Temperature-Aware DVFS Scheduling for Deep-Drilling RISC-V Processors

Nico Rumpeltin*§, Tim-Marek Thomas†§, Malte Rücker‡, Malte Hawich*, Daniel Lohmann†, Holger Blume*
*Institute of Microelectronic Systems, Leibniz University Hanover, Hanover, Germany
†Institute of Systems Research and Architecture, Leibniz University Hanover, Hanover, Germany
‡Drilling Services, Baker Hughes, Celle, Germany
Accepted Version/Authors copy.

***Abstract*—Deep drilling systems must process sensor data and make decisions on site because the communication channel to the surface provides less than 10 bit/s, too slow for offloading. At the same time, the downhole processor must operate under extreme and varying ambient temperatures (up to 175 °C) while drawing from a limited power budget. In the target 180 nm SOI technology, static leakage accounts for less than 1% of total power, making dynamic power the dominant energy term and DVFS-based task stretching more effective than race-to-completion.**

We present a temperature-aware hardware–operating-system co-design for a RISC-V processor that enables the OS to select safe operating points based on ambient temperature and available task slack, using a model of temperature-dependent timing behavior. Evaluations with industrial FreeRTOS and synthetic AUTOSAR workloads show that the approach reduces energy consumption beyond conventional sleep and temperature-unaware DVFS strategies. In a simulated several-hour drilling process, temperature-aware scheduling reduces total energy by 43.3 percent over a WFI-only baseline, directly extending battery life in long-running downhole deployments.

***Index Terms*—RISC-V, Harsh Environment, Energy-aware scheduling.**

I. Introduction

Deep drilling systems must operate autonomously kilometers below the surface, where the communication channel provides less than 10 bit/s [1]. This is far too slow to stream raw sensor data or outsource time-critical decisions; sensing, signal processing, and actuator control must therefore be executed on site by the downhole processor itself.

But on-site processing is not the only constraint. The downhole environment exposes electronics to extreme temperatures (up to 175 °C, with temporary spikes beyond), high pressure, and vibration. Conventional power sources are unavailable; the system relies on mud-flow generators during drilling and switches to primary batteries during downtimes. Energy efficiency therefore directly determines how long the system can operate autonomously.

The combination of tight energy budgets and wide thermal variation creates a challenge that conventional energy-saving strategies cannot fully address. In typical processors, self-heating dominates, and dynamic thermal management aims to keep the chip cool. In our setting, the reverse is true: the ambient temperature exceeds self-heating by an order of magnitude,

§These authors contributed equally to this work.

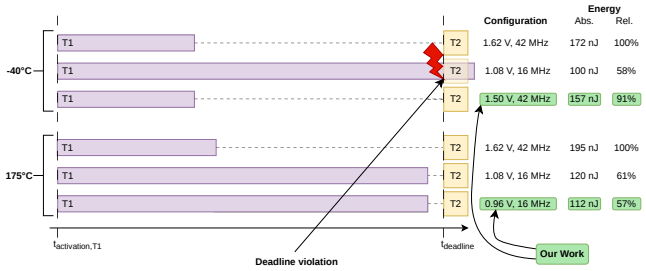


Fig. 1: Comparison between two corners and our approach. While the energy conserving corner may fail deadlines, the performance corner wastes energy. In contrast, our temperature-aware scheduling reduces energy by 43% at −40 °C while meeting the deadline.

and the maximum safe operating frequency itself depends on temperature. A voltage-frequency pair that is safe at 25 °C may cause timing violations at 175 °C, while an overly conservative configuration wastes energy at lower temperatures.

The target technology further inverts conventional wisdom. We use the X-FAB XT018 180 nm SOI process, which is optimized for robust high-temperature operation. In this technology, static leakage accounts for less than 1% of total power, even at 175 °C. Dynamic power is the dominant term. This makes race-to-completion, the standard approach in leakage-dominated technologies, the wrong strategy: finishing early and idling saves almost no energy [2][3]. Instead, we can reduce energy by stretching task execution over available slack and lowering the voltage-frequency operating point.

But exploiting this opportunity requires information that hardware alone cannot provide: task deadlines, execution history, available slack, and ambient temperature. The OS is the only layer where all of this information converges. We therefore present a hardware–operating-system co-design that extends a RISC-V processor with DVFS controls and FreeRTOS with a temperature-aware scheduler. The scheduler selects safe operating points at runtime based on a model of temperature-dependent timing behavior, and reclaims slack by stretching task execution toward their deadlines.

In summary, this paper makes the following contributions:

- A characterization of the deep-drilling environment and its implications for processor energy optimization.

