The CSI Journal on Computer Science and Engineering

A 6-monthly publication of Computer Society of Iran (CSI)

The CSI Journal on Computer Science and Engineering

Vol. 13 **No. 2** **2016**

CONTENTS

The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 1-22
Regular Paper

An Energy-Optimal Real-Time Scheduling Algorithm for Unrelated DVS-Enabled Parallel Machines

Mahmood Gholipour[1] Mehdi Kargahi[1, 2] Heshaam Faili[1]
Shahbaz Youssefi[3] Hadi Ravanbakhsh[1]

[1]School of Eectrical and Computer Engineering, University of Tehran,Tehran, Iran
[2]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran. Iran
[3]Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genova, Italy

Abstract

The category of unrelated parallel machines is one of the most realistic models of heterogeneous multiprocessor systems. In this model, the pair of task and machine (processor) jointly specifies the required execution time and energy to complete the task. In this paper, we consider the problem of scheduling real-time tasks on a set of unrelated parallel machines with the capability of voltage selection through Dynamic Voltage Scaling (DVS). The tasks can migrate among the processors and each processor is also permitted to switch among a set of discrete voltage levels (and thus, task-specific speed levels) to minimize the overall system energy usage. A polynomial-time Energy-Optimal Real-Time Scheduling Algorithm (EORTSA) is proposed: At first, the problem of energy-optimal task-to-machine and voltage-level-to-task assignments are formulated as a linear program and used as a polynomial-time schedule ability test for periodic tasks. Second, task sets which are passed the schedule ability test are feasibly scheduled on the machines using a matching-based algorithm. We prove that the worst number of migratory tasks is 2m, where m is the number of machines. Also, the worst-case total number of migrations, preemptions and voltage-level switches is of $O(m^2)$ in each schedule e period, which is a period of time between two arbitrary consecutive task releases in the system. Comparisons with the PCG algorithm show up to 60% energy saving and reveal that EORTSA outperforms PCG in terms of average number of task migrations and preemptions, especially in systems with large number of tasks.

Keywords: Dynamic Voltage Scaling (DVS), Energy Optimization, Migration, Real-Time Systems, Scheduling, Unrelated Parallel Machines.

1. Introduction

Real-time embedded applications need predictable computing platforms and deep information about their behavior with respect to different patterns of usage to satisfy their time-sensitive requirements. Such applications include safety-critical decision making systems such as those related to earthquake, Air Traffic Control (ATC), transportation, militarism, and many others which mostly have real-time and distributed natures. Recent advances in multiprocessor and distributed system design (multi/many-core processors, MPSoCs, clouds, etc.) have resulted great speedups through the execution of different tasks in parallel. The parallel processing elements in the mentioned systems may have either equivalent capabilities (e.g., multi-core processors or homogeneous multi-server systems) or different capabilities and characteristics (e.g., asymmetric multiprocessors or heterogeneous multi-server systems with different services as well as cloud computing systems).

Even systems which are supposed to have homogeneous processors may actually be heterogeneous due to reasons such as process variation [1] and processor aging [2]. In

overall, the diversity of available computer systems which can be connected through interconnection networks, and the distributed nature of the above-mentioned applications, confirm that most systems are inherently heterogeneous. These heterogeneities may be observed in both the computing capability and power consumption of the processing elements, with an emphasis to the fact that one major concern in the design and management of such systems is their energy usage.

As energy is an expensive resource nowadays, many studies in the context of real-time systems have been concentrated on the methods of energy saving in centralized [3, 4, 5] and distributed [6, 7] real-time systems with the consideration of the fact that the performance of such systems should remain in an acceptable level. A well-known method to do such management is to use dynamic voltage scaling (DVS) [5], which is available in most modern microprocessors [8, 9].

Meanwhile, another important property of the heterogeneous distributed real-time systems is that at a specific voltage-level of a DVS-enabled processor, the speed and energy usage are task-specific, namely they depend on the task running on the processor. This property, which is valid for unrelated parallel machines (as defined formally in Section 2), is a realistic consideration for today's computing systems. Such systems may contain different processing units (e.g., graphics co-processors, math co-processors, CPUs with different architectures, etc.), where each unit may better work for specific tasks from both aspects of computation speed and power usage. (Throughout this paper, we use the two terms machine and processor interchangeably).

In the current study, we present a scheduling algorithm for a relatively complex multiprocessor system which reflects a general behavior of the distributed real-time systems mentioned above. We consider a distributed system with periodic real-time tasks, where the system processors are heterogeneous from both aspects of processing capability and power usage. Furthermore, we suppose that speed and energy usage at a specific voltage-level of the processor are task-specific.

We consider a multi-criteria optimization problem which within the real-time tasks should meet their deadline and the energy usage should be minimized. Also, some methods will be proposed to reduce the number of task migrations as well as preemption and voltage-level switches. To the best of our knowledge, we are unaware of any previous study on a similar problem with the properties stated in this paper.

The rest of this paper is organized as follows. The next section reviews some previous work on identical, uniform, and unrelated parallel machines. Then, we present a precise definition of the problem considered in the current study in Section 3. In Section 4, the concept of a schedule period is introduced to reduce the problem to an abstract version. Next, in Section 5, the proposed scheduling algorithm and the respective schedule ability test are described, whereas its optimality and correctnessare proved. In Section 6, we discuss our method to reduce the degree of migrations as well as the number of preemptions and voltage-level switches. Section 7 reports the simulation results and Section 8 concludes the paper and indicates some future works.

2. Background

In this section, we review the properties and the respective studies of three categories of multiprocessor systems, namely identical, uniform, and unrelated parallel machines.

2.1. Identical Parallel Machines

The set of machines which are all identical and have the same computational power is called a set of identical parallel machines. In 1969, Muntz and Coffman [10] presented a level algorithm for systems with two identical processors. A strict fairness constraint for such systems, called P-fair, and a scheduling algorithm based on P-fair has been presented by Baruah et al. [11]. According to this algorithm, the resources are given to the tasks in proportion to their weights. Due to its strict fairness, this algorithm needs to reschedule tasks many times, which results in many task switches and migrations, and reduces the efficiency of the algorithm. A similar idea, namely the Fluid schedule is also given by Holman and Anderson [12].

