applied sciences

MDPI

Article

An Analysis of the Impact of Gating Techniques on the Optimization of the Energy Dissipated in Real-Time Systems

Ernest Antolak * and Andrzej Pułka

Department of Electronics, Electrical Engineering and Microelectronics, Faculty of Automatic Control, Electronics, and Computer Science, Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland; andrzej.pulka@polsl.pl

* Correspondence: ernest.antolak@polsl.pl

Abstract: The paper concerns research on electronics-embedded safety systems. The authors focus on the optimization of the energy consumed by multitasking real-time systems. A new flexible and reconfigurable multi-core architecture based on pipeline processing is proposed. The presented solution uses thread-interleaving mechanisms that allow avoiding hazards and minimizing unpredictability. The proposed architecture is compared with the classical solutions consisting of many processors and based on the scheme using one processor per single task. Energy-efficient task mapping is analyzed and a design methodology, based on minimizing the number of active and utilized resources, is proposed. New techniques for energy optimization are proposed, mainly, clock gating and switching-resources blocking. The authors investigate two main factors of the system: setting the processing frequency, and gating techniques; the latter are used under the assumption that the system meets the requirements of time predictability. The energy consumed by the system is reduced. Theoretical considerations are verified by many experiments of the system's implementation in an FPGA structure. The set of tasks tested consists of programs that implement Mälardalen WCET benchmark algorithms. The tested scenarios are divided into periodic and non-periodic execution schemes. The obtained results show that it is possible to reduce the dynamic energy consumed by real-time applications' meeting their other requirements.

Keywords: real time; multitask; energy-efficient; time-predictable; safety systems; hardware design; FPGA; clock gating; resource-usage optimization

Citation: Antolak, E.; Pułka, A. An Analysis of the Impact of Gating Techniques on the Optimization of the Energy Dissipated in Real-Time Systems. *Appl. Sci.* **2022**, *12*, 1630. https://doi.org/10.3390/app12031630

Academic Editor: Steve Beeby

Received: 7 December 2021
Accepted: 31 January 2022
Published: 4 February 2022

1. Introduction

Real-time embedded systems represent one of the most important segments of the modern electronics market. They are one of the crucial parts of the safety systems operating in medicine, military, control systems, etc. One can observe the constant growth of the expectations of real-time systems over the years. Users require from such systems the highest reliability and long-lasting, failure-free operation. As real-time systems can now be found in mobile and remote applications, a new challenge arises—low power consumption. Thus, new techniques and design methodologies that allow minimizing the energy required by embedded systems have been intensively investigated for the last few decades.

The present paper concerns a low-level approach to time-predictable systems. The authors focus on the hardware design of real-time system architectures. The research analyzes different solutions of PRET architectures [1] with respect to the various cost parameters. The main effort has been toward the minimization of the total energy consumed by the system. All investigations and practical experiments have been carried out on original architecture, developed in the Department of Electronics, Electrical Engineering and Microelectronics (formerly, the Institute of Electronics) [2–4].

The paper is organized in the following way: the second section discusses the related work and briefly describes alternative solutions concerning hardware, as well as software,

elements of time-predictable real-time systems. Section 3 presents different architectures of time-predictable systems, wherein the authors analyze the main building blocks of the solutions that are tested in this research. The fourth section contains theoretical analyses of time-predictable systems and formulates the main quality metrics that are used in the analyses. Section 5 presents selected results of the practical experiments carried out under different multitask systems' configurations. The authors emphasize the analysis of the energy consumed by the solutions that meet timing requirements, i.e., in which the deadlines of all tasks are satisfied. Finally, the last section summarizes the paper, discusses the results, draws conclusions, and formulates areas for further research.

2. Related Work

One of the main requirements of real-time systems is their predictability, i.e., given a set of some hard tasks [5,6], they must meet their deadlines. The problem has been recognized by researchers from all over the world [1,3,5,7–10] for many years now. During the DAC 2007 conference, Edwards and Lee [1] formulated the PRET (precision time machines) philosophy and, since then, the problem of time predictability has been intensively investigated and many interesting solutions in the field have been proposed. The authors of the PRET idea [1] suggest the introduction of pipeline processing with a thread-interleaving mechanism [11] that allows reducing hazards. Wilhelm, in [10], gives a detailed overview of many issues concerning the design and modeling of real-time systems; among others, he pointed out that '*pipeline analysis is a highly complex part of the overall analysis because*, (. . .) *most pipelines do not have compact, efficiently updatable abstract domains*'. This survey outlines that the time predictability of embedded systems requires a detailed analysis of many issues covering hardware, software design, communication between modules and system layers, and task scheduling. Many authors have pointed out how important is analysis of the critical paths (cases) of a system. In [12], the reader can find a very detailed analysis of many aspects of worst cases involving single tasks and various cases of their execution, task-mapping processes, operating systems, and memory operations (especially caches); the authors introduce special traces for this purpose. Wilhelm [10] describe various techniques of WCET analysis, from single-core architectures to multi-core approaches, proposing a very interesting cache analysis. Another paper, [3], describes an interesting time-predictable architecture where the worst-case analysis is performed dynamically during the system run, and, based on a set of factors, the scheduling of tasks is corrected online. This analysis is carried out in the high- and system-level abstraction using the SystemC language. Authors of [7] present MPSoCBench, a simulation toolset that allows the simulation of a system consisting of a set of scalable, virtual multiprocessors on-chip (MPSoCs). This environment has enabled the development and testing of new applications, methodologies, parallel software, and hardware components. Moreover, the toolset is provided with algorithms and mechanisms that allow dynamic voltage and frequency scaling (DVFS) and, thus, controlling the overall system throughput and power consumption. The authors of the T-CREST project [9] have developed a time-predictable, multi-core platform consisting of Patmos processors [13] connected by a network-on-a-chip. They provide a set of worst-case timing (WCET) analysis tools and a dedicated compiler, allowing the parallelization of data processing. The description is available in opensource code (Java). The entire platform has been tested in communication use-cases of examples of practical applications.

An important factor that determines the quality of contemporary ICs is power consumption. In general, the power consumption is affected by circuit currents (leakage, standby, short-circuit, etc.) [14]. There are four basic techniques for minimizing power dissipation in embedded electronic systems: voltage scaling, frequency scaling, power gating, and clock gating. As is well known, many theoretical considerations, as well as practical experiments, have shown that the energy consumed by electronic circuits strongly depends on the operating frequency [2,8,9,15–18].

The experiments also showed that very often, the need to reduce energy consumption entails a reduction in system performance [2]. In the case of real-time applications, when

task deadlines are crucial, making the final design decision is very difficult. Usually, it is a trade-off between energy and system performance [9]. Kim et al. [15] presents experiments on ARM's LITTLE real-time architecture that show a strong correlation between the energy and the system performance; its authors try to accommodate real-time task deadlines, working in mobile applications, into the system throughput. Li has made some interesting theoretical considerations in his paper [8] regarding energy and time-constrained task scheduling on multiprocessor computers. He proposes several algorithms working with discrete clock-frequency and voltage-supply levels. Li also proves that the problem of minimizing schedule length with energy consumption constraints and the problem of minimizing energy consumption with schedule length constraints are NP-hard, even on a uniprocessor computer with only two speed levels. In [16] one can find a set of interesting experiments carried out on different structures of real-time systems. The authors show that the correlation between the real-time system speed and overall system power dissipates and some factors thereof, such as thread-level parallelism, communication models, and the microarchitectures of general-purpose processors. Xie et al. [17] proposes a set of task-scheduling algorithms based on the combination of two different approaches, global DVFS (dynamic voltage and frequency scaling called, GDES) that moves tasks to processor slacks, generating minimum dynamic energy consumption, and non-DVFS (NDES), energy-efficient scheduling based on the concept of deadline slacks analysis. Eventually, there are some scheduling techniques based on AI algorithms and heuristics, such as machine learning [15] or linear programming [18].

Coherency and organization of the memory systems have a strong impact on time predictability [1,5,13]. Generally, it is recommended to avoid caches in real-time systems. As many works have reported, the scheduling of tasks and their mapping to available resources also may balance the system loading and better use its capacities [2,19–22]. The methodology proposed in [19] uses a group-based, energy-efficient dual-priority scheduling (GEDP) that isolates different types of tasks and, thus, allows avoiding disruption and minimizing the context switches between tasks. On the other hand, [20] presents a hierarchical approach to task scheduling (HDA), in which the mapping process takes into account the dependencies between tasks, which, as a result, allows better management of resources and organization of communication in the system. The authors of [21] suggest the usage of a dynamic mechanism that analyzes the resource load during tasks' execution. Specialized resource load and resource power managers allow controlling the dynamic power consumption via a on/off switching mechanism. The industrial approach is presented in [22]; the authors analyze many aspects of the practical applications of real-time systems and point out the limitations of multicore architectures. Many other interesting techniques may improve the processing quality of a multitask system, such as using the benchmark-based approach [7] or the appropriate synchronization of system clusters [23]. The authors of [24] have discussed and shown a very interesting methodology of clock gating that allows the efficient use of system modules and reducing the overall system energy [25]. Present clock gating applied to a group of flip-flops, and the entire procedure is preceded by a detailed analysis of the activity of the blocks. The authors of [26] also show a clock-gating strategy that is successfully applied to dataflow designs based on streaming processing and implement it in FPGA structures. [27] deals with FPGA implementations of embedded systems as well, but the authors primarily attend to the power gating that allows dynamically controlling the power supply of the utilized modules. They propose a special CAD flow methodology, which is tested in a robotic application used in endoscopy. [28] presents an approach based on state-retentive power gating and the cyclic switching on/off of the power supplying the system modules. The authors of [29] propose a selective state-retention power-gating (SSRPG) mechanism based on circuit analysis at the gate level. This approach allows up to a 78% reduction in the used resources.

3. Different Configurations of Real-Time Systems

The research on time-predictable architectures has been carried out in the Faculty of Automatic Control, Electronics and Computer Science for several years. The first papers therefrom were devoted to real-time systems' dynamic scheduling of tasks. In 2009, the proposal of a time-predictable architecture was presented [4]. That solution met all PRET assumptions [1] and contained a flexible thread-interleaving mechanism [11] (Figure 1). The architecture was implemented as a high-level behavioral VHDL model and implemented in Xilinx Virtex 5 FPGA platform. Another approach has been developed in SystemC as an abstract model of time-predictable architecture [3] with the Threads' State Controller (Figure 2) responsible for the generation of threads' identifiers. The thread switching (control) mechanism was based on tracing several factors reflecting the dynamic progress of tasks and approaching deadlines. For a few years, the group has been working on a low-level approach, i.e., on hardware architectures of time-predictable systems. Recently the authors have proposed a few algorithms that allow directing the synthesis towards energy-efficient solutions [2]. In this paper, the different multi-task architectures of safety-critical systems are compared. The analysis starts with the classical approach based on the scheme of a single task per single core and ends with a pipeline processor with interleaved threads.

