Integration of Minimum Energy Point Tracking and Soft Real-Time Scheduling for Edge Computing

***Abstract*—In the upcoming Internet of Things era, reducing energy consumption of embedded processors is highly desired. Minimum Energy Point Tracking (MEPT) is one of the most efficient methods to reduce both dynamic and leakage energy consumption of a processor. Previous works proposed a variety of MEPT methods over the past years. However, none of them incorporate their algorithms with practical real-time operating systems, although edge computing applications often require low energy task execution with guaranteeing real-time properties. The difficulty comes from the time complexity for identifying MEP and changing voltages, which often prevents real-time task scheduling. This paper proposes an approximated MEPT algorithm, which reduces the complexity of identifying MEP down to that of Dynamic Voltage and Frequency Scaling (DVFS). We also propose a task scheduling algorithm, which adjusts processor performance to the workload, and provides a soft real-time capability to the system. With these two methods, MEPT became a general task, and the operating system stochastically adjusts the average response time of a processor to be equal to a specified deadline. The experiments using a fabricated test chip show that the proposed algorithm introduced the energy loss by only $0.5\,\%$ at most without sacrificing the fundamental real-time properties.**

***Index Terms*—Minimum Energy Point Tracking (MEPT), Dynamic Voltage and Frequency Scaling (DVFS), Adaptive Body Biasing (ABB), Real-Time Operating System (RTOS), Soft Real-Time Scheduling**

I. Introduction

In embedded systems, reducing energy consumption is raising its importance since the Internet of Things applications, e.g., sensor networks, become popular and take on a reality. Such applications are typically equipped with a tiny battery and evaluate sensor information with wireless communication. To maximize the battery cycle, the processor must reduce its energy consumption as low as possible. On the other hand, such applications deal with sensors or actuators, which requires real-time capability.

For reducing energy consumption, Dynamic Voltage and Frequency Scaling (DVFS) has been widely studied. It dynamically controls supply voltage (V_{DD}) and the frequency of a processor. Since dynamic energy is proportional to the square of V_{DD}, DVFS can drastically reduce the active processor's energy consumption. Motivated by the significant energy reduction and simple implementation, many researchers integrates DVFS into real-time systems [1], [2], [3], [4], [5], [6]. However, in the near- or sub-threshold region, leakage energy becomes dominant, and DVFS only reduces it linearly, which limits the efficiency of DVFS.

Leakage energy is proportional to the power of threshold voltage, which can be controlled linearly through body bias voltage (V_{BB}). Therefore, supplying the optimal V_{BB} for V_{DD} reduces overall energy consumption significantly. Such an operating point is called Minimum Energy Point (MEP), and several MEP tracking (MEPT) methods are proposed over the years to overcome the limitation of DVFS [7], [8], [9].

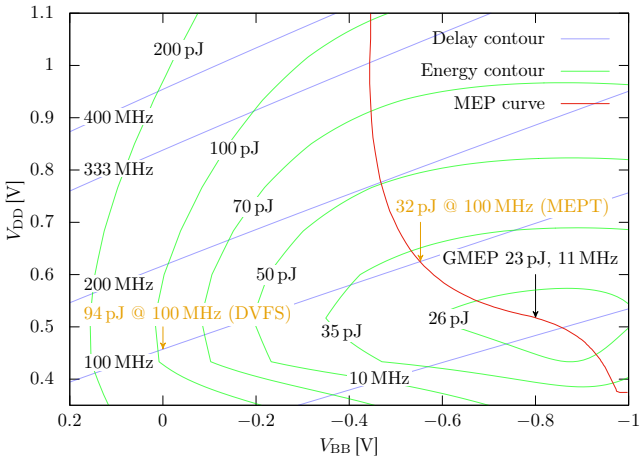

Fig. 1. Contours and MEP curve

Figure 1 shows the outline of delay contours, energy contours, and an MEP curve of a processor. MEP is the point where the delay and energy contours are tangent [10]. The center of the energy contour is the operating point, which globally minimizes the processor's energy consumption. We refer to this point as a global minimum energy point (GMEP) since it is treated as a reference point in our proposal. Note that we cannot reduce energy any more than GMEP even if we reduce the processor's frequency. Thus, we do not take care of the MEP curve below the GMEP.

