MAFin: Maximizing Accuracy in FinFET based Approximated Real-Time Computing

Shounak Chakraborty**
shounak.chakraborty@ntnu.no
Norwegian university of Science and Technology, Trondheim, Norway

Sangeet Saha*
sangeet.saha@essex.ac.uk
University of Essex
Colchester, UK

Magnus Själander
magnus.sjalander@ntnu.no
Norwegian university of Science and Technology, Trondheim, Norway

Klaus D. McDonald-Maier
kdm@essex.ac.uk
University of Essex
Colchester, UK

ABSTRACT

We propose *MAFin* that exploits the unique temperature effect inversion (TEI) property of a FinFET based multicore platform, where processing speed increases with temperature, in the context of approximate real-time computing. In approximate real-time computing platforms, the execution of each task can be divided into two parts: (i) *the mandatory part*, execution of which provides a result of acceptable quality, followed by (ii) *the optional part*, that can be executed partially or fully to refine the initially obtained result in order to increase the result-accuracy (QoS) without violating deadlines. With an objective to maximize the QoS for a FinFET based multicore system, *MAFin*, our proposed real-time scheduler first derives a task-to-core allocation, while respecting system-wide constraints and prepares a schedule. During execution, *MAFin* further increases the achieved QoS, while balancing the performance and temperature on-the-fly by incorporating a prudential temperature cognizant frequency management mechanism and guarantees imposed constraints. Specifically, *MAFin* exploits the TEI property of FinFET based processors, where processor-speed is enhanced at the increased temperature, to reduce the execution time of the individual tasks. This reduced execution-time is then traded off either to enhance QoS by executing more from the tasks' optional parts or to improve energy efficiency by turning off the core. While surpassing prior art, *MAFin* achieves 70% QoS, which is further enhanced by 8.3% in online, with a maximum EDP gain of up to 12%, based on benchmark based evaluation on a 4-core based system.

CCS CONCEPTS

• **Computer systems organization** → **System on a chip**; **Multicore architectures**; **Real-time systems**; • **Hardware** → **Thermal issues**.

*Both authors contributed equally to this research.

KEYWORDS

FinFET, Energy Efficiency, Thermal Management, SHE, TEI, Real-Time Systems, Approximate Computing

ACM Reference Format:
Shounak Chakraborty, Sangeet Saha, Magnus Själander, and Klaus D. McDonald-Maier. 2024. MAFin: Maximizing Accuracy in FinFET based Approximated Real-Time Computing. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3655985

1 INTRODUCTION

In real-time computing, the correctness depends both on the result-accuracy and on the time at which the results are obtained. For such time-critical scenarios, approximated results generated on-time are preferred to accurate results produced after deadline [3]. For example, in case of target tracking, an approximated estimation of the target's location produced before deadline is better than an accurate location, obtained too late. In approximate real-time computing, each individual task is logically decomposed into a mandatory and an optional part [14]. The entire mandatory part must be finished before deadline to generate the minimally acceptable quality of service (QoS), followed by a partial or complete execution of the optional part, subject to availability of resources. The QoS improves with the number of execution cycles spent on the optional part.

In 2018, approximate computing was introduced for multicore based real-time systems while considering the energy budget as a system-wide constraint for the independent task set [8]. L. Mo et al. proposed energy efficient scheduling of the dependent approximate tasks with dynamic voltage and frequency scaling (DVFS) of the cores [14]. In recent works [9, 16], thermal efficient task scheduling was proposed for dependent approximate real-time tasks, where the offered QoS in offline mode is enhanced by employing online architectural techniques for MOSFET based systems. The recent shift from MOSFET to FinFET in state-of-the-art technology nodes enables a more precise control of the transistor channel, reducing unwanted current leakage and lowering power consumption [13]. Moreover, FinFETs experience a noticeable increase in circuit speed at a higher temperature, known as temperature effect inversion (TEI) [11]. However, the FinFET channel is encapsulated within a thermal insulator that obstructs heat dissipation and can lead to circuit failure due to the self heating effects (SHEs) [2]. Here, we argue for a novel approximate real-time scheduling strategy that leverages these unique properties of FinFET based multicores and prudently exploits the temperature effect inversion to improve QoS of approximated real-time tasks while combating SHEs.

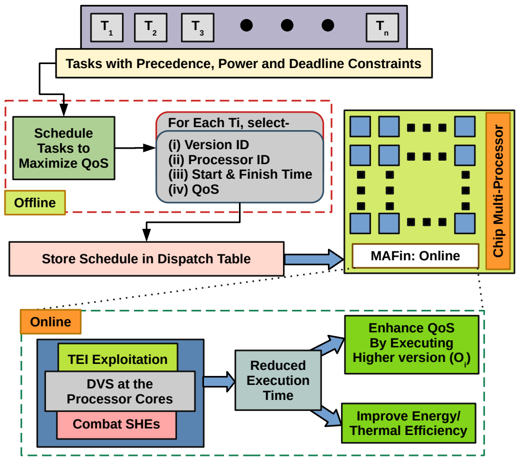
T1
T2
T3
Tn
Tasks with Precedence, Power and Deadline Constraints
Schedule Tasks to Maximize QoS
For Each Ti, select-
(i) Version ID
(ii) Processor ID
(iii) Start & Finish Time
(iv) QoS
Offline
Chip Multi-Processor
Store Schedule in Dispatch Table
MAFin: Online
Online
TEI Exploitation
DVS at the Processor Cores
Combat SHEs
Reduced Execution Time
Enhance QoS By Executing Higher version (Oi)
Improve Energy/ Thermal Efficiency

