Energy Conscious Scheduling for Fault-Tolerant Real-Time Distributed Computing Systems

Savina Bansal, Rakesh Kumar Bansal, and Kiran Arora

1 Introduction

An integration between computing and physical world with upcoming advances in technology has made it possible to fulfill the growing computational demands and needs of industry and individuals. Pervasive computing devices employ controllers to read physical inputs through sensors, perform data processing, and feed tangible outputs to actuators. Real-time functions especially based on artificial intelligence such as computer vision and sensor fusion are gaining popularity due to cost-effective availability of needed hardware owing to advances in VLSI and related technologies. Real-time applications, as in avionics and aerospace engineering, automobile sectors, mission and safety-critical application in defense and medical fields, for which timely completion within a given time deadline is very crucial along with logical accuracy, demand usage of real-time systems. Timeliness is essential for real-time application as beyond the specific time window or time instant (also referred as task deadline of a task) even a logically correct outcome is of no use. Failing to honor deadline can lead to serious consequences—from loss of signal quality, as during video-conferencing, to some bigger financial loss or may even cost human lives [37, 49]. Real-time systems are capable of producing accurate results within the given deadline provided the tasks are scheduled properly over

S. Bansal · R. K. Bansal
Department of Electronics and Communication Engineering, Giani Zail Singh Campus College of Engineering & Technology, Bathinda, India
e-mail: savina.bansal@gmail.com; drrakeshkbansal@gmail.com

K. Arora (✉)
Department of Computer Science and Engineering, Baba Hira Singh Bhattal Institute of Engineering & Technology, Lehragaga, India
e-mail: erkiranarora@gmail.com

S. Pandey et al. (eds.), *Role of Data-Intensive Distributed Computing Systems in Designing Data Solutions*, EAI/Springer Innovations in Communication and Computing, https://doi.org/10.1007/978-3-031-15542-0_1

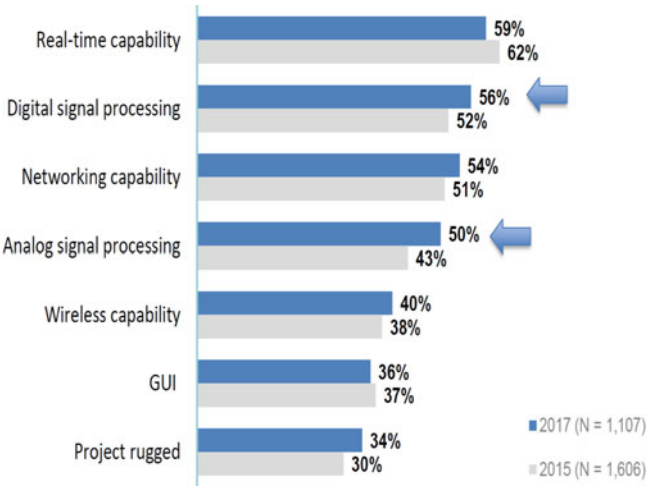

Fig. 1 Embedded system market [14]

them. Scheduling, in general, relates to allocation and assignment of incoming tasks to the underlying computing processor/s such that deadlines of all tasks get honored. The major concern of this work is on scheduling of real-time systems.

The development of systems with real-time capability is increasing at a fast pace (Fig. 1) to satisfy the needs of our day-to-day lives. More specifically, these systems have application that affects our social and personal lives directly or indirectly such as bank transactions, automobiles, traffic signal controller, medical care, video-conferencing, smart home, and firefighting [40]. As per the new Global Info Research study, it is projected that worldwide market growth for embedded systems will rise from 86,500 million US dollar in 2020 to 11,620 million US dollar in 2025, at a compound annual growth rate of approximately 6.1% [39]. For instance, contemporary cars have hundreds of processing units equipped to provide basic features such as vehicle control to specialized facilities for safety and comfort. To recognize its surroundings, perception subsystem in the vehicle should be able to process enormous data that demands huge computational power necessitating the use of multicore or multiprocessor systems [35] in order to achieve higher throughput, reduced response time, and increased reliability.

Substantial advancement in the performance of present-day computing systems has led to considerable rise in power consumption. In fact, the amount of heat generated by them is quickly growing to level equivalent to nuclear reactors as shown in Fig. 2 [46, 57]. As projected by Moore's law, energy utilization of computing systems has increased at an exponential rate from last few decades. Such rise in energy consumption results in ecological and monetary problems due to which energy management has turn out to be a prime design concern for computing systems [1, 13, 20, 38]. In the scientific and technical literature, the interrelation between energy, economy, and environment is recognized with "3E"

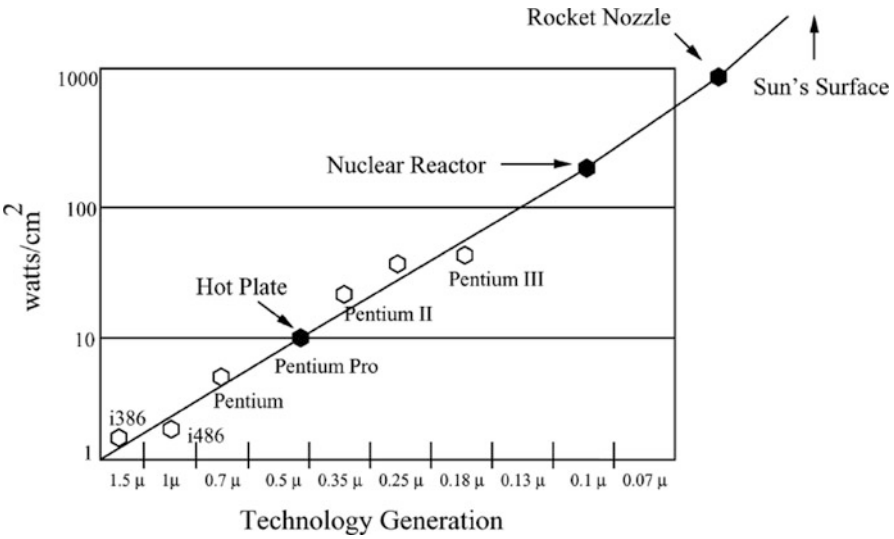

Fig. 2 Rise in power densities [46]

[56]. United Nations Development Organization proposed at the beginning of the twenty-first century that this triple bond is of global importance and is among the eight Millennium Development Goals (MDGs) in the global economic scenario. The evolution of objective of ensuring environmental sustainability promotes the importance of energy management. In light of this, 3E is more conspicuous today than ever before, making energy management a premier research problem for computational systems.