- A RISC-V single-core processor with exposed DVFS controls and temperature observability.
- A temperature-aware scheduling that selects safe operating points based on ambient temperature and slack.
- An evaluation on industrial FreeRTOS workloads and synthetic AUTOSAR applications.

The remainder of the paper is organized as follows: Section II presents constraints and considerations resulting from the harsh environment. In Section III we describe the co-design, present our scheduling algorithm, and how we model the operating points. We evaluate and compare the proposed methods in Section IV, discuss the applicability of our findings and possible extensions in Section V, present in Section VI relevant literature, and finally conclude in Section VII.

II. Environment Considerations

Electronics deployed in deep drilling applications operate under extreme environmental conditions. Ambient temperatures usually reach up to 175 °C, pressures can rise up to 200 MPa, and communication bandwidth to the surface is severely limited [4][5], data must be transmitted via mud-pulse telemetry [1], which fundamentally constrains the achievable bit rate. As a consequence, processors must execute control, signal processing, and actuator management tasks autonomously and reliably on site, while operating under sustained thermal and mechanical stress.

Downhole temperature is primarily determined by the geothermal gradient, which is typically in the range of 25 °C/km to 35 °C/km and leads to formation temperatures well above 150 °C at depths of several kilometers [6]. In addition to this high steady-state temperature, operational changes such as mud circulation or pump shutdowns introduce dynamic thermal variations on shorter timescales. For the computing platform, this means that the relevant operating temperature is not fixed, but changes over the lifetime of the deployment and must therefore be considered at runtime. Crucially, both the gradual geothermal trend and the operational transients are slow compared to task-execution timescales, making temperature-aware scheduling practical: the ambient temperature can be treated as constant within each hyperperiod.

This is particularly important at the semiconductor level. Elevated temperature affects transistor switching behavior and thereby the maximum safe operating frequency of the processor. In high-temperature *Silicon-on-Insulator (SOI)* technologies, such as the 180 nm process targeted in this work, these timing effects are especially relevant because the technology is optimized for robust operation under harsh conditions rather than for maximum performance [7]. At the same time, the process technology and the additional isolation measures keep static leakage currents below 1% of total power consumption, as confirmed by our post-synthesis power analysis (see Section IV). As a result, dynamic power dominates the overall processor power consumption in our setting. Its influence further increases with clock frequency, making the current operating point a key factor for overall energy demand.

Self-heating itself is only of secondary importance in our setting. Across multiple projects on our in-house fabricated chips, we observe typical operating power in the range of 100 mW to 200 mW [8]. Using an effective junction-to-ambient thermal resistance of approximately 40 °C/W for our package and board setup, a first-order estimate with $\Delta T \approx P \cdot R_{\mathrm{th}}$ yields only about 4 °C to 8 °C temperature rise for this power range [9]. For deep drilling, the chip temperature therefore mainly follows the environment, i.e., the dominant thermal challenge is the high and varying ambient temperature rather than power-induced self-heating [10].

The energy supply situation further strengthens the need for efficiency. During regular operation, energy is typically provided by a generator driven by the mud flow. However, drilling operations must pause occasionally, e.g., when extending the drill string or during measurements that would otherwise be distorted by vibrations. During these phases, the system relies on primary batteries, due to the high ambient temperature. Reducing energy consumption therefore directly translates into longer operating time during generator downtimes and reduced battery replacement requirements.

In summary, downhole processor design is constrained by high and varying ambient temperatures, limited communication bandwidth, and restricted local energy supply. For the processor, the key implication is not primarily self-heating, but that safe voltage–frequency operating points depend on the current temperature. This makes temperature awareness a prerequisite for both reliable operation and effective energy optimization, motivating the hardware–operating-system co-design.

III. System Design

Figure 1 highlights the goal of our design. While a fixed voltage and frequency can provide a safe operating point, it might fail due to changing ambient temperature. In addition, by choosing the safe, commonly not most efficient, operating point we waste energy in deployment.

Thus, to enable safe and energy conserving operation, we need to implement several changes to the target system. First, we need to adjust the *System-on-Chip (SoC)* and add a *Dynamic Voltage and Frequency Scaling (DVFS)* module. Then, we need to model temperature dependencies of the operation points, in particular the possible frequencies and voltages per temperature. At last, we need to modify the *Real-Time Operating System (RTOS)* scheduler to change the HW configuration online, based on the modeled values and bookkeeping.