If we assume DVFS uses $V_{\mathrm{BB}} = 0$ only without loss of generality, MEPT is about three times more energy-efficient than DVFS in our experiment as shown in Fig. 1. The processor only consumes $32\,\mathrm{pJ}$ per cycle with MEPT while it consumes $94\,\mathrm{pJ}$ with DVFS, when the frequency is $100\,\mathrm{MHz}$.

However, searching for MEP at runtime is not trivial. The difference between DVFS and MEPT is their complexity of control. DVFS only scales the V_{DD}. In contrast, MEPT needs to adjust both the V_{DD} and V_{BB}. This additional tuning knob makes MEPT much more complicated. Since MEP shifts widely due to the temperature change or process variation, V_{DD} and V_{BB} need to be appropriately controlled. Otherwise, a large amount of energy loss may be involved. On the other hand, accurate MEPT relies on energy measurement, which requires considerable time. Moreover, the overhead of changing V_{DD} and V_{BB} also prevents real-time task scheduling. Therefore, implementing an accurate MEPT algorithm on real-time systems is challenging.

This paper proposes a concept of approximated MEPT,

where the complexity is almost equivalent to DVFS. The key idea is to approximate the MEP curve by concatenated straight lines. The fact that maximum operating frequency on the processor's MEP curve is almost linear to V_{DD} also plays an important role. Thanks to the approximation, the measurement phase of MEPT is skipped, and the system can find near MEP immediately on the request of performance change. The points for deriving the lines are accurately measured at the boot phase or when the chip temperature changes.

The method mentioned above identifies MEP in a real-time manner. However, shifting V_{DD} and V_{BB} still includes overhead, which affects real-time processing. To implement the MEPT algorithm on a real-time operating system (RTOS), we also propose a task scheduling algorithm. The key idea is to introduce an M/M/1 model of queuing theory into soft real-time scheduling, which stochastically adjusts processor frequency to the workload. Since it asymptotically controls average response time to be equal to a specified deadline to provide a soft real-time capability to the system, it does not require any rapid voltage and frequency change. It makes MEPT more practical to integrate into real-time systems. This algorithm can be implemented by applying some minor changes to an existing RTOS kernel. To the best of our knowledge, this is the first approach that integrates a MEPT algorithm into a real-time task scheduling framework.

The rest of the paper is organized as follows. Section II summarizes previous works and introduces the main contributions of this work. Section III describes the theory behind MEPT. Section IV proposes the method of approximated MEPT. Section V introduces the task scheduling algorithm of implementing MEPT to an RTOS. Section VI shows the experimental result on an actual chip. Section VII concludes this paper.

II. Related Work and Our Contribution

As mentioned earlier, several MEPT algorithms are proposed over the years.

A self-tuning MEPT processor is proposed in [9]. It automatically adjusts circuit voltages by hardware so that the processor always operates on the MEP condition. However, it consumes considerable time for power measurement required to identify MEP. Additionally, since the hardware automatically controls voltages with timing overhead, no real-time capability of software is guaranteed.

An approximated MEPT algorithm is proposed in [11]. It approximates the MEP curve by concatenated straight lines to achieve real-time identification of MEP. However, it does not discuss the integration of MEPT into a real-time operating system. Even if MEP is identified in a real-time manner, shifting V_{DD} and V_{BB} still consumes considerable time, which may prevent the tasks from real-time processing. Besides, it does not take care of the GMEP, which may induce a situation that reducing operating frequency unexpectedly increases energy consumption. Moreover, it does not propose any concrete method to deal with the change of temperature.

Our methodology overcomes the two issues mentioned above, the overhead of identifying MEP and changing voltage. The method for addressing the former is similar to [11], but ours tracks GMEP without approximation. It is accurately tracked at once exceptionally and does not induce overhead usually. Additionally, our task scheduling model handles the MEPT operation as a general workload and provides a soft real-time capability of an operating system.

III. Minimum Energy Point Tracking

This section explains the theoretical background of MEPT and a conventional method to identify MEP accurately. The analytical solution of the MEP is formed from circuit delay and energy consumption. The following subsections describe how V_{DD} and V_{BB} affect factors of the MEP.