However, the main target for energy saving in such systems is processor, as a major fraction of total power is consumed by CPU alone. As the complexity and computational power of real-time computing systems grow, it leads to high operating temperature generation due to excessive transistor integration on small size chips. Miniaturization further aggravates energy consumption of processors. State-of-the-art processors consume a substantial amount of energy. For example, Intel Core i7-975 drains estimated 83 W of power in idle state, and AMD FX 8350 processor has a peak power consumption of 210 W [41]. Various assessments [5, 8, 72] recommended that main focus should be on power efficiency while designing complex real-time systems. Hence, it becomes necessary to consider energy management as a mandatory parameter for real-time multiprocessor scheduling algorithms.

Along with the timing precision, real-time systems must be reliable, but a precisely designed system may fail, which can lead to unexpected situations. Massive heat dissipation adversely affects reliability and performance of semiconductor devices as well and also contributes to global warming [17]. Another serious threat to reliability is caused by high operating temperature, which is a direct consequence of high power consumption generated owing to excessive transistor integration in

small size. As reported by Srinivasan et al. [53], maximum rise in temperature realized by 180 nm processor is 15 °K lesser than realized by 65 nm processor, and scaling to such small value leads to 316% growth in error rate. For safety-critical real-time computing systems, reliability is a vital feature because faults may cause deadline violations, which can also be disastrous at times. To avoid this, fault detection and tolerance features should be incorporated in the system to achieve high reliability so that it can operate proficiently even in case of faults. Therefore, reducing energy consumption while maintaining reliability of a real-time system is a challenging problem and requires consideration.

2 Energy Management

The presence of miniaturized electronic components and chips in the contemporary computing systems makes energy consumption scenario worst ever. The most prevailing digital electronic technology is Complementary Metal-Oxide Semiconductor (CMOS), whose peak power dissipation occurs during state transitions of transistors. To handle power consumption of CMOS circuits, static power dissipation (based on leakage voltage) and dynamic power dissipation (based on supply current) need to be minimized [31–33]:

- **Static power**: It arises as an after-effect of leakage current flowing through transistors. Leakage current increases exponentially with reduced thickness of insulating region and leads to rise in static power.
- **Dynamic power**: It is a consequence of repeated charging and discharging of capacitance of several hundreds of gates in contemporary chips.

To reduce static and dynamic power consumption, commonly used techniques are dynamic power management (DPM) and dynamic voltage and frequency scaling (DVFS), respectively. These techniques are overviewed in the following subsections.

2.1 Dynamic Power Management

Intel, HP, and Microsoft presented an enhanced framework, called Advanced Configuration and Power Interface (ACPI) for device configuration and monitoring [7, 60]. Basically, ACPI provides a simple and adaptable interface to operating system for configuring and discovering peripherals. The motive of ACPI-based power management is to put the whole system or devices that are unused or less used into low-power states when possible.

Due to arbitrary workloads during operation time, DPM attains energy efficiency in the system by judiciously lowering the performance of system components and by switching off the processor in idle periods, thereby saving energy. However, putting

system component to power-saving state and taking back to the active state involve energy and time overhead.

Every processor has some minimum transition time from one state to another called break-even time b_0. When an idle interval is greater than the break-even time, only then processor is put to sleep mode to take advantage of DPM for reducing energy consumption. DPM technique involves taking decision of putting the system components to a power-saving state based on the size of forthcoming idle period. For example, energy budget of switching between states will be larger than the energy saved in the sleep state if the idle period is relatively small. Thus, transition to power-saving state must be done only when idle interval is greater than break-even time. The smallest value of b_0 is the one that consumes exactly an equal amount of energy if kept in active state or transition it from active to power-efficient state.

2.2 Dynamic Voltage and Frequency Scaling

Growing computing capabilities demand usage of higher operating frequency of processors, which lead to higher energy consumptions. To sustain necessary processor performance by using higher operational frequencies, a number of integrated transistors per chip are growing day by day [12]. Fast switching of a large number of transistors increases the frequency of a processor and also makes them dissipate more dynamic power.

Dynamic power consumption of a processor and supply voltage have quadratic relation between them such that:

$$\rho_{dyn} = \wp \zeta_{ef} \upsilon^2 f, \tag{1}$$

where ρ_{dyn} is the dynamic power, $\wp$ is the gate activity factor, ζ_{ef} is the switched capacitance, υ is the supply voltage, and f is the operating frequency. DVFS dynamically adjusts voltage/frequency to reduce processor's power consumption; however, it trades energy with performance since reducing frequency will in turn increase execution time of application. The challenge for DVFS technique for real-time applications is how to preserve the feasibility of a system while reducing voltage so that all deadlines can be honored and energy consumption is decreased. So, care must be taken while using DVFS for real-time applications, as they have stringent timing constraint.

Nowadays, processors being launched in the market have DVFS capabilities enabled on it, such as an AMD R-series [2]. Thus, in contemporary processors, it is possible to dynamically regulate the supply voltage and operational frequency to cut down dynamic power consumption using DVFS but at the price of elongated circuit delay [6, 9]. Real-time DVFS techniques can be differentiated based on time of speed adjustment as inter-task and intra-task.

– **Inter-task DVFS**: With inter-task DVFS techniques, a job runs at the same frequency level until it finishes its execution after being dispatched or is preempted by a high-priority job. The speed may be readjusted when it restarts execution after preemption depending on the available slack at that particular time. A majority of DVFS algorithms are based on inter-task technique as it has low run-time overhead.
– **Intra-task DVFS**: The intra-task algorithms adjust the speed at the well-determined power-management points (PMPs) at run time and focus on reducing dynamic energy consumption. But they involve extra energy and time overhead owing to a large number of speed changes.