A. SoC adjustments

We employ a simplified SoC adapted from an industrial deep-drilling application as a system platform. The SoC comprises a custom RISC-V processor core, 8 kB of data memory, 64 kB of instruction memory, and a Core-Local Interrupt Controller module that provides timer-based interrupts. The architecture is kept minimal to emphasize deterministic execution and low complexity. This enables the systematic exploration of voltage–frequency operating points and their interaction with

varying environmental conditions. In this design, SRAM occupies 84.7 % of the area, while logic occupies only 15.3 %.

Since dynamic power consumption dominates in our use-case, reducing it is the most effective way to reduce overall power consumption. Therefore, as dynamic power scales quadratic with supply voltage and linearly with frequency, DVFS emerges as the most effective mechanism. For a fixed amount of work over a fixed time window we can maximize the savings by stretching the task executions over the available slack and running the processor at the lowest possible voltage-frequency operating point.

In order to enable frequency scaling on the system platform, the `mtime` counter [11] was relocated to a separate frequency domain. The frequency itself is adjusted using programmable clock dividers, which allow rapid reconfiguration of the operating frequency, with only minimal delay. In addition, the SRAMs, which are supplied at the foundry-recommended operating voltage of 1.62 V, cannot be voltage-scaled. Due to this, they were relocated to a separate static voltage domain, which was connected to the voltage-scalable domain by level shifters. Voltage scaling has to take place in a target system via an external voltage regulator.

Both voltage and frequency can be controlled directly by the *Operating System (OS)* via a memory-mapped register interface. The hardware only provides the mechanisms required to adjust the configuration at runtime. The responsibility for selecting a suitable voltage–frequency operating point is fully delegated to the OS.

B. Modelling temperature dependence

For the selection of the suitable operating point for each task, the OS needs to know at which temperature which voltage–frequency combination is viable. This creates the need to model the temperature dependence.

While the temperature dependence of leakage currents has already been discussed in Section II, the wide temperature range encountered in deep drilling applications also affects the circuit's switching behavior and therefore the maximum operating frequency. This effect originates from the temperature-dependent reduction of threshold voltage and degradation of charge carrier mobility in the transistors. While the reduction of the threshold increases switching speed, the opposite degradation of charge carrier mobility typically dominates at higher temperatures, resulting in an overall reduction in maximum frequency. This relationship between the maximum achievable frequency and temperature is depicted in Figure 2a. It can also be seen that the closer the supply voltage reaches the threshold voltage, the effect of threshold voltage reduction becomes significantly more important.

This effect can be exploited by operating the circuit at a higher clock frequency that is adapted to the temperature, rather than using the minimum achievable frequency across the entire temperature range. This directly leads to an increase in the computational power of the processor without having to increase the supply voltage. Conversely, ignoring temperature dependence forces the designer to choose between two suboptimal strategies: using a conservative frequency rated for the worst-case temperature wastes performance at lower temperatures, while using a frequency rated for nominal temperature risks timing violations when the chip heats up. For example, at 1.08 V, the maximum safe frequency drops from 21 MHz at 175 °C to 16 MHz at −40 °C, a 24% penalty if a single worst-case operating point is used.

However, it is important to note that the circuit must never be operated at a frequency that is too high for the current operating temperature. In order to solve this problem, it is necessary to predict both the frequency at which the circuit can operate at a given temperature and the temperature at which the circuit will operate. Since the chip temperature in deep-drilling operation is dominated by the ambient temperature, the maximum chip temperature can be predicted comparatively easily.

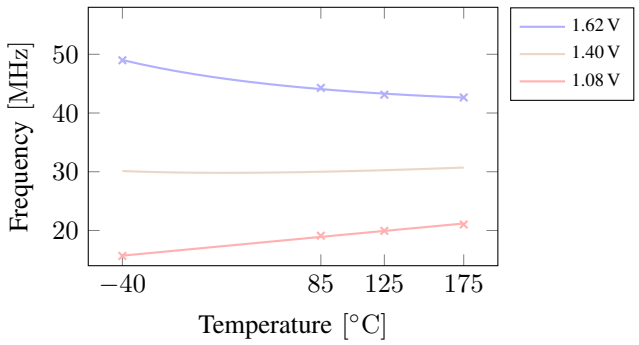
To determine safe voltage–frequency operating points for a given temperature, the maximum achievable operating frequency must be known. A common method to obtain this information is *Static timing analysis (STA)*, which determines the critical path delay of the circuit for a given voltage and temperature. However, this requires temperature, voltage, and process libraries for each corner (i.e., , each combination of temperature, voltage, and process variation) that is to be analyzed. Since the number of corners provided by the foundry is limited, the number of obtainable data points is also limited, necessitating a model based approach. Equation 1 models the relationship between maximum operating frequency, voltage, and temperature [12].