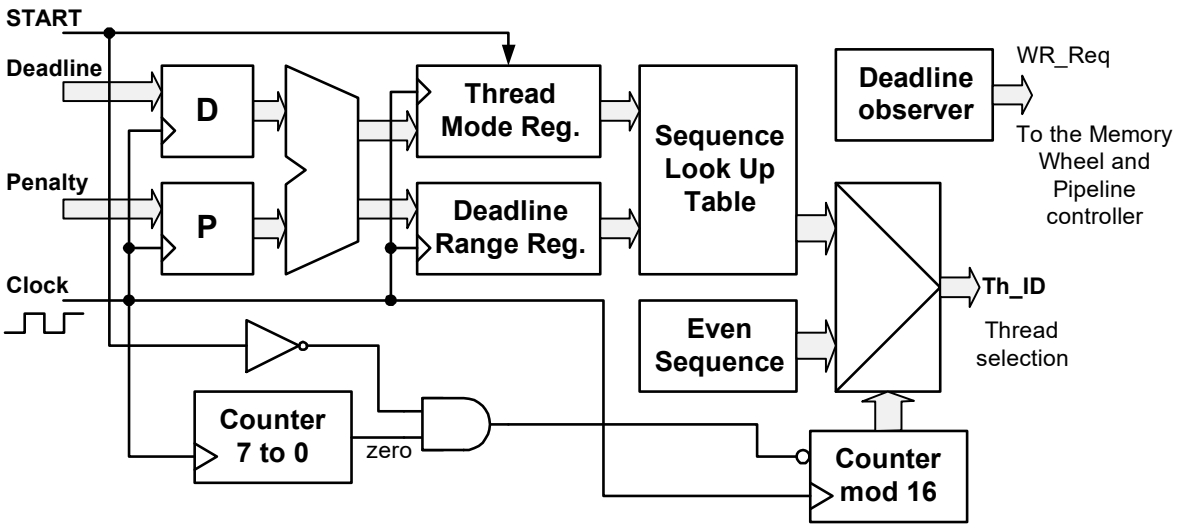

Figure 1. Dynamically scheduled PRET architecture with thread-interleaving and memory developed in IE SUT [4].

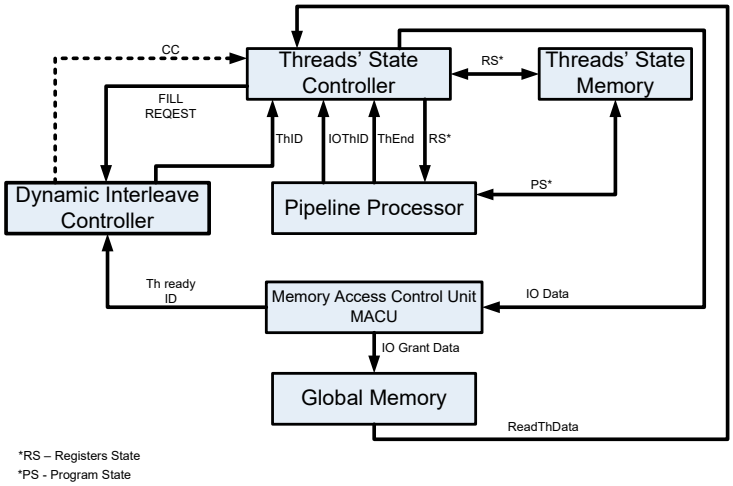

Figure 2. Architecture of a single-core SystemC model of the PRET architecture [3].

3.1. Classical Simple Core Architecture (STP)

Figure 3 presents the idea of a simple processor [30] that needs five clock cycles to execute a single instruction. The architecture consists of five phases: instruction fetch (IF), instruction decoding (ID), execution (EXE), memory operations (MEM), and write-back-to-the-register files (WR) (Figure 3a). The diagram presented on the right side (Figure 3b) shows the occupation of the processor stages (phases) by a given task (Th1) during the subsequent clock cycles. In the paper, this architecture is called STP (single-thread processing). It realizes the following scheme: a single task processed by a single core. In the case of the first simple safety systems, it had been a very reliable solution, ensuring time predictability because the executed task is not disturbed by any additional program or interruption to be handled by a given core. However, in contemporary embedded systems, such a solution is very seldom and very expensive. The main drawback of this solution is that the available resources are not efficiently used during the calculations, i.e., the processor's phases are utilized, at most, at 20% of is capacity.

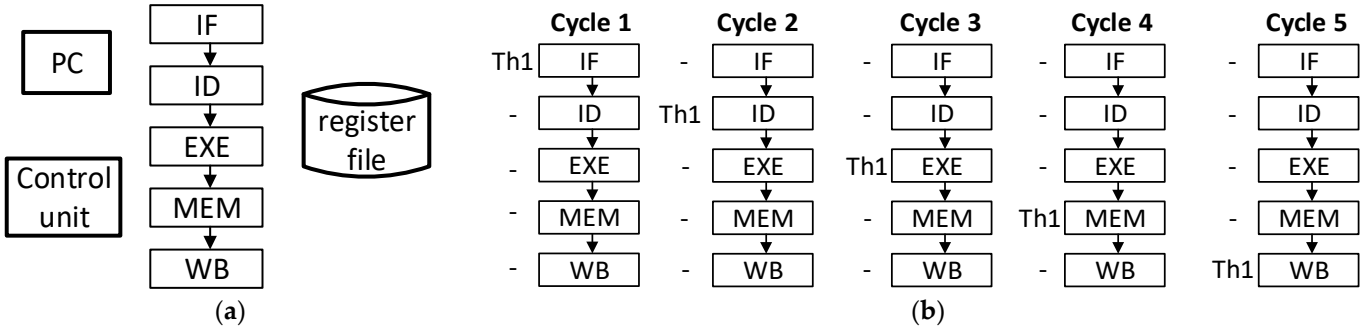

Figure 3. Structure of a classical simple processor (**a**) and subsequent instruction's cycles (**b**).

3.2. Simple Core with Clock-Gating Mechanism (STP CG)

The modification of the previous solution is depicted in Figure 4. In this structure, a clock gating technique [14,24] was used to efficiently use only those resources that are needed in a given phase of the processing. In the presented structure the clock signal is selectively blocked, i.e., only the modules utilized in the currently performed phase of the instruction cycles are clocked, whilst the clock signal is blocked to the rest of those processor resources belonging to inactive phases in a given moment. As is presented below, this modification allows radically reducing the dynamic power dissipated in the core.

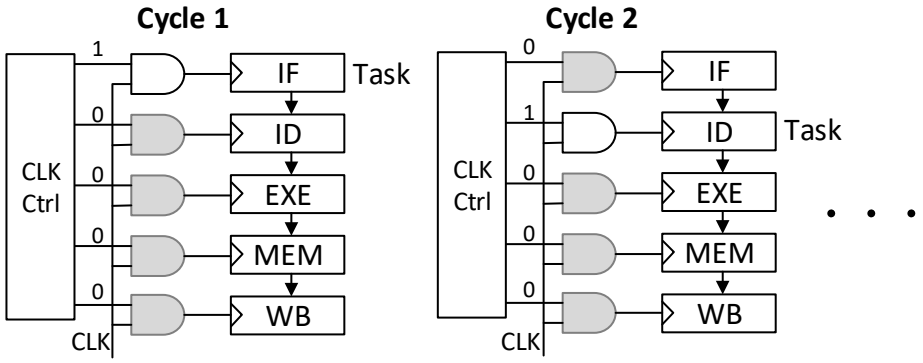

Figure 4. Structure of a classical simple processor with the clock gating of unused phases.

3.3. Multithread Core with Pipeline Processing (MTP)

Recently, an interesting multicore architecture has been published by the team from the Faculty of Automatic Control, Electronics and Computer Science [2]. This structure consists of many cores that can process many threads (tasks). Figure 5 presents the structure of a single core that processes different threads on its pipeline stages, i.e., it is able to switch between replicated resources. Moreover, it is shown that the architecture of a single core

is flexible and the length of the pipeline (the number of its stages) can be adjusted to the properties of the executed programs (tasks). The length of the pipeline can vary from 5 to 12 stages (Figure 6a,b). The five-staged pipeline consists of the same phases as the above-described simple core (STP). The fully extended architecture, depicted in Figure 6b, consists of 12 stages. The original stage instruction fetch had been divided into three sub-stages: select bank and instruction address (SBIA), responsible for determining the next instruction (every thread has its own program counter); proper IF; and select instruction (SI), in which an appropriate output register (corresponding to the appropriate thread identifier) is selected. Similarly, the instruction decoding (ID) and the memory access (MA) phases have also been split into a few sub-stages, in which a given memory (MEM) and general-purpose registers (GPR) are addressed. However, in the case of processing based on thread interleaving, one needs to remember that the frequency of the appearance of a given thread in the pipeline is limited. This problem has been discussed in many papers [2,11]. As a consequence of this fact, to ensure appropriate performance of the system and the meeting of deadlines, the operating frequency must be increased. Taking into account one of the main goals of the paper, which is reducing the total energy consumed by the system and considerations thereof in the next section, the experiments have been limited to the shortest pipeline. Figure 5 also illustrates the problem of data exchange between a given thread and its memory and between different threads. This mechanism is described in [2], but, here, it is omitted because, for the current paper, it is not the main issue concerning energy-efficient design.