Decrease in processor frequency leads to reduction in frequency-dependent power, but it increases execution time of task, which in turn results in rise in static and independent power. To overcome this problem, a critical frequency f_{crit}, also called energy-efficient frequency, has been proposed in the literature [29], below which the DVFS does not remain effective. So, tasks should not be executed at frequency lower than f_{crit}.

3 Fault Tolerance

Rapid advancement in scale and complexity of real-time multiprocessor computing systems has made reliability an increasingly challenging issue. Due to the aggressive scaling of transistors, CMOS devices become more susceptible to extrinsic effects such as high-energy radiations and electromagnetic interference. Thus, computing systems have become prone to various types of faults that may introduce some errors in results. In a combinational logic circuit from 600 nm to 50 nm feature size [52], the soft error rate (SER) per chip increases by nine orders of magnitude. If scaling process remains at the same pace, then for 16 nm technology, per day per computer chip will have at least one failure [23, 34].

Despite being designed perfectly, a system may fail abnormally owing to unpredictable fault occurrence. A fault is a situation of unusual response due to some defect in the system. A fault may be a hardware defect or an implementation flaw in the software. In other words, a system is supposed to have failure when service provided by it diverges from the desired service. For example, a computing system that observes the state of critical patients in the hospital must take an action as soon as the patient's state changes. A remedial measure must be taken if patient's blood pressure decreases/increases beyond a specific threshold, such as giving an alarm or injecting medicine in patient's body. This process must be performed strictly in a defined time limit (or deadline). Thus, computing system employed in hospitals especially in intensive care units (ICU) should guarantee that even if the processor incurs fault, the task is executed within its deadline [18]. Another example is in flight control systems where often tasks are activated by the controllers depending

on the output seen on screen. However, if system incurs a fault, it should be able to handle fault before the deadline [36].

Processor faults are broadly classified as permanent and transient faults:

- **Permanent faults**: Hardware failures lead to permanent faults caused due to manufacturing defects at the time of fabrication or due to wear and tear because of aging. The sole way to tolerate permanent faults is hardware redundancy (to employ additional hardware components). Permanent faults cause damages to processors and can hamper its working.
- **Transient fault**: This type of fault generates soft errors (or single-event upsets (SEU)), which is not persistent and may cause errors in computation or corruption in data. Moreover, with continuing scaling of CMOS technologies, approximately all digital systems are prone to transient faults along with systems that work in outer space [70]. Studies showed that transient faults appear more often as compared to permanent faults [11, 19].

Many techniques have been proposed for detecting faults based on hardware and software [30, 42]. The well-known error detection mechanisms are fail-signal processors, alarms or watchdogs, signatures, and acceptance tests (ATs) [10, 23, 45].

3.1 Fault-Tolerant Techniques

Fault tolerance is basically concealing error by switching to another unit of work at the time of fault occurrence. Redundancy is generally applied in the form of extra resources to mask faults for preserving required levels of performance in the system [19]. To integrate fault tolerance in the computing system, approaches have been suggested to tolerate faults that are generally based on redundancy of various resources such as hardware, software, time, and information.

- **Hardware redundancy** is achieved by deploying extra hardware in the system for the replacement of a faulty component.
- **Software redundancy** employs substitute implementations of program that can be used in case the initial version encounters a fault at run time.
- **Information redundancy** techniques are used to handle faults that occur while transferring or storage of data such as error detection and correcting codes.
- **Time redundancy** uses extra CPU time for re-execution of a faulty task or executes a secondary task in case of a fault.

To tolerate permanent faults, hardware redundancy is essential, but repeating the execution of task fully or partially helps in tolerating a transient fault [48]. Re-execution and checkpointing are two most commonly used time redundancy-based techniques for tolerating transient faults that repeat task fully and partially, respectively.

- **Checkpointing**: This technique saves the snapshot of current state of system to stable storage during the execution at regular intervals called checkpoints, where every checkpoint comprises all the context information required to restart process from that point of time. On detecting a fault, system re-executes faulty segment from the most recent correct checkpoint. This technique is able to tolerate g-faults in a task.
- **Re-execution**: Under this technique, re-execution of original task in a failure situation is done and is widely used to tolerate transient fault.

If the system is safety critical, task duplication/replication is used to tolerate transient faults to provide required reliability level. However, redundancy increases resource overhead such as rise in energy consumption. Owing to the rising concern for energy management and reliability in contemporary world, energy-saving techniques must be incorporated in fault-tolerant real-time task scheduling.

4 Joint Optimization of Energy and Fault Management

Computing systems are nowadays affecting almost every facet of our everyday life. Due to the increased responsibilities, it becomes essential that computer systems should provide both safety and reliability. For many years, researchers have addressed the emerging problems of system reliability, which come along with this thriving evolution of VLSI technology and raised it as prime design concern for real-time systems. Energy management has also become as an essential design parameter for real-time systems due to various environmental, economic, technical, and social issues such as hike in green-house gas emission, cooling infrastructure cost due to more heat dissipation, and damage to public health. If not judiciously handled, high energy consumption and degraded reliability will restrict the advancements to be made to real-time multiprocessor computing systems in upcoming future.

Systems such as avionics, defense, and space exploration with real-time constraints need to be reliable as well as energy-efficient. Conventional approaches focused solely on timing constraints, whereas recently additional design issues such as thermal, energy, and reliability have gained attention, which has made the scheduling problem more complex. Hence, it is desirable that task scheduling algorithms for real-time systems must consider different constraints such as timing, energy, and reliability and be designed systematically to accomplish the specified design objectives.

Together, reliability and energy management are conflicting design goals for a real-time system. Redundancy-based reliability/fault-tolerance enhancement techniques increase energy consumption due to overhead of the additional resources/computation. Researchers have also observed that there exists an inverse relationship between supply voltage and the rate of transient faults. As a result, reducing energy consumption makes the system more vulnerable to transient faults.

The ability to achieve timeliness for real-time applications increases on multiprocessor systems with an increased number of computing units. With a greater number of processing units, the possibility of enhancing reliability/fault tolerance improves by having increased prospects for replication of tasks. However, redundancy of tasks raises energy consumption due to multiple executions. Owing to this reason, energy-aware task scheduling algorithms remain a pressing concern for the fault-tolerant real-time multiprocessor system. However, it is challenging to reduce energy while tolerating faults because both are conflicting issues having a trade-off between them. These concerns have motivated the research for joint optimization of energy and system reliability.

