Testbed for thermal and performance analysis in MPSoC platforms

Michal Sojka*, Ondřej Benedikt*†, Zdeněk Hanzálek*
Czech Technical University in Prague, Czech Republic
*Czech Institute of Informatics, Robotics and Cybernetics
†Faculty of Electrical Engineering
Email: michal.sojka@cvut.cz

Pavel Zaykov
Honeywell International s.r.o.,
Advanced Technology Europe, Brno, Czech Republic
Email: pavel.zaykov@honeywell.com

***Abstract*—Many modern computing platforms in the safety-critical domains are based on heterogeneous Multiprocessor System-on-Chip (MPSoC). Such computing platforms are expected to guarantee high-performance within a strict thermal envelope. This paper introduces a testbed for thermal and performance analysis. The testbed allows the users to develop advanced scheduling and resource allocation techniques aiming at finding an optimal trade-off between the peak temperature and the achieved performance. This paper presents a new, open-source Thermobench tool for data collection and analysis of user-defined workloads. Furthermore, a methodology for shortening the time needed for the data collection is proposed. Experiments show that a significant amount of time can be saved. Specifically, time reduction from 60 minutes to 15 minutes is achieved with the i.MX8 MPSoC by NXP while running a set of user-defined benchmarks that stress CPU, GPU, and different levels of the memory hierarchy.**

I. Introduction

HIGH-PERFORMANCE computing platforms are composed of heterogeneous Multi-Processor System-on-Chip (MPSoC). The heterogeneity in the MPSoC is the key for delivering high-performance as each hardware (HW) component has its strengths for specific user workloads. Exemplary heterogeneous computing resources in an MPSoC are various types of Central Processing Units (CPUs), Graphical Processing Units (GPUs), and Field-Programmable Gate Arrays (FPGAs).

In recent years, safety-critical domains such as automotive and aerospace have experienced a significant increase in the Software (SW) complexity and functionality that led to the gradual adoption of the heterogeneous MPSoCs. Examples of successfully deployed heterogeneous MPSoCs are infotainment platforms and autonomous driving computers in the recent car generations.

Besides guaranteeing the high-performance, the safety-critical platforms shall also operate under harsh environmental conditions such as dust, vibrations, and extended thermal ranges. In the context of safety and reliability, it is vital to preserve the MPSoC thermal envelope. Thus, it is necessary to keep the peak temperature under a predefined threshold. One of the most popular methods for thermal management is the active cooling that is commonly implemented by forced air. The active cooling significantly complicates the mechanical design. In some cases, it might not be available as electronics are so closely placed that only a limited airflow is available. An alternative to the active cooling is the passive cooling, which is commonly implemented by heat-sinks. The passive cooling is less thermal efficient than the active cooling and requires additional space and adds additional weight. Therefore, the safety-critical domains are interested in complementary techniques to reduce the peak temperatures in MPSoCs chips.

This paper paves the road towards the development of efficient peak-temperature reduction techniques based on scheduling and resource allocation. We introduce a testbed for thermal and performance analysis of various user-defined workloads executed on a selected MPSoC platform, NXP i.MX8QuadMax [1]. We focus our effort on building the testbed using a real hardware platform rather than on working with simulators. As the thermal behavior of the real platform is rich and influenced by a huge amount of factors like computer architecture, physical chip and board layout, and ambient environment, we propose tools and methods that shall be applicable to a wide-range of real-world conditions and hardware platforms.

More specifically, in this paper, we study a reproducible way to measure platform temperatures while running various user-defined workloads. The goal is to eliminate various random temperature-influencing factors as much as possible without using expensive special-purpose equipment such as thermal chambers. A part of our study is an investigation of minimal experiment length that achieves both reproducible results and good precision. To that end, we derive a thermal model and try to use its knowledge to shorten the experiments.

The contributions of this paper are as follows:

1) We introduce Thermobench tool – a new open-source tool that helps with benchmarking, collection of statistical (performance and thermal) data, and their analysis. It also contains multiple ready-to-use user-defined workloads.
2) We propose a method to shorten the length of the measurements needed to predict the steady chip temperature (from 60 to 15 minutes), and we evaluate the precision of this method.
3) We present selected results measured on our testbed with the proposed tooling, showing the relation of thermal and performance properties.

The remainder of the paper is organized as follows. In Section II, we analyze the works most related to ours. Section III introduces the components in our hardware setup and Section IV outlines the functionality of the Thermobench tool. In Section V, we provide details on the data analytics and outline the conducted experiments for the model fitting. Experimental results from the user-defined workloads are presented in Section VI. The paper concludes with Section VII.

II. Related work

Many researchers analyze the thermal properties of computer systems. A common approach is to create a model and examine the thermal behavior using system simulation [2]–[4]. The disadvantage of such an approach is that the simulation precision highly depends on the input parameters. Examples of input parameters are the floor plans of the chip and details about the computer architecture [2]. However, such input parameters are rarely available for the modern MPSoC chips.

An alternative approach, pursued in this paper, is an experiment-based analysis performed on real hardware. The experimental approaches can be divided into two groups based on the physical quantity being measured: i) electrical power/ energy and ii) temperature. These two quantities are related and measuring each one of them has its own advantages and disadvantages. For example, authors of [5]–[9] analyze the power consumption of various workloads aiming at decreasing it. The advantage of measuring the power consumption rather than the temperature is the instantaneous response. Nevertheless, there are also some disadvantages, especially when temperature is the primary quantity of interest:

- in case the power measurement circuitry is not integrated into the system, an invasive modification have to be made to the board [10], which is not always possible. An alternative is to measure the total input power, but this way, it is not possible to distinguish between the power consumers (e.g., MPSoC, displays and communication interfaces).
- power consumption has fluctuations. Thus, the peaks in the power consumption have to be captured with high sampling rate [6]. Execution of such high-frequency sampling on the target system increases the power consumption and execution on an external system makes it harder to correlate measurements with activities on the target system.

In contrast to the availability of power measurements, almost all modern MPSoCs have on-chip temperature sensors that are accessible without the need of any special hardware. Also, thermal measurements allow measuring interesting effects not visible when only the power is measured [11].

The thermal properties obtained as a result of this work can be used to optimize scheduling or resource allocation of the workload. This topic is addressed by other authors [12]–[15], but in most cases their work is not applicable to our platform and safety-critical requirements, either because they consider single-core platforms or because their scheduling model is not compatible with our target domain – avionics.

