processors," in *Proceedings of International Symposium on Quality Electronic Design*, 2015.
- [16] D. Sulaiman, I. Hamarash, and M. Ibrahim, "Microprocessors optimal power dissipation using combined threshold hopping and voltage scaling," *IEICE Electronics Express*, vol. 14, no. 24, pp. 20171046–20171046, 2017, doi: 10.1587/elex.14.20171046.
- [17] D. Sulaiman, I. Hamarash, and M. Ibrahim, "Adaptive supply and body voltage control for ultra-low power microprocessors," *IEICE Electronics Express*, vol. 14, no. 12, pp. 20170306–20170306, 2017, doi: 10.1587/elex.14.20170306.
- [18] Y. Xie and W. Hung, "Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design," *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 45, no. 3, pp. 177–189, Dec. 2006, doi: 10.1007/s11265-006-9760-y.
- [19] J. Kang, "Scheduling algorithms for energy minimization," University of Florida ProQuest Dissertations, 2008.
- [20] M. J. Walker, S. Diestelhorst, A. Hansson, D. Balsamo, G. V. Merrett, and B. M. Al-Hashimi, "Thermally-aware composite run-time CPU power models," in *2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2016, pp. 17–24. doi: 10.1109/PATMOS.2016.7833420.
- [21] M. Awadalla, "Processor speed control for power reduction of real-time systems," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, no. 4, pp. 701–713, Aug. 2015, doi: 10.11591/ijece.v5i4.pp701-713.
- [22] N. S. Kim *et al.*, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003, doi: 10.1109/MC.2003.1250885.
- [23] Z. Wang, "Thermal-aware task scheduling on multicore processors," University of Florida, 2012.
- [24] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011, doi: 10.1145/2024716.2024718.
- [25] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 469–480.
- [26] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205–222, Dec. 2005, doi: 10.1016/j.cam.2004.07.034.
- [27] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, Sep. 2006, doi: 10.1145/1186736.1186737.





- [28] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, 2001, pp. 3–14. doi: 10.1109/WWC.2001.990739.
- [29] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," in *Proceedings of 30th Annual International Symposium on Microarchitecture*, 1997, pp. 330–335. doi: 10.1109/MICRO.1997.645830.

## BIOGRAPHIES OF AUTHORS



**Mohammed Parwez**     received B.Sc. in Electrical Engineering from Salahaddin University, Erbil, Iraq, in 2015. Currently, he is an M.Sc. student of the Computer and Control branch at Salahaddin University. His research interests include operating systems, multicore architectures, OS schedulers, and CPU thermal management. He can be contacted at email: mohammad.perweze@gmail.com and LinkedIn: <https://www.linkedin.com/in/mohammad-parweze>



**Diary R. Sulaiman**     Professor of Electronics and Computer Engineering, Department of Electrical Engineering, College of Engineering, Salahaddin university-Erbil, Iraq. He has been working in higher education for more than 30 years and he taught both undergraduate and postgraduate students at Department of Electrical Engineering. He has gained the CISCO certifications and Information System Professional certifications. His current research interests include power/thermal management of microprocessors, advanced digital design, and CMOS circuit design. Diary R. SULAIMAN has authored many publications on electronics and computer hardware design, CMOS circuit design, and microprocessors power/thermal management. He has more than 50 published articles in international journals and conferences. He can be contacted at email: diary.sulaiman@su.edu.krd, Academic Website: <https://academics.su.edu.krd/diary.sulaiman>, and LinkedIn: <https://www.linkedin.com/in/diary-sulaiman/>