Several schemes are available in the literature that deals with the joint problem of power and reliability management on single as well as multiprocessor platforms. Re-execution, checkpointing, and replication along with voltage scaling and shutdown methods are frequently used strategies to preserve desired level of reliability/fault tolerance and power management in the system. Not only the task ordering for execution on a given processor but task mapping to various processors also affects energy consumption and reliability of the system. Hence, there are various aspects of fault-tolerant task scheduling on a real-time multiprocessor system where energy efficiency can be improved. The research fraternity has shifted to examine the problems at the intersection of fault tolerance and power management in recent past. Task scheduling techniques for joint management of fault tolerance and energy efficiency are discussed below as per the classification shown in Fig. 3.

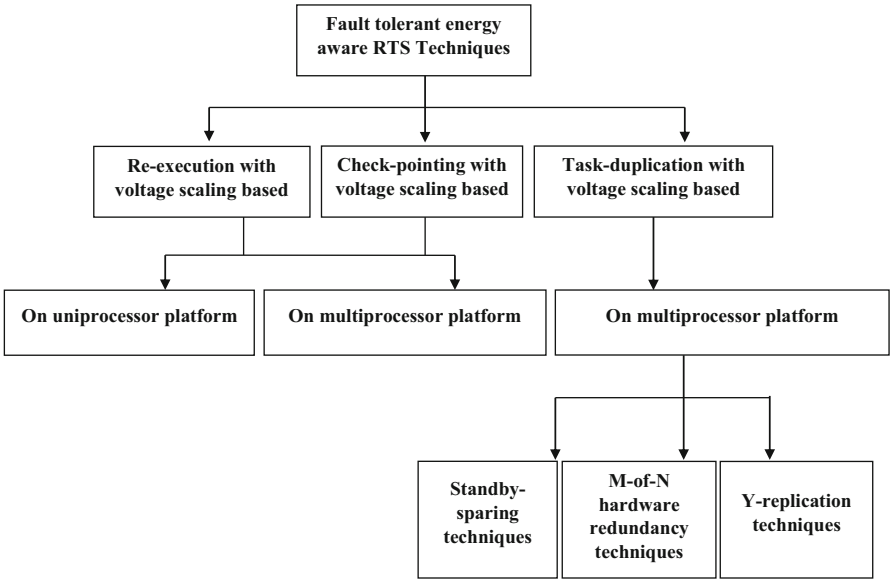

Fig. 3 Classification of real-time scheduling techniques with joint management of energy and reliability

Re-execution with Voltage Scaling A combination of time redundancy and voltage scaling is used to tackle the joint problem of fault tolerance and energy management on uniprocessor system. Based on re-execution strategy, reliability-aware power management (RA-PM) refers to the unified approach of energy management and fault tolerance based on time redundancy and has been explored in the literature with different aspects. It refers to the notion of original reliability, which is the probability of successfully executing all real-time tasks at maximum CPU speed with no transient fault. RA-PM works by utilizing the available slack for slowing down the tasks with DVFS policy as well as for executing backup copy of scaled tasks in case of fault [69]. Zhu et al. [69] proposed RA-PM over periodic real-time tasks by considering both EDF [69] and RM [71] as underlying scheduling algorithms and showed that RA-PM approach maintains the original reliability of all tasks while saving energy.

In another work based on aperiodic tasks, Zhu [70] exploited dynamic slack for further lowering the frequency of tasks and to assign backup tasks to enhance reliability with RA-Greedy algorithm. He also proposed checkpointing for utilizing dynamic slack when recovery placement is not possible due to small size of available slack. The energy-constrained version of reliability-aware power management (ECRM) has been presented by Zhao et al. [65], where they focus on achieving maximum reliability for a real-time system that works in a limited energy budget.

For fixed-priority real-time system with weakly hard QoS constraint, Niu et al. [44] proposed reliability conscious energy-aware scheduling (FPRMK-EM) algorithm by reserving space for recovery of mandatory jobs in case of failure and reducing frequency of other tasks for energy efficiency.

Zhang et al. [61] targeted to improve energy savings of real-time system with shared resources under the constraint of reliability with EDF/DDM as underlying scheduling algorithm. They proposed Dynamic Low-Power Scheduling Algorithm for Periodic Tasks with Shared Resources (DLPSR) algorithm that exploits dynamic slack for reliability preservation and energy conservation.

Considering the preemption overhead, Xu et al. [59] proposed reliability-aware power-management algorithms that effort to reduce execution time and energy consumption of real-time tasks by minimizing the number of preemptions. They proposed greedy energy efficiency scheduling algorithm (GEE) based on greedy strategy of maximally utilizing slack time. Further, GEEPU and GLEEPU have been proposed that reduce frequency based on processor utilization, and DGAET exploits dynamic slack for improving energy saving.

Zhao et al. [66] proposed Generalized Shared Recovery (GSHR) technique, where in spite of separate recovery copies for scaled tasks, one or more global shared recovery blocks are reserved, which can be used by any task at whatever time in the situation of fault. In case a task encounters a fault, it uses the recovery block, and the rest of the tasks are then executed at the maximum speed. This scheme improves the reliability of a system to great extent due to the ability to tolerate multiple faults by same task with multiple shared recovery blocks. Thus, it can be used for safety-critical systems where it is essential to maintain certain arbitrary level of reliability in an energy-efficient manner. The authors proposed shared recovery

scheme for common-deadline-based tasks (Incremental Reliability Configuration Search—IRCS) [64, 66], periodic tasks [67], and precedence constrained tasks (SHR-DAG) [68], where frequency of tasks and the number of recovery blocks are determined based on the given reliability target.

Qi et al. [50] proposed global-scheduling-based reliability-aware power-management scheme for individual and shared recovery schemes on multiprocessor system. They showed that it is NP-hard problem to find an optimal solution for the selection of subgroup of tasks for energy and reliability management. Algorithms to exploit dynamic slack have also been proposed by them to improve energy savings.

Reliability-aware dynamic power management (RA-DPM) has been presented by Fan et al. [17] with shared recovery blocks on single processor system. As soon as a minimum number of tasks execute successfully, time reserved for recoveries is used for reducing frequency for extending energy savings dynamically.