A. Switching Energy Consumption

Logic circuits are composed of switching MOSFET gates. These can be modeled as charging or discharging capacitors. Supplying voltage V_{DD} to a capacitor C consumes energy $E_C = C{V_{\mathrm{DD}}}^2/2$. Summing up these energies for the entire chip, we obtain dynamic energy consumption per cycle E_{d}.

$$E_{\mathrm{d}} = \sum E_C = k_1 {V_{\mathrm{DD}}}^2. \tag{1}$$

Here, k_1 is a fitting parameter which represents the active gate capacitances of the entire processor.

B. Threshold Voltage and Leakage Current

Threshold voltage of an FET (V_{TH}) can be approximated as:

$$V_{\mathrm{TH}} \simeq V_{\mathrm{TH0}} - k_\gamma V_{\mathrm{BB}}. \tag{2}$$

Here, V_{BB} is the body bias voltage of the FET. V_{TH0} is the threshold voltage at $V_{\mathrm{BB}} = 0$, and k_γ is the constant depending on process technology and temperature.

Sub-threshold leakage current of an FET (I_{ds}) is modeled as:

$$I_{\mathrm{ds}} = I_{\mathrm{ds0}} \exp\left(\frac{V_{\mathrm{gs}} + \eta V_{\mathrm{ds}} - V_{\mathrm{TH}}}{n\phi_T}\right)\left\{1 - \exp\left(-\frac{V_{\mathrm{ds}}}{\phi_T}\right)\right\}. \tag{3}$$

Here, I_{ds0} is the leakage current at threshold voltage. V_{ds} is the drain-source voltage, η represents drain-induced barrier lowering (DIBL) coefficient [12], and n is the ideal coefficient. Furthermore, ϕ_T is the thermal voltage defined as $\phi_T = kT/q$ using boltzmann constant k, temperature T, and elementary charge q.

C. Static Energy Consumption

Static leakage current of a processor (I_{s}) is the summation of sub-threshold leakage, gate leakage, and junction leakage. Static energy consumption per cycle (E_{s}) can be expressed as:

$$E_{\mathrm{s}} = V_{\mathrm{DD}} I_{\mathrm{s}} T_{\mathrm{clk}}. \tag{4}$$

Here, T_{clk} is the clock period. In this paper, we approximate I_{s} to be equal to sub-threshold leakage.

Sub-threshold leakage is the leakage current of an off-state FET, which is equal to the situation $V_{\mathrm{gs}} = 0$ in (3). The

parameters except for V_{TH} can be approximated as constants. Thus, $I_{\mathrm{s}} \propto \exp(-V_{\mathrm{TH}}/N_{\mathrm{s}})$ holds. Besides, the clock is controlled so that its period is equal to the critical path delay D. In this case, (4) can be transformed into

$$E_{\mathrm{s}} = V_{\mathrm{DD}} I_{\mathrm{s}} D = k_2 D V_{\mathrm{DD}} \exp\left(-\frac{V_{\mathrm{TH}}}{N_{\mathrm{s}}}\right) \tag{5}$$

where k_2 is a fitting parameter, and $N_{\mathrm{s}} = n\phi_T$.

D. Delay

Consider discharging a capacitor C of an FET gate by constant current I. When the gate's voltage reaches half of V_{DD}, we assume the switching is done. In this scenario, the switching time τ is expressed as $\tau = CV_{\mathrm{DD}}/(2I)$.

In the super-threshold region ($V_{\mathrm{DD}} \gg V_{\mathrm{TH}}$), I can be modeled as $I \propto (V_{\mathrm{DD}} - V_{\mathrm{TH}})^{\alpha}$ using a constant α. In the other case, substituting V_{DD} to V_{gs} in (3) yields $I \propto \exp\{(V_{\mathrm{DD}} - V_{\mathrm{TH}})/N_{\mathrm{s}}\}$.

Summing up τ for the critical path, the delay of the processor can be expressed as:

$$D = \sum \tau = \begin{cases} \frac{k_3 V_{\mathrm{DD}}}{(V_{\mathrm{DD}} - V_{\mathrm{TH}})^{\alpha}} & (V_{\mathrm{DD}} \gg V_{\mathrm{TH}}) \\ k_4 V_{\mathrm{DD}} \exp\left(-\frac{V_{\mathrm{DD}} - V_{\mathrm{TH}}}{N_{\mathrm{s}}}\right) & (\text{otherwise}) \end{cases}. \tag{6}$$

Here, k_3 and k_4 are fitting parameters.