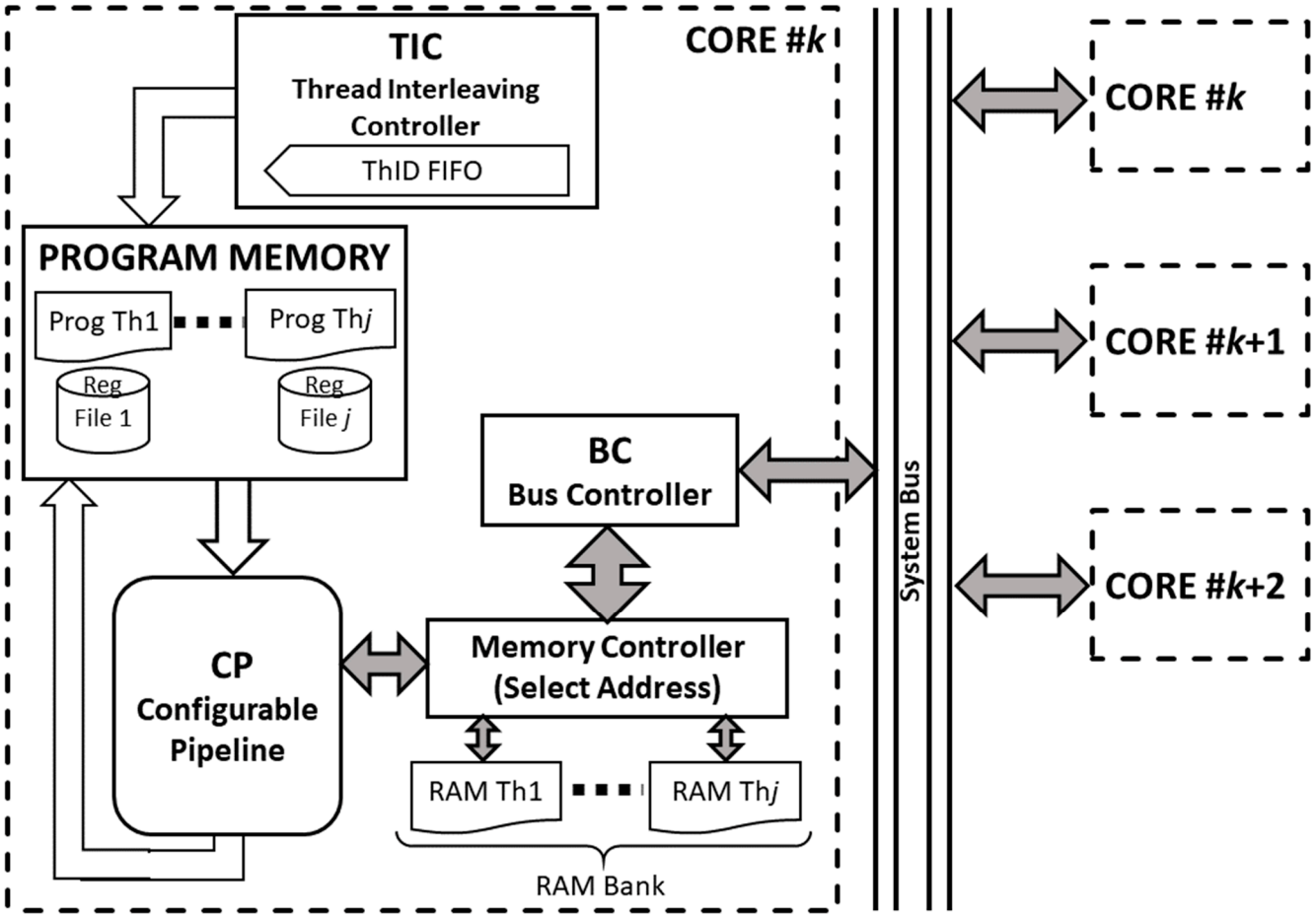

Figure 5. Architecture of a single core with pipeline processing and a thread-interleaving mechanism.

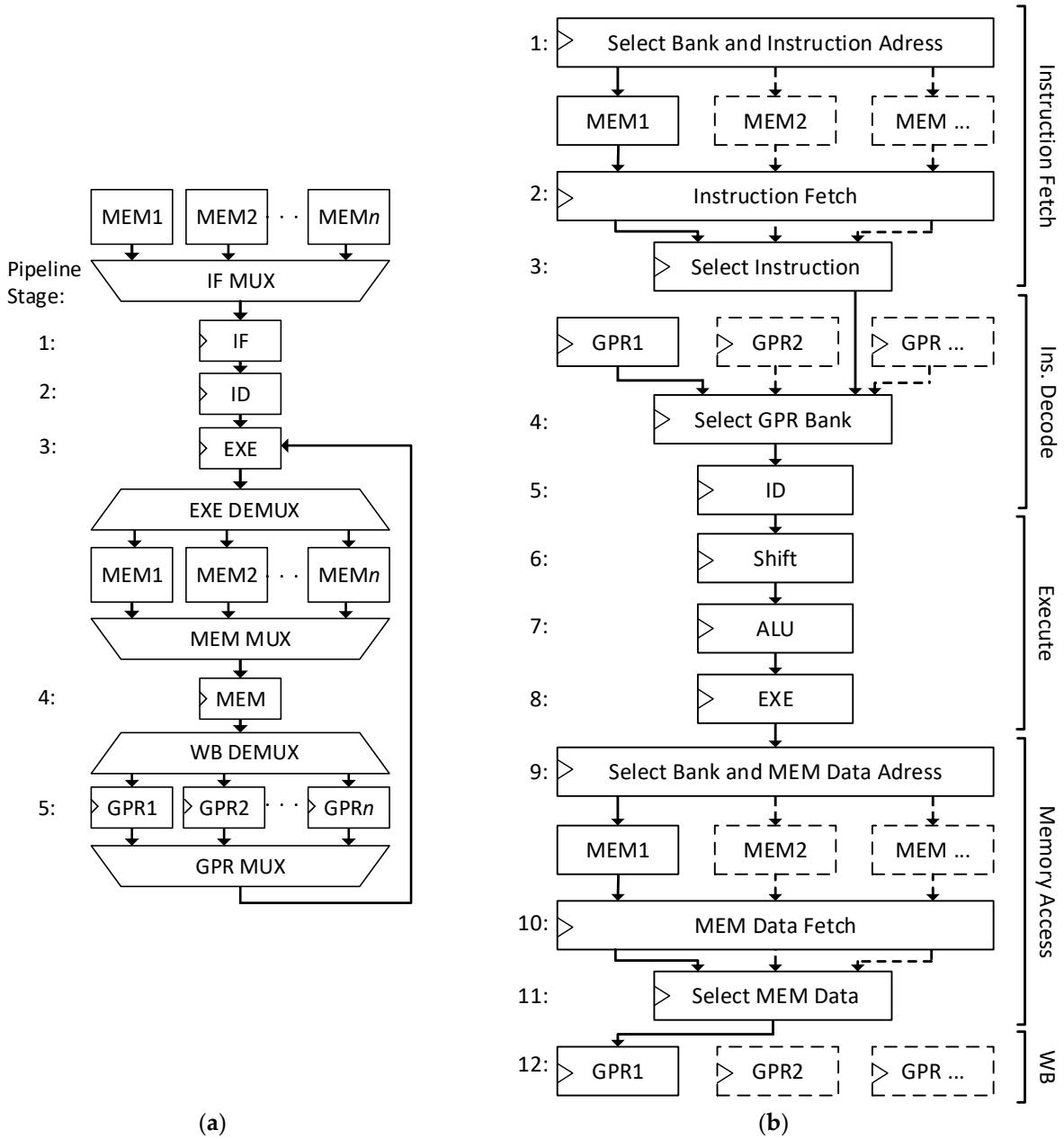

Figure 6. Basic 5-staged pipeline (**a**) and the most advanced 12-staged pipeline (**b**) of the multithread processor.

3.4. Multithread Pipeline Processors with Memory-Gating Mechanisms (MTP RG)

The fourth analyzed architecture (Figure 7) is supported by the special control unit responsible for efficient memory resources. This unit controls the enabling signal delivered to the RAM block of the threads. In the case of the five-staged pipeline, only two stages, IF and MEM, have unlocked access to the memory (Figure 7). The threads occupying the remaining three stages TH2, TH3, and TH5 (shaded rectangles), in a given moment do not need to communicate with the memory, so these memories can be set to an idle state. As further experiments show, the RAM-gating (RG) mechanism reduces the power consumed by the memory blocks during their idle cycles when the blocks are not used.

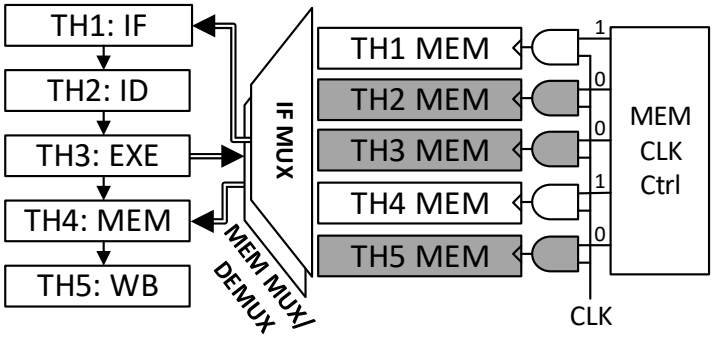

Figure 7. Multi-thread core architecture with pipeline processing and thread interleaving, wherein memory-gating mechanisms are implemented.

3.5. The Main Motivation of the Research and Proposed Methodology

The main objective of the paper is to present an energy-efficient approach to the design of multitask, real-time systems. As is mentioned above, the authors have developed their own original real-time system architecture and proposed a set of effective, energy-efficient scheduling techniques that allow meeting task deadlines [2]. In this paper, other techniques commonly used in the design flow of energy-efficient embedded systems, namely gating techniques, have been tested. For this purpose, the authors have developed a set of different models and proposed various system configurations (scenarios). Then, a series of empirical experiments are done, to find clear criteria when the proposed solution, based on multitasking pipelining processing, is competitive with a single-task processing unit. The impact of the interleaving mechanism on reducing the system performance is also investigated and its limitations are analyzed. Then, all obtained results are compared. Finally, some design recommendations concerning the configuration of the system architecture are formulated.

4. Metrics Used

As is mentioned above, the main requirement of real-time systems is their predictability, i.e., all hard tasks [5] must meet their deadlines. This is the first and the strongest criteria of the design process. To make quantitative analysis possible, the appropriate model of a processed task is necessary. In [2] the following task model is proposed:

$$T_i = \{C_i,\ M_i,\ D_i\} \tag{1}$$

where:

C_i—the number of instructions of the program to execute the i-th task (except M_i);
M_i—the number of memory access instructions of the i-th task; and
D_i—the deadline of the i-th task.

For a given system configuration another parameter, *M_dur* [2], is defined. It reflects the worst-case delay time for the memory operations. *M_dur* stands for the number of clock cycles necessary for memory operations and it is important in case of non-independent tasks, i.e., those threads that exchange data with one and other. From the task model (1) one can express the task frequency, *TF*, factor that describes the processing rate for the i-th task:

$$TF_i = \frac{C_i + M_i \cdot M\_dur}{D_i} \tag{2}$$

Few algorithms presented in [2] show how to map tasks to available processing cores based on *TF* factors and meeting assumed design constraints (throughput, power, size, etc.). Of course, in the case of the simple architecture (Section 3.1), the task frequency multiplied by the number of the clock cycles corresponding to the instruction cycle (here, five) is equivalent to the minimum working frequency of the core processing the task. Meeting this requirement ensures the time predictability of the simple architecture.

In the case of MTP (multitasking processing cores) architectures, other issues need to be considered and worst cases need to be analyzed. For the multitask architectures analyzed in this paper, the following requirement for the deadline of the i-th task is used:

$$D_i \leq \frac{C_i \cdot Min_{indistance}}{F_{sys}} \quad (3)$$

where:

D_i—the deadline of the the i-th task;

F_{sys}—the operating frequency of the system; and

$Min_{indistance}$—the minimal interleaving distance between pipeline stages of the same task.

$Min_{indistance}$, used in Equation (3), corresponds to the minimum number of steps that must separate successive instructions of the same task. For the pipeline processing, $Min_{indistance}$ also equals the minimum number of clock cycles separating successive instructions of the program. This requirement is a condition for avoiding hazards in pipelined processing [11]. As to the operating frequency of the system F_{sys}, its value depends on a few factors [2]: a core loading, i.e., how many tasks are mapped to it, the total sum of tasks' frequencies, and how many cores the overall system architecture consists of.