$$f \propto \frac{(V_{DD} - (V_{th} + k \cdot (T - T_{ref})))^{\xi}}{V_{DD} \cdot T^{\mu}} \tag{1}$$

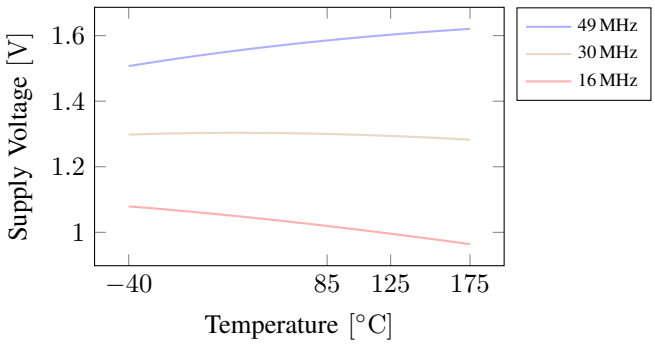
V_{DD} denotes the supply voltage, T the temperature, and V_{th} the threshold voltage at a reference temperature T_{ref}. The parameters k, ξ, and μ are empirical, technology-specific constants. To obtain the values of these constants, the model was curve-fitted against eight data points obtained from the STA, using the Levenberg–Marquardt algorithm. As it can be seen in Figure 2a, the model closely fits the data points.

To further ensure the validity of the model beyond the discrete synthesis points, the approximation was verified against the SPICE models provided by the foundry. For this purpose, the critical path of the design was recreated, and the resulting delay was simulated for several discrete corners. This comparison confirms that the fitted model captures the temperature-dependent frequency behavior with sufficient accuracy for runtime operating-point selection.

Instead of increasing the clock frequency, the operating voltage can also be reduced while keeping the frequency constant. Since the dynamic power consumption increases quadratically with voltage but only linearly with clock frequency, this is effective for reducing dynamic power. However, the circuit must remain within a stable operating region and continue to meet its timing constraints. We use Equation 1 to determine suitable operating points under these conditions.



(a) Max. frequency of the processor core over voltages.



(b) Supply voltage required for fixed frequencies over temperature.

Fig. 2: a) The three curves are generated by the model; crosses mark values from front-end synthesis results. The model reproduces these accurately, demonstrating that it reliably predicts safe operating frequencies across the full temperature range. b) The voltage can be reduced by up to 0.12 V over the total temperature range, corresponding to a dynamic-power reduction of up to 21% while safely maintaining the same clock frequency.

In Figure 2b, the voltage that is necessary to operate the circuit constantly at a fixed operating frequency is depicted. Since the circuit slows down at high supply voltages as the temperature rises, the voltage must be increased to compensate. However, at the low baseline supply voltage of 1.08 V, the voltage can be reduced by up to 0.12 V as the temperature rises, which corresponds to a reduction of up to 21 % of the dynamic power due to its quadratic dependence.

To make runtime adaptation feasible, the platform requires access to the current ambient temperature. In the considered drilling scenario, this information is already available, since temperature sensors are part of the drill string instrumentation.

As voltage and frequency control are intentionally delegated to the OS, safe operation can no longer be ensured by hardware alone. Instead, the OS must select operating points according to the current temperature and guarantee that the system remains in a stable region. Temperature awareness is therefore not merely beneficial for efficiency, but a prerequisite for correct system operation under varying environmental conditions.

C. OS adjustments

As we have seen in Section II, not only the hardware but also the system software needs to consider the environmental conditions. The OS is the only place where all information needed for the implementation of temperature-aware scheduling is available. In particular, following is needed at the time of a task switch:

1) $conf_{hw}$: current hardware configuration
2) $conf_{avail}$: *possible* hardware configurations
3) T_{amb}: ambient temperature
4) t_{T_x}: expected runtime of the xth to-be-dispatched task.
5) $t_{T_{x,i}}$: deadline of the to-be-dispatched ith task instance.

We track each task's execution times over the lifetime of the application; as deadlines we assume the arrival of the next task instance. While that does not result in a guaranteed optimal schedule, it allows us to deploy the approach to arbitrary applications. When the *Worst-Case Execution Times (WCETs)* and deadlines for each task are given, e.g., gathered by a static analysis [13], we could predict a schedule which considers future task activations. However, static analysis is a tedious process and, as the main goal is a safe and accurate WCET estimation, prone to pessimism. In our case, even a dynamic analysis is quite costly to perform, as we would need to measure task executions for all possible configurations at every possible ambient temperature, resulting in 172 runs per application. Thus, we base our decisions on on-premise bookkeeping.