Figure 1: *MAFin*: Process Overview

In *MAFin*, we devise an integer linear programming (ILP) based scheduling technique for a set of approximated real-time tasks on a FinFET based chip-multiprocessor (CMP), where scheduling is constrained by the power consumption of the CMP, task-dependency, and deadline. Each task is equipped with multiple versions defined by a diverse set of QoS, based on the respective amount of the optional part that is executed. With an objective to enhance the QoS while considering the given constraints, our ILP based scheduling provides the following information for task-execution: which versions of each task (*Version ID*) to be executed on which core (*Processor ID*), along with the task's start and finish times (*Start & Finish Time*) and the overall obtained QoS. During execution, we divide the entire execution span of each task into multiple slices, where on completion of each slice, the core temperature is evaluated, and the core frequency is subsequently updated with a certain voltage level to maximize *TEI exploitation*, while respecting time constraint and thermal safety to curtail SHEs. *MAFin* trades off the reduced execution time caused by TEI exploitation to increase the result-accuracy by executing a higher optional version of the task (subject to availability), or by turning off the core to improve energy and thermal efficiency of the CMP. The entire mechanism is depicted in Figure 1.

The contributions of *MAFin* are as follows:

- We schedule a set of dependent approximated real-time tasks on a FinFET based CMP with an objective to maximize the result-accuracy (QoS), by employing an ILP based strategy (detailed in Sec. 3.1).
- We apply a TEI and SHE cognizant dynamic thermal management (DTM) strategy that prudently reduces execution time of the individual task by judiciously exploiting TEI, while combating SHE, and generates slack by accelerating the processing speed through TEI exploitation (detailed in Sec. 3.2), which we have empirically validated and reported in Sec. 5.
- We exploit slack in either or both of the following ways:
  (1) to maximize QoS by executing higher version from the optional part of the tasks (based on availability),
  (2) to improve thermal and energy efficiency of the CMP by power gating the cores.

We argue and empirically validate the significance of the task-scheduling approach of *MAFin* in combination with online TEI exploitation and SHE combating mechanisms (Sec. 5). We achieve 70% QoS with our ILP based scheduling technique for a set of dependent tasks with 65% workload, for which a state-of-the-art [14] achieves a QoS of 55%. Our evaluation on a 4 core based CMP shows, *MAFin: Online* enhances achieved QoS by 8.3% with a maximum energy delay product (EDP) gain of 12%. *MAFin is the first scheduling mechanism that considers TEI exploitation in a FinFET based CMP to enhance QoS of dependent approximated real-time tasks, while maintaining both real-time constraints and thermal safety.*

2 SYSTEM MODEL

Our CMP consists of m homogeneous cores, denoted as $P = \{P_1, P_2, ..., P_m\}$. A real-time approximate computing (AC) based application ($\mathcal{A}$) with deadline D_{PTG} is modelled, as a precedence constrained task-graph (PTG) (as shown in Figure 2), $G = (T, E)$, where T is the set of tasks ($T = \{T_i \mid 1 \le i \le n\}$) and E is the set of directed edges ($E = \{\langle T_i, T_j\rangle \mid 1 \le i, j \le n; i \ne j\}$). The worst-case execution length (len_i) for each task T_i ($1 \le i \le n$) is logically decomposed into M_i cycles for its mandatory part, and O_i, the cycles for its optional part. We further assume that a task T_i might have k_i different versions, that is, $T_i = \{T_i^1, T_i^2, \ldots, T_i^{k_i}\}$, which are distinct by their individual execution lengths of their respective optional parts (O_i), denoted as $O_i^1, O_i^2, ..., O_i^{k_i}$, where O_i^p achieves a higher result-accuracy than O_i^q, if $p > q$. The execution length ($EL(i,j)$) of the j^{th} version of task T_i (i.e. T_i^j where $1 \le j \le k_i$) can now be defined as: $EL(i,j) = M_i + O_i^j$. The result-accuracy Acc_i^j of the T_i^j is basically the executed optional part of the task, O_i^j (i.e., $Acc_i^j = O_i^j$). Thus, the overall system level result-accuracy is defined as the sum of the executed cycles of O_i^j for all of our tasks [8] and can be stated as: $QoS(\mathcal{A}) = \sum_{i=1}^{n} O_i^j \mid T_i = T_i^j$.