The Thermobench tool presented in this work already includes several ready-to-be-used benchmarks. Still, it is possible to use our tool with other benchmarks, such as Rodinia benchmark suite [16].

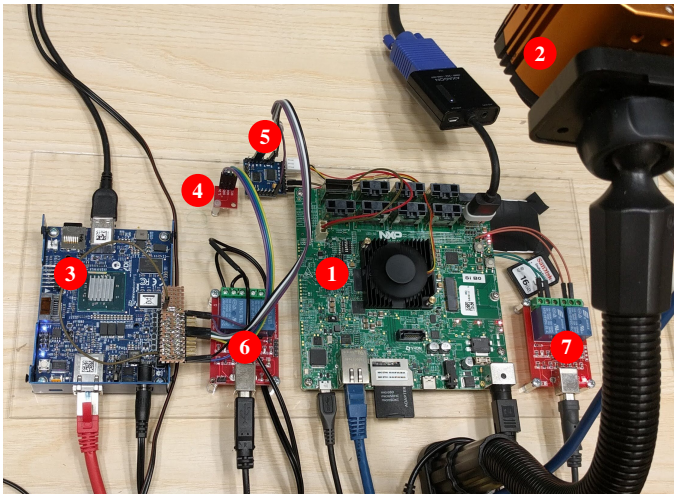

Fig. 1. Testbed for thermal measurements.

III. Testbed hardware setup

This section describes the hardware setup in the testbed. Figure 1 depicts the designed testbed where each label corresponds to one of the following hardware components:

1) i.MX8 Multi-sensory Enablement Kit (MEK) [1];
2) Workswell thermal camera WIC 336 [17];
3) MinnowBoard Turbot (x86 architecture) [18];
4) HTU21D ambient temperature sensor [19];
5) WeMos D1 mini TB6612FNG fan motor controller [20];
6) USB-controlled relay connected to the MinnowBoard;
7) USB-controlled relay connected to the i.MX8 board reset and power buttons.

The target device for our thermal and performance measurements is the i.MX8 MEK board by NXP [1] (hereafter referred to as i.MX8 board). We choose this particular MPSoC because it is the latest generation of the i.MX family, which has successfully demonstrated itself as a prominent computing platform for a wide range of applications, including the on-board infotainment systems in the automotive domain.

The MPSoC in the i.MX8 board is equipped with two CPU clusters. The first one has four ARM Cortex-A53 cores, while the second has two ARM Cortex-A72 cores. The MPSoC also contains two Vivante GC7000 GPUs.

For the convenience of the testbed users and to make the validation easier with various software stacks and OS kernels, we choose to boot up the i.MX8 board over the network rather than from the SD card. The network booting process relies on the features provided by the U-Boot bootloader. The Linux kernel is loaded via TFTP protocol, and the root file system

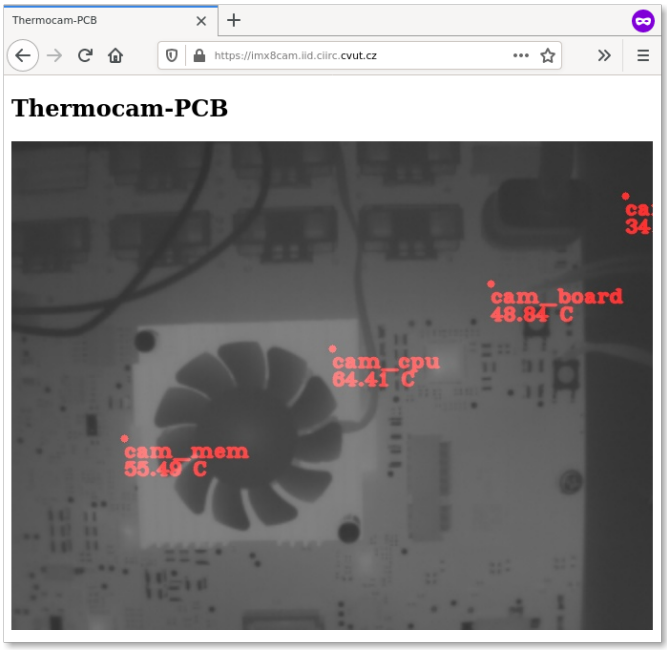

Fig. 2. *Thermocam* application – Web interface.

is mounted via the NFS protocol. The network boot process is automated with the help of the novaboot[1] tool.

We employ an external Pulse Width Modulation (PWM) motor controller to command the Revolutions Per Minute (RPM) for the on-board CPU fan as the i.MX8 board does not allow us to command the fan speed directly. The CPU fan speed is controlled from the software in the i.MX8 board by remotely running (via an SSH session) a command on the Turbot board. Then, the Turbot board controls the WeMos minifan controller to which the CPU fan is attached.

Many of the experiments are executed for prolonged periods of time. During this time, the ambient temperature changes, which may negatively impact the thermal benchmarks' precision. The temperature deviations caused by the fluctuations in the ambient temperature has to be compensated. A way to achieve such a compensation is to record the ambient temperature and later consider it in the analysis.

The ambient temperature sensor is attached to the Turbot development board. The software on the i.MX8 board may read the ambient temperature sensor by remotely running a command on the Turbot board. The command records the ambient temperature every 10 seconds.

The thermal behavior of the i.MX8 board is monitored by an external thermal camera. In our testbed, the thermal camera is attached to the Turbot board and controlled by a custom application referred to as *thermocam*. The *thermocam* application processes the images from the camera and makes them available over a web interface (see Figure 2).

[1]https://github.com/wentasah/novaboot

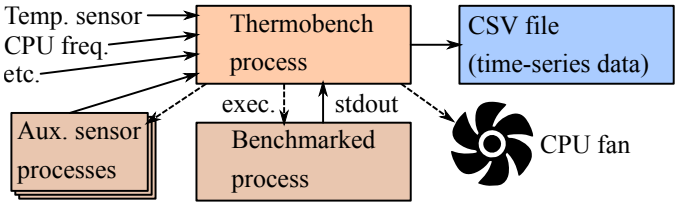

Fig. 3. Data flows in the Thermobench tool

Finally, the MinnowBoard Turbot board serves as a "gateway" for the i.MX8 board in the testbed. The MinnowBoard services provide access to the CPU fan, the ambient temperature sensor, the relays, and the thermal camera.