The Fluid schedule, which is an ideal policy, assumes that each task executes at a constant rate. This algorithm is almost inefficient due to its high number of rescheduling. However, in the study, the Stagger model is also proposed which improves the P-fair algorithm.

Dertouzos and Mok [13] presented a model for expressing the relation between the laxity and remaining computation time of a task, called Laxity and Computation plane (L-C plane). The laxity of a task is a measure of urgency, which is represented on the x-axis and its remaining computation time is shown on the y-axis. Cho et al.[14] extended the idea of representing tasks as tokens in the L-C plane and introduced Time and Local Execution Time Domain Planes (T-L planes), where time and the remaining execution time are represented on the x-axis and y-axis, respectively. They proposed Largest Local Remaining Execution Time First (LLREF) scheduling algorithm based on the T-L plane and proved that it is an optimal online scheduling algorithm for identical parallel machines.

Megel et al. [57] introduced a novel approach to decrease preemption and migration count in optimal global real-time scheduling on identical multiprocessors. Their approach is composed of an offline and an online step. In the offline step, a linear program decides about placing jobs on intervals. Scheduling of associated jobs within each interval is done dynamically by an online local scheduler. Their approach suffers from severe issues: the proposed mixed integer linear programming of offline part is computationally complex since the problem is solving in a hyper period and the number of MILP variables scale with the task count, processor count and length of the hyper period.

For the sake of energy efficiency, different approaches have been investigated which usually consider continuous voltage-levels for the identical machines. Lee [15], through Dynamic Voltage and Frequency Scaling (DVFS) in a lightly-loaded multi-core system with periodic tasks, achieved up to 64% energy saving comparing to some simple methods. Chen et al. [16] devised a 1.13-approximation algorithm for scheduling migratory periodic tasks on identical multiprocessors, assuming each processor can take arbitrary operating frequencies. Later, Chen and Kuo [17]

proposed an optimal algorithm for migratory tasks and a 1.412-approximation when for non-migratory tasks, considering that the tasks have different power characteristics. Also, Chen et al. [18] proposed an algorithm with a 1.283-approximation bound for energy usage to schedule periodic tasks over identical DVS-enabled multiprocessor, considering both dynamic and static sources of power consumptions. If the overhead of turning processors on and off is not negligible, though, their algorithm is within a 2-approximation bound.

2.2. Uniform Parallel Machines

A set of processors which are solely characterized by their computation speeds is called a set of uniform parallel machines. Therefore, a job that executes for t time units on a processor with the computation speed s performs $s \times t$ units of execution. Horvath et al. [19] extended the idea of the level algorithm proposed in [10] to uniform heterogeneous multiprocessors. Gonzalez and Sahni [20] proposed an offline optimal$O(n + m)$ scheduling algorithm with no more than $2(m - 1)$ preemptions, where n is the number of tasks and m is the number of processors.

Other works have also been done on the online scheduling. Hochbaum and Shmoys [21] presented a polynomial approximation algorithm to solve the minimum makespan problem on uniform heterogeneous multiprocessors. Baruah and Goossens [22] tried to adapt the Rate Monotonic (RM) scheduling algorithm [23] to find a static-priority online scheduling algorithm for uniform multiprocessors.

They also presented a RM-feasibility test for this problem, generalizing the RM-feasibility test of identical heterogeneous multiprocessor systems. In a different study, Funk et al. [24] presented a feasibility condition for periodic task sets to be executed on uniform heterogeneous multiprocessors. This condition (which is called FGB in [25]) is based on the Earliest Deadline First (EDF) scheduling algorithm [23].

In the same study, they also presented an EDF-feasibility test; however, no efficient algorithm was found to schedule the task sets satisfying the FGB condition. Chen and Hsueh [25] proposed an optimal online $O(n^2 \log n)$ scheduling algorithm called PCG with at most n rescheduling, where n is the number of tasks. In their study, T-L_{er} plane was introduced and a greedy algorithm was proposed. They also proved the optimality of their algorithm based on the FGB condition.

In regard of energy efficiency in uniform parallel machines, Funaoka et al. [54] proposed a real-time static voltage and frequency scaling (RT-SVFS) technique based on an optimal real-time scheduling algorithm (LLREF) by means of T-L Plane transformation, which is a technique to apply processor frequency scaling to LLREF scheduling. They proposed a uniform RT-SVFS algorithm and then extended it to an independent RT-SVFS for uniform platforms.

Later, in [55] they extended this work and proposed a real-time dynamic voltage and frequency scaling (RT-DVFS) techniques based on RT-SVFS. Despite the significant run-time overhead associated to it, RT-DVFS obtains better energy saving than RT-SVFS.

2.3. Unrelated Parallel Machines

In unrelated parallel machines, there is an execution rate s_{ij} associated with each task-processor pair (T_i, P_j), with the interpretation that task T_i performs $s_{ij} \times t$ units of execution if executed on processor P_j for t time units. Two common methods for solving such scheduling problems in parallel machines are task partitioning and global scheduling. In the former method, certain tasks are only allowed to be executed on certain machines and only partial migration is allowed. Using this type of static decision has some performance advantages such as reducing the migration overhead. For example, Yu and Prasanna [26] explored minimizing the energy usage of periodic real-time tasks on unrelated parallel machines by task partitioning and proposed a polynomial-time scheduling algorithm by relaxing integer linear programming. The method of task partitioning is generally sub-optimal. Due to the NP-hard nature of such scheduling problems in their general form [12, 27], it is convenient to propose heuristics or efficient algorithms for special forms of the problems. Accordingly, the latter method, namely global scheduling is desired in many cases to feasibly schedule the task sets, in which an important criterion is to keep low the number of migrations.