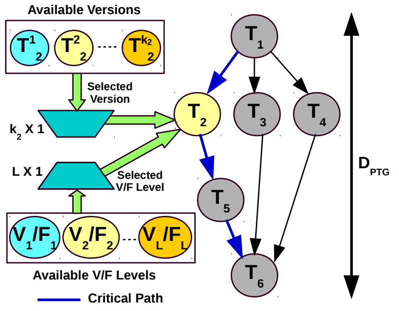

Figure 2: Task graph

3 *MAFIN*: PROPOSED TECHNIQUE

MAFin prioritises accuracy. It first creates an offline schedule (Sec. 3.1) and then uses an online technique (Sec. 3.2) to exploit TEI for even better results.

3.1 *MAFin*: Offline

We present a scheduling strategy based on integer linear programming (ILP). For this purpose, we first introduce an integer decision variable $S_i \in \mathbb{Z}^+$ to capture start time of each task T_i, where $\mathbb{Z}^+$ denotes the set of positive integers. We further define a binary decision variable, $Z_{ik\eta}$, where, $i = 1, 2, ..., n$; $k = 1, 2, ..., k_i$; and $\eta = 1, 2, ..., m$; Here indices, i, k and η denote task ID, corresponding version ID and the processor ID, respectively. $Z_{ik\eta} = 1$, if the k-th version of T_i (i.e. T_i^k) executes on processor η, otherwise 0. We define another binary variable Y_{ij}, where $Y_{ij} = 1$, if task T_i starts before T_j, else 0. Note that, all tasks are assumed to be executed at a fixed core-frequency. To model our scheduling strategy, the constraints on the decision variable are stated as follows:

(1) Each task T_i is assigned to exactly one processor with a particular version

$$\sum_{k=1}^{k_i} \sum_{\eta=1}^{m} Z_{ik\eta} = 1 \tag{1}$$

(2) The application $\mathcal{A}$ must meet its end-to-end absolute deadline D_{PTG}. Hence, the sink node T_n should be finished by D_{PTG}, which is represented as-

$$S_n + \sum_{k=1}^{k_n} \sum_{\eta=1}^{m} (EL(n,k) \times Z_{nk\eta}) - 1 \leq D_{PTG} \tag{2}$$

(3) The peak power consumption of the defined system must not exceed the stipulated power budget. Let Pow_{peak} represents the peak power consumption of the system-

$$Pow_{peak} = max\{Pow_{sys}\} \tag{3}$$

where,

$$\forall t | Pow_{peak} \leq Pow\_BGT \tag{4}$$

Pow_{sys} is dynamic plus static power consumption of all the busy processor-cores built in 14nm FinFET technology node, and is the summation of power consumption of all the tasks executing at the time instant t. By considering computational loads of M_i and O_i, we derived the power consumption of M_i and O_i's of each task (as shown in our example in Table 1).

(4) Precedence constraints between the tasks must also be satisfied. The execution of T_j should commence only after the completion of its predecessor T_i.

$$\forall (\langle T_i, T_j \rangle) \in E, \ (S_i + \sum_{k=1}^{k_i} \sum_{\theta=1}^{m} EL(i,k) \times Z_{ik\eta}) \leq S_j \tag{5}$$

(5) In order to avoid overlapping between tasks executing at the same processors, the following inequalities need to be satisfied: $\forall (\langle T_i, T_j \rangle) \in \mathcal{A}$, where $i \neq j$,

$$Y_{ij} + Y_{ji} > 0 \tag{6}$$

$$Y_{ij} + Y_{ji} \leq 1 \tag{7}$$

$$(S_i + \sum_{k=1}^{k_i} \sum_{\eta=1}^{m} EL(i,k) \times Z_{ik\eta}) \leq (S_j + (1 - Y_{ij}) \times M) \tag{8}$$

Equation 8 avoids time-wise overlap of any pair of tasks on the same processor, i.e. T_j should start after completion of T_i, if T_i is predecessor of T_j. If tasks are executed in reverse order, we use big-M nullification for deactivating the constraint.

(6) **Objective.** Our objective of the formulation is to select the feasible solution, that maximizes QoS of the application. Hence, the objective can be represented as follows:

$$Maximize \ QoS(\mathcal{A}) \tag{9}$$

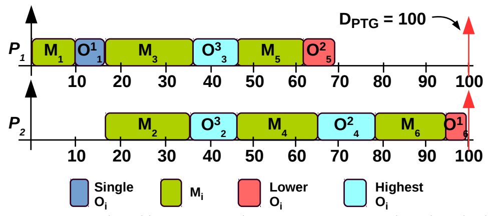

Figure 3: Task allocation by constrained scheduling.