Since the work presented here deals with energy optimization, it is also necessary to analyze the issues that affect the dissipated dynamic power. A commonly used equation [5] describing the relationship between the dynamic power consumed by the embedded systems and its parameters is as follows:

$$P_{dynamic} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot F_{sys} \quad (4)$$

where:

α—expresses the switching activity of the system (it can be controlled);

C_L—a constant that denotes the switching capacitance; and

V_{DD}—the value of the voltage supply.

Formula (4) shows how the dynamic power consumed by the system can be controlled. In the case of programmable devices (FPGA) where voltage supply and capacitance usually cannot be changed, two factors can be modified: switching activity (α) and the frequency of the system (F_{sys}). Previously published experiments [2] have been carried out with the different scheduling algorithms, proving that both the reduction in frequency and the associated need for more resources help to reduce the energy consumed by the system. Thus, this research has been extended and the possibility of reducing the parameter α is also investigated.

Moreover, because different structures are compared and the cores can process only a single task, another coefficient, AP—the average power necessary to execute a single task—is introduced.

Additional metrics that allow comparing and analyzing various architectures concern the amount of the utilized FPGA resources.

5. Experimental Results

In order to validate the approach to the multithread system design and support the theoretical analysis, the authors carried out a series of practical experiments on hardware. Each of the tested architectures was implemented and synthetized to the FPGA Xilinx Virtex 7 platform [31]. A set of tasks based on the Mälardalen WCET benchmarks [32] was implemented. Every case was precisely analyzed, and their main power and timing parameters were estimated. The experiments were divided into several groups. In the beginning, the elementary structures, based on the scheme *one task per single processor*, for various scenarios and structural solutions were tested. The clock gating of the unused phases was investigated and the dynamic power savings thereof were estimated. Then, the authors investigated properties of the original architecture [2] based on multithread pipeline processing with thread interleaving. The possibility of blocking some resources

and the impact of this treatment on energy savings was analyzed. As in the first group of the experiments, the basic factors describing the system quality were estimated. Finally, the properties of all structures were compared and some recommendations are made as to when a particular solution should be used.

5.1. Simple Architecture Testing

First, the basic architecture (STP) was examined. The averaged results, obtained for various WCET benchmarks [32] during these experiments, are gathered in Table 1. A structure consisting of multiple processors, according to a scheme in which each core processes only one task, was implemented. The first part of Table 1 presents these results and the notation STP $\times$ N refers to a structure consisting of N cores. Then, structures were modified (as described in Section 3.2), i.e., a clock-gating mechanism to block unused stages of the processors was added (the second part of the table—rows denoted by the symbols STP CG $\times$ N). These results showed that the proposed clock-gating mechanism for idle core phases allowed us to achieve significant energy savings. The diagrams depicted in Figure 8 present the averaged results of the power savings achieved for the selected structures by clock gating, which are compared with the appropriate STP structures without clock-gating mechanisms for different frequencies. In turn, Figure 9 shows the percentage power savings, depending on the size of the structure (the number of cores). To make the results more objective, these graphs show the average power per task.

Table 1. Average results of the experiments with the simple architectures testing.

Scenario	Frequency (F_{sys} [MHz])	Total Power (P_{all} [mW])	Cores' Power (P_{CPU} [mW])	FPGA Utilization			
				Slice LUT	Slice Reg	F7 Muxes	F8 Muxes
STP × 5	25	175	54	7053	5206	1170	505
STP × 5	50	22	108	7053	5206	1170	505
STP × 5	100	328	215	7053	5206	1170	505
STP × 5	150	453	327	7053	5239	1170	505
STP × 10	25	231	11	14,091	10,376	2340	1010
STP × 10	50	332	22	14,091	10,376	2340	1010
STP × 10	100	551	438	14,091	10,376	2340	1010
STP × 10	150	786	66	14,091	10,439	2340	1010
STP × 20	25	34	219	28,167	20,716	4680	2020
STP × 20	50	549	437	28,166	20,716	4680	2020
STP × 20	100	986	873	28,167	20,716	4680	2020
STP × 20	150	1441	1315	28,167	20,790	4680	2020
STP CG × 5	25	125	23	7038	5303	1170	505
STP CG × 5	50	149	45	7038	5303	1170	505
STP CG × 5	100	198	9	7038	5303	1170	505
STP CG × 5	150	245	137	7038	5334	1170	505
STP CG × 10	25	147	45	14,061	10,573	2340	1010
STP CG × 10	50	192	88	14,061	10,573	2340	1010
STP CG × 10	100	284	176	14,061	10,573	2340	1010
STP CG × 10	150	379	271	14,061	10,634	2340	1010
STP CG × 20	25	19	88	28,107	21,113	4680	2020
STP CG × 20	50	279	175	28,107	21,113	4680	2020
STP CG × 20	100	456	348	28,107	21,113	4680	2020
STP CG × 20	150	644	536	28,107	21,190	4680	2020

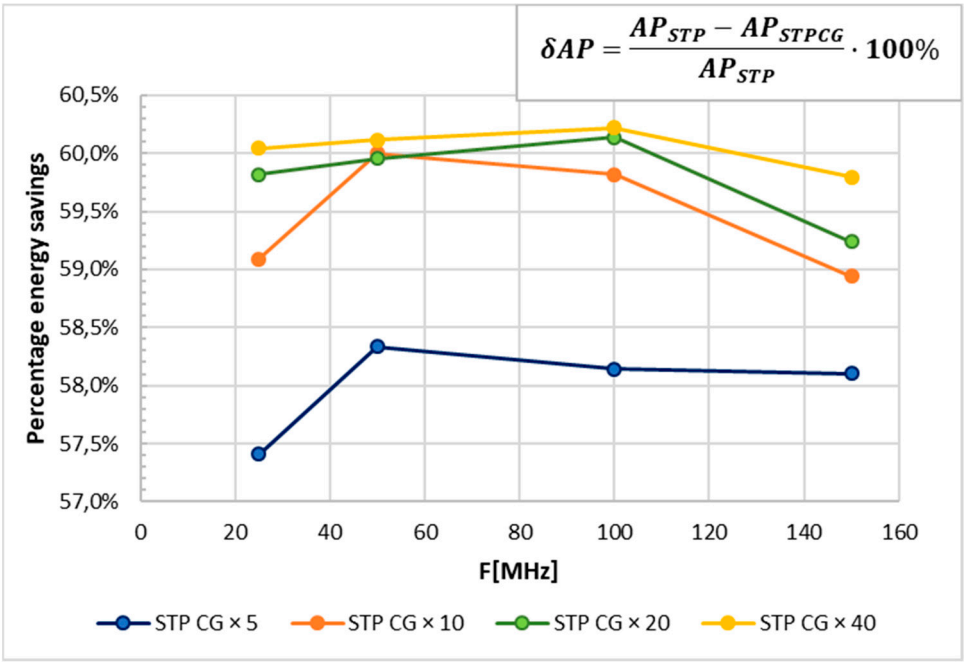

Figure 8. The average power savings per task for the architecture with a clock-gating mechanism compared with the appropriate STP structure for different frequencies.

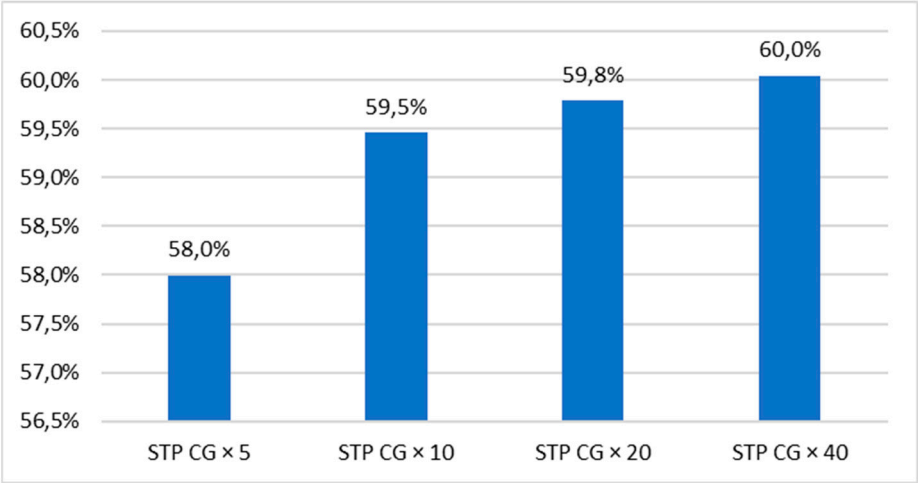

Figure 9. The average power savings per task for the architecture with clock-gating mechanism compared with the appropriate STP structure for the number of tasks (cores).

It is worth noting that these savings were achieved with virtually the same hardware resources with only a slight increase in Slice Registers (less than 2%).

5.2. Experiments with Multitasking Cores (MTP)

The next group of experiments concerned multitasking architectures (MTP) with thread-interleaving mechanism [2]. During the experiments, the same (as previously) set of WCET benchmarks was used. The most representative results were selected and are presented in Table 2. Again, the first experiments were conducted with the original multitasking architecture [2]—the first five rows of the table contain symbols of the form: MTP *K* p $i \times n$. *K* denotes the number of pipeline stages; *i* stands for the number of cores; and *n* is the number of tasks processed by a single core. The second part of Table 2 (rows 6–10) gathers results for the structure built of multitasking cores with the memory operations-

blocking mechanism (Figure 7). These architectures are denoted by symbols of the form: MTP RG K p $i \times n$. The meaning of symbols is the same as for the MTP architectures.

Table 2. Selected results of the experiments with the multitask cores with interleaved pipelines.

Scenario	Frequency (F_{sys} [MHz])	$Min_{indistance}$	Cores' Power (P_{CPU} [mW])	FPGA Utilization			
				Slice LUT	Slice Reg	F7 Muxes	F8 Muxes
MTP 12p 1 × 30	150	30	1028	25,208	18,494	6850	3030
MTP 5p 2 × 15	75	15	646	26,039	16,888	6712	3030
MTP 5p 3 × 10	50	10	427	34,596	17,450	6741	3030
MTP 5p 5 × 6	30	6	262	35,795	18,489	6215	3030
MTP 5p 6 × 5	25	5	217	36,687	19,024	6240	3030
MTP RG 12p 1 × 30	150	30	963	25,008	18,521	6763	3030
MTP RG 5p 2 × 15	75	15	613	26,274	16,922	6654	3030
MTP RG 5p 3 × 10	50	10	391	34,908	17,493	6741	3030
MTP RG 5p 5 × 6	30	6	243	36,202	18,541	6210	3030
MTP RG 5p 6 × 5	25	5	200	36,982	19,026	6240	3030

The analysis of the results shows that, similar to the previous case (STP), the number of resources used in both solutions, i.e., MTP and MTP RG, are comparable. Figure 10 presents the averaged results of the power savings achieved for selected structures with RAM gating compared with appropriate MTP structures for different frequencies, while Figure 9 shows the percentage power savings by structure configuration (the number of tasks per core). As in the STP case, the results presented in all diagrams shows the average power per task. In Section 4, another coefficient, *AP*, was defined as the average power necessary to execute a single task. Figure 11 presents some results concerning the dependence of relative *AP* savings on frequency for MTP RG architectures compared with the basic MTP scenario. These results showed that, in the range of tested frequencies, the differences for individual configurations (scenarios) did not exceed 2.5%. The characteristics had some local extrema, which were related to the matching with the FPGA chip resources during the high-level synthesis. After a certain threshold, all configurations showed a monotonic increase of the consumed power ratio with frequency.