Lenstra et al. [28] presented a polynomial approximation algorithm for unrelated parallel machines to find a schedule that minimizes the makespan. They guarantee that it is not longer than twice the optimal value for the case where the number of processors is fixed. They also proved that no polynomial algorithm could achieve a worst-case ratio less than 1.5 unless P = NP. For the case of two processors, Gonzalez et al. [29] introduced a linear time algorithm to construct optimal scheduling which have at most two preemptions. Jansen and Porkolab [30] gave a fully polynomial-time approximation algorithm for the problem of scheduling tasks in the case where each task can be executed only on one processor. Srivastava [31] developed a tabu search heuristic for minimizing makespan in that problem. This algorithm produces near-optimal results with reasonable amounts of computational time for problems of reasonable size. Baruah [32] proposed a polynomial-time feasibility test for the global heterogeneous multiprocessor scheduling. They also presented a scheduling solution as a proof for correctness of the test; such schedule however suffers from high preemption and migration cost.

The lack of an optimal scheduling algorithm with reasonable cost motivated us to propose an optimal scheduling algorithm; however optimality is only one of aspect of our work and we also consider the energy optimality of scheduling algorithm. Recently, Cucu-Grosjean and Goossens [56] proposed two exact schedule ability tests for task-level fixed-priority and job-level fixed-priority earliest deadline first scheduler on specific tasks executing on unrelated parallel machines. Also, for a special case of heterogeneous parallel machines which includes two type of processors, Raravi et al. [58] proposed two scheduling algorithms with time complexity of $O(n\log(n))$ for implicit deadline sporadic task set. They consider that task migration is only possible among processors of the same type.

Many works have also been done on energy-constrained unrelated parallel machines. Hsu et al. [33] and Chen and Kuo [34] tried to find a solution for cost minimization of machines and allocation, respectively under the given timing

and energy usage constraints. Chu et al. proposed a near-optimal [35] and later an optimal [36] solution to the voltage-setup problem; their work is the assignment of voltage-levels to processors with the limitation that each processor assumes one voltage-level throughout the system lifetime. Luo and Jha [37] devised both static and dynamic variable voltage scheduling algorithms to minimize the energy cost of scheduling dependent periodic and aperiodic tasks on unrelated parallel machines.

Some studies consider discrete voltage-levels for non-migratory task, and thus, they use different approaches to solve a NP-Complete problem. Ding et al. [38] approached the problem using the ant colony optimization in order to obtain energy efficient scheduling of partially dependent tasks on DVS-enabled processors, even with coarse-grained voltage modes. Lin et al. [39] addressed the problem when systems in question work on rechargeable batteries. They solve the problem based on four heuristics including genetic algorithm and ant colony optimization.

Yang et al. [7] used a fully polynomial-time approximation scheme for such problems. Yu and Prasanna [26] approximated the energy efficiency when assigning non-migratory tasks to DVS-enabled processors. They formulated the problem as an extended generalized assignment problem and also hired a heuristic to approximately solve the NP-Complete problem. They used a model in which the DVS-enabled machines have discrete voltage-levels, where both speed and energy usage of machines depend on the voltage-levels as well as the tasks they are running.

The above studies, although valuable, suffer from at least one of the following problems: (i) They do not take the advantages of DVS for runtime energy management into account [35, 36], (ii) Machine speeds are available in a continuous spectrum [35, 36, 37], (although power-supply electronic may provide systems with continuous voltage spectrum in the future, it is a fact that most of the currently available DVS-enabled microprocessors (e.g. Trans meta, Intel, and AMD) use a few discrete voltage-levels [40]), (iii) Energy usage of a machine depends either only on its specification (speed and/or workload) or only on the task that is running, rather than both [35, 36, 37, 38, 34, 7, 39, 33], and (iv) No task migration is permitted [26, 35, 36, 37, 38, 34, 7, 39, 33].

Although task migration in unrelated parallel machines offers potential energy/performance gains, there are also high costs and complexities involved in the migration process. Hence, it has not attracted considerable attention from researchers. However, according to recent researches [41, 42, 43, 44], this cost has became reasonable and task migration in unrelated parallel machines has become a feasible option. We consider task migration to better utilize the available heterogeneity of the platform as well as to achieve better power-performance efficiency.

3. System Model, Problem Definition and Conceptual Solution[1]

In this section, we introduce the model of the system. Also, we present the precise definition of the problem under study a general view to the solution method.

3.1. Task Model

We consider n periodic tasks shown as the task set $T = \{T_1, \ldots, T_n\}$. Each task T_i, i=1,…,n is independent of the other tasks and is described as (p_i,e_i), where p_i is the task period, e_i is its execution requirements and the respective relative deadline is again p_i. Additionally, there is a dummy task represented by T_0 with an execution time and period equal to the sum of the processor idle times and hyper period respectively.

3.2. Processor Model

The system consists of m unrelated parallel machines shown as the set $M = \{M_1,\ldots,M_m\}$ with no resource contentions. Machine M_j, j=1,…,m has k_j voltage-levels as $V_j = \{V_{j1},\ldots,V_{jk_j}\}$. At the l^{th} voltage-level of M_j, namely V_{jl}, l=1,…,k_j, a maximum speed S_{jl}, $(S_{i1} < S_{i2} < \cdots < S_{ik})$ is achievable and a maximum power Pow_{jl}, $(Pow_{j1} < Pow_{j2} < \ldots < Pow_{jk_j})$ may be consumed. Based on the characteristics of the running task at voltage-level V_{jl}, the attainable speed and the respective power usage may be far less than S_{jl} and Pow_{jl}, respectively.

3.3. Task-Processor Model

As also indicated in the processor model above, we consider m unrelated parallel machines (note that uniform and identical parallel machines are special cases of unrelated parallel machines). Therefore, each task T_i, i=1,…,n if assigned to a processor M_j, j=1,…,m which works at the voltage-level V_{jl}, l=1,…, k_j, can be run with a speed of S_{ijl} which certainly is a fraction of $S_{jl}(0 < S_{ijl} \le S_{jl})$. Meanwhile, the task-specific power usage of processor M_j working at voltage-level V_{jl}, when task T_i is run upon which, will be Pow_{ijl}, which again is a fraction of Pow_{jl} $(0 < Pow_{ijl} \le Pow_{jl})$. In addition, Pow_{0jl} is the power usage of processor M_j when it is idle, or equivalently executes the dummy task T_0, at voltage-level V_{jl}. In this regards, if T_i runs for duration of length t on machine M_j working at voltage-level V_{jl}, the energy usage of the task-processor pair in the duration can be calculated as:

$$E_{ijl}(t) = Pow_{ijl} \times t \tag{1}$$

To obtain the task execution times at different voltage-levels of the unrelated parallel machines, we consider the following information. Source code of each task is compiled separately for every supported processor in the set of unrelated parallel machines. The generated binary codes will be executed on the corresponding processor, and thus, the worst-case number of CPU cycles is determined. For each voltage-level of the DVS-enabled processor, a similar scenario is followed. For each task, the processor with the longest execution, which is less than or equal to the task deadline, will be considered as the base processor. We consider the execution speed of the base processors as 1 and execution speed of the task on the other processors are normalized with respect to this execution time. In our notation, we refer to the task execution time on base

processor as task execution time e_i in our task model.