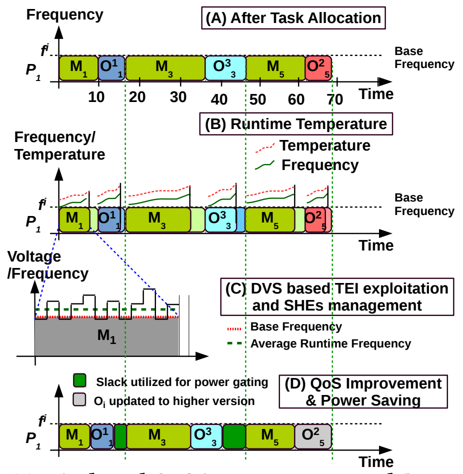
Frequency
(A) After Task Allocation
fⁱ
P1
M1
O¹1
M3
O³3
M5
O²5
Base Frequency
10 20 30 40 50 60 70
Time
(B) Runtime Temperature
Frequency/ Temperature
Temperature
Frequency
Voltage /Frequency
(C) DVS based TEI exploitation and SHEs management
Base Frequency
Average Runtime Frequency
Slack utilized for power gating
Oi updated to higher version
(D) QoS Improvement & Power Saving

Figure 4: TEI induced QoS improvement and Power Saving.

Here, in the context of ILP formulation, $QoS(\mathcal{A})$ will be:

$$QoS(\mathcal{A}) = \sum_{\eta=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{k_i} Z_{ik\eta} \times O_i^k \tag{10}$$

subject to the constraints presented in Equation 1 to 8.

Table 1: Parameters and their values, for example task-set

Task	M_i	Pow_{M_i}	O_i	Pow_{O_i}	Task	M_i	Pow_{M_i}	O_i	Pow_{O_i}
T_1^1	10	20	6	15	T_4^1	20	30	6	25
T_2^1	20	30	5	20	T_4^2	20	30	12	30
T_2^2	20	30	7	20	T_5^1	16	20	6	20
T_2^3	20	30	10	25	T_5^2	16	20	7	20
T_3^1	20	20	4	20	T_5^3	16	20	9	25
T_3^2	20	20	8	20	T_6^1	18	20	3	10
T_3^3	20	20	10	25	T_6^2	18	20	5	15

Let us consider an example with the task-set given in Table 1. These tasks have to be scheduled on two processors ($m = 2$), with a deadline $D_{PTG} = 100$ time units. By choosing different versions of the tasks, our algorithm generates the feasible schedule, which is depicted in Figure 3. Here, T_5 and T_6 are executed with lower versions to satisfy the deadline. Our total obtained QoS value is 48.

3.2 *MAFin*: Online

The entire *MAFin: Online* mechanism to exploit TEI, while combating SHEs is given in Algo 1. The dispatch table stores scheduled tasks, including task ID, assigned core, execution order, and the processor's base frequency, which must be maintained throughout task execution to meet deadlines and ensure thermal safety.

The tasks are fetched from the dispatch table by each processor (P_j) in parallel along with the base frequency ($F^{M_i}\_Base[i]$) and the current core temperature ($Temp$) is also noticed (line 2 to 3). Considering $Temp$, the supply voltage (V_{in}) is determined by employing a prior TEI frequency model proposed by Cai and

Marculescu [7], so that the core frequency should become at least $F^{M_i}\_Base[i]$ and the execution is subsequently initiated (line 4). The entire execution span of each tasks is evenly sliced into some time-quantum, representing a number of consecutive clock cycles, denoted as Δ_{slice}. Once the current cycle count ($cycle\_cntr$) reaches a Δ_{slice} value, indicating end of a time-quantum, the frequency ($F^{M_i}\_Base[i]$) and temperature ($Temp_{slice}$) of the last Δ_{slice} is noticed (line 7). If the $Temp_{slice}$ violates the higher temperature threshold ($Temp^{Hi}_{thr}$), the V_{in} will be set to the lowest possible voltage ($V_{dd}[1]$) to combat the SHEs (line 8 to 9). While $Temp_{slice}$ is lesser than the lower temperature limit, implying lower TEI exploitation, the V_{in} is set to the highest possible value of $V_{dd}[L]$ (line 10 to 11). In case $Temp_{slice}$ is within the range of allowable temperature values, the V_{in} is set to the maximum possible value so that, average frequency should not be lower than $F^{M_i}\_Base[i]$ to maximize TEI exploitation. The newly assigned frequency along with the current voltage and $Temp_{slice}$ is next used to predict the temperature at the end of the subsequent slice to ensure that, this predicted temperature ($Temp_{derived}(V[m], F^{avg}, Temp_{slice})$) will not violate the critical temperature ($Temp_{Crit}$)) of the core, so that SHEs can be tackled. The entire mechanism is illustrated in line 12 to 18. Once the frequency will be assigned at the end of a slice, it will also be added to a frequency array so that the overall average frequency can be tracked to make sure that timing constraint is respected (line 19 to 21). Note that, our $cycle\_cntr$ is incremented during the slice (line 23).