IV. Thermobench Tool

The Thermobench tool is an open-source software hosted on GitHub[2]. The Thermobench tool is developed to configure the testbed and capture the execution profiles of user-defined workloads, as depicted in Fig. 3. Its three main components are designed to be portable and are described as follows:

- A C++ application that captures the execution profile of user-defined workloads and stores them in a file. Examples of execution profiles might be thermal and performance measurements.
- A Julia[3] package that has data analytics capabilities (refer to Section V for further details), and allows the user to generate various graphs.
- User-defined workloads that expose the thermal and performance profiles of the MPSoC under test. Currently, we provide the following examples of user-defined workloads: CPU micro-benchmarks, CPU memory subsystem benchmarks, and GPU benchmarks. Further details and results are provided in Section VI.

Other notable features of the Thermobench tool are:

- inserting a cool-down time between the execution of two consecutive user-defined workloads,
- controlling the fan speed, and
- the possibility to specify the collected statistics via an external *sensor file*. This feature makes it easier to collect data from all relevant sensors available on a given board.

In a nutshell, the Thermobench tool runs a user-defined workload and periodically collects its execution profile (most importantly measured temperatures), which are then stored to a file for later processing. Examples of recorded statistics are:

- Timestamps,
- Temperatures from Linux thermal-zone sensors,
- CPU clock frequencies,
- CPU load,
- Standard output of the benchmarked program,
- Output (or just selected values) from specific commands (e.g., reading temperatures from the ambient temperature sensor and from the thermal camera).

[2]https://github.com/CTU-IIG/thermobench
[3]https://julialang.org/

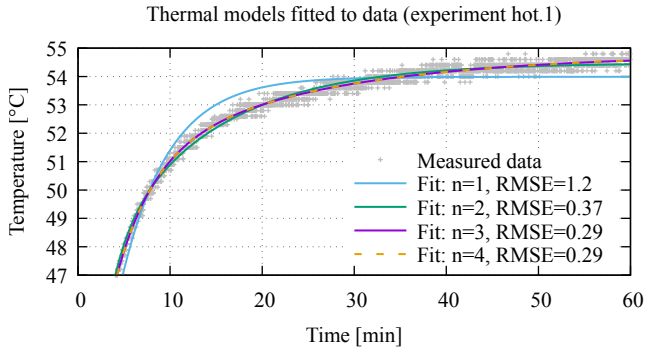

Fig. 4. Measured data and fitted models (data < 47 °C not visible). Exact equations of the fitted models are shown in (3)–(6).

By default, the Thermobench tool records the statistics once per second. With this setting, we have experimentally verified that the Thermobench tool's impact on the system's thermal and performance profile is negligible.

V. Data analytics

The data analytics in the Thermobench tool allows to compare the thermal and performance profiles of various user-defined workloads in the testbed. The thermal profile is based on the *heat flow* that is produced during the execution of user-defined workloads on the MPSoC. The heat flow is the amount of heat energy passed out of the MPSoC. The heat flow is denoted by $\dot{Q}$ and measured in Watts. Unfortunately, the heat flow cannot be directly measured. Therefore, we do estimate it by alternative means – namely using the temperature of the chip. In the rest of this Section, we describe how to estimate the heat flow and evaluate the estimation's precision.

The heat flow $\dot{Q}$ produced by a chip is proportional to the *relative temperature* ΔT_{ss} [21]:

$$\dot{Q} \propto \Delta T_{\text{ss}} = T_{\text{ss}} - T_{\text{amb}}, \tag{1}$$

where T_{ss} is the steady-state chip temperature, and T_{amb} is the ambient temperature. Therefore, to compare the heat flows of various user-defined workloads in an MPSoC, it is sufficient to compare their ΔT_{ss}. However, waiting for the chip temperature to stabilize may require a prolonged period of time, while the user may expect quick and precise ΔT_{ss} estimates.

A naïve approach to estimate the ΔT_{ss} is to average multiple adjacent temperature samples. Unfortunately, the temperature readings are noisy and may significantly compromise the ΔT_{ss} precision. A more robust approach is proposed in the sections to follow, where we also discuss the influence of the ambient temperature on the ΔT_{ss} estimates. Finally, we conclude the section by proposing a way to shorten the time necessary for the data capturing of the temperature readings.

A. Thermal model fitting

Figure 4 visualizes the temperature measurements collected with the Thermobench tool with sampling period of 1 second for a 60-minutes experiment running arithmetic CPU computations. The temperature measurements are captured with a switched-off CPU fan as required by our target applications. To overcome the noise in the temperature measurements, the T_{ss} is estimated by fitting the thermal model to the measured data and by computing the T_{ss} as T_∞ from the thermal model described by (2). We use least-squares fitting implemented by Levenberg-Marquardt algorithm. In this paper, the applied *thermal model* [21] follows the evolution of the chip temperature as a function of time:

$$T_n(t) = T_\infty + \sum_{i=1}^{n} k_i e^{-\frac{t}{\tau_i}}, \tag{2}$$

where n is the order of the model and τ_i are the *time constants* of the model. The time constants specify "how fast" the temperature reacts to changes of the heat flow. We selected such a thermal model because it has the same result as a solution of a set of linear differential equations that is typically used in the modeling of thermal systems. By fitting the thermal model (2) to the temperature data measured with the Thermobench tool, we manually select n and discover the constants T_∞, k_i, and τ_i.

In Figure 4, we demonstrate how the thermal models of different orders (n) may fit the data. The first and the second-order models do not fit well – see their root-mean-square errors (RMSE). The third and the fourth-order models fit better. The difference between them is negligible, thus we conclude that the 3rd order model is sufficient. Numerically, the models for different orders are as follows:

$$T_1(t) = 54.0 - 17.9e^{\frac{-t}{5.2}} \tag{3}$$

$$T_2(t) = 54.5 - 13.6e^{\frac{-t}{1.9}} - 8.4e^{\frac{-t}{11.6}} \tag{4}$$

$$T_3(t) = 54.8 - 7.3e^{\frac{-t}{0.9}} - 11.2e^{\frac{-t}{4.1}} - 4.6e^{\frac{-t}{20.1}} \tag{5}$$

$$T_4(t) = 54.8 - 0.3e^{\frac{-t}{0.02}} - 7.3e^{\frac{-t}{0.9}} - 11.2e^{\frac{-t}{4.1}} - 4.6e^{\frac{-t}{20.3}}. \tag{6}$$

We observe in (5) that for $n = 3$, the $T_\infty = 54.795 \pm 0.075$ °C (95% confidence intervals are the output of the fitting algorithm). Depending on the thermal model order, the estimated T_∞ differ by approximately one degree. For the third-order thermal model, the time constants are $\tau_1 = 0.91 \pm 0.08$, $\tau_2 = 4.1 \pm 0.3$ and $\tau_3 = 20.1 \pm 2$ minutes. The longest time constant τ_3 is particularly important, because it determines the experimental time for the temperature to reach a steady-state – the exponential term reaches 95% of its contribution k_i in $3 \cdot \tau_i$. In case of τ_3, the experimental time is ≈ 60 minutes. In Section V-C below, we examine how this time can be reduced.