E. Minimum Energy Point

MEP is a pair of V_{DD} and V_{TH} which minimizes overall energy $E_{\mathrm{d}} + E_{\mathrm{s}}$ at the given delay D_0. The solution of this optimization problem satisfies (7) with some constant λ.

$$\begin{cases} \frac{\partial}{\partial V_{\mathrm{DD}}}(E_{\mathrm{d}} + E_{\mathrm{s}}) + \lambda \frac{\partial D}{\partial V_{\mathrm{DD}}} = 0 \\ \frac{\partial}{\partial V_{\mathrm{TH}}}(E_{\mathrm{d}} + E_{\mathrm{s}}) + \lambda \frac{\partial D}{\partial V_{\mathrm{TH}}} = 0 \\ D - D_0 = 0 \end{cases}. \tag{7}$$

By substituting (1), (5), and (6), (7) can be transformed into (8) in the super-threshold region.

$$\frac{V_{\mathrm{DD}} E_{\mathrm{s}}}{(2E_{\mathrm{d}} + E_{\mathrm{s}}) N_{\mathrm{s}}} = \frac{\alpha V_{\mathrm{DD}}}{\alpha V_{\mathrm{DD}} - (V_{\mathrm{DD}} - V_{\mathrm{TH}})}. \tag{8}$$

Accurate MEPT based on (8) can be realized with integrated on-chip sensors [10]. We assume that accurate MEPT can be achieved at runtime.

IV. Approximated MEPT

This section proposes the methodology of approximating an MEP curve and identifying MEP in real-time.

A. Methods to identify the GMEP

Before approximating the MEP curve, we describe the method to identify the GMEP. It can be found analytically by solving the point where the derivative of the energy contour is zero. In other words, the energy contour at the GMEP is horizontal and vertical at the same time, as shown in Fig. 2. The condition which makes the energy contour to be horizontal is given by (9).

$$\frac{\partial}{\partial V_{\mathrm{TH}}}(E_{\mathrm{d}} + E_{\mathrm{s}}) = 0. \tag{9}$$

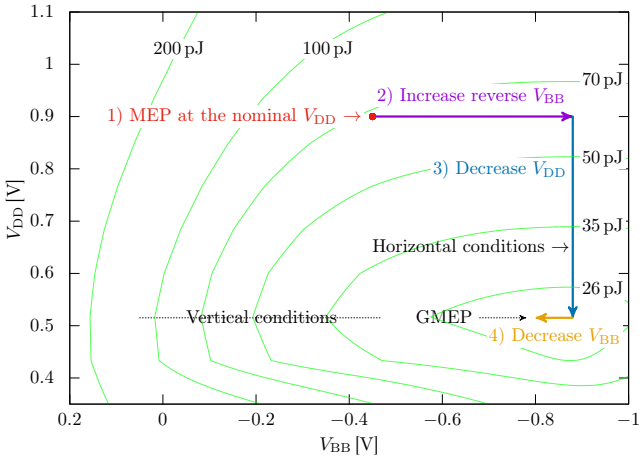

Fig. 2. GMEP search method

In the super threshold region, (9) can be transformed into

$$V_{\mathrm{DD}} - V_{\mathrm{TH}} = \alpha N_{\mathrm{s}}. \tag{10}$$

On the other hand, (9) holds true so that the energy contour is always horizontal in the near- or sub-threshold region.

Based on the discussions above, the following method can be a practical solution to identify the GMEP.

1) Find a V_{BB} value on the MEP curve where the V_{DD} is the nominal voltage of the target technology.
2) Increase V_{BB} value with preserving the V_{DD} value until energy is no longer reduced, i.e, the energy contour is horizontal. This step put the processor into the near-threshold region. If such a V_{BB} does not exist, supply the maximum allowed V_{BB}.
3) Fix V_{BB} and decrease V_{DD} to identify the GMEP. Since the energy contour is always horizontal in this region, scaling V_{DD} to make the contour vertical can find the operating point whose energy consumption is equal to the GMEP, but the frequency is lower.
4) Fix V_{DD} and decrease V_{BB} until energy increases. This makes the processor speed faster as possible while preserving energy consumption.