Upon completion of M_i of a task (line 24) with our dynamic TEI exploitation mechanism, the spare cycles are calculated and added to the scheduled cycles left for running O_i (line 25). If the highest O_i is scheduled, it will be executed otherwise, if the cycles left for running O_i is sufficient to execute a higher version, O_i is updated and will be executed next (line 27 to 30). As TEI exploitation will finish the task earlier, a slack can be generated at the end of each task (line 31). Upon completion of the execution of a task, $Extended\_end\_Time(T_i)$ is computed for T_i, which is either the starting time of the next task on P_j, or deadline if T_i is the last task on P_j. Next $Slack\_after\_T_i$ is derived and if found higher than the core's $Break\_Even\_Time$, the core will be turned off and it will be turned on again, once the $Slack\_after\_T_i$ interval is over (line 33 to 36). Figure 4 shows the gains of TEI exploitation, including reduced execution time, improved energy efficiency, and enhanced QoS.

Implementation Overhead: Implementation of Algo 1 utilizes thermal sensors for temperature tracking, commonly found in commercial CMPs [12]. For voltage scaling and core power management, conventional on-chip VRs and per-core-power-gating circuitry are integrated at each core, respectively, typical features in modern CMPs [6]. Notably, the implementation of *MAFin* does not necessitate any additional hardware.

Algorithm 1: *MAFin*: Exploiting TEI with DTM

```
Input: $F\_Base[1 : |\mathbb{T}|], Temp\_Init, \Delta, dispatch\_table, V_{dd}[1 : L], Temp^{Hi}_{thr}, Temp^{Low}_{thr}, Break\_Even\_Time$
Output: A schedule with enhanced accuracy and thermal/energy efficiency
1  $Temp = Temp\_Init$ ;
2  for each processor $P_j$ (do in parallel) do
3      # Fetch the task $\mathbb{T}^i$ from $dispatch\_table$ along with its assigned base frequency
         ($F^{M_i}\_Base[i]$) and current temperature ($Temp$) ;
4      # Set $V_{in}$ to fix the frequency, so that, $Freq(V_{in}, Temp) \geq F^{M_i}\_Base[i]$, and
         start execution and $cycle\_cntr = 0$ ;
5      while $\mathbb{T}^i$ is executed do
6          if $cycle\_cntr == \Delta_{slice}$ then
7              # Get the frequency ($F^{M_i}_{slice-1}$) during $\Delta_{slice-1}$ and temperature
                 ($Temp_{slice}$) at the end of $\Delta_{slice-1}$ ;
8              if $Temp_{slice} \geq Temp^{Hi}_{thr}$ then
9                  $V_{in} = V_{dd}[1]$ ;
10             if $Temp_{slice} \leq Temp^{Low}_{thr}$ then
11                 $V_{in} = V_{dd}[L]$ ;
12             if $Temp^{Hi}_{thr} > Temp_{slice} > Temp^{Low}_{thr}$ then
13                 for $m = 2$ to $(L - 1)$ do
14                     $F^{M_i}_{slice} = getFreq(V[m], Temp_{slice})$ ;
15                     $F^{avg} = (F^{M_i}_{slice-1} + F^{M_i}_{slice})/2$ ;
16                     if ($F^{avg} \geq F^{M_i}\_Base[i]$ and
                         $Temp_{derived}(V[m], F^{avg}, Temp_{slice}) \leq$
                         $Temp_{Crit}$) then
17                         $V_{in} = V_{dd}[m]$ ;
18                         break ;
19             $F^{M_i}_{slice} = getFreq(V_{in}, Temp)$ ;
20             Add $F^{M_i}_{slice}$ to $FreqArray^{M_i}\{\}$ ;
21             $cycle\_cntr = 0$ ;
22         else
23             $cycle\_cntr++$ ;
24         if $cycle\_cntr == TotCycle(M_i)$ then
25             $SpareCycle^{O_i} = \frac{F^{M_i}\_Base[i] - Mean(FreqArray^{M_i}\{\})}{TotCycle(M_i)}$ ;
26             $CycleLeft^{O_i} = SpareCycle^{O_i} + SchedTotCycle^{O_i}$ ;
27             if Highest version of $O_i$ is scheduled then
28                 Fetch the scheduled $O_i$ and execute ;
29             else
30                 Find the highest possible $O_i$ having $TotCycle^{O_i} \leq$
                     $CycleLeft^{O_i}$, and start execution ;
31         $Slack\_after\_T_i = Extended\_end\_Time(T_i) - CurrTime$ ;
32         if ($cycle\_cntr == TotCycle(O_i)$) && ($Slack\_after\_T_i \geq$
             $Break\_Even\_Time$) then
33             # Turn off the core ;
34             while $Slack\_after\_T_i > Break\_Even\_Time$ do
35                 $Slack\_after\_T_i$-- ;
36             #Turn on the core ;
```

4 SIMULATION FRAMEWORK FOR *MAFIN*:ONLINE

We simulated a tiled homogeneous CMP with 4 x86 OoO cores using the gem5 full system simulator[5]. Each replicated tile includes a processor core, data, and instruction local L1 caches. A shared L2 cache is centrally located, and a 2D-mesh-NoC connects the tiles and the L2 cache. Performance traces of multithreaded PARSEC benchmarks [4] from gem5 were analyzed using McPAT-monolithic [10] for power simulation. Power traces were then sent to HotSpot 6.0 [19] to generate temperature traces, utilizing thermal properties of the 14nm FinFET technology [7]. A 1ms interval (Δ_{slice}) was used for collecting periodic performance traces from gem5. Although the TEI effect in FinFET results in frequency no longer fixed at various temperatures, for our simulation, we assume a fixed temperature during the entire Δ_{slice} [7]. The default parameters are listed in Table 2.