B. Suppression of ambient temperature changes

The ambient temperature influences the on-chip temperature. The experiments may run for prolonged periods of time, and the ambient temperature may easily vary among the experiments or even during a single experiment. Such variations often result in non-reproducible temperature readings.

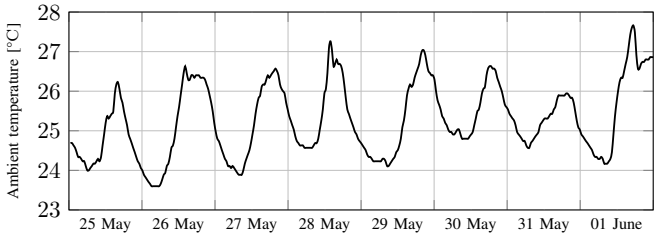

Fig. 5. Evolution of the ambient temperature over a period of one week.

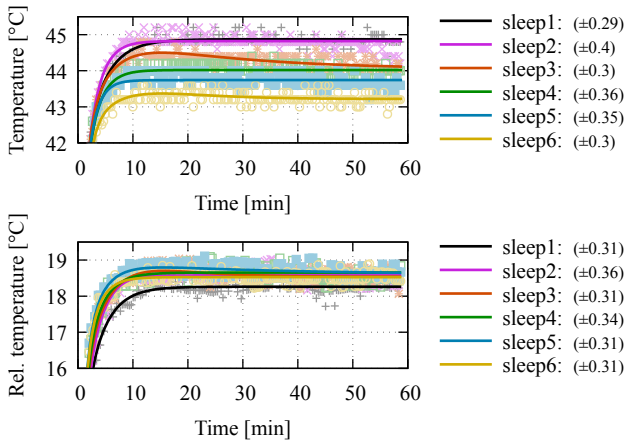

Fig. 6. Comparison of thermal model fitting with (bottom) and without (top) ambient temperature compensation on a series of identical experiments. Fit error (RMSE) is given after ± sign in parentheses.

Figure 5 visualizes the evolution of the ambient temperature as measured by our testbed over a period of one week.

To suppress the effect of ambient temperature changes, it might be best to have a model that counts the impact of the ambient temperature on the on-chip temperature readings and can produce delays of the heat propagation from the ambient environment to the chip. Such a model is referred to as a *transfer function* and can be estimated by system identification methods based on models such as OE, ARX, and ARMAX[4]. It is a well-known fact that for these methods to have good fit, it is necessary to have a high variation on the system inputs. Small changes in the ambient temperature over a long period of time (as in Figure 5), together with relatively high measurement noise, rendered these methods to be ineffective.

Due to the lack of a better model, we do compensate for the ambient temperature changes by simply subtracting the actual ambient temperature $T_{\text{amb}}(t)$ from the other measured temperatures. As a result, the T_∞ in model (2) represents the estimate of ΔT_{ss}. Figure 6 visualizes the results of the fitting thermal models with and without the ambient temperature compensation. The graphs show data from an experiment called "sleep", which was repeated six times. It can be seen

[4]https://www.mathworks.com/help/ident/ug/what-are-polynomial-models.html

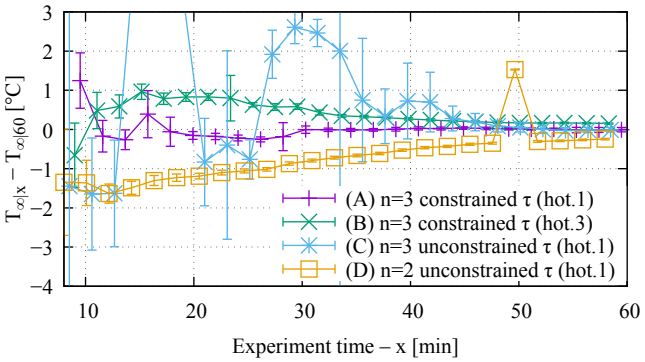

Fig. 7. Relation between the length of the experiment and estimation of T_∞. Vertical axis shows the difference between T_∞ estimated from data of length x ($T_{\infty|x}$) and from full 60 minutes of data ($T_{\infty|60}$). Error bars represent 95% confidence intervals.

TABLE I
PARAMETERS OF THE 3RD ORDER THERMAL MODEL WITH UNCONSTRAINED τ FOR EXPERIMENTS HOT.1 AND HOT.3.

	T_∞ [°C]	k_1 [°C]	τ_1 [min]	k_2 [°C]	τ_2 [min]	k_3 [°C]	τ_3 [min]
hot.1	54.8	−7.3	0.9	−11.2	4.1	−4.6	20.1
hot.3	54.9	−3.4	0.6	−8.3	1.6	−7.8	7.1

that the standard deviation of T_∞ estimates is $\approx 0.6\,°\text{C}$ without the compensation and $\approx 0.1\,°\text{C}$ with it.

The proposed compensation for the ambient temperature changes also helps with the model fitting. The mean value of the fit errors (RMSE) from the examined experiments is 4% lower after the compensation, and the maximum fit error is even 12% lower.

C. Reduction of the experimental time

As we have demonstrated, the user-defined workloads have to run for at least 60 minutes, which turns to be impractical and time-consuming. In this section, we investigate the possibility of fitting the thermal model from a shorter experimental data set and estimating the T_∞ afterward. The difference between the estimates from short and full 60 minute experiments can be seen in Figure 7.

We compare three ways for the thermal model fitting:

- 3rd order model with known time constants, i.e., τ_i are constrained to $\pm 1\%$ of the values estimated from 60 minutes of data.
- 3rd order model with unconstrained time constants, i.e., time constants are fully estimated from the shorter data,
- 2nd order model with unconstrained time constants.