Moreover, each task can run on at most one machine at each instant of time and can arbitrarily migrate across different machines during its execution. Without loss of generality in the proposed algorithm, we assume negligible overheads for task migrations, preemptions, and voltage-level switches in the current study. By the way, we propose some techniques which significantly reduce the number of such events, and thus, reduce the respective overheads.

3.4. Problem Definition

The problem considered in this paper is to schedule a feasible task set T on a given set of machines M with the goal of minimizing the power consumption. We need to divide the task instances (jobs) into job slices. A job slice is defined by a vector $(T_i, M_j, V_{jl}, t_s, t_f)$, meaning that task T_i is scheduled on machine M_j working at voltage-level V_{jl} from time t_s to time t_f. Furthermore, it is considered that $[t_s, t_f)$ is a period where the task is executed uninterruptedly, i.e., with no task migration, preemption, or voltage-level switching during that period. Equivalently, we use $(T_0, M_j, V_{jl}, t_s, t_f)$ to denote that M_j is in idle mode in the period of $[t_s, t_f)$.

Let define the average system energy usage in a time interval $[a,b)$ as:

$$\overline{E(a,b)} = \frac{(\sum_{i=0}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k_j} Pow_{ijl} \times \hat{t}_{ijl})}{(b-a)} \quad (2)$$

where $\hat{t}_{ijl} \le (b-a)$ is the total length of time during time interval $[a,b)$ in which task T_i runs on machine M_j at voltage-level V_{jl}. The resulting schedule is to minimize the average system energy usage throughout the system lifetime. This can be achieved by minimizing the average system energy usage in a hyperperiod of length H, i.e., $(\overline{E(0,H)})$, where the resulting energy is shown as $\overline{E(H)}_{\min}$. This goal can be achieved by appropriate determination of the job slices along with the consideration of all other constraints.

3.5. Conceptual Solution

The proposed solution to the defined problem can be summarized as follows:

1. We first reduce the scheduling problem in a hyperperiod to scheduling the scaled tasks in Schedule Periods (SPs) of unit length (defined in Section 4) that is easier to solve.
2. Then, we model the scheduling problem in the SPs with Linear Programming (LP). If a solution to this LP exists, it specifies the required amounts of time that each task needs to be run at every voltage-level on each machine. These amounts of execution minimize the system energy usage in a manner that all deadlines are met. The algorithm for this step can be independently used for schedulability test of real-time periodic tasks on unrelated parallel machines and is explained in detail in Section 5.1.
3. Tasks are assigned to processors based on the results of Step 2. An extension to Lemma 2 of [32] is a solution for task scheduling; such a solution suffers from large number of preemptions, migrations and voltage-level switches. Hence, we use this solution only to prove the existence of a feasible schedule. Instead, we have proposed a scheduling algorithm which aims to reduce the number of migrations and preemptions. Our algorithm is composed of the following steps:
   - 3.1.1 Each SP interval is divided into several time intervals with different lengths. The procedure to obtain lengths of these intervals is explained in Section 5.
   - 3.1.2 For each interval, task to processor assignments is found by solving a custom model of minimum-cost network flow problem. This matching is selected in a way that it leads to significant decrease in the number of preemptions and migrations in the hyper period. Procedure from Steps 3.1.1 and 3.1.2 implicitly prevents the overlaps of execution of segments of a task on different processors.
   - 3.1.3 Number of migrations, preemptions and voltage-level switches is further reduced by use of mirroring technique which discussed in Section 5.2.3.

   3.2 Then, the remaining tasks, i.e., non-migratory ones, are scheduled on the machines. The main point about scheduling non-migratory tasks is that we are not concerned about the parallel execution of job slices of a task. Hence they can be scheduled using an algorithm such as EDF in the remaining free intervals of the machines which prevents excessive preemptions and voltage-level switches. Subsection 5.3 describes this step, which completes the solution.

4. Schedule Period and Feasible Scheduling

To address the aforementioned problem, in this section we introduce the concept of schedule period, which has been borrowed from Fluid scheduling method [12] and is used to convert the scheduling problem into an equivalent but simpler one. The idea of Fluid scheduling is to execute the tasks at a constant rate in a way that the tasks execution are finished exactly at their deadline [12]. In a T-L plane, as introduced in Section 2, tokens representing the tasks move over time. The trajectory of such a token during the execution of the respective task according to its Fluid schedule is called the task Fluid path.

Therefore, the Fluid path is simply a straight line between the coordinates (r, e) and $(r + p, 0)$ in the respective T-L plane, where r is the task release time, e is its execution requirement, and p is the respective relative deadline (see figure 1). The actual path is the result of concatenation of execution lines of the task. The slope of each execution line depends on S_{ijl} for task T_i when it is running on machine M_j at the voltage-level V_{jl}. This slope is zero when the task is not running on any machines.

We define a Schedule Period (SP) as a period between two arbitrary consecutive task releases in the system. Scheduling is done on a per SP basis. Consider n tasks $T_1, T_2, ..., T_n$ with their Fluid paths as shown in figure 2. In the time period between two task releases, namely in SP_k (with start time of s_k and finish time of f_k), each task T_i, according

to its Fluid schedule, executes for a certain amount of time, shown as e_i^k. This piece of a task is called a scaled task. Then the relative deadlines for the execution of all the scaled tasks located in SP_k are equal to the length of the respective SP (shown as $|SP_k|$ in this case). Accordingly, a new set of scaled tasks is resulted which we call it scaled task set. Thus, the scaled task set is shown as $\{ST_1, ST_2, ..., ST_n\}$ with each scaled task ST_i having an execution requirement of e_i^k and all the scaled tasks having a relative deadline of length $|SP_k|$.