Figure 2 describes the method. Step 2 and 3 can be accelerated by applying a binary search algorithm.

B. Linear approximation of the MEP curve

We approximate the MEP curve as shown in Fig. 3. The method runs in the following sequence. All MEPs except for GMEP is tracked based on (8).

1) Search for MEP at the nominal V_{DD}. In this case, it is 0.9 V.
2) Search for GMEP.
3) Apply V_{BB} of the point obtained by 1) and V_{DD} of the point obtained by 2), and measure the delay. We refer to this delay as D_1.
4) Search for MEP on the D_1 contour.
5) Calculate the average frequency of 1) and 2). We refer to the delay corresponding to this average frequency as D_2.

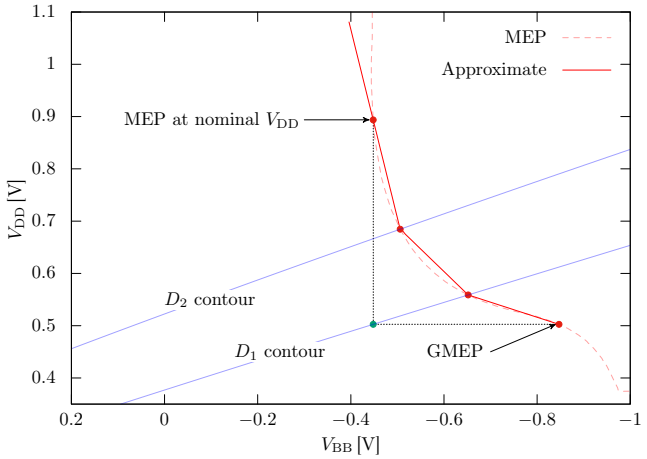

Fig. 3. MEP curve approximation method

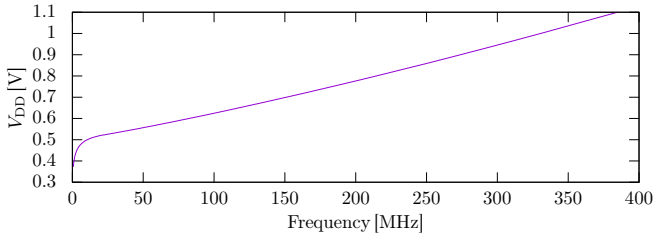

Fig. 4. Relationship between f and V_{DD}

6) Search for MEP on the D_2 contour.
7) Connect the points obtained by 1), 2), 4), 6) with three straight lines.

C. Approximation of V_{DD}

From (6), it is shown that the operating frequency f (inverse of D) is proportional to ${V_{DD}}^{\alpha-1}$ in the super-threshold region. Taylor series of f around $V_{DD} = V_0$ is

$$f = \frac{a_0}{V_0} + \frac{a_1}{1!\,{V_0}^2}(V_{DD} - V_0) + \frac{a_2}{2!\,{V_0}^3}(V_{DD} - V_0)^2 + \cdots \quad (11)$$

where a_n is a polynomial proportional to ${V_{DD} - V_0}^{\alpha-n}$. This implies that f can be approximated with a linear function of V_{DD} if V_0 is large enough.

Figure 4 shows the plot of f and V_{DD}. It clearly shows the linearity of the relationship. We approximate this curve by concatenating three straight lines which connect four points used in the MEP curve approximation.

V. OS Based MEPT

This section proposes an algorithm of task scheduling that treats MEPT as a general workload to provide the soft real-time capability.

Firstly, we assume the system follows an M/M/1 model as shown in Fig. 5. The parameters are described in the following paragraphs. We assume that tasks are activated following the exponential distribution, and the CPU also processes them following the exponential distribution. The number of CPUs is assumed to be one in this paper.

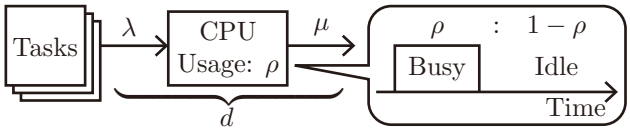

Fig. 5. The M/M/1 model of task scheduling

We define the average arrival time interval of tasks as $1/\lambda$, and the average processing time of tasks as $1/\mu$. λ can be estimated by measuring the average time interval of task activations.