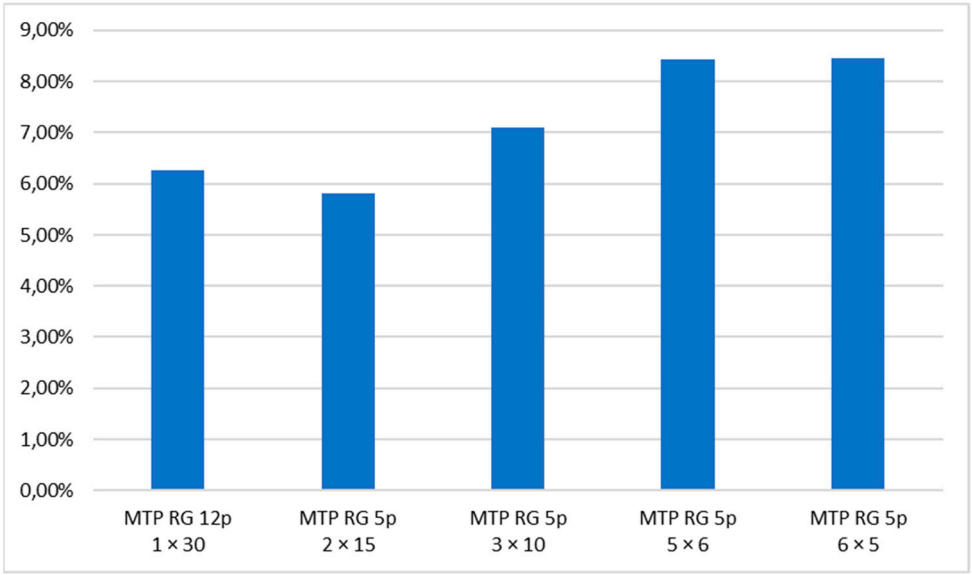

Figure 10. The average power savings per task for the architecture with a RAM-gating mechanism compared with the appropriate MTP structure.

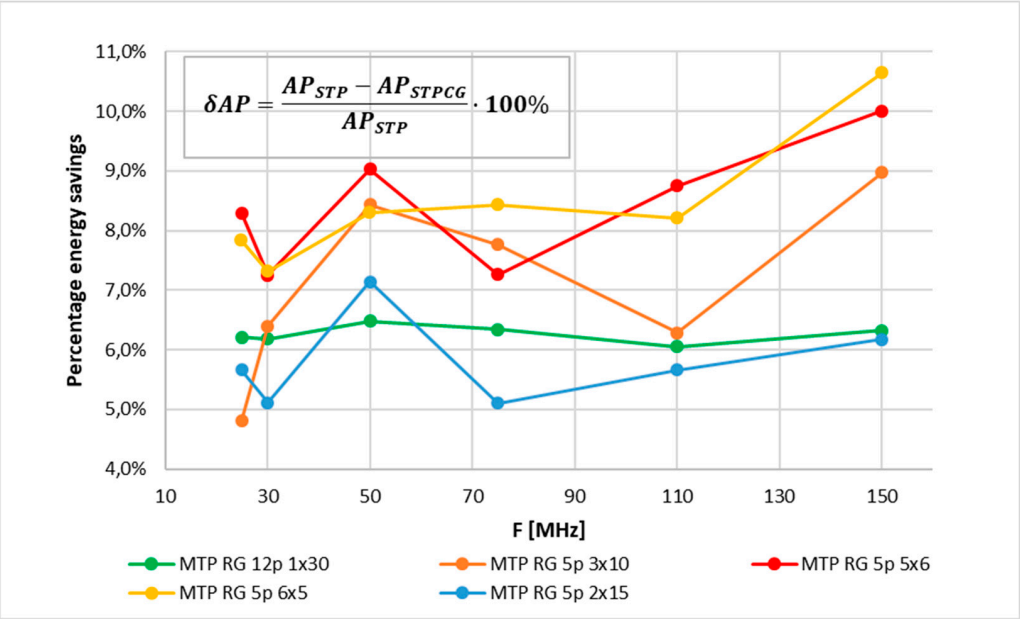

Figure 11. The average power savings per task for the architecture with a RAM gating mechanism compared with the appropriate MTP structure for different frequencies.

5.3. Global Comparisons of All Tested Structures

These group of experiments concerned comparisons and evaluations of different structures that executed the same work, i.e., the same set of tasks. For this purpose, another factor—*SP*, the system productivity—was defined. It is expressed by:

$$SP = \frac{N_T \cdot F_{sys}}{Min_{indistance}} \quad (5)$$

where N_T is the number of processed tasks.

Table 3 presents the results for eight different structures with the same productivity, *SP* = 150. Additionally, the power and resource savings are presented in separate diagrams, Figures 12 and 13, respectively. In the latter case, all structures contained the same amount of switching resources (F8 Muxes), which is why they are not presented in the diagram. In both diagrams, the basic STP × 30 structure was used as a reference architecture. The results depicted in the first diagram (Figure 12) concern the basic system's processing of up to 30 tasks, showing that the best results were obtained from the simple architecture with a clock-gating mechanism. Quite good results were obtained for pipelined (multitasking) solutions when properly configured, i.e., when multiple cores with appropriately chosen (reduced) frequencies were used.

Table 3. Results obtained for different systems with the same productivity: *SP* = 150.

Scenario	Frequency (F_{sys} [MHz])	$Min_{indistance}$	Cores' Power (P_{CPU} [mW])	FPGA Utilization			
				Slice LUT	Slice Reg	F7 Muxes	F8 Muxes
STP × 30	25	5	322	42,243	31,057	7020	3030
STP × 30 CG	25	5	130	42,251	31,660	7020	3030
MTP 5p 3 × 10	50	10	427	34,596	17,450	6741	3030
MTP 5p 5 × 6	30	6	262	35,795	18,489	6215	3030
MTP 5p 6 × 5	25	5	217	26,274	16,922	6654	3030
MTP RG 5p 3 × 10	50	10	391	34,908	17,493	6741	3030
MTP RG 5p 5 × 6	30	6	243	36,202	18,541	6210	3030
MTP RG 5p 6 × 5	25	5	200	36,982	19,026	6240	3030

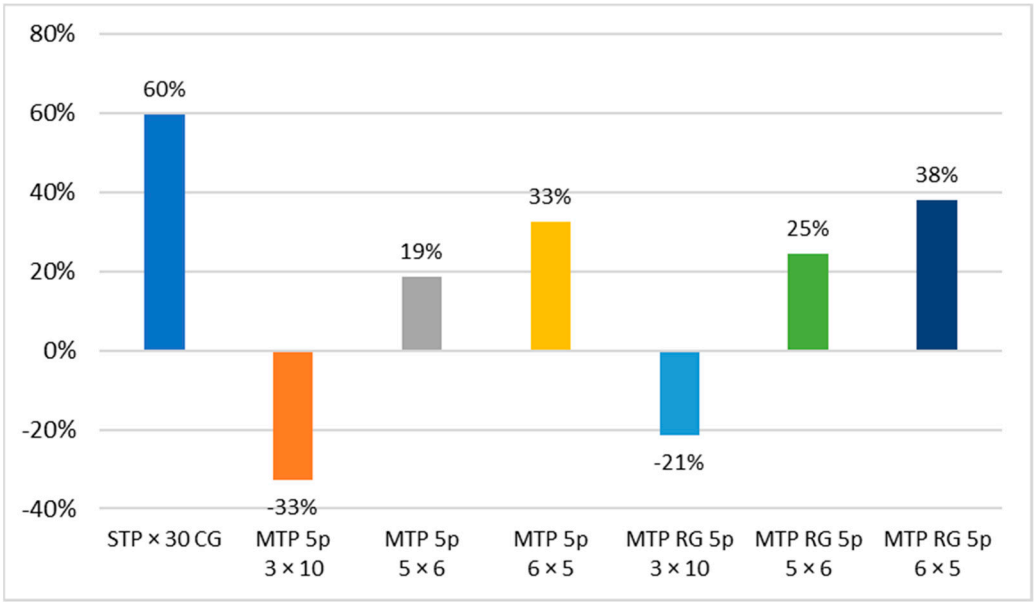

Figure 12. The comparison of power savings per task relative to the architecture consisting of single-task cores (STP) for different structures with $SP = 150$.

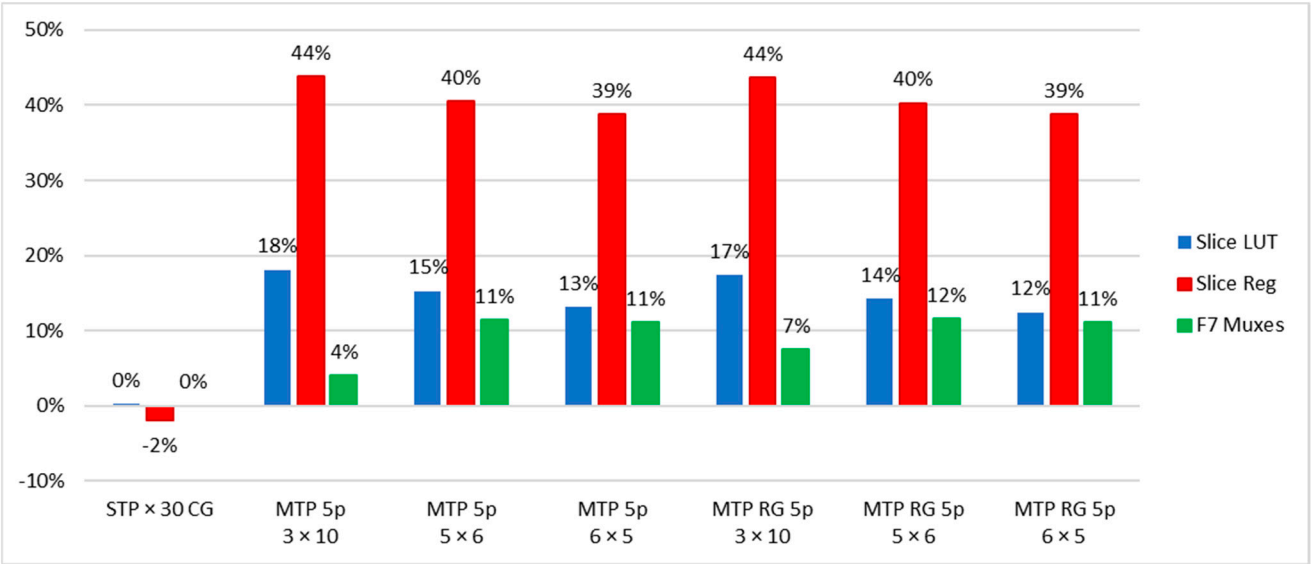

Figure 13. The comparison of resources savings relative to the architecture consisting of single-task cores (STP) for different structures with $SP = 150$.