In Figure 7, the 3rd order model and the constrained τ is depicted by curve (A). Curve (A) suggests that the thermal model is able to predict the chip temperature with unsatisfactory precision ($> \pm 0.5$°C) for an experiment executed for less than 20 minutes. Curve (B) suggests that the same method applied to the same user-defined workload but executed 2

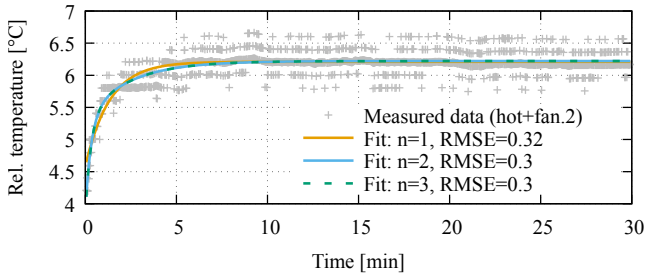

Fig. 8. Measured data and fits with the fan switched on. Note that the ripples in the measured data are caused by the compensation for the ambient temperature. The second order thermal model is $6.2 - 1.5e^{\frac{-t}{0.3}} - 0.9e^{\frac{-t}{2.4}}$.

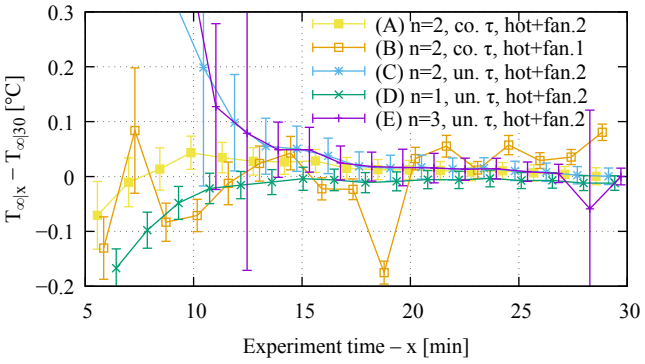

Fig. 9. Relation between the length of the experiment and estimation of T_∞. Vertical axis shows the difference between T_∞ estimated from data of length x ($T_{\infty|x}$) and from full 30 minutes of data ($T_{\infty|30}$). Error bars represent 95% confidence intervals.

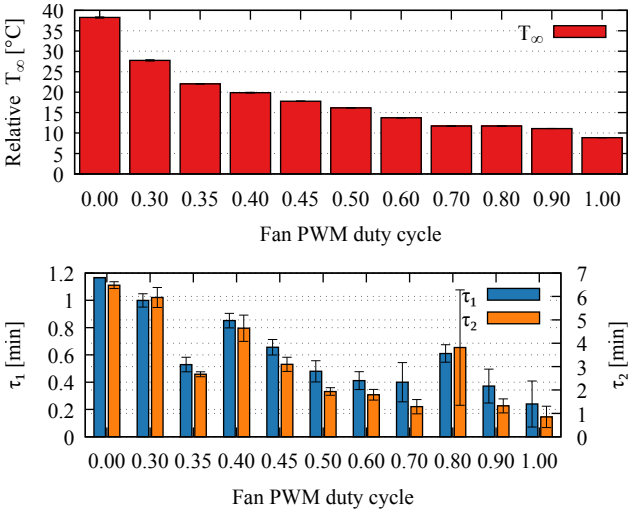

Fig. 10. Relation between different fan speeds and estimated thermal model parameters. Note that 0.3 is the lowest PWM duty cycle that makes the fan move. Error bars represent 95% confidence intervals.

hours later (hot.3) produces even worse results. A satisfactory estimate is achieved for an experiment longer than 30 minutes.

In Figure 7, curve (C) shows the results of fitting the "shortened" data without constraining the time constants close to the correct value. We observe a high number of outliers and convergence to the constrained case for experiments longer than 50 minutes. Also, the time constants have high variation when the fits of the different runs of the same experiments are compared – see Table I. We attribute this variation to the fact that our thermal model presents a lumped-parameter system, where the spatial distribution of heat production and transfer is ignored, whereas the thermal model in a real MPSoC is a distributed-parameter system where spatial dimension matters.

Finally, in Figure 7, curve (D) shows the results of fitting the 2nd order model with a systematic estimation error.

To conclude, a satisfactory estimate T_∞ is achieved for experiments longer than $1.5 \max_i(\tau_i) \approx 30$ minutes. To have satisfactory temperature estimates for shorter experiments, it is necessary to decrease the time constants of the tested system. One of the possible ways is described in Section V-D.

D. Using the CPU fan to decrease time constants

In thermal models, the time constant τ can be computed as $\tau = RC$, where R is the *thermal resistance* between two objects with different temperatures, and C is the *thermal capacity* of the object whose temperature is being measured. In our case, the object is the MPSoC chip. The time constant (τ_i) can be reduced by:

- reducing the thermal resistance R. It can be achieved by regulating the CPU fan speed – higher RPM of the fan motor results in lower thermal resistance between the heat sink and the surrounding environment, or
- reducing the capacity C, e.g., by removing the heat sink from the MPSoC chip.

We decided to pursue the former alternative with the CPU fan as it is easy to implement. Figure 8 visualizes the temperature variations from the same workload as in Figure 4, but with the fan running at full speed. It can be seen that the steady-state temperature is only 6.2 °C above ambient temperature and that the time constants are much lower: 0.3 and 2.4 minutes, respectively. Also note that in this case, the 3rd order model is not necessary as it provides the same result as the 2nd order model.

When we try to estimate the T_∞ from a shorter period of time, we observe (Figure 9) that satisfactory results are obtained for experiments longer than 13 minutes with unconstrained (un.) τ estimations. Constraining (co.) τ constants leads to systematic errors for data from different experiments (curve B). The 1st order model error is slightly below zero even for $x \to 30$ (curve D). The 3rd order model (curve E) gives the same results as 2nd order model, but with few outliers.