It is proven in Theorem 1 below, which a task set is schedulable with average energy usage E, if and only if the respective scaled task set in a SP is schedulable with the same amount of average energy usage. Therefore, the problem is reduced to scheduling the scaled tasks in only one SP. The resulting schedule can then be used to schedule the scaled tasks in every SP as well as the task set in a hyper period, and as a result, throughout the system lifetime.

Theorem 1. A periodic task set is schedulable on a set of unrelated parallel machines with an average energy usage E, if and only if the respective scaled task set is schedulable in a SP with the same average energy usage on the same set of machines.

Proof. Suppose the scaled task set is schedulable in a SP, namely SP_k, with average energy usage E. Consider the tasks and their Fluid paths as in figure 2. For each task T_i, the slope of its Fluid path is e_i/p_i. In the schedule period SP_k, each scaled task ST_i has an execution requirement of:

$$e_i^k = |SP_k| \times e_i / p_i \tag{3}$$

In this regards, for another schedule period $SP_{k'}$ with the length $|SP_{k'}|$, we can scale the schedule obtained for SP_k by a factor of $|SP_{k'}|/|SP_k|$. Then we obtain a new schedule for $SP_{k'}$, within which each scaled task S T_i has a execution requirement of $e_i^{k'}$ as:

$$|SP_k| \times e_i/p_i \times |SP_{k'}|/|SP_k| = |SP_{k'}| \times e_i/p_i \tag{4}$$

In general, schedule scaling for SP_k by a factor of K means that corresponding to each job slice $(T_i, M_j, V_{jl}, s_k + t_s, s_k + t_f)$ with an average energy usage of $E(s_k + t_s, s_k + t_f) = Pow_{ijl}$ in the primary schedule, there is a job slice $(T_i, M_j, V_{jl}, s_{k'} + K \times t_s, s_{k'} + K \times t_f)$ in the scaled schedule with the average energy usage of $E(s_{k'} + K \times t_s, s_{k'} + K \times t_f) = Pow_{ijl}$. Clearly, the average energy usage does not change for any job slices in the scaled schedule and all job slices' lengths are scaled by the same factor K. Thus, the average energy usage of $SP_{k'}$ would be $\overline{E(s_{k'}, f_{k'})} = \overline{E(s_k, f_k)}$.

However, this new execution requirement (resulting from (4)) is the same as the execution requirement for task T_i in the schedule period $SP_{k'}$, i.e., $e_i^{k'}$. Therefore, to obtain a schedule for an arbitrary schedule period $SP_{k'}$ with length $|SP_{k'}|$, it is sufficient to multiply each job slice in the given scheduling of SP_k by $|SP_{k'}|/|SP_k|$.

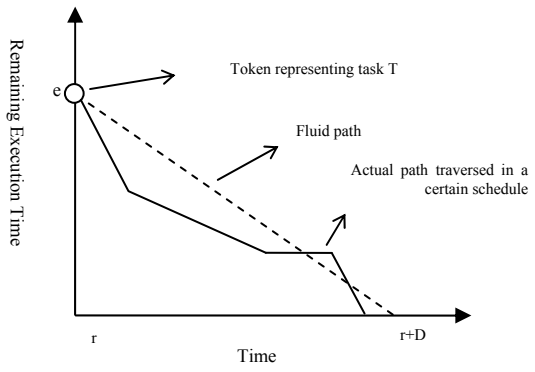


Figure 1. A sample T-L plane

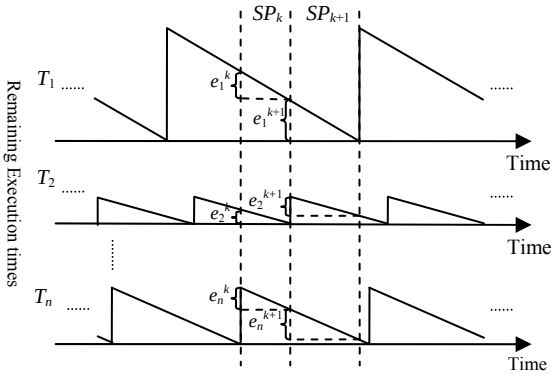


Figure 2. Schedule periods

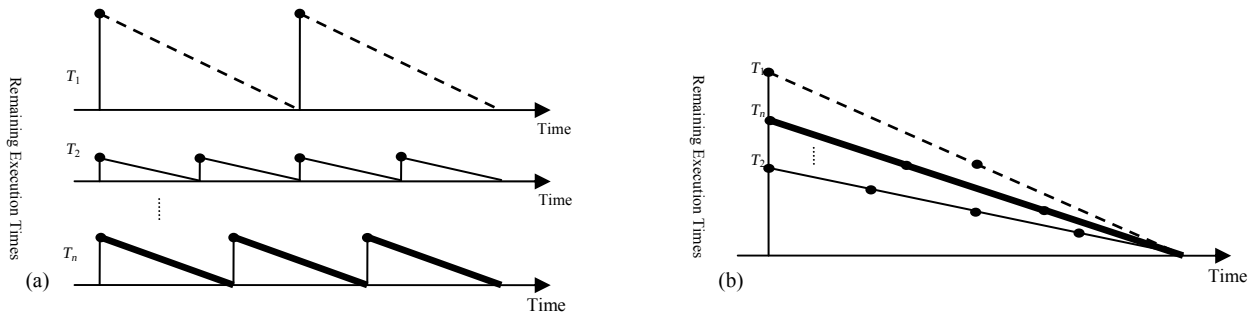


Figure 3. Relaxing the problem of scheduling in a SP to one hyper period: a) Fluid schedule of periodic tasks in one hyper period, b) Cumulated execution times of the tasks in the hyper period

In this way, the task set would be schedulable in all the SPs. Scheduling the scaled tasks in each SP results in each scaled task token in the respective T-L plane to meet the task's Fluid path at the end of the SP. Since all the scaled task deadlines are located at the end of their respective SPs, all deadlines are met by individual scheduling of each SP. Besides, average energy usage in all the SPs is the same and equal to E. Thus, the average energy usage in a hyper period would be the same. Thus, the periodic task set is schedulable with the average energy usage E.