As to Figure 13, describing the resource savings, one can observe that the best results could be achieved for the low number of multitasking cores processing many tasks. However, in such cases, it is usually necessary to increase the frequency, and this entails increasing the power margin. This was due to the fact that, in the case of STP-type architectures, the number of cores had to be increased, while, for MTP structures, the frequency had to be increased in order to meet all hard deadlines. This is also shown in in Figure 14, which presents the absolute values of the average power consumed by a single task in different scenarios. Definitely, the best solution was this scenario: one task per a single core with clock gating (STP CG). However, because of the natural limitations of such a solution, in many cases, multitasking equivalents should be considered.

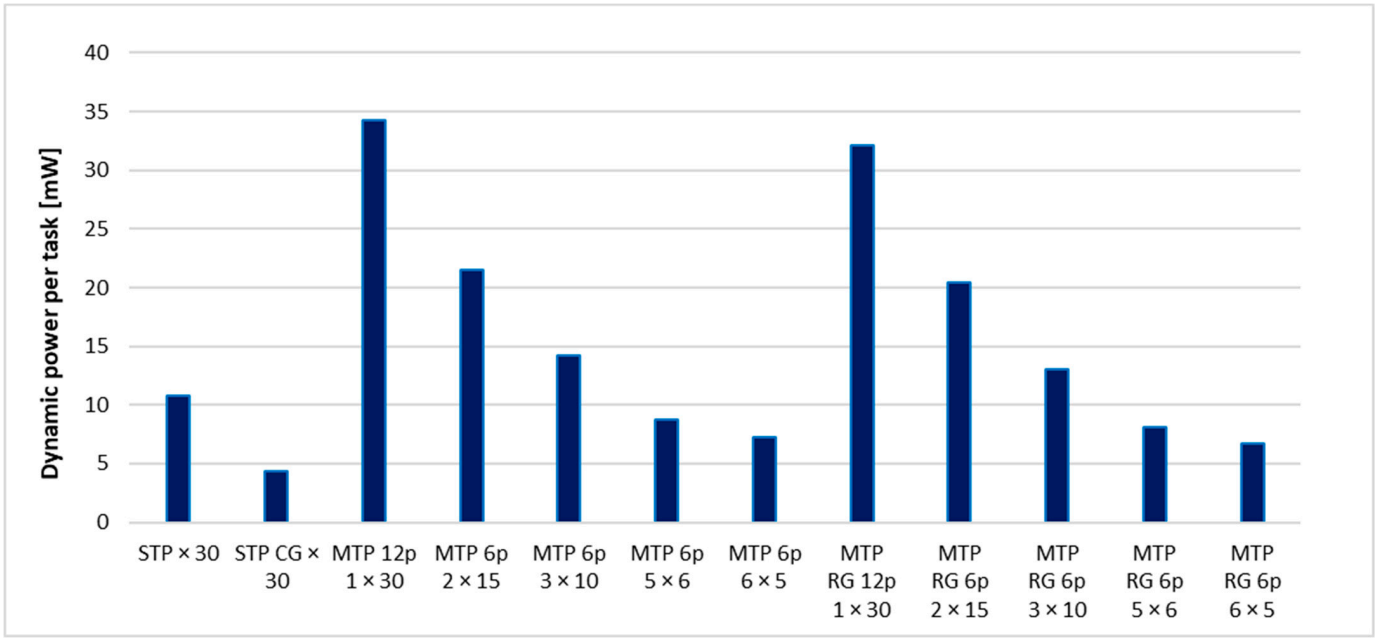

Figure 14. The comparison of the average dynamic power consumed by a single task by different structures with *SP* = 150.

5.4. Quantitative Analysis—The Maximum Processed Tasks

Figure 15 presents the hypothetical (theoretical estimation) limits of each of the tested configurations. These limitations become more evident when we are dealing with a safety system, wherein resources are replicated. In such cases, the same task may be executed concurrently by two or more identical processors. Of course, we must remember that when we decide to switch to the multitasking pipelined architecture (MTP), in order to maintain predictability, the operating frequency must be increased. Table 4 contains comparisons between the main parameters of different MTP scenarios.

Table 4. Results obtained for different systems with the same productivity: *SP* = 150.

Scenario	Frequency (F_{sys} [MHz])	Maximum Number of Tasks	Maximum Number of Cores	Minimum Total Average Power per Core (*AP* [mW])
MTP RG core 30 task	150	360	12	32.10
MTP RG core 15 task	75	345	23	20.43
MTP RG core 10 task	50	260	26	13.03
MTP RG core 6 task	30	250	42	8.10
MTP RG core 5 task	25	245	49	6.67
STP CG	25	215	215	4.33

Figure 16 shows the relationship between the minimum achievable power per task and the maximum system capacity corresponding to the maximum number of tasks that the system can perform. Each point of the graph was further described by a configuration (scenario) for which this minimum could be achieved. The characteristics showed a clear upward trend, i.e., if we want to process more and different tasks, it is necessary to change the architecture to multitasking, and this involves raising the system frequency. As a result, the average dynamic power consumption increased. This should be considered while multitasking embedded systems with strong power constraints are designed.

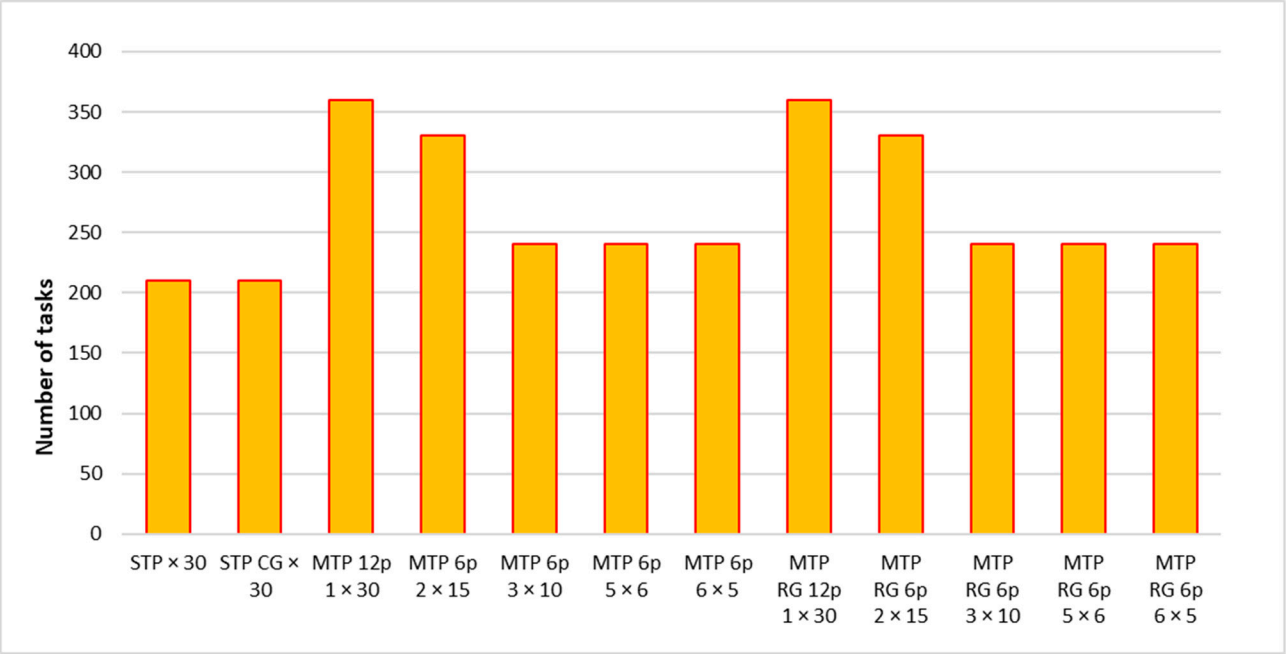

Figure 15. Comparison of the hypothetical structure capacity—estimation of the maximum number of tasks that can be executed in various scenarios (the system configurations).

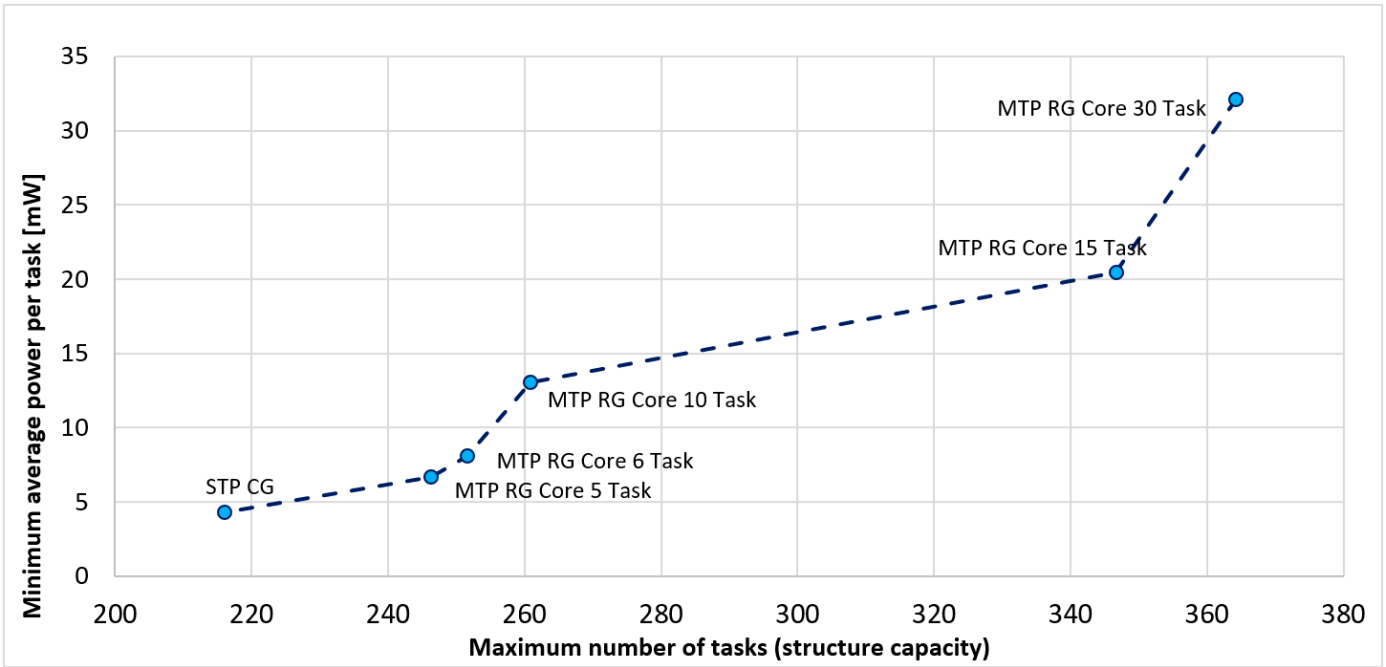

Figure 16. Relationship between the maximum number of processed tasks (system capacity) and the minimum possible value of the power per task for a Xilinx Virtex 7 device. The points are denoted by scenario type (configurations).