Table 2: System Parameters

Parameters	Values
Number of Cores	4 (each 2 cores represent a single P_i (in Figure 3))
Core Model (Technology)	x86 (14nm FinFET)
Nominal Frequency	3.5GHz
$V_{dd}[1]$, $V_{dd}[L]$ (at cores)	0.65v, 0.75v
Private L1 D/I Cache	64KiB, 4W SA, LRU
Shared L2 Cache	8MiB, 16W SA, LRU
DRAM	8GiB
Ambient Temperature	40 °C

Table 3: Tasks formation with PARSEC. (Acronyms: Blackscholes (*Black*), Bodytrack (*Body*), Canneal (*Can*), Dedup (*Ded*), Fluidanimate (*Fluid*), Freqmine (*Freq*), Streamcluster (*Stream*), and X264 (*X264*)). The execution lengths (*EL*s) are in million cycles. *Black (2)* implies 2 copies of *Black*, which is the same for others.

Tasks	**Benchmarks (M_i, O_i)**	**EL ([M_i], [O_i])**	**Sel. Oi [EL]**
T_1	*Black (2), Body (2)*	[100], [60]	#1 [60]
T_2	*Stream (2), Can (2)*	[200], [50, 70, 100]	#3 [100]
T_3	*Ded (2), Fluid (2)*	[200], [40, 80, 100]	#3 [100]
T_4	*Fluid (2), Freq (2)*	[200], [60, 120]	#2 [120]
T_5	*Body (2), X264 (2)*	[160], [60, 70, 90]	#2 [70]
T_6	*X264 (2), Ded (2)*	[180], [30, 50]	#1 [30]

We set the threshold values used in Algo 1 as follows: $Temp_{thr}^{Hi}$ = 80 °C, $Temp_{thr}^{Low}$ = 77 °C, $V_{dd}[L] = 0.75v$ and $V_{dd}[1] = 0.65v$ [7]. We assumed a maximum safe temperature of 82 °C, hence, we set $Temp_{thr}^{Hi}$ = 80 °C to restrict the thermal overshoot beyond 82 °C. By considering different technology nodes and process variations, both temperature values for determining hotspots and the threshold temperatures can be tuned, which we intend to explore in our future work. To maintain an average frequency of 3.7GHz, we set V_{in} so that we can maintain the lowest and the highest frequencies at 3.0GHz (for $Temp_{thr}^{Low}$, V_{Lo}) and 3.9GHz (for $Temp_{thr}^{Hi}$, V_{Hi}), respectively. Note that all of our thresholds can be adjusted by considering different technical specifications of the underlying circuitry and/or technology nodes. However, our employed on-chip VR is assumed to be installed per core, and each VR has a switching speed of 20 mV/ns [9], and the respective area and power overheads are based on a prior model [18].

We have evaluated the following core-based techniques:

- **Baseline** – the default model with system parameters according to Table 2;
- **MAFin** – the proposed techniques;
- **ENPASS** – prior DVFS technique for FinFET-CMPs [15].

By considering prior arts, where PARSEC [4] can be used in an approximated computing paradigm [1, 17], we constructed our task-set by defining each of the 6 tasks[1] with multiple copies of PARSEC applications with large input sets (detailed in Table 3).

5 RESULTS AND ANALYSIS

We will now discuss the efficacy of *MAFin: Offline* (Sec. 3.1) and the benchmark based evaluation of *MAFin: Online* (Sec. 3.2).

5.1 *MAFin*: Offline

We define **N**ormalized **A**chieved **Q**oS (NAQ) as the ratio between the actually achieved QoS for the DAG, and the maximum achievable QoS by running the highest versions of all tasks. NAQ can be formulated as: $NAQ = \frac{\sum_{i=1}^{n} Acc_i^j}{\sum_{i=1}^{n} Acc_i^{k_i}}$, where k_i is the highest version of task T_i. We compared our strategy with prior arts, $Task\_Deploy$ [14] and *Prepare* [9], and the results are shown in Figure 5. Towards a fair comparison with $Task\_Deploy$, the overall power constraint is based on the considered higher temperature limit (82 °C) of the experimental framework of *MAFin*, which we have also used to evaluate *Prepare*. Next, we considered our comparison by uniformly choosing M_i of the tasks between 20% to 80% of len_i. As execution demand of individual tasks goes up (due to increase in Sys_{WL}), *MAFin* maintains improved QoS by achieving higher NAQ than $Task\_Deploy$. *MAFin* is able to maintain 70% QoS at 65% workload where $Task\_Deploy$ achieves 55% QoS, as the considered overall power budget in $Task\_Deploy$ would scale up with the higher Sys_{WL}. Moreover, $Task\_Deploy$ also allows unlimited tasks migration, that incurs additional overhead. However, for all workloads, NAQs obtained by *Prepare* are surpassed by *MAFin*, which attributes to the fact that *Prepare* employs DVFS policy to maintain the power budget and thus, often chooses the lower version of O_i.