To prove the theorem in the inverse direction, suppose the task set is schedulable. Consider the length of the tasks hyper period is H and the average energy usage is E. In this regards, task T_i is executed H/p_i times in the hyper period, each with the length of e_i. Therefore, in each hyper period, task T_i has a cumulative execution requirement of length $H \times e_i / p_i$. The problem of scheduling tasks with $H \times e_i / p_i$, i=1,…,n execution requirements, all having a relative deadline equal to H, is a relaxed version of the original problem (see figure 3).

In other words, we can scale down the scheduling in a hyper period by factor $|SP_k|/H$ to address the problem of scheduling the respective scaled task set in SP_k for an arbitrary value of k. The reason is that each task T_i must be executed for $|SP_k| \times e_i / p_i$ which is equal to $H \times e_i / p_i \times |SP_k|/H$. Besides, the average energy usage remains equal to E, because scaling a scheduling does not change its average energy usage. Thus, the proof is complete.

Let $\overline{E(SP)}_{\min}$ show the minimum value of the average energy usage of the scaled task set in a SP that can be obtained through proper scheduling. Then we have:

Corollary 1. The minimum average energy usage for a set of periodic tasks on a set of unrelated parallel machines is E if and only if the minimum average energy usage for scheduling the corresponding scaled task set in an arbitrary SP (e.g., SP_k) on the same set of machines is E, or equivalently, $\overline{E(SP)}_{\min} = \overline{E(H)}_{\min}$.

Proof. We use contradiction for the proof. First, consider that there is a scheduling with an average energy usage $\overline{E(SP)}_{\min}$ ($< \overline{E(H)}_{\min}$) for the scaled task set in a SP. However, according to Theorem 1, we were able to schedule the periodic task set with the average energy usage $\overline{E(SP)}_{\min}$, which is in contradiction to the optimality of $\overline{E(H)}_{\min}$. Similarly, for the inverse direction, consider that there is a scheduling with average energy usage $\overline{E(H)}_{\min}$ ($< \overline{E(SP)}_{\min}$) for the periodic task set. Again, according to Theorem 1, we were able to schedule the scaled task set with the average energy usage $\overline{E(H)}_{\min}$ in the SP, which is in contradiction to the optimality of $\overline{E(SP)}_{\min}$.

In the following, we provide a scheduling algorithm for the scaled task set in a SP of unit length to minimize the energy usage. Then we use the obtained schedule to construct a global schedule with minimal average energy usage for the task set.

5. Energy-Optimal Task Scheduling on Unrelated Parallel Machines

The proposed Energy-Optimal Real-Time Scheduling Algorithm (EORTSA) has three steps. First, to assure that all the deadlines can be met and the energy usage is minimized, we find the required amounts of time that each task needs to be run on each machine, as well as the length of time that it needs to spend at each voltage-level on that machine. The algorithm for this step, explained in detail in Section 5.1, can be independently used for schedule ability test of real-time periodic tasks on unrelated parallel machines. Second, we schedule the tasks which need migration, namely migratory tasks, on the appropriate machines. The details of this step are discussed in Section 5.2. Finally, the remaining tasks, i.e., non-migratory ones, are scheduled on the machines determined in the first step, only in the remaining free intervals of the machines, which completes the solution as discussed in Section 5.3.

5.1. Schedule Ability Test

In this subsection, we use linear programming to assign the system tasks to the unrelated parallel machines and set their speeds in a manner that the average system energy usage is minimized and all the deadlines are met. This assignment can then be used as a schedule ability test for the system. Given n tasks and m machines, the amounts of time that each task T_i, i=1,…,n needs to be run on machine M_j, j=1,…,m, at voltage-level V_{jl}, l=1,…,k_j, namely t_{ijl} in a SP of unit length is derived. The objective function here is to minimize the average energy usage in a SP, which is equivalent to minimizing the total energy usages of the machines (based on Corollary 1). Thus, the optimization problem is formulated as follows (notice that the length of SP is considered 1):

Minimize

$$\overline{E} = \sum_{j=1}^{m} \left(\sum_{i=0}^{n} \left(\sum_{l=1}^{k_j} Pow_{ijl} \times t_{ijl} \right) \right) \quad (5)$$

Subject to

$$\sum_{j=1}^{m} \sum_{l=1}^{k_j} s_{ijl} \times t_{ijl} = e_i / p_i, \quad \text{i=1,...,n}$$

$$\sum_{i=0}^{n} \sum_{l=1}^{k_j} t_{ijl} = 1, \quad \text{j = 1,...,m}$$

$$\sum_{j=1}^{m} \sum_{l=1}^{k_j} t_{ijl} \le 1, \quad \text{i = 1,…,n}$$

$$t_{ijl} \ge 0, \quad \text{i = 1,...,n, j = 1,...,m, l = 1,...,}k_j$$

The first constraint above indicates that the total normalized execution time of task T_i with respect to the speed levels of the machines at different voltage-levels should be equal to its normalized expected execution requirement. The second constraint indicates that the idle times of machine M_j and the total execution time of the job slices assigned to that machine should be equal to the length of the respective SP. Also, the third constraint mentions that the total execution time of task T_i on different machines

working at different voltage-levels should not violate the scaled task deadline, which is again the length of the SP (i.e., the task cannot be executed for longer than the SP length). The last constraint simply indicates that all the times are nonnegative. Solving this LP problem, the t_{ijl}s, i = 1,…,n, j = 1,…,m, l = 1,…,k_j, are obtained which specify how long each task T_i should be executed on machine M_j at voltage-level V_{jl} to minimize the overall system energy usage.

In other words, we model the energy-optimal scheduling as a LP problem. This LP model could be used as a schedule ability test as follows. If there is not any solution for scheduling the given task set on the given set of parallel machines, then the LP problem does not have any solution. In addition, it is guaranteed that if the task set is schedulable, then the LP problem certainly has a solution.