5.5. Analyses of Mixed System Configurations

In the next stage, the authors conducted experiments with a system containing mixed structures and configurations. The frequencies were scaled, i.e., different cores could operate at different speeds and cores processed different numbers of tasks. Such a situation is obviously possible from the point of view of a time-predictable system only under the assumption that tasks have been grouped [19], i.e., tasks processed by different cores do not

communicate with one another. Figure 17 presents the relationship between the average power consumed by a single task and the operating frequency for the tested configuration, while Figure 18 shows the results obtained for the mixed scenarios, i.e., different frequencies and different tasks allocated to cores. All cases presented in Figure 18 have the same *SP* factor. These diagrams show that the best results were obtained for the system working with a single frequency, i.e., all cores clocked at the same frequency.

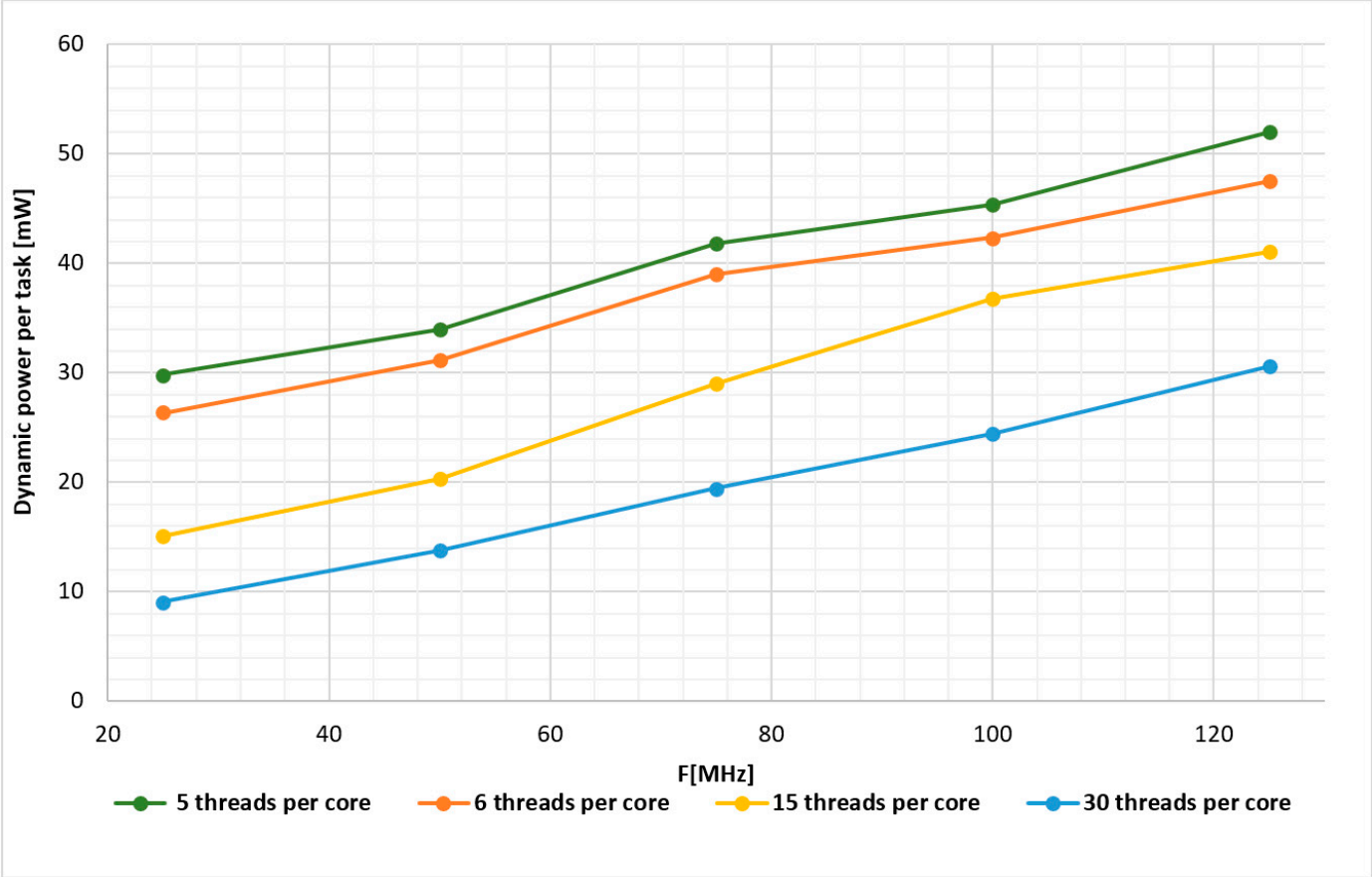

Figure 17. Relationship between the average power per task and the operating frequency for different MTP RG structures (various numbers of tasks processed by a single core).

5.6. Supplementary Discussion of Results

In this paper, several architectures of time-predictable systems were compared. All the structures were implemented in hardware. The possibility of reducing the energy consumed by the system was examined using two techniques: frequency scaling and clock gating. The proposed hardware structures were the original solution, based on thread interleaving. Making a simple comparison with the solutions presented in other works is difficult from this point of view, because either those works were implemented on commercial architectures or they dealt with dedicated applications. Table 5 contains a comparison with the research presented in [26]. That work was also implemented on the same FPGA chip, a Xilinx Virtex 7, and the authors of [26] also used the gating method for energy reduction. The percentage resource consumption and the percentage energy reduction using the gating technique were compared. It turned out that, for both STP- and MTP-type structures, the results reported in the current paper were better. The results (shown in Table 5) indicated a significant increase in hardware requirements when using the gating technique in the solution presented in [26]; up to more than 38% for Slice LUTs and about 17% for the other resources. In the solution presented here, these changes were negligible. Furthermore, in the case of reducing the dynamic power dissipated in the system, the current solution gave at least twice-better results. This is due to the fact that,

despite operating at a low hardware level, in the presented solution the gating procedure was implemented at the instruction level rather than at the signal level.

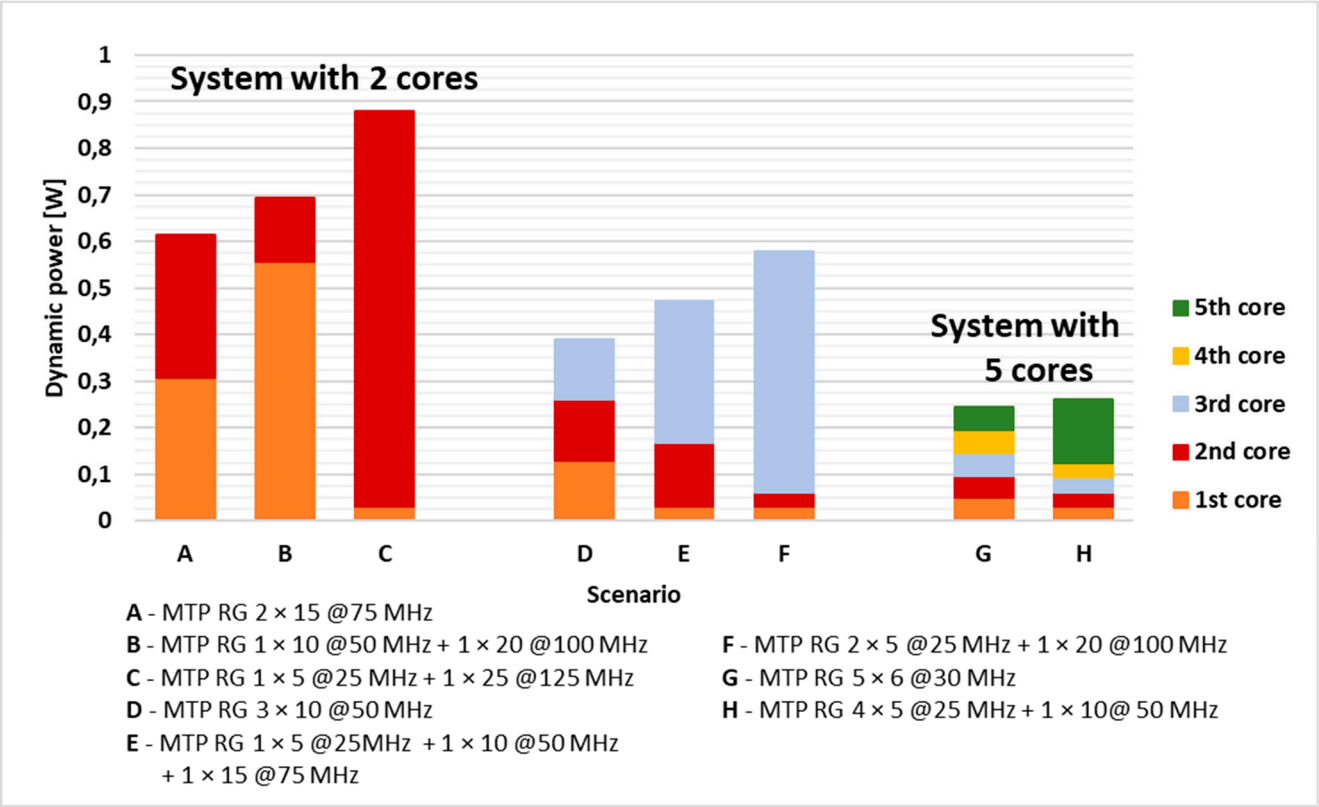

Figure 18. Comparison of the total dynamic power consumed by mixed structures for a different number of cores and various configurations of cores (all cases for *SP* = 150).

Table 5. Comparison of the results obtained for STP CG and MTP RG with [26].