5.2 *MAFin*: Online

First, we evaluated the impact of TEI on our considered benchmarks (Table 3), while employing Algo 1. For compute intensive applications, (e.g. Black, X264, etc.), the peak temperature of the processor cores is comparatively higher, so is the TEI exploitation, rather than the memory intensive ones (e.g. Freq, Stream, etc.). Overall, by exploiting TEI, Algo 1 is able to improve frequency by 6.6% on an average while the range is in 6.1% to 7.1%. This enhanced frequency although improves the performance, but higher frequency potentially incurs power as well as thermal overheads which is also taken care by our SHE combating mechanism of Algo 1. We also compared our SHE cognizant TEI exploitation with ENPASS [15], a TEI agnostic DVFS based thermal management technique for FinFET based CMPs. The aggressive DVFS approach of ENPASS offers better thermal efficiency than *MAFin* by lowering core frequency up to 10% (with an average reduction of 8%), but *MAFin*'s TEI cognizant thermal management ensures thermal safety while improving performance through TEI exploitation. Figure 6 shows how *MAFin* surpasses ENPASS and Baseline systems in terms of operational frequency. Lower frequency maintains lower peak temperature for ENPASS (a temperature range of 76.3 °C to 78.5 °C) than Baseline and *MAFin*, which is depicted in Figure 7. *MAFin* maintains peak temperature within a range of 79.6 °C to 80.3 °C, which confirms no temperature overshoot beyond 82.0 °C. Note that, both ENPASS and *MAFin* shows better thermal efficiency over Baseline, ranging the peak temperature between 81.6 °C and 83.0 °C).

Improving frequency by employing Algo 1 trims the execution lengths of both M_i and O_i of individual tasks (as depicted in Figure 4), which generates scope either to execute higher version of O_i or to power gate the cores or both. To showcase the efficacy of *MAFin*, we scale the execution of our example shown in Figure 3), and the respective execution lengths are given in Table 3. In this example, apart from T_5 and T_6, all other tasks are executed with their highest versions. By applying Algo 1 to this task-set while considering our schedule of Figure 3, we observed that, slacks have been generated after end of each tasks, and Algo 1 is also able to execute highest versions of both T_5 and T_6. Due to execution of highest versions for both T_5 and T_6, the slack spans have been reduced, however, these spans are sufficient (i.e. higher than break-even-time of the cores) to power gate the respective cores. The updated schedule is illustrated in Figure 8, and the overall QoS improvement of 8.3% is shown in Table 4. By power gating the respective cores at the

[1]For real-time systems, the executed applications must be known to assure meeting deadlines.

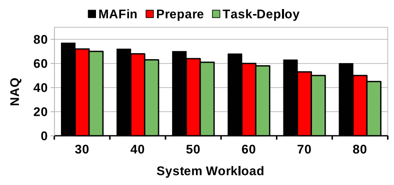

Figure 5: Sys_{WL} vs. NAQ.

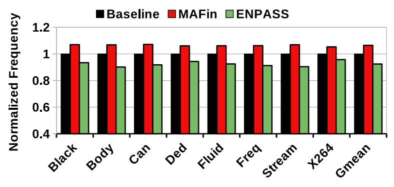

Figure 6: Impact of TEI on Frequency.

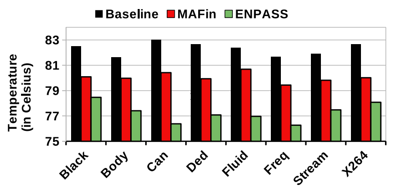

Figure 7: Peak Temperature.

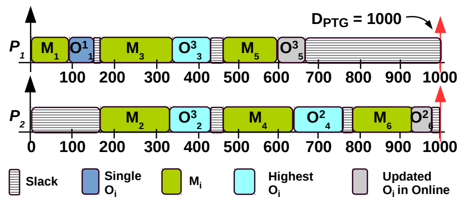

Figure 8: Updated Schedule by employing Algo 1. Timeline is based on Table 3. Cores are power-gated during slacks.

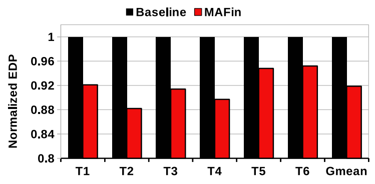

Figure 9: EDP Gains.

end of each task, we achieved an average EDP gain of 8.1%, with a maximum gain of 12%, which are plotted in Figure 9.