The average utilization ratio of the M/M/1 model is given by $\rho = \lambda/\mu$. This can be estimated by measuring the ratio of times that the processor is busy or idle. Therefore, we can obtain $\mu = \lambda/\rho$.

We propose an algorithm to control the average response time $r = 1/(\mu - \lambda)$ to be equal to the deadline d. Firstly, we assume $\mathrm{d}\mu/\mathrm{d}f = 1$, i.e., scaling the frequency f by a factor a makes average processing time $1/\mu$ to be $1/(a\mu)$. In this scenario, our aim is at holding the following equation true:

$$d = \frac{1}{a\mu - \lambda} \Leftrightarrow a = \frac{1 + \lambda d}{\mu d} = \rho\left(\frac{1}{\lambda d} + 1\right). \quad (12)$$

Periodically scaling f by a adjusts the processor performance for the workload automatically to achieve the best trade-off between response time and energy consumption.

This methodology can be implemented with a minor change applied to an existing RTOS, as shown in Algorithm 1 and 2, and a task to perform MEPT, as shown in Algorithm 3. Although these algorithms do not depend on RTOS, they are written for TOPPERS ASP3 [13], an ITRON based RTOS as a prototype.

Algorithm 1 is an extended dispatcher. In addition to the context switching routine, it measures the time that the processor is active, which is referred to as T_{busy}. When waking up from sleep, it starts a timer at line 5. Moreover, when entering to sleep, it stops the timer and updates T_{busy} at line 9.

Algorithm 2 is an extended function to activate a task. In general, this function provides a task for the ready queue and updates the scheduled task if needed. The extended part from line 6 to 8 measures the arrival time intervals of tasks and update the parameter $1/\lambda$.

Algorithm 3 is a task to perform MEPT. This is a general task of the RTOS and periodically executes the same routine. Firstly, it calculates ρ and a from T_{busy} and $1/\lambda$ obtained from the kernel. After that, it adjusts the performance of the processor and change V_{DD} and V_{BB}. This operation involves a specific overhead; however, since it is one of the general tasks and the overhead can be considered as included in the time for measurements of T_{busy} and $1/\lambda$, this overhead is treated as a workload.

In addition to these routines, the characterization of the MEP curve is necessary. It can be done once at the boot phase. If the effect of temperature is considerable, it should be performed periodically according to the time constant of temperature change. This is also taken into account as a kind of workload.

Algorithm 1: An extension of dispatcher

```
Function dispatcher(void):
    Data: p_runtsk: A global variable pointing to a
          task control block(TCB) of the currently
          running task.
    Data: p_schedtsk: A global variable pointing to a
          TCB of the currently scheduled task.
    Description: Perform context switching. Save the
          context of p_runtsk and restore the
          context of p_schedtsk.
    if p_runtsk != NULL then
        Save context;
    else
        Start T_busy timer;
    end
    p_runtsk ← p_schedtsk;
    if p_runtsk == NULL then
        Stop T_busy timer;
        Update T_busy by adding the timer value;
        Enter sleep;
    else
        Restore context;
    end
end
```

Algorithm 2: The function to make a task runnable

```
Function make_runnable(p_tcb):
    Data: p_tcb: A pointer to a TCB to make
          runnable.
    Description: Add p_tcb to the ready queue and
          update p_schedtsk if needed.
    Add p_tcb to the ready queue;
    if p_tcb has the highest priority then
        p_schedtsk ← p_tcb;
    end
    Stop 1/λ timer;
    Update 1/λ;
    Reset and start 1/λ timer;
end
```

VI. Experiment

This section shows the experimental results obtained with a fabricated test chip running in a thermostat chamber to evaluate our linear approximation method for MEP curves. They are compared with the precise ones to verify that the energy loss is negligible.

A. Target Processor

We performed experiments on an actual embedded processor. It is fabricated with 55 nm deeply depleted channel CMOS, and the core is based on 32-bit RISC-V instruction set architecture. It is equipped with an integrated on-chip leakage monitor, a temperature sensor, and performance counters to track minimum energy points accurately.