According to the optimization, if there are two voltage-levels l and l' for machine M_j in an order that $t_{0jl} > 0$ and $t_{0jl'} > 0$ while $Pow_{0jl'} \geq Pow_{0jl}$, the energy usage can be reduced by setting t_{0jl} to $t_{0jl} + t_{0jl'}$ and $t_{0jl'}$ to zero, which has no effect on the satisfaction of the other constraints. Applying this method iteratively, we can reach to a situation in which there are no two voltage-levels for a machine M_j so that $t_{0jl} > 0$ and $t_{0jl'} > 0$, and therefore, machines use only one voltage-level when they are idle. Example 1 illustrates how this optimization works.

Example 1: Suppose a machine set $M = \{M_j \mid 1 \leq j \leq 4\}$, where M_1 and M_4 have two voltage-levels and the other machines work in only one voltage-level. The execution requirements and periods of the tasks of $T = \{T_i \mid 1 \leq i \leq 7\}$ are listed in table 1. Given the task power consumptions (Table 2) and execution speeds (Table 3) on different voltage-levels, the non-zero variables have been obtained by solving the LP, as shown in table 4.

Corresponding to each $t_{0jl} > 0$ $(i > 0)$, we define a task segment (TS_{ijl}) of length t_{ijl}, with the interpretation that there is a segment of task $T_i$to be executed upon machine M_j at voltage-level V_{jl}. For the SP under discussion, we divide these segments into two categories: segments which belong to the tasks running over more than one machine in that SP, referred to as segments of migratory tasks, and segments running over only one machine during the SP, referred to as segments of non-migratory tasks.

In Sections 5.2 and 5.3, respectively, scheduling of migratory and non-migratory task segments over the processors is discussed.

Table 1. Execution requirements and periods of tasks

Task Characteristics	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_7**
E	40	90	40	20	75	8	10
P	40	100	50	25	100	10	12

Table 2. Power usage of different tasks on different voltage-levels

Voltage-level	**T_0**	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_7**
V_{11}	1	5	7	2	2	2	3	3
V_{12}	2	7	9	3	4	4	6	5
V_{21}	1	2	4	6	3	4	5	3
V_{31}	1	5	6	2	2	2	11	5
V_{41}	1	2	2	2	2	2	4	4
V_{42}	2	4	5	5	5	3	6	6

Table 3. Execution speed of different tasks on different voltage-levels

Voltage-level	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_7**
V_{11}	1	0.5	1	0.5	0	1	0.5
V_{12}	2	1	2	1	0	2	1
V_{21}	0.5	2	0	0	0	2	2
V_{31}	2	1	0.5	2	2	0	0.5
V_{41}	0	1	1	0	0	0.5	1
V_{42}	0	2	2	0	0	1	2

Table 4. List of non-zero variables at optimal poin

Variable	**Value**	**Type**
t_{112}	$0.16\overline{904761}$	Segment of a migratory task
t_{121}	$0.4\overline{238095}$	Segment of a migratory task
t_{131}	0.225	Segment of a migratory task
t_{241}	0.9	Segment of a non- migratory task
t_{312}	0.35	Segment of a migratory task
t_{341}	0.1	Segment of a migratory task
t_{431}	0.4	Segment of a non- migratory task
t_{531}	0.375	Segment of a non- migratory task
t_{611}	$0.4\overline{809523}$	Segment of a migratory task
t_{621}	$0.15\overline{952380}$	Segment of a migratory task
t_{721}	$0.41\overline{6}$	Segment of a non- migratory task

5.2. Scheduling Migratory Tasks

In this subsection, we discuss the method of scheduling segments of migratory tasks in a SP of unit length. Without loss of generality, we perform the scheduling in the interval [0,1). All the segments of migratory tasks, i.e., TS_{ijl}s should be scheduled on their corresponding machines (M_j) with no parallel execution of different job slices of the same task. This scheduling is done through an iterative algorithm, namely Algorithm 1, as shown in Algorithm 1.

Algorithm 1. Scheduling algorithm for migratory tasks

1.	For each t_{ijl} ; i=1,…,n;j =1,…,m; l=1,…,k_i
1.1.	if T_i is a migratory taskthent'$_{ijl}$=t_{ijl}
1.2.	else t'$_{ijl}$=0
2.	$\gamma = 1$
3.	prevX = ∅
4.	nextX = ∅
5.	while$\gamma > 0$
5.1.	U = {urgenttasks}
5.2.	F = {fullmachines}
5.3.	if (U ≠ ∅ or F ≠ ∅)
5.3.1	nextX= matching (prevX)
5.4.	determine value of σ// σ is the time that is scheduled in this iteration
5.5.	for each $(T_i, M_j) \in$ nextX
5.5.1	an arbitrary task segment TS_{ijl} is scheduled in the interval of $[\gamma-\sigma,\gamma)$
5.5.2	t'$_{ijl}$ = t'$_{ijl}$– σ
5.6.	prevX = nextX
5.7.	$\gamma = \gamma - \sigma$

In the beginning of each iteration, all machines have idle periods in the range of $[0,\gamma)$. In each iteration, some part of scheduling is done in the time interval $[\gamma-\sigma,\gamma)$, where the length of such interval, i.e. $\sigma > 0$, is specified through the determination of some task-machine pairs and their respective voltage-levels. Before going further, we introduce some notations and terminologies. Having the value of γ at the beginning of an iteration and t'_{ijl}s as defined in Algorithm 1, we define machine M_j as full if

$$\sum_{i=1}^{n}\sum_{l=1}^{k_j} t'_{ijl} = \gamma \quad (6)$$

which means machine M_j should execute migratory tasks for all the remaining time and has no other time to be idle or execute non-migratory tasks during the interval $[0,\gamma)$. We use F to refer to the set of such full machines. Similarly, we call a migratory task T_i as urgent if

$$\sum_{j=1}^{m}\sum_{l=1}^{k_j} t'_{ijl} = \gamma \quad (7)$$

namely, task T_i should be executed continuously in the remaining period $[0,\gamma)$ in order to be completed on time. We use U to refer to the set of urgent tasks. In each iteration, full machines and urgent tasks are selected to be scheduled. This decision is made to prevent unnecessary preemptions and migrations. For this purpose, in each iteration, we need to find a bijective function $\chi: D \rightarrow R$, where D and R are set of tasks and machines, respectively, satisfying conditions $U \subseteq D \subseteq T$ and $F \subseteq R \subseteq M$. More precisely, χ is a matching that maps each urgent task to a (possibly full) machine and some other tasks to the remaining full machines.