Listing 1: Pseudocode for temperature-aware scheduling

```
1  procedure schedule_next_task(T_{x,i}):
2    slack := t_{T_{x,i}} - t_{T_x} - t_cur
3
4    if slack > 0 then
5      stretch_factor := (t_{T_x} + slack) / t_{T_x}
6    else
7    conf_target := match_frequency(conf_avail, stretch_factor)
8    if not meets_deadline(conf_target, t_{T_x}, t_{T_{x,i}}) then
9      conf_target := max_frequency(conf_avail) # Fallback
10   end if
11   end if
12
13   set_hw_configuration(conf_target)
14   context_switch(T_{x,i})
15 end procedure
```

Listing 1 describes our proposed scheduling. As the current hardware implementation only provides two possible frequencies, the algorithm remains relatively simple. If more were available, we could fit more fine-granular to the available slack.

In addition, we can base our decision on the shortest, average, average with a safety margin, or longest runtime we recorded of the tasks. As we want to highlight potential savings, we employ the most optimistic approach in our evaluation, i.e., the shortest seen runtime so far. However, we did not encounter any deadline misses during our evaluation.

Safety can be expected as long as the shortest-observed-runtime assumption holds (i.e., when task execution times remain stable or decrease over time). For workloads with high variability or sporadic interrupts, the safety margin can be increased by using the average or longest observed runtime instead of the shortest, at the cost of reduced energy savings.

If the calculated slack is insufficient, we fall back to the highest safe frequency. Please note, that the implementation is still based on the scheduler of FreeRTOS, which is a best-effort scheduler. The decision *which task needs to run* stays the same, we only change the hardware configuration on a task basis.

More technically, we introduce hooks into the task switch function of FreeRTOS, one pre-switch and one post-switch. We take timestamps in each to calculate task durations, and track them in the TaskApplicationTag. Before we switch into a task, we execute the described algorithm and change the hardware configuration accordingly. Costs for switching can be tracked as well, and considered in the algorithm. Note that the reported energy numbers in our evaluation are based on steady-state power at each operating point and do not include voltage-regulator efficiency, DVFS transition latency, or transition energy; these costs depend on the specific hardware.

IV. Evaluation

For the evaluation, a front-end synthesis using Cadence Genus of the system described in III was performed (the SoC has been synthesized but not yet fabricated; power figures are therefore based on post-synthesis simulation). A mix of five algorithms (Gaussian elimination, prime sieve, Fourier transformation, AES encryption, and matrix multiplication) was then simulated on the netlist to obtain a vector for the power analysis. These algorithms were chosen because they provide a close representation of the computational load during deep drilling [14]. In addition to this, a vector representing the behavior in the WFI state was recorded. These vectors were then used to calculate the power consumption for both sleep and load states using the Cadence Joules power analysis tool. This analysis was run for all possible corners (−40 °C, 25 °C, 85 °C, 125 °C, 175 °C, 1.62 V and 1.08 V) for all possible configurations. Using these values, the model from Section III was then used to determine the values for each operating point not covered by the foundry provided corners. Based on these results, we want to answer the following questions:

1) What is the overhead introduced by the bookkeeping?
2) How well do our scheduler adjustments perform?
3) In a realistic setting, what are reachable savings?

For applications, we were provided with a small industrial application implemented in FreeRTOS [15]. It consists of 6 tasks, including an idle task, 3 queues, and only includes purely periodic task activations. To gauge how different percentages of idle time affect the savings, we made the task delays (i.e., the task activations) adjustable, so we can scale to 10%, …, 90% spent in the idle task resulting in 8 applications. The applications run on the soft-core for 1000 task switches and the timings of each task and idle periods are gathered.

In addition, we implemented an application generator based on Kramer et al. [16] for AUTOSAR [17]. AUTOSAR stems from the automobile industry; however, especially the task timings reflect embedded applications in general and are applicable to our use case. We generate 9 applications, each with a different number of tasks (one to nine), and simulate them to gather timing information. Thus, we evaluate our approach on 18 applications in total.

First, we want to take a look at the *overhead* introduced by our bookkeeping in FreeRTOS. As we need some memory to track execution times, we allocate – mainly at the system start – 16 B of memory in the ApplicationTaskTag for each task. The overhead for this is negligible in comparison to the task creation itself, we require on average 166 ± 0 cycles, while the task creation needs $7\,782.17 \pm 2\,308.2$ cycles. In addition, on every task switch we need to a) save start and end timestamps (58.69 ± 11.81 cycles) and b) calculate the stretching factor for the new task (571.38 ± 290.2 cycles).