Algorithm 3: MEPT Task

```
Task mept(void):
    Description: Periodically update the operating
          condition of the processor.
    while true do
        Compute ρ = T_busy/MEPT_PERIOD;
        Compute a;
        loc_cpu();  // Disable interrupts
        if a > 1 then
            Increase V_DD and V_BB along linearized
             MEP plot;
            Increase f;
        else
            Decrease f;
            Decrease V_DD and V_BB along linearized
             MEP plot;
        end
        unl_cpu();  // Enable interrupts
        /* Delay task execution for
            MEPT_PERIOD.                    */
        dly_tsk(MEPT_PERIOD);
    end
end
```

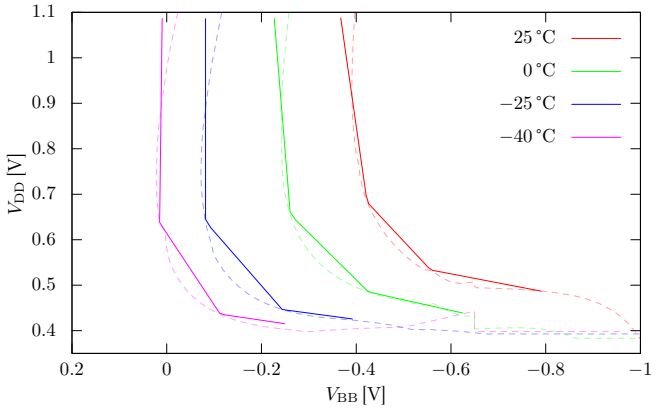

Fig. 6. MEP curve and linear approximations

B. Result

Figure 6 shows MEP curves over the temperature ranging from 25 °C down to −40 °C. When the temperature goes down, the MEP curve shifts towards the lower-left of the graph. This behavior is explained from (5). Leakage energy E_s is exponentially related to the temperature. Therefore, in a lower temperature condition, leakage energy E_s is much smaller than that of an original temperature condition. In contrast, since switching energy E_d is independent of the temperature, E_d is dominant in the low-temperature condition. Thus, the total energy consumption can be reduced toward MEP by lowering both V_{DD} and V_{TH} without degrading the performance of the processor. This is the reason why the MEP curve shifts to the lower-left when the temperature is lowered. Our method appropriately approximates the MEP curve against

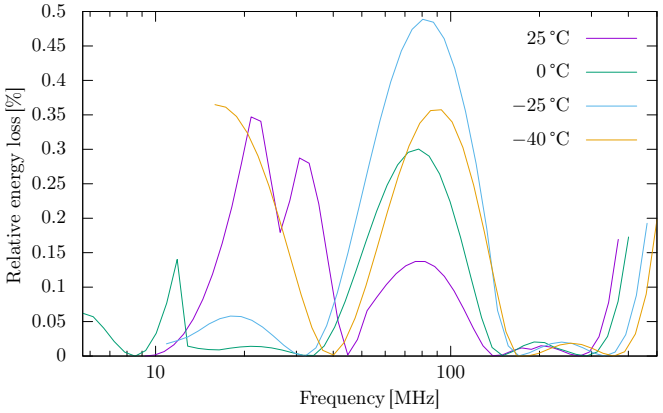

Fig. 7. Energy loss of approximation

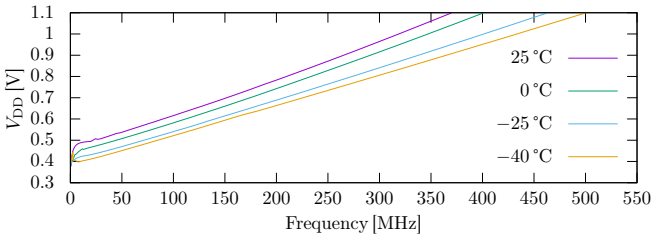

Fig. 8. f and V_{DD} on the real chip

the change of its shape.

Figure 7 shows the relative energy loss introduced by our approximation method. There are four points for each temperature condition where the losses are zero since they are on the accurate MEP curve. The maximum loss is only $0.5\,\%$ at $80\,\mathrm{MHz}$, $-25\,^{\circ}\mathrm{C}$, which shows that the effect of the approximation is negligible.

Figure 8 shows the relationship between f and V_{DD} for each temperature condition. The range of frequency is around $10\,\mathrm{MHz}$ to $400\,\mathrm{MHz}$. In the super-threshold region, all conditions have high linearity, which shows that the linear approximation can be applied for wide range of the temperature.