For each $(T_i, M_j) \in \chi$, there is at least one segment of a migratory task TS_{ijl} while $t'_{ijl} > 0$. This means that task T_i still must be executed on machine M_j for some amounts of time. Our proposed approach for finding the matching will be discussed in detail in Subsection 5.2.1.

Two approaches are followed in the matching algorithm to reduce the number of migrations, preemptions, and voltage-level switches:

1. In each iteration, to find the matching χ, the algorithm gets feedback from the task-to-processor assignments of the previous iteration. In other word, we intend to select the same edges that have been selected in previous iteration, or equivalently, assign tasks to the same processors to which they have been assigned in the previous iteration. This strategy leads to improvement in the number of migrations and preemptions as well as voltage-level switches.
2. We postpone scheduling tasks and processors as late as their equivalent nodes become critical. In each iteration, critical nodes must participate in scheduling, or otherwise, corresponding tasks or processors to critical nodes cannot satisfy their timing constraints, namely constraints (5.1) and (5.3). By postponing task scheduling, idle time segments of processors in schedule period, which are used to schedule non-migratory tasks, will connect together, leading to minimum preemption and voltage-level switches for non-migratory tasks.

The first two lines of Algorithm 1 force the schedule to be composed exclusively from migratory task segments. In each iteration, prevX contains the edges from the previous iteration and nextXholds the current set of edges. Each edge in the sets indicates the mapping of one task from migratory task set to one processor from the processor set. At the beginning of each iteration, current full machines and urgent tasks is identified. Then, if there is no urgent task or no full machine, the task to processor assignment is postponed as late as possible. Otherwise, based on Algorithm 2, a task to processor assignment with the minimum possible changes (with respect to the immediate previous iteration) will be obtained.

After finding χ, for each $(T_i, M_j) \in \chi$, an arbitrary task segment TS_{ijl} is scheduled in the interval $[\gamma-\sigma,\gamma)$, i.e., task T_i is scheduled on machine M_j at voltage-level V_{jl} from γ – σ to γ. σ defines the interval length, and respective procedure to calculate it is discussed in Section 5.2.2. At the end of each iteration, we change the remaining range of scheduling by setting γ to γ – σ; decrement each t'$_{ijl}$ by σ for the scheduled task segments TS_{ijl}s; and replace the set of edges used in the previous iteration with the set of edges obtained in the current iteration to determine the matching χ in the next iteration. Iterations will be stopped when γ becomes zero. Based on this approach, we select TS_{ijl} which was scheduled in the immediate previous iteration. This strategy leads to less preemptions, migrations and voltage-level switches. Next section describes the implementation of the approach used to find the matching.

5.2.1. Finding the Tasks to Machines Assignment in Each Iteration

To find task to machine assignment (called matching χ in

Algorithm 1), in each iteration, we define a graph with m′vertices for full machines and the machines that are adjacent to urgent tasks and n′ vertices for the urgent tasks and adjacent tasks to full machines. We refer to the vertices corresponding to full machines as well as urgent tasks as critical vertices. In the graph, vertex of each task T_i is adjacent to vertex of machine M_j if there is a segment of a migratory task TS_{ijl} for some values of l while $t'_{ijl} > 0$. Therefore, matching functionχ corresponds to a minimum weighted bipartite matching which covers all critical vertices on the resulting bipartite graph by using as much edges as possible from previous iteration matching. Now, we describe the procedure to find this matching.

First, the adjacency matrix for migratory tasks and machines is initialized. Using the adjacency matrix, a bipartite graph is built which its vertices are full machines and machines which are adjacent to urgent tasks in one part and urgent tasks and adjacent tasks with full machines at the other part. Then, in Lines 13 and 14 from Algorithm 2, we add edges to set E if there is adjacency between the corresponding vertices. Then a network flow graph will be built from the bipartite graph. For this purpose, two additional nodes are inserted in the bipartite graph node set. The first node, namely the source node will be connected to all task nodes. Similarly, the second node, i.e. the sink node will be connected to all processor nodes. Positive supply value of 1 will be assigned to the nodes which fall in the category of urgent tasks (supply node).

Also, negative supply value of 1 is assigned to each full machine nodes (demand nodes). To keep our special minimum cost network flow model solvable, the sum of supply and demand flows must remain equal. In this regard, additional flow, which is equal to the difference of supply flows and demand flows, will be added to the source or sink nodes. All arcs have capacity of one except arcs between source node and urgent task nodes and arcs between full machine nodes and the sink node, which their capacity is set to 0. Arcs which are connected to source and sink have the cost of 1. The costs of other arcs are determined by the algorithm presented in figure 6.

This algorithm assigns cost a to arcs which are selected in the previous iteration and have the most importance. It assigns cost b to second most importance arcs, the arcs for which one of its adjacent nodes belongs to category of urgent tasks and the other is a full machine. The costs of all remaining arcs which have the least importance are set to c. Although the exact values of a, b, and c depend on the size of the problem, but the relation $a < b < c$ holds among them. These values should be selected in such a way that enables the matching to select as much as possible arcs with higher priority while finding the minimum total cost. Obviously, it is possible to leave some arcs with costs of a or b uncovered in the matching due to the problem constraints which prevent overlapped execution of difference segments of each task.

As shown in Algorithm 3, cost function is used for assigning edge cost and is computed according to the set of input parameters including E, U, F, and prev X. By solving minimum cost network flow problem, if flow of every urgent task and full machine nodes are zero, then it represents a matching that covers all urgent task and all full machine nodes. Otherwise, in each iteration, flow value of source node will be incremented by 1 and flow value of sink node will be decremented by 1. This loop will be terminated by finding a matching that covers all the full machines and urgent tasks. In such a matching, flow of every node equals to zero. Proof of the fact that such matching always exists and so the loop will be stopped eventually is an extension to what proposed in [32] and is described below.

Algorithm 2. Matching χ

1.	Matching χ :
2.	{
3.	Input: prevX
4