Next, we want to highlight the gains from different mechanisms, especially of the *temperature-aware scaling*. Figure 3a depicts for all configurations of the FreeRTOS application the gains in comparison to an implementation including only the lightweight sleep mode. As expected, the slowdown approach already yields significant savings, even without scaling the voltage. Once again, this is the case for a fixed time window for a fixed amount of work. In the best case, we reduce needed energy by 61.96 pp at −40 °C and 59.29 pp at 175 °C.

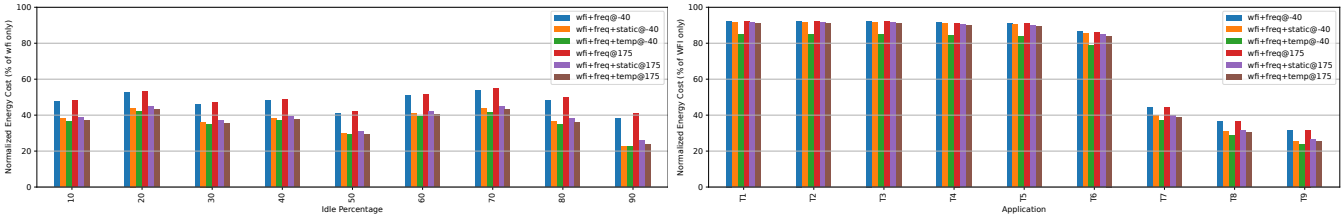
It is further reduced when voltage scaling is introduced with two fixed voltages. For that configuration, the needed energy reduced by further 15.24 pp at −40 °C and 14.71 pp at 175 °C in the best case. When we allow freely adjusting the voltage we can optimize the savings by a minimum of 0.19 and up to 2.44 pp.

In addition, Figure 3a depicts an interesting data point: while the time spent in the idle task increases, it does not always translate to higher savings (see 50% vs 60%). The reason for this stems from the way we change the application configuration to achieve the different idle percentages; we adjust the task delays and thus the "slots" of slack available to slow down task executions. While we spend more time in the idle task, the available slack is too short to slow down the execution and thus a higher idle percentage can result in less savings.

Figure 3b depicts the AUTOSAR applications and a bad case for our approach, as almost no task executions can be slowed down. Only in idle phases we can, in addition to the WFI-sleep mode, reduce the frequency and voltage supply. In this case our savings are highly dependent on the load of the processor. For TASK1 the load is roughly 15%, TASK2 20%, TASK3 25%, and for TASK{4-9} between 60% and 70%. However, even at the highest load (TASK9) our temperature-aware DVFS can reduce the costs by additional 6.8 percent points at −40 °C.

The additional energy savings from the temperature-aware scheduling already provide an extension of battery lifetime over the whole drilling process. To emphasize this, we simulate *a several hour-long drilling process*. We assume that we drill to the depth of 7000 m, with 600 m/h [18]. As application, we assume the task activation pattern of the 40 percent idle time FreeRTOS application. For each hyperperiod we chose the closest working point from our front-end synthesis (steps of 5 °C) and calculated the costs.

As we can see in Table I, we reduce the costs to 43.33 percent, which saves significant energy over the very long



(a) FreeRTOS application configurations

(b) AUTOSAR applications

Fig. 3: (a) Normalized costs for all configurations of the FreeRTOS industrial application. (b) Normalized costs for all AUTOSAR applications. We depict the two temperature extremes −40 °C and 175 °C. The baseline is a configuration containing only a *Wait For Interrupt (WFI)* sleep mode. Temperature-aware DVFS consistently reduces energy beyond conventional "static" DVFS and WFI-only baselines across all workloads, down to 22.8 percent at low temperatures.

	WFI (baseline)	WFI+freq	WFI+freq +static	WFI+freq +temp
Abs. costs in kJ	16.27	10.51	9.49	9.22
Rel. savings (%)	0	35.41	41.66	43.33

TABLE I: Savings for a several hour-long drilling process. Each extension incrementally improves savings; temperature-aware voltage selection adds the largest cumulative benefit over the full drilling process.

lifetime of the application, extending the battery lifetime correspondingly. Also, please note that our scheduling is not yet perfect. From the 2 000 ticks for a single hyperperiod we are still left with 160 idle ticks, while we can slow down the execution for 1 040 timer ticks. The idle ticks are left over as no task execution fits into an idle phase.