It may happen that for less CPU/memory intensive user-defined workloads, the T_∞ is even lower than 6 °C. In that case, the resolution of the temperature sensors might not be sufficient to provide precise temperature estimates. A possible mitigation is to lower the CPU fan speed. The results of the 2nd order thermal model fitting with different CPU fan speeds are presented in Figure 10. The T_∞ temperature clearly decreases

with increasing fan speed and the same trend can be observed for time constants τ_1 and τ_2 except for few outliers (0.35 and 0.8 PWM duty cycle). For some experiments, mostly with high CPU fan speeds (in Figure 10, it is visible from error bar size for 0.8 PWM duty cycle) the fitting algorithm is not able to precisely estimate τ_2, because the slow mode is almost not visible in the measured data, i.e., the 1st order model would be more appropriate there. From non-outlier experiments, we observe that τ_2 drops from 6 to about 0.8 minutes. The outliers in τ estimates are the reason why constraining the time constants to "correct" values during the model fitting, as described in Section V-C, does not always lead to satisfactory results.

The results presented in this section follow from laws of physics and as such, they should apply universally to different hardware platforms. We validated them on a second platform – NVIDIA Jetson Xavier – and the measured trends were similar as those described in this paper.

E. Summary

Analyzing the data from the Thermobench tool measurements is not fully automatic and may require a few manual steps. The manual steps are mainly related to the selection of the thermal model order, the experiment duration, and the appropriate CPU fan speed. Additionally, after fitting the thermal model, one has to check that the model fitting algorithm did not end up in a local minimum, which may result in imprecise T_∞ estimations. After the manual steps are completed, the methods described in this section give a reasonably precise estimate for the T_∞. The estimate T_∞ is proportional to the heat flow $\dot{Q}$ generated by the executed workload.

By fitting the thermal model to the measured data, we obtain more robust estimates for T_∞. Furthermore, we have demonstrated the role of the CPU fan to shorten the experimental time. With the CPU fan switched on, it is sufficient to run the experiments for as short as 15 minutes instead of the initial 60 minutes. Note that the results obtained from the fan-enabled testbed are still applicable to fan-less operation in target applications after scaling the temperatures and time constants up.

VI. Experimental results

The Thermobench repository contains multiple user-defined workloads that might be used to assess the thermal and performance characteristics of the selected MPSoC in the testbed. In what follows, we introduce three types of user-defined workloads. Each one of the workloads has been validated on the i.MX8 board.

A. CPU computation-intensive workloads

The `benchmarks/CPU/instr` folder contains various CPU micro-benchmarks that perform mostly arithmetic operations. With these custom benchmarks, we compare the thermal efficiency and the performance of the CPUs and the CPU clusters. Note that by thermal efficiency, we refer to heat flow, is proportional to steady-state temperature T_∞.

In Figure 11, we list the multiplication operations. The top row compares the single-core performance of A53 and A72 CPUs. The A72 core offers higher performance for non-SIMD and floating-point SIMD instructions. Surprisingly, the integer SIMD instructions are faster on A53. The CPU temperature does not depend significantly on a particular type of operation. On average, the A72 produces $51.1 \pm 5.7\%$ more heat than the A53, while delivering only $92.9 \pm 1.6\%$ of the A53 performance.

The bottom row compares the multi-core performance, where the same benchmark was running on all CPUs within a single cluster. For the A53 cores, the comparison between the single-core and the multi-core execution suggests that the performance increases four times while the temperature rises by only $31.9 \pm 8.0\%$. For the A72, the comparison between the single-core and the multi-core execution suggests that the performance is increased by $2\times$ while the temperature is raised by $27.8 \pm 4.4\%$. The experiments suggest that if all cores in a cluster are used, the A53 cluster is always faster than the A72 cluster, while the A72 cluster dissipates $46.4 \pm 8.6\%$ more heat.

B. CPU memory-intensive workloads

In the Thermobench repository, the CPU memory-intensive workloads are referred to as `membench` benchmark. The `membench` benchmark stresses each level of the memory hierarchy and measures the available memory bandwidth.

The memory bandwidth achieved by various numbers of CPUs can be seen in Figure 12. As expected, the highest bandwidth is for L1 cache memory, i.e., for working set size (WSS) $\leq$ 32 KiB, followed by the L2 cache bandwidth (32 KiB < WSS < 1 MiB) and the lowest bandwidth is, unsurprisingly, available for the DRAM accesses (WSS > 1 MiB). The drops in the performance are aligned with the size of the caches. The further the cache from the processor core is, the lower the performance is. One may also observe, that the DRAM bandwidth available to the 2x A72 cores is slightly lower than the DRAM bandwidth available to the 4x A53 cores.

Figure 13 visualizes the temperature effects of accessing different parts of the memory hierarchy. Clearly, the L1 cache accesses are the most thermal efficient, whereas the DRAM accesses are the least thermal efficient.

C. GPU-intensive workloads

User-defined workloads for the GPU are based on OpenCL-based benchmarks. In OpenCL, the compute work is divided into *work items*. The total number of work items is referred to as the *global size*. The work items are being worked on by *kernel* code running in so-called *work groups*. Each work group processes a certain number (called *local size*) of work items in parallel. The work groups are executed on the GPU either sequentially or in parallel, depending on its (local) size and the size of the GPU.