Figure 9 shows the absolute error of V_{DD} introduced by the approximation. The maximum error is only on the order of mV, and especially around $30\,\mathrm{MHz}$ to $200\,\mathrm{MHz}$, which is mostly used, the error is less than $\pm 5\,\mathrm{mV}$. It is negligible compared to the precision of the DC-DC converter.

VII. Conclusion

Throughout this paper, we proposed an approximated MEPT method. Our approximation of the MEP curve reduces the complexity of MEPT down to that of DVFS with a small amount of energy loss and makes it possible to integrate into RTOS. The method is verified on an actual microprocessor chip fabricated with commercial $55\,\mathrm{nm}$ deeply depleted channel CMOS. Although our approximated MEPT method does not guarantee that the processor's operating frequency is exactly the same as the requested frequency, our scheduling algorithm adjusts processor speed to meet the soft real-time deadline as much as possible. Additionally, the MEPT operation itself is treated as a general task of RTOS. This makes

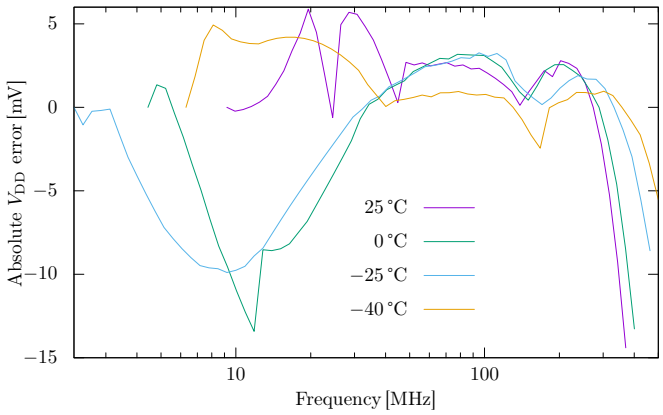

Fig. 9. Absolute V_{DD} error of approximation

the kernel to handle MEPT as a workload and prevent other tasks from violating the deadline.

References

[1] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, ser. FOCS '95. USA: IEEE Computer Society, 1995, p. 374.

[2] T. Pering, T. Burd, and R. Brodersen, "Voltage Scheduling in the IpARM Microprocessor System," in *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, ser. ISLPED '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 96–101.

[3] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems," in *Proceedings of the 41st Annual Design Automation Conference*, ser. DAC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 275–280.

[4] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, 2000.

[5] W. Yuan and K. Nahrstedt, "Integration of Dynamic Voltage Scaling and Soft Real-Time Scheduling for Open Mobile Systems," in *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 105–114.

[6] S. Li and F. Broekaert, "Low-Power Scheduling with DVFS for Common RTOS on Multicore Platforms," *SIGBED Rev.*, vol. 11, no. 1, p. 32–37, Feb. 2014.

[7] K. Nose and T. Sakurai, "Optimization of VDD and VTH for Low-Power and High Speed Applications," in *Proceedings of the 2000 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 469–474.

[8] D. Markovic, V. Stojanovic, B. Nikolic, M. A. Horowitz, and R. W. Brodersen, "Methods for true energy-performance optimization," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1282–1293, 2004.

[9] J. Lee, Y. Zhang, Q. Dong, W. Lim, M. Saligane, Y. Kim, S. Jeong, J. Lim, M. Yasuda, S. Miyoshi, M. Kawaminami, D. Blaauw, and D. Sylvester, "A Self-Tuning IoT Processor Using Leakage-Ratio Measurement for Energy-Optimal Operation," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 87–97, 2020.

[10] J. Shiomi, S. Hokimoto, T. Ishihara, and H. Onodera, "Minimum Energy Point Tracking with All-Digital On-Chip Sensors," *Journal of Low Power Electronics*, vol. 14, pp. 227–235, 06 2018.

[11] K. Kiyawat, Y. Masuda, J. Shiomi, and T. Ishihara, "Real-Time Minimum Energy Point Tracking Using a Predetermined Optimal Voltage Setting Strategy," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2020, pp. 415–421.

[12] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[13] "TOPPERS: Toyohashi OPen Platform for Embedded Real-Time Systems." [Online]. Available: http://www.toppers.jp