Huang et al. [28] proposed energy-efficient fault-tolerant mapping and scheduling for precedence constrained tasks with mixed-integer linear programming formulation on heterogeneous multiprocessor system. They proposed List-based Binary Particle Swarm Optimization (LBPSO) algorithm that is based on particle swarm optimization to obtain high-quality solution in terms of energy saving and reliability.

Checkpointing with Voltage Scaling In order to guarantee reliability and energy efficiency, an adaptive checkpointing scheme (ADT-DVS) has been presented by Zhang et al. [63] assuming Poisson fault model. They adjust checkpoint intervals dynamically to tolerate a fixed number of faults for a set of periodic tasks with EDF scheduling policy on a single processor system.

For fixed-priority scheduling algorithm, Zhang et al. [62] proposed a unified approach for checkpointing and DVFS (both task-level and application-level speed scaling) for tolerating g-transient faults while lessening energy consumption for periodic real-time task sets. The authors used genetic-algorithm-based approach (GA) to find the optimal frequency assignment with exhaustive search, which is computationally unfeasible for heavy workload applications on the processor with a large number of available discrete frequency levels. Using adaptive checkpointing technique, work was done by Wei et al. [58] based on the behavior of tasks and fault rate at run time while complying with tasks' deadline constraints. Two offline DVFS scheduling algorithms—application-level DVS (A-DVS) and task-level DVS (T-DVS)—were proposed for fixed-priority real-time tasks by exploring dynamic slack to minimize energy consumption.

Another non-uniform checkpointing technique combined with DVFS for power management has been presented by Melhem et al. [43], which has an advantage of improved energy saving over uniform checkpointing. They considered EDF scheduling algorithm for periodic tasks on a single-core processor with the constraint of having at most one failure in the system. To reduce the number of checkpoints for the sake of minimizing energy consumption, two-state checkpointing (TsCp) concept has been introduced by Salehi et al. [51] where non-uniform

checkpointing is applied in the fault-free scenario but as soon as the fault occurs, system shifts to uniform checkpointing policy.

Duplication with Voltage Scaling A majority of strategies that have been proposed in the literature for fault-tolerant energy-aware task scheduling based on duplication of task copies are for homogeneous platform. Research works done on multiprocessors using duplication of task have been divided into three categories based on the number of duplicate/replicated copies of tasks as follows:

– **Standby-sparing techniques**: Standby-sparing strategy uses one level of replication, such that each task has exactly one replica to execute for fault handling on dual processor system. The workload handled by this technique is not greater than the maximum utilization bound of single processor because extra processor is just employed to provide fault tolerance by scheduling duplicate task copies on it.

  For independent periodic tasks with common deadline, Ejlali et al. [15] proposed that instead of using standby-sparing scheme with hot or cold spares, Low-Energy Standby-Sparing (LESS) is more effective in saving energy while providing reliability. LESS reduces voltage of primary tasks by applying DVFS and delays backup tasks maximally keeping the deadline constraint fulfilled. They considered reduced energy model and reliability model by considering energy and time overheads as well as static-energy consumption.

  Aminzadeh et al. [3] did the comparative analysis of system-level energy-management schemes based on DVFS and DPM for standby-sparing systems and proposed a Markov model to analyze their energy and reliability parameters. They proposed that hybrid method of postponing secondary tasks and frequency reduction of primary and backup tasks on standby-sparing system always save more energy as compared to simple DVFS and DPM methods.

  For fixed-priority scheduling, Haque et al. [26] suggested that executing primary tasks at lower voltage and backing up tasks at maximum voltage maintain reliability of the system as well as save energy. They proposed Standby-Sparing Fixed-Priority (SSFP) algorithm for periodic tasks that uses dual-priority scheduling approach on spare processor to delay backup tasks and applied deallocation strategy as well to save energy by canceling backup tasks whose corresponding primary tasks have finished successfully. Dynamic slack has been exploited to enhance energy saving by reducing speed of tasks on main processor and further delaying of backup tasks at run time.

  Ansari et al. [4] followed the similar concept of [26] for energy- and reliability-aware scheduling on standby-sparing system by using dual-priority strategy for earliest deadline first scheduling algorithm. They presented a new Adaptive Dual-Queue scheduling (AdDQ) algorithm and showed that their work saves 14% more energy than ASSPT and CSSPT algorithms [24].
– **M-of-N hardware redundancy techniques**: Optimistic TMR has been proposed by Elnozahy et al. [16] to reduce the energy consumption for conventional TMR systems. Two out of three machines run at lower frequency and their result is

matched. The output is released, but in case of deviation in results, output of third machine that was slower than other ones is used as tie-breaker.

Benefit of multiprocessor platform has been exploited by Salehi et al. [51] with N-modular redundancy to improve reliability by masking errors on multiple processing units; however, it imposes substantial energy overhead. They suggested to work in two phases to carry out N-modular redundancy. Half-plus-one copies for every task are executed in the first indispensable phase, and the rest of the copies are executed in the on-demand phase only if fault appeared in the earlier phase thereby saving energy in fault-free scenario.

– **Y-replication techniques**: For a set of independent periodic tasks, Unsal et al. [55] proposed an energy-aware fault-tolerance technique with primary–backup scheme, which defers the execution of backup tasks as late as possible to minimize overlap between the execution of primary and backup copies. Energy consumption is reduced by canceling the backup copy on successful completion of primary copy.

For the heterogeneous systems, Tosun [54] proposed energy- and reliability-aware task scheduling and achieved 62% energy saving against energy-oblivious schemes. He presented an integer linear-programming-based framework for mapping and scheduling tasks to heterogeneous multiprocessor system on chip (HMPSoC) for periodic real-time tasks.

For highly safety-critical systems, to achieve target reliability level, a certain number of replicas are required. But to generate an energy-efficient schedule, tasks must be executed at reduced frequency value. For preemptive periodic real-time applications, Haque et al. [27] analyzed the interplay between the energy, replication, frequency, and reliability. They proposed a method to create energy–frequency–reliability (EFR) table [25] and then how to use it for determining the extent of replication and frequency reduction for lowering energy consumption with the help of energy-efficient replication (EER) algorithm.