Method	Resources Utilization Related to Non-Gated Solution [%]		Dynamic Power Reduction Related to the Non-Gated Solution [%]
	Slice LUTs	Slice Registers	
STP CG	100	101.9	59.6
MTP RG	100.8	100	7.8
[26]	138.7	116.9	4.0

6. Conclusions and Further Work

In this paper, the authors presented some considerations and practical experiments concerning the recently proposed multitasking architecture for a real-time system [2]. Some switching techniques that allowed efficiently handling the available resources and reducing the power consumed by the system were tested. These proposals were compared with a classical solution in a safety system based on the scheme of one dedicated processor per a single task (STP). The first technique was based on the introduction of a clock-gating mechanism to the classical STP architecture and it gave the best results—the greatest energy savings. Next, a series of experiments were conducted with a multitasking architecture based on pipelined processing with task interleaving (MTP). This structure was modified by adding memory-resource switching (RAM gating). This modification also yielded a radical reduction of power.

However, in the case of the scenario consisting of simple cores, wherein each core executed only a single task (STP), there were limitations. First of all, as the number of concurrently and independently working cores grew, the reliability of the system decreased. This problem became more evident when different tasks needed to cooperate with one

another. Moreover, the capacity of the structure was limited and when the number of different tasks was too great it was impossible to use the STP architecture.

In the authors' opinion, the main contributions of the paper are:

- the development of an original, energy-efficient, multicore architecture and its flexible models;
- the extension of the developed multitasking structure with mechanisms and modules controlling the gating procedures;
- the proposing of a set of metrics for analyzing the basic properties of real-time systems; and
- having conducted a series of experiments, analyses of the obtained results, and the formulation of design guidelines that allowed selecting the optimal system configuration (scenario).

The research presented in the paper concerned mainly independent tasks. Different configurations (scenarios) of the system, which investigated the impact of two mechanisms on dynamic power reduction, were analyzed. However, there were many issues not addressed in this paper, e.g., cooperating tasks that need to exchange data, synchronization of tasks, complex memory operations, buses with energy-efficient protocols, cyclic execution of tasks, and the further gating of idle resources. In the literature, an additional power optimization technique was published, power gating [27–29]. However, the authors decided to neglect this problem in this research. The main reason was that the paper dealt with FPGA structures, in which this technique is not common. Admittedly, in the structure of the chip on which the experiments were conducted, a Virtex 7, Xilinx, introduced the possibility of power gating [33], but the application of this technique for real-time systems is debatable. Problems with state retention and additional delays are not acceptable for time-predictable systems. Yet another idea for future consideration is a mixed system, consisting of STP CG cores and MTP RG processors. The authors also plan to extend their research into GALS (globally asynchronous, locally synchronous) circuits, with cores working at different frequencies. All of these issues provide directions for further research.

Author Contributions: Conceptualization, E.A. and A.P.; Methodology, E.A. and A.P.; Software, E.A. Validation, E.A. and A.P.; Investigation, E.A. and A.P.; Resources, E.A. Writing—Original draft preparation, A.P.; Writing—Review and editing E.A. and A.P.; Visualization, E.A. and A.P.; Supervision, A.P.; Funding acquisition, E.A. All authors have read and agreed to the published version of the manuscript.

Funding: The work reported in the paper is partially supported by the European Social Funds; project no POWR.03.02.00-00-I007/17-00 "CyPhiS—the program of modern PhD studies in the field of Cyber-Physical Systems" and the Ministry of Science and Higher Education funding for statutory activities, BKM-663/RAu-11/2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Simulation data and practical measurements were saved to files with formats compatible with the software and hardware environment used during the described research performed by the author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Edwards, S.A.; Lee, E.A. The Case for the Precision Timed (PRET) Machine. In Proceedings of the 44th ACM/IEEE Design Automation Conference, New Orleans, LA, USA, 27–30 May 2007; pp. 264–265.
2. Antolak, E.; Pułka, A. Energy-Efficient Task Scheduling in Design of Multithread Time Predictable Real-Time Systems. *IEEE Access* **2021**, *9*, 121111–121127. [CrossRef]
3. Golly, Ł.; Milik, A.; Pulka, A. High Level Model of Time Predictable Multitask Control Unit. *IFAC Pap.* **2015**, *48*, 348–353. [CrossRef]
4. Pułka, A.; Milik, A. Multithread RISC Architecture Based on Programmable Interleaved Pipelining. In Proceedings of the IEEE ICECS 2009 Conference, Medina-Hammamet, Tunisia, 13–16 December 2009; pp. 647–650.

5. Buttazzo, G.C. *Hard Real-Time Computing Systems*; Springer: New York, NY, USA, 2011.
6. Ruiz, P.A.; Rivas, M.A.; Harbour, M.G. Non-Blocking Synchronization Between Real-Time and Non-Real-Time Applications. *IEEE Access* **2020**, *8*, 147618–147634. [CrossRef]
7. Duenha, L.; Madalozzo, G.A.; Santiago, T.; Moraes, F.G.; Azevedo, R. MPSoCBench: A benchmark for high-level evaluation of multiprocessor system-on-chip tools and methodologies. *J. Parallel Distrib. Comput.* **2016**, *95*, 138–157. [CrossRef]
8. Li, K. Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels. *J. Parallel Distrib. Comput.* **2016**, *95*, 15–28. [CrossRef]
9. Schoeberl, M.; Abbaspour, S.; Akesson, B. T-CREST: Time-predictable multi-core architecture for embedded systems. *J. Syst. Archit.* **2015**, *61*, 449–471. [CrossRef]
10. Wilhelm, R. Real time spent on real time. *Commun. ACM* **2020**, *63*, 54–60. [CrossRef]
11. Lee, E.; Messerschmitt, D. Pipeline interleaved programmable DSP's: Architecture. *IEEE Trans. Acoust. Speech Signal Process* **1987**, *35*, 1320–1333. [CrossRef]
12. Davis, R.I.; Altmeyer, S.; Indrusiak, L.S.; Maiza, C.; Nélis, V.; Reineke, J. An extensible framework for multicore response time analysis. *Real Time Syst.* **2018**, *54*, 607–661. [CrossRef]
13. Schoeberl, M.; Schleuniger, P.; Puffitsch, W.; Brandner, F.W.; Probst, C. Towards a Time-predictable Dual-Issue Microprocessor: The Patmos Approach. In Proceedings of the First Workshop on Bringing Theory to Practice: Predictability and Performance in Embedded Systems (PPES 2011), Grenoble, France, 18 March 2011; pp. 11–21.
14. Pedram, M. Power Minimization in IC Design, Principles and Applications. *ACM Trans. Des. Automat. Electron. Syst.* **1996**, *1*, 3–56. [CrossRef]
15. Kim, D.; Ko, Y.; Lim, S. Energy-Efficient Real-Time Multi-Core Assignment Scheme for Asymmetric Multi-Core Mobile Devices. *IEEE Access* **2020**, *8*, 117324–117334. [CrossRef]
16. Lorenzon, A.F.; Cera, M.C.; Beck, A.C. Investigating different general-purpose and embedded multicores to achieve optimal trade-offs between performance and energy. *J. Parallel Distrib. Comput.* **2016**, *95*, 107–123. [CrossRef]
17. Xie, G.; Zeng, G.; Xiao, X.; Li, L.; Li, K. Energy-Efficient Scheduling Algorithms for Real-Time Parallel Applications on Heterogeneous Distributed Embedded Systems. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 3426–3442. [CrossRef]
18. Chniter, H.; Mosbahi, O.; Khalgui, M.; Zhou, M.; Li, Z. Improved Multi-Core Real-Time Task Scheduling of Reconfigurable Systems with Energy Constraints. *IEEE Access* **2020**, *8*, 95698–95713. [CrossRef]
19. Ge, Y.; Liu, R. A Group-Based Energy-Efficient Dual Priority Scheduling for Real-Time Embedded Systems. *Information* **2020**, *11*, 191. [CrossRef]
20. Huang, C. HDA: Hierarchical and dependency-aware task mapping for network-on-chip based embedded systems. *J. Syst. Archit.* **2020**, *108*, 101740. [CrossRef]
21. Rehman, A.U.; Ahmad, Z.; Jehangiri, A.I.; Ala'Anzy, M.A.; Othman, M.; Umar, A.I.; Ahmad, J. Dynamic Energy Efficient Resource Allocation Strategy for Load Balancing in Fog Environment. *IEEE Access* **2020**, *8*, 199829–199839. [CrossRef]
22. Salloum, C.; Elshuber, M.; Höftberger, O.; Isakovic, H.; Wasicek, A. The ACROSS MPSoC—A new generation of multi-core processors designed for safety-critical embedded systems. *Microprocess. Microsyst.* **2012**, *37*, 1020–1032. [CrossRef]
23. Glaser, F.; Tagliavini, G.; Rossi, D.; Haugou, G.; Huang, Q.; Benini, L. Energy-Efficient Hardware-Accelerated Synchronization for Shared-L1-Memory Multiprocessor Clusters. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 633–648. [CrossRef]
24. Geier, M.; Brändle, M.; Chakraborty, S. Insert & Save: Energy Optimization in IP Core Integration for FPGA-based Real-time Systems. In Proceedings of the 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS), Nashville, TN, USA, 18–21 May 2021; pp. 80–91. [CrossRef]
25. Wimer, S.; Koren, I. Design Flow for Flip-Flop Grouping in Data-Driven Clock Gating. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 771–778. [CrossRef]
26. Bezati, E.; Casale-Brunet, S.; Mattavelli, M.; Janneck, J.W. Clock-Gating of Streaming Applications for Energy Efficient Implementations on FPGAs. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2017**, *36*, 699–703. [CrossRef]
27. Bsoul, A.A.M.; Wilton, S.J.E.; Tsoi, K.H.; Luk, W. An FPGA Architecture and CAD Flow Supporting Dynamically Controlled Power Gating. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *24*, 178–191. [CrossRef]
28. Çakmak, I.Y.; Toms, W.; Navaridas, J.; Luján, M. Cyclic Power-Gating as an Alternative to Voltage and Frequency Scaling. *IEEE Comput. Archit. Lett.* **2016**, *15*, 77–80. [CrossRef]
29. Greenberg, S.; Rabinowicz, J.; Tsechanski, R.; Paperno, E. Selective State Retention Power Gating Based on Gate-Level Analysis. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 1095–1104. [CrossRef]
30. Stallings, W. Reduced instruction set computer architecture. *Proc. IEEE* **1988**, *76*, 38–55. [CrossRef]
31. VC707 Evaluation Board for the Virtex-7 FPGA. Available online: https://www.xilinx.com/support/documentation/boards_and_kits/vc707/ug885_VC707_Eval_Bd.pdf (accessed on 1 November 2019).
32. Mälardalen: WCET Benchmark Programs. Available online: http://www.mrtc.mdh.se/projects/wcet/benchmarks.html (accessed on 1 March 2021).
33. Hussein, J.; Klein, M.; Hart, M. *Lowering Power at 28 nm with Xilinx 7 Series Devices*; Xilinx White Paper, WP389; 2015. Available online: https://www.xilinx.com/support/documentation/white_papers/wp389_Lowering_Power_at_28nm.pdf (accessed on 5 December 2021).