V. Discussion

As we have seen in Section IV, in the drilling process simulation, our approach still has room for improvement. If we had full knowledge at every single decision and over several hyperperiods into the future, we could reduce the amount of ticks spent in the idle task to virtually zero. For this, an analysis needs to be performed; we could use a whole-system analyzer like ARA [19] in conjunction with gathered *Worst-Case Response Times (WCRTs)* using a tool like FRET [20]. Still, these analyses are needed for every application, every version, and every configuration. Our bookkeeping approach in contrast is practical and easily deployable.

Another aspect that we did not yet consider, is how we can handle the temperature range above the specification. As described in Section II, above 200 °C we observed a noticeable increase in leakage until the SRAMs no longer provided reliable access, resulting in a device crash. However, the system was able to resume normal operation after a cooldown phase.

We want to exploit this observation by handling execution above the crucial temperature similar to intermittent computing. Normally, intermittent computing handles transient energy sources and the fact that it cannot be predicted *when energy is available* [21]. In our case, we cannot predict when the temperature cools down enough to continue computation. Thus, we want to optimize a *temperature budget*.

VI. Related Work

Approaches for *reclaiming slack* for energy savings are not new per se. Aydin et al. [22] published scheduling algorithms, which reclaim slack by slowing down the execution using DVFS. They mostly focus on reclaiming the difference between the static schedule calculated using WCET estimates and the actual execution time. There exists a plethora of similar works, too many to highlight everyone individually [23]. In contrast, we consider temperature-dependent operating points.

Temperature-aware scheduling, or in literature more commonly dynamic thermal management, aims to ensure that the self-heating of the board stays under a predefined threshold. Several techniques can be employed, mostly DVFS, global clock gating, and instruction-level parallelism. However, in all cases the self-heating is managed [24][12]. The temperature range that occurs in the related work is rather small compared to the variation in ambient temperature during deep-drilling operations. Our work focuses on adjustments according to the current temperature to reduce power consumption.

VII. Conclusion

We presented a temperature-aware hardware–operating-system co-design for a RISC-V processor in high-temperature 180 nm SOI technology, combining DVFS hardware with a FreeRTOS scheduler that selects safe operating points based on ambient temperature and task slack. On industrial FreeRTOS and synthetic AUTOSAR workloads, the approach consistently reduces energy beyond sleep-only and temperature-unaware-DVFS baselines; in a simulated several-hour drilling process, temperature-aware scheduling achieves 43.33 percent total energy reduction over a WFI-only baseline. These savings translate directly into extended battery life in deployments where the cost of replacement is high, demonstrating that temperature-aware slack reclaiming is the appropriate energy-optimization strategy for high-temperature SOI systems.

Future work will focus on extending the approach beyond the CPU to include peripheral subsystems and clock-tree management in order to exploit additional system-level power savings. Furthermore, we plan to investigate strategies for operating beyond the nominal temperature range by introducing temperature-budget–aware scheduling concepts inspired by

intermittent computing, enabling graceful degradation and recovery in extreme thermal conditions.

Acknowledgment

We thank Mareike Burg for her help with the AUTOSAR task generator.

References

[1] T. Stuckenberg, M. Rücker, M. Gottschlich, N. Rother, R. Nowosielski, F. Wiese, and H. Blume, "Design and evaluation of a 180 nm powerline communication ASIC for harsh environment," *Microprocess. Microsystems*, vol. 97, p. 104742, 2023.

[2] N. Rumpeltin, T.-M. Thomas, M. Rücker, M. Hawich, D. Lohmann, and H. Blume, "A hardware/operating system co-design approach for energy optimization in harsh environments," in *2025 Kleinheubach Conference*, Sep. 2025.

[3] N. Rumpeltin, T.-M. Thomas, M. Rücker, M. Hawich, D. Lohmann, and H. Blume, "Cpu-power optimization in harsh environments using temperature-aware scheduling," in *2026 IEEE 37th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2026, pp. 163–164.

[4] M. Hawich, M. Rücker, T. Stuckenberg, and H. Blume, "The HOT-RISC-V: A high-operating-temperature RISC-V architecture for extreme conditions," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS) – Proceedings of the 25th International Conference*, ser. Lecture Notes in Computer Science. Springer, 2025.

[5] J. Clegg, "New advances in high temperature electronics and sensors for geothermal drilling," *GeoTHERM Journal*, vol. 2, 2024.

[6] R. Brinkmann, *Abriss der Geologie*. Enke 1977-1980, Stuttgart, 1977, no. 2.