Poursafaei et al.[47] used EFR table and presented an algorithm that works in two phases. The first phase is offline replication in which extent of replication and frequency reduction is determined for every task depending on the given reliability target. Later on, at run time, the online phase prevents the execution of redundant copies of task whose one of the copies has finished successfully.

By extending the concept of standby-sparing scheme to multiprocessor system, Guo et al. [22] proposed paired-SS and generalized-SS task configuration schemes for independent periodic real-time tasks with dynamic-priority scheduling algorithm. They used worst-fit decreasing strategy for task allocation and showed that generalized-SS is a more energy-efficient configuration for dynamic-priority task set on multiprocessor system. Later on, they extended the concept for mixed scheduling where in spite of standby-sparing configuration, tasks are allocated in mixed manner, such that every processor has a mixture of primary and backup tasks provided that copies of same task are not allocated to the same processor with POED-Mix algorithm. POED algorithm is used to schedule primary tasks with ASAP preference and backup tasks in ALAP manner [21] to save energy by delaying backup tasks for reducing overlap in the execution of two copies of same task as well as minimizing the number of executed backup.

5 Conclusion

With the growing availability of multiprocessor technology, hardware redundancy has emerged as a suitable candidate for providing fault tolerance in real-time systems. Duplicating tasks on separate processing units has turned up as a fitting technique to meet stringent reliability requirements. But efficient scheduling techniques are required to handle the after-effect of replicating task resulting in increased energy consumption. Use of dynamic voltage scaling and dynamic power-management techniques has been the choice of researchers for designing energy-efficient scheduling algorithms. However, in fault-tolerant systems, careful application of energy-management schemes is required, as execution on processor at lower voltage raises fault rate.

References

1. Agarwal, M. M., Govil, M. C., Sinha, M., & Gupta, S. (2019). Fuzzy based data fusion for energy efficient internet of things. *International Journal of Grid and High Performance Computing, 11*(3), 46–58. https://doi.org/10.4018/ijghpc.2019070103
2. AMD. 2nd generation AMD embedded R-series APU. https://www.amd.com/en/products/embedded-r-series-2nd-gen-apu (2nd). Accessed 20 March 2020
3. Aminzadeh, S., & Ejlali, A. (2011). A comparative study of system-level energy management methods for fault-tolerant hard real-time systems. *IEEE Transactions on Computers 60*(9), 1288–1299 (2011). https://doi.org/10.1109/tc.2011.42
4. Ansari, M., Safari, S., Poursafaei, F. R., & Salehi, M. (2017). AdDQ: Low-energy hardware replication for real-time systems through adaptive dual-queue scheduling. *The CSI Journal on Computer Science and Engineering, 15*(1), 31–38.
5. Attia, K. M., El-Hosseini, M. A., & Ali, H. A. (2017). Dynamic power management techniques in multi-core architectures: A survey study. *Ain Shams Engineering Journal, 8*(3), 445–456. https://doi.org/10.1016/j.asej.2015.08.010
6. Aydin, H., Melhem, R., Mosse, D., & Mejia-Alvarez, P. (2004). Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers, 53*(5), 584–600. https://doi.org/10.1109/tc.2004.1275298
7. Bambagini, M. (2014). Energy Saving in Real-Time Embedded Systems. Ph.D. Thesis, ReTiS Lab, TeCIP Institute, Pisa, Italy.
8. Bambagini, M., Marinoni, M., Aydin, H., & Buttazzo, G. (2016). Energy-aware scheduling for real-time systems. *ACM Transactions on Embedded Computing Systems, 15*(1), 1–34. https://doi.org/10.1145/2808231
9. Burd, T. D., & Brodersen, R. W. (1995). Energy efficient CMOS microprocessor design. In *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences* (Vol. 1, pp. 288–297). https://doi.org/10.1109/HICSS.1995.375385
10. Campbell, A., McDonald, P., & Ray, K. (1992). Single event upset rates in space. *IEEE Transactions on Nuclear Science, 39*(6), 1828–1835. https://doi.org/10.1109/23.211373
11. Castillo, X., McConnel, S. R., & Siewiorek, D. P. (1982). Derivation and calibration of a transient error reliability model. *IEEE Transactions on Computers, C-31*(7), 658–671. https://doi.org/10.1109/tc.1982.1676063
12. Cong, J., Nagaraj, N. S., Puri, R., Joyner, W., Burns, J., Gavrielov, M., Radojcic, R., Rickert, P., & Stork, H. (2009). Moore's law: Another casualty of the financial meltdown? In *2009 46th ACM/IEEE Design Automation Conference* (pp. 202–203).