Table 4: Outputs of *MAFin: Offline* and *MAFin: Online*

Tasks	Mapped Core	Scheduled Version (Offline)	Updated Version (Online)	Amount of Slack
T_1	P_1	1	1	6.9%
T_2	P_2	3	3	7.0%
T_3	P_1	3	3	6.1%
T_4	P_2	3	3	6.2%
T_5	P_1	2	3	2.5%
T_6	P_2	1	2	1.7%
Improvement in Achieved QoS			8.3%	

Discussion: Both offline and online techniques of *MAFin* outperform prior arts [9, 14, 15] in terms of performance. SHE cognizant TEI exploitation of *MAFin* incurs no noticeable implementation cost, unlike these prior arts. In Prepare [9], that considers a MOSFET based CMP, the performance improvements are achieved by implementing some hardware mechanisms to detect memory stalls, whereas, ENPASS [15] considers a FinFET based CMP, but its aggressive DVFS is TEI agnostic, misses the chance of TEI induced performance improvements. From offline perspective, *Task_Deploy* [14] proposes a power aware task migration, potentially aggravating the performance. However, *MAFin*'s TEI aware combined offline-online scheduling mechanism exploits higher temperature as a service while guaranteeing thermal safety, which enables one to finish the tasks early with improved accuracy as well as energy efficiency.

6 CONCLUSIONS

In *MAFin*, we introduce a novel hybrid offline-online *scheduling strategy* for approximate real-time tasks in a FinFET based CMP platform. *MAFin* generates a schedule for a dependent task-set with an objective to maximize the QoS, while respecting other system-wide constraints. At runtime, while ensuring thermal safety, *MAFin* exploits the TEI property of FinFET based processor cores, where operating frequency increases at higher temperature, thus reducing the execution time of the individual tasks. This reduced execution-time will further be employed either to enhance QoS by executing more of the tasks' optional parts or to improve energy efficiency by turning off the core. Simulation results show that, *MAFin* surpasses the prior art for a variety of simulation scenarios.

ACKNOWLEDGMENTS

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through grants $EP/X015955/1$, $EP/V000462/1$. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

REFERENCES

[1] S. Achour and M. C. Rinard. 2015. Approximate Computation with Outlier Detection in Topaz. *SIGPLAN Not.* (2015).
[2] W. Ahn et al. 2018. Integrated modeling of Self-heating of confined geometry (FinFET, NWFET, and NSHFET) transistors and its implications for the reliability of sub-20nm modern integrated circuits. *Microelectronics Reliability (Elsevier)* (2018).
[3] H. Aydin et al. 2001. Optimal reward-based scheduling for periodic real-time tasks. *IEEE TC* (2001).
[4] C. Bienia et al. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *PACT.*
[5] N. Binkert et al. 2011. The Gem5 Simulator. *SIGARCH CAN* (2011).
[6] E. A. Burton et al. 2014. FIVR — Fully integrated voltage regulators on 4th generation Intel® Core™ SoCs. In *APEC.*
[7] E. Cai and D. Marculescu. 2017. Temperature Effect Inversion-Aware Power-Performance Optimization for FinFET-Based Multicore Systems. *IEEE TCAD* (2017).
[8] K. Cao et al. 2019. QoS-Adaptive Approximate Real-Time Computation for Mobility-Aware IoT Lifetime Optimization. *IEEE TCAD* (2019).
[9] S. Chakraborty et al. 2021. Prepare: Power-Aware Approximate Real-Time Task Scheduling for Energy-Adaptive QoS Maximization. *ACM TECS* (2021).
[10] A. Guler and N. K. Jha. 2020. McPAT-Monolithic: An Area/Power/Timing Architecture Modeling Framework for 3-D Hybrid Monolithic Multicore Systems. *IEEE TVLSI* (2020).
[11] W. Lee et al. 2014. Dynamic thermal management for FinFET-based circuits exploiting the temperature effect inversion phenomenon. In *ISLPED.*
[12] K. A. A. Makinwa. 2018. *Temperature Sensor Performance Survey.* http://ei.ewi.tudelft.nl/docs/TSensor_survey.xls
[13] S. Mei et al. 2016. New understanding of dielectric breakdown in advanced FinFET devices — physical, electrical, statistical and multiphysics study. In *IEDM.*
[14] L. Mo et al. 2019. Approximation-aware Task Deployment on Asymmetric Multicore Processors. In *DATE.*
[15] K. Neshatpour et al. 2018. Enhancing Power, Performance, and Energy Efficiency in Chip Multiprocessors Exploiting Inverse Thermal Dependence. *IEEE TVLSI* (2018).
[16] S. Saha et al. 2022. ACCURATE: Accuracy Maximization for Real-Time Multi-core systems with Energy Efficient Way-sharing Caches. *IEEE TCAD* (2022).
[17] S. Sidiroglou-Douskos et al. 2011. Managing Performance vs. Accuracy Trade-Offs with Loop Perforation. In *ACM SIGSOFT.*
[18] W Kim et al. 2008. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *HPCA.*
[19] R. Zhang et al. 2015. HotSpot 6.0: Validation, Acceleration and Extension.. In *University of Virginia, Tech. Report CS-2015-04.*

Received 20 February 2024; revised 12 March 2024; accepted 5 June 2024