[7] D. K. Schroder, *Semiconductor Material and Device Characterization*, 3rd ed. John Wiley & Sons, 2006.

[8] T. Stuckenberg, M. Rücker, N. Rother, R. Nowosielski, F. Wiese, and H. Blume, "Powerline Communication System-on-Chip in 180 nm Harsh Environment SOI Technology," in *2021 IEEE Nordic Circuits and Systems Conference (NorCAS)*. IEEE, 2021, pp. 1–5.

[9] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and A. S. Lavine, *Fundamentals of Heat and Mass Transfer*, 8th ed. John Wiley & Sons, 2018.

[10] (2007) Integrated circuits thermal test method environmental conditions—natural convection (still air). JEDEC Solid State Technology Association. [Online]. Available: https://www.jedec.org/standards-documents/

[11] *The RISC-V Instruction Set Manual – Volume II: Privileged Architecture, Document Version 20211203"*, Dec. 2021. [Online]. Available: https://drive.google.com/file/d/1EMip5dZlnypTk7pt4WWUKmtjUKTOkBqh/view?usp=drive_link

[12] M. Bao, A. Andrei, P. Eles, and Z. Peng, "On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration," in *Proceedings of the 46th Annual Design Automation Conference*, 2009, pp. 490–495.

[13] M. Lv, N. Guan, Y. Zhang, Q. Deng, G. Yu, and J. Zhang, "A survey of WCET analysis of real-time operating systems," in *2009 International Conference on Embedded Software and Systems*, 2009, pp. 65–72.

[14] M. Hawich, N. Rumpeltin, M. Rücker, T. Stuckenberg, and H. Blume, "High performance instruction fetch structure within a RISC-V processor for use in harsh environments," in *Embedded Computer Systems: Architectures, Modeling, and Simulation - 23rd International Conference, SAMOS 2023, Samos, Greece, July 2-6, 2023, Proceedings*, ser. Lecture Notes in Computer Science, C. Silvano, C. Pilato, and M. Reichenbach, Eds., vol. 14385. Springer, 2023, pp. 255–268. [Online]. Available: https://doi.org/10.1007/978-3-031-46077-7_17

[15] "FreeRTOS homepage," http://freertos.org/.

[16] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *Proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded Real-Time Systems (WATERS '15)*. Washington, DC, USA: IEEE Computer Society Press, 2015.

[17] AUTOSAR, "Specification of operating system (version 5.1.0)," Automotive Open System Architecture GbR, Tech. Rep., Feb. 2013.

[18] N. K. Snyder, C. F. Visser, E. I. W. Alfred, W. Baker, J. Tucker, R. Quick, T. Nagle, J. Bell, S. Bell, D. Bolton *et al.*, "Geothermal drilling and completions: Petroleum practices technology transfer," National Renewable Energy Laboratory (NREL), Golden, CO (United States), Tech. Rep., 01 2019. [Online]. Available: https://www.osti.gov/biblio/1490999

[19] G. Entrup, B. Steinmeier, and C. Dietrich, "Ara: Automatic instance-level analysis in real-time systems," in *Proceedings of the 15th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT '19)*, Jul. 2019.

[20] A. Berger, S. Schuster, P. Wägemann, and P. Ulbrich, "Dynamic fuzzing-based whole-system timing analysis," in *IEEE Real-Time Systems Symposium, RTSS 2025, Boston, MA, USA, December 2-5, 2025*. IEEE, 2025, pp. 420–433. [Online]. Available: https://doi.org/10.1109/RTSS66672.2025.00041

[21] T. Häberlein, E. Dengler, P. Raffeck, and P. Wägemann, "Watwaos: A framework for worst-case-aware tailoring and whole-system analysis of energy-constrained real-time systems," in *IEEE Real-Time Systems Symposium, RTSS 2025, Boston, MA, USA, December 2-5, 2025*. IEEE, 2025, pp. 82–95.

[22] H. Aydin, P. Mejía-Alvarez, D. Mossé , and R. G. Melhem, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," in *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001), London, UK, 2-6 December 2001*. IEEE Computer Society, 2001, pp. 95–105.

[23] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms," in *Proceedings of the 13th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '07)*. IEEE, 2007, pp. 28–38.

[24] R. Jayaseelan and T. Mitra, "Temperature aware scheduling for embedded processors," in *VLSI Design 2009: Improving Productivity through Higher Abstraction, The 22nd International Conference on VLSI Design, New Delhi, India, 5-9 January 2009*. IEEE Computer Society, 2009, pp. 541–546.