13. Dewangan, B. K., Agarwal, A., Venkatadri, M., & Pasricha, A. (2019). Energy-aware autonomic resource scheduling framework for cloud. *International Journal of Mathematical, Engineering and Management Sciences, 4*(1), 41–55. https://doi.org/10.33889/ijmems.2019.4.1-004
14. EETimes, Staff, E. (2017). 2017 Embedded Market Survey (2017). Accessed 21 May 2020.
15. Ejlali, A., Al-Hashimi, B. M., & Eles, P. (2012). Low-energy standby-sparing for hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 31*(3), 329–342. https://doi.org/10.1109/tcad.2011.2173488
16. Elnozahy, E., Melhem, R., & Mosse, D. (2002) Energy-efficient duplex and TMR real-time systems. In *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002*. IEEE Comput. Soc. https://doi.org/10.1109/real.2002.1181580
17. Fan, M., Han, Q., & Yang, X. (2017). Energy minimization for on-line real-time scheduling with reliability awareness. *Journal of Systems and Software, 127*, 168–176. https://doi.org/10.1016/j.jss.2017.02.004
18. Ghosh, S., Melhem, R., & Mosse, D. (1997). Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems, 8*(3), 272–284. https://doi.org/10.1109/71.584093
19. Ghosh, S., Melhem, R., Mossé, D., & Sarma, J. S. (1998). Fault-tolerant rate-monotonic scheduling. *Real-Time Systems, 15*(2), 149–181. https://doi.org/10.1023/a:1008046012844
20. Goyal, N., Dave, M., & Verma, A. K. (2016). Energy efficient architecture for intra and inter cluster communication for underwater wireless sensor networks. *Wireless Personal Communications, 89*(2), 687–707. https://doi.org/10.1007/s11277-016-3302-0
21. Guo, Y., Su, H., Zhu, D., & Aydin, H. (2015). Preference-oriented real-time scheduling and its application in fault-tolerant systems. *Journal of Systems Architecture, 61*(2), 127–139. https://doi.org/10.1016/j.sysarc.2014.12.001
22. Guo, Y., Zhu, D., Aydin, H., Han, J. J., & Yang, L. T. (2017). Exploiting primary/backup mechanism for energy efficiency in dependable real-time systems. *Journal of Systems Architecture, 78*, 68–80. https://doi.org/10.1016/j.sysarc.2017.06.008
23. Han, Q., Wang, T., & Quan, G. (2015). Enhanced fault-tolerant fixed-priority scheduling of hard real-time tasks on multi-core platforms. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE. https://doi.org/10.1109/rtcsa.2015.22
24. Haque, M. A., Aydin, H., & Zhu, D. (2011). Energy-aware standby-sparing technique for periodic real-time applications. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*. IEEE. https://doi.org/10.1109/iccd.2011.6081396
25. Haque, M. A., Aydin, H., & Zhu, D. (2013). Energy-aware task replication to manage reliability for periodic real-time applications on multicore platforms. In *2013 International Green Computing Conference Proceedings* (pp. 1–11). IEEE. https://doi.org/10.1109/igcc.2013.6604518
26. Haque, M. A., Aydin, H., & Zhu, D. (2015). Energy-aware standby-sparing for fixed-priority real-time task sets. *Sustainable Computing: Informatics and Systems, 6*, 81–93. https://doi.org/10.1016/j.suscom.2014.05.001
27. Haque, M. A., Aydin, H., & Zhu, D. (2017). On reliability management of energy-aware real-time systems through task replication. *IEEE Transactions on Parallel and Distributed Systems, 28*(3), 813–825. https://doi.org/10.1109/tpds.2016.2600595
28. Huang, K., Jiang, X., Zhang, X., Yan, R., Wang, K., Xiong, D., & Yan, X. (2018). Energy-efficient fault-tolerant mapping and scheduling on heterogeneous multiprocessor real-time systems. *IEEE Access, 6*, 57614–57630. https://doi.org/10.1109/access.2018.2873641
29. Jejurikar, R., Pereira, C., & Gupta, R. (2001). Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st Annual Design Automation Conference, DAC '04* (pp. 275–280). ACM. https://doi.org/10.1145/996566.996650
30. Jhumka, A., Hiller, M., Claesson, V., & Suri, N. (2002). On systematic design of globally consistent executable assertions in embedded software. *ACM SIGPLAN Notices, 37*(7), 75. https://doi.org/10.1145/566225.513843

31. Kaur, N., Bansal, S., & Bansal, R. K. (2016). Energy conscious scheduling with controlled threshold for precedence-constrained tasks on heterogeneous clusters. *Concurrent Engineering, 25*(3), 276–286. https://doi.org/10.1177/1063293x16679001
32. Kaur, N., Bansal, S., & Bansal, R. K. (2016). Energy efficient duplication-based scheduling for precedence constrained tasks on heterogeneous computing cluster. *Multiagent and Grid Systems, 12*(3), 239–252. https://doi.org/10.3233/MGS-160252
33. Kaur, N., Bansal, S., & Bansal, R. K. (2017). Duplication-controlled static energy-efficient scheduling on multiprocessor computing system. *Concurrency and Computation: Practice and Experience, 29*(12), e4124. https://doi.org/10.1002/cpe.4124
34. Khudia, D. S., & Mahlke, S. (2014). Harnessing soft computations for low-budget fault tolerance. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE. https://doi.org/10.1109/micro.2014.33
35. Kim, J., Kim, H., Lakshmanan, K., & Rajkumar, R. (2013). Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. In *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)* (pp. 31–40).
36. Lala, J., & Harper, R. (1994). Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE, 82*(1), 25–40. https://doi.org/10.1109/5.259424
37. Leveson, N. G. (1986). Software safety: Why, what, and how. *ACM Computing Surveys, 18*(2), 125–163. https://doi.org/10.1145/7474.7528
38. Li, K. (2016). Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels. *Journal of Parallel and Distributed Computing, 95*, 15–28. https://doi.org/10.1016/j.jpdc.2016.02.006
39. Market, E.S. (2020). Embedded system market by hardware (MPU, MCU, application-specific integrated circuits, DSP, FPGA, and memories), software (middleware, operating systems), system size, functionality, application, region—global forecast to 2025. Accessed 21 May 2020.
40. Marwedel, P. (2018). *Embedded system design*. Springer International Publishing. https://doi.org/10.1007/978-3-319-56045-8
41. Masiero, M., & Roos, A. (2012). Power consumption—CPU charts 2012: 86 processors from AMD and Intel, tested (2012). Accessed 02 Jan 2020.
42. Meixner, A., Bauer, M. E., & Sorin, D. (2007). Argus: Low-cost, comprehensive error detection in simple cores. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. IEEE. https://doi.org/10.1109/micro.2007.18
43. Melhem, R., Mosse, D., & Elnozahy, E. (2004). The interplay of power management and fault recovery in real-time systems. *IEEE Transactions on Computers, 53*(2), 217–231. https://doi.org/10.1109/tc.2004.1261830
44. Niu, L., & Li, W. (2016). Reliability-conscious energy management for fixed-priority real-time embedded systems with weakly hard QoS-constraint. *Microprocessors and Microsystems, 46*, 107–121. https://doi.org/10.1016/j.micpro.2016.03.005
45. Oh, S. K., & Macewen, G. H. (1992). Toward fault-tolerant adaptive real-time distributed systems.
46. Pollack, F. J. (1999). New microarchitecture challenges in the coming generations of CMOS process technologies (keynote address) (abstract only). In *Proceedings of the 32Nd Annual ACM/IEEE International Symposium on Microarchitecture, MICRO 32* (p. 2). IEEE Computer Society.
47. Poursafaei, F. R., Safari, S., Ansari, M., Salehi, M., & Ejlali, A. (2015). Offline replication and online energy management for hard real-time multicore systems. In *2015 CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE. https://doi.org/10.1109/rtest.2015.7369847
48. Pradhan, D. K. (1996). *Fault-tolerant computer system design*. Prentice-Hall.
49. Punnekkat, S. (1997). Schedulability Analysis for Fault Tolerant Real-time Systems. Ph.D. Thesis, University of York, UK.