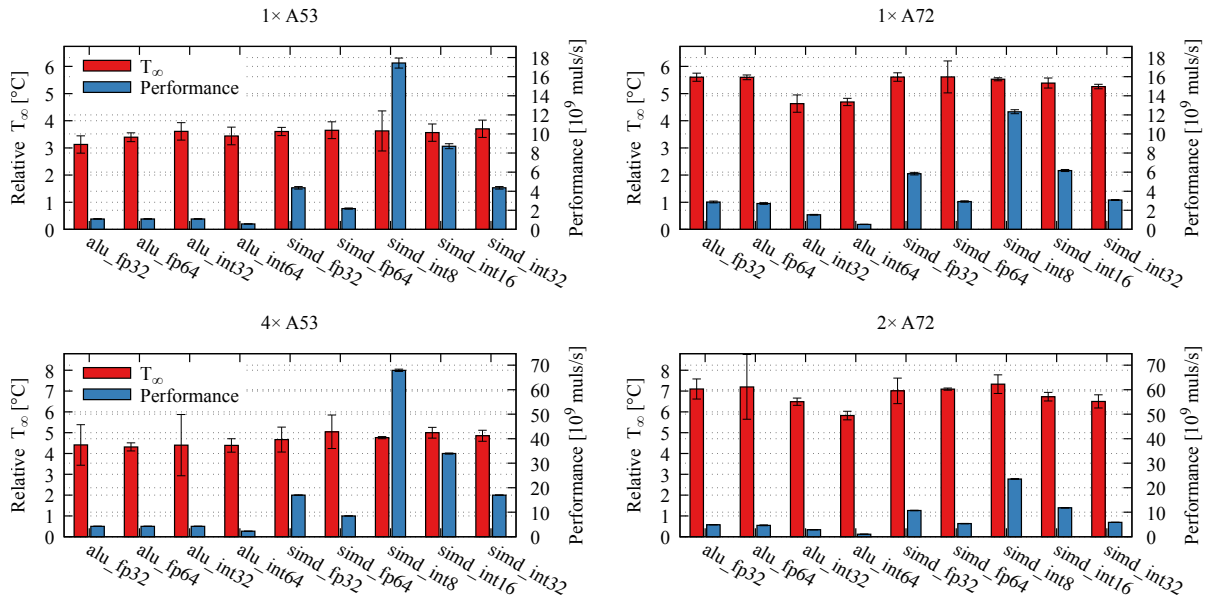

Fig. 11. Comparison of relative steady state temperature T_∞ and performance of multiplication instructions (number of performed multiplications per second) on different CPUs.

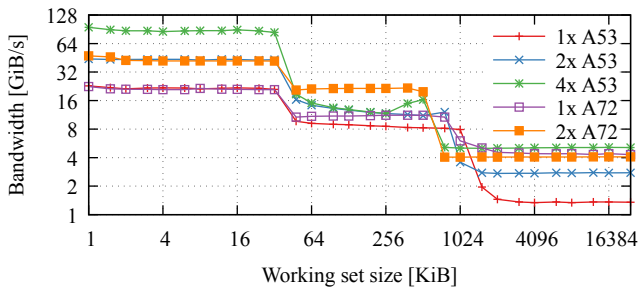

Fig. 12. Memory bandwidth with sequential access.

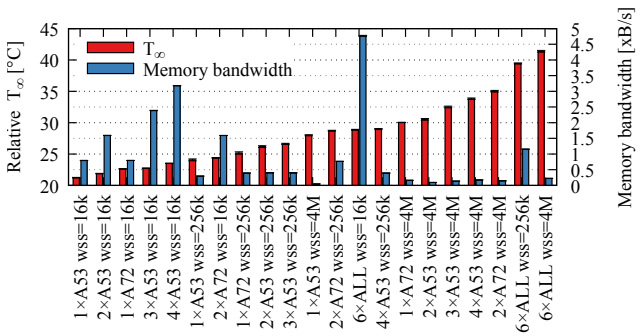

Fig. 13. Memory performance and temperature (measured with the fan switched off).

Figure 14 lists the results from the OpenCL `mandelbrot` (compute-bound) benchmark. The left graph shows the results from the experiments with different global sizes and the same local size. The maximum performance is reached for a global size of 512 or higher (with negligible performance loss for a size of 1024). Smaller global sizes cannot reach the full GPU parallelism, and hence the computation takes a longer time. The steady-state temperature T_∞ decreases with decreasing performance.

In Figure 14, the graph on the right side shows that there is no significant difference in both temperature and performance when the same amount of work items is divided into differently sized work groups.

Figure 15 shows the results of the OpenCL memory-bound benchmark that reads dummy data from the DRAM memory. The left graph suggests that the increased parallelism (global size) leads to decreased performance because the memory bandwidth is the limiting factor. The temperature slightly decreases with performance reduction. The right graph shows that varying the local size makes no difference.

VII. Conclusions

We demonstrated the functionality of our testbed and of the open-source Thermobench tool for processing the data gathered from the testbed. The Thermobench tool allows us to automate the measurement process and estimate the steady state temperature from a fitted thermal model. We evaluated the achieved precision and found that using the fan allows to shorten the experiments form 60 to 15 minutes. The presented experimental results with computation and memory intensive CPU and GPU benchmarks demonstrate the types of experiments that can be obtained from the testbed.

In the future, we intend to leverage the findings of the current work and develop advanced scheduling and resource allocation techniques aiming at finding an optimal trade-offs between the dissipated heat and the achieved performance. The currently proposed testbed will be used for assessing the effectiveness of the proposed techniques.

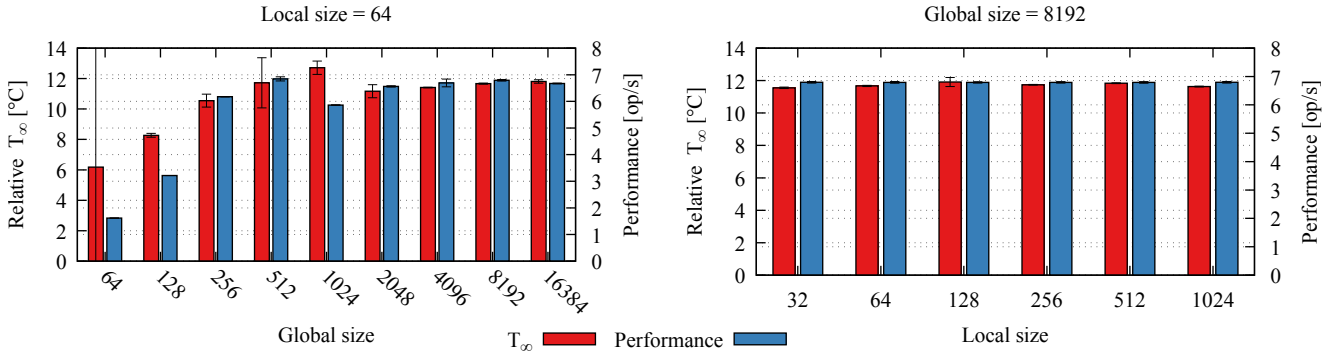

Fig. 14. Performance of a compute-bound GPU benchmark.

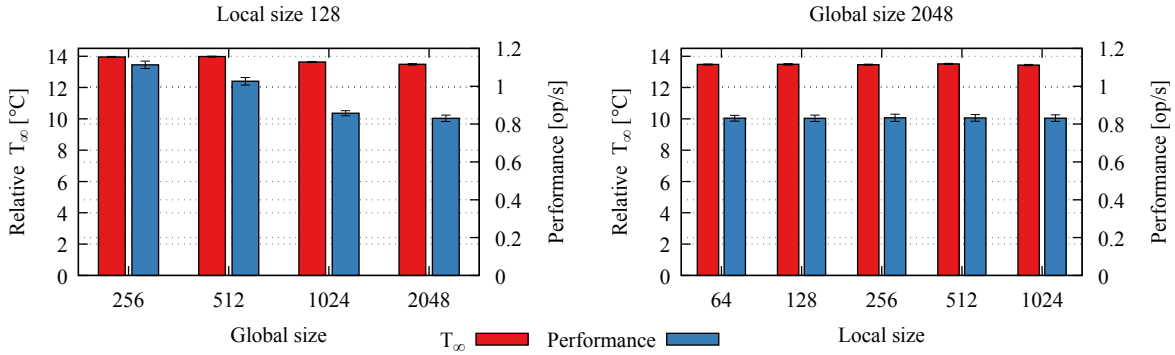

Fig. 15. Performance of a memory-bound GPU benchmark.