50. Qi, X., Zhu, D., & Aydin, H. (2011). Global scheduling based reliability-aware power management for multiprocessor real-time systems. *Real-Time Systems, 47*(2), 109–142. https://doi.org/10.1007/s11241-011-9117-x
51. Salehi, M., Ejlali, A., & Al-Hashimi, B. M. (2016). Two-phase low-energy n-modular redundancy for hard real-time multi-core systems. *IEEE Transactions on Parallel and Distributed Systems, 27*(5), 1497–1510. https://doi.org/10.1109/tpds.2015.2444402
52. Shivakumar, P., Kistler, M., Keckler, S., Burger, D., & Alvisi, L. (2002). Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proceedings International Conference on Dependable Systems and Networks*. IEEE Comput. Soc. https://doi.org/10.1109/dsn.2002.1028924
53. Srinivasan, J., Adve, S., Bose, P., & Rivers, J. (2004). The impact of technology scaling on lifetime reliability. In *International Conference on Dependable Systems and Networks, 2004*. IEEE. https://doi.org/10.1109/dsn.2004.1311888
54. Tosun, S. (2011). Energy- and reliability-aware task scheduling onto heterogeneous MPSoC architectures. *The Journal of Supercomputing, 62*(1), 265–289. https://doi.org/10.1007/s11227-011-0720-3
55. Unsal, O. S., Koren, I., & Krishna, C. M. (2002). Towards energy-aware software-based fault tolerance in real-time systems. In *Proceedings of the 2002 International Symposium on Low Power Electronics and Design* (pp. 124–129). ACM Press. https://doi.org/10.1145/566408.566442
56. Uribe-Toril, J., Ruiz-Real, J., Milán-García, J., & de Pablo Valenciano, J. (2019). Energy, economy, and environment: A worldwide research update. *Energies, 12*(6), 1120. https://doi.org/10.3390/en12061120
57. Venkatachalam, V., & Franz, M. (2005). Power reduction techniques for microprocessor systems. *ACM Computing Surveys, 37*(3), 195–237. https://doi.org/10.1145/1108956.1108957
58. Wei, T., Mishra, P., Wu, K., & Zhou, J. (2012). Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems. *Journal of Systems and Software, 85*(6), 1386–1399. https://doi.org/10.1016/j.jss.2012.01.020
59. Xu, H., Li, R., Zeng, L., Li, K., & Pan, C. (2018). Energy-efficient scheduling with reliability guarantee in embedded real-time systems. *Sustainable Computing: Informatics and Systems, 18*, 137–148. https://doi.org/10.1016/j.suscom.2018.01.005
60. Zahaf, H. E. (2016). Energy efficient scheduling of parallel real-time tasks on heterogeneous multicore systems. Ph.D. Thesis, Lille 1 University of Science and Technology, France.
61. Zhang, Y. W., Zhang, H. Z., & Wang, C. (2017). Reliability-aware low energy scheduling in real time systems with shared resources. *Microprocessors and Microsystems, **52***, 312–324. https://doi.org/10.1016/j.micpro.2017.06.020
62. Zhang, Y., & Chakrabarty, K. (2006). A unified approach for fault tolerance and dynamic power management in fixed-priority real-time embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25*(1), 111–125. https://doi.org/10.1109/tcad.2005.852657
63. Zhang, Y., & Chakrabarty, K. (2004). Dynamic adaptation for fault tolerance and power management in embedded real-time systems. *ACM Transactions on Embedded Computing Systems, 3*(2), 336–360. https://doi.org/10.1145/993396.993402
64. Zhao, B., Aydin, & H., Zhu, D. (2009). Enhanced reliability-aware power management through shared recovery technique. In *Proceedings of the 2009 International Conference on Computer-Aided Design* (pp. 63–70). ACM Press. https://doi.org/10.1145/1687399.1687412
65. Zhao, B., Aydin, H., & Zhu, D. (2010). On maximizing reliability of real-time embedded applications under hard energy constraint. *IEEE Transactions on Industrial Informatics, 6*(3), 316–328. https://doi.org/10.1109/tii.2010.2051970
66. Zhao, B., Aydin, H., & Zhu, D. (2011). Generalized reliability-oriented energy management for real-time embedded applications. In *Proceedings of the 48th Design Automation Conference on—DAC '11*. ACM Press. https://doi.org/10.1145/2024724.2024815

67. Zhao, B., Aydin, H., & Zhu, D. (2012). Energy management under general task-level reliability constraints. In *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium* (pp. 285–294). IEEE. https://doi.org/10.1109/rtas.2012.30
68. Zhao, B., Aydin, H., & Zhu, D. (2013). Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints. *ACM Transactions on Design Automation of Electronic Systems, **18***(2), 1–21. https://doi.org/10.1145/2442087.2442094
69. Zhu, D., & Aydin, H. (2009). Reliability-aware energy management for periodic real-time tasks. *IEEE Transactions on Computers, 58*(10), 1382–1397. https://doi.org/10.1109/TC.2009.56
70. Zhu, D. (2010). Reliability-aware dynamic energy management in dependable embedded real-time systems. *ACM Transactions on Embedded Computing Systems, 10*(2), 1–27. https://doi.org/10.1145/1880050.1880062
71. Zhu, D., Qi, X., & Aydin, H. (2007). Priority-monotonic energy management for real-time systems with reliability requirements. In *2007 25th International Conference on Computer Design*. IEEE. https://doi.org/10.1109/iccd.2007.4601963
72. Zhuravlev, S., Saez, J. C., Blagodurov, S., Fedorova, A., & Prieto, M. (2013). Survey of energy-cognizant scheduling techniques. *IEEE Transactions on Parallel and Distributed Systems, 24*(7), 1447–1464. https://doi.org/10.1109/tpds.2012.20