ACKNOWLEDGMENTS

This research has received funding from the Clean Sky 2 Joint Undertaking under the European Union's H2020 research and innovation programme under grant agreements No 807081 and No 832011.

REFERENCES

[1] NXP. (2020). "i.MX 8QuadMax/QuadPlus Multisensory Enablement Kit," [Online]. Available: https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/i-mx-8quadmax-multisensory-enablement-kit-mek:MCIMX8QM-CPU (visited on 07/02/2020).

[2] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, 2006. DOI: 10.1109/TVLSI.2006.876103.

[3] A. Kanduri, M. Haghbayan, A. M. Rahmani, M. Shafique, A. Jantsch, and P. Liljeberg, "adBoost: Thermal Aware Performance Boosting Through Dark Silicon Patterning," *IEEE Transactions on Computers (TC)*, vol. 67, no. 8, pp. 1062–1077, 2018. DOI: 10.1109/TC.2018.2805683.

[4] Y. Chandarli, N. Fisher, and D. Masson, "Response Time Analysis for Thermal-Aware Real-Time Systems under Fixed-Priority Scheduling," in *Proc. of the Int'l Symposium on Real-Time Distributed Computing (ISORC)*, 2015, pp. 84–93. DOI: 10.1109/ISORC.2015.34.

[5] N. Bombieri, F. Busato, and F. Fummi, "Power-aware Performance Tuning of GPU Applications Through Microbenchmarking," in *Proc. of the Annual Design Automation Conference (DAC)*, ACM, 2017, 66:1–66:6. DOI: 10.1145/3061639.3062304.

[6] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, e4143, 2017. DOI: 10.1002/cpe.4143.

[7] J. Lucas and B. Juurlink, "MEMPower: Data-Aware GPU Memory Power Model," in *Proc. of the Architecture of Computing Systems (ARCS)*, 2019, pp. 195–207. DOI: 10.1007/978-3-030-18656-2_15.

[8] B. Johnston, B. Lee, L. Angove, and A. Rendell, "Embedded Accelerators for Scientific High-Performance Computing: An Energy Study of OpenCL Gaussian Elimination Workloads," in *Proc. of the Int'l Confer-*

ence on Parallel Processing Workshops (ICPPW), 2017, pp. 59–68. DOI: 10.1109/ICPPW.2017.22.

[9] F. Muslim, A. Demian, L. Ma, L. Lavagno, and A. Qamar, "Energy-efficient FPGA implementation of the k-nearest neighbors algorithm using OpenCL," in *Annals of computer science and information systems*, vol. 9, 2016, pp. 141–145. DOI: 10.15439/2016F327.

[10] C. Schlaak, M. Fakih, and R. Stemmer, "Power and Execution Time Measurement Methodology for SDF Applications on FPGA-based MPSoCs," in *Proc. of the Int'l Workshop on High Performance Energy Efficient Embedded Systems (HIP3ES)*, 2017. arXiv: 1701.03709.

[11] K. Dev, I. Paul, W. Huang, Y. Eckert, W. Burleson, and S. Reda, "Implications of Integrated CPU-GPU Processors on Thermal and Power Management Techniques," 2018. arXiv: 1808.09651.

[12] Y. Lee, K. G. Shin, and H. S. Chwa, "Thermal-Aware Scheduling for Integrated CPUs–GPU Platforms," *ACM Transactions on Embedded Computing Systems (TESC)*, 18th ser., pp. 1–25, 2019. DOI: /10.1145/3358235.

[13] J. Perez Rodriguez and P. Meumeu Yomsi, "Thermal-aware schedulability analysis for fixed-priority non-preemptive real-time systems," in *Proc. of the Real-Time Systems Symposium (RTSS)*, 2019, pp. 154–166. DOI: 10.1109/RTSS46320.2019.00024.

[14] S. Hosseinimotlagh and H. Kim, "Thermal-Aware Servers for Real-Time Tasks on Multi-Core GPU-Integrated Embedded Systems," in *Proc. of the Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Apr. 2019, pp. 254–266. DOI: 10.1109/RTAS.2019.00029.

[15] S. Pagani, H. Khdr, J. Chen, M. Shafique, M. Li, and J. Henkel, "Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon," *IEEE Transactions on Computers (TC)*, vol. 66, no. 1, pp. 147–162, 2017. DOI: 10.1109/TC.2016.2564969.

[16] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, L. Wang, and K. Skadron, "A characterization of the rodinia benchmark suite with comparison to contemporary CMP workloads," in *Proc. of the Int'l Symposium on Workload Characterization (IISWC)*, 2010, pp. 1–11. DOI: 10.1109/IISWC.2010.5650274.

[17] Workswell. (2020). "Workswell infrared camera," [Online]. Available: https://workswell-thermal-camera.com/workswell-infrared-camera-wic/#specifications (visited on 07/02/2020).

[18] Intel. (2020). "Minnowboard turbot," [Online]. Available: https://software.intel.com/content/www/us/en/develop/topics/iot/hardware/minnow-board-turbot.html (visited on 07/02/2020).

[19] TE Connectivity. (2020). "Htu21d digital high accuracy rh/t sensor," [Online]. Available: https://www.te.com/global-en/product-CAT-HSC0004.html (visited on 07/02/2020).

[20] Toshiba. (2020). "TB6612FNG," [Online]. Available: https://toshiba.semicon-storage.com/ap-en/semiconductor/product/motor-driver-ics/brushed-dc-motor-driver-ics/detail.TB6612FNG.html (visited on 07/02/2020).

[21] F. T. Brown, *Engineering System Dynamics: A Unified Graph-Centered Approach, Second Edition*, 2 edition. Boca Raton, FL: CRC Press, 2006, 1059 pp., ISBN: 978-0-8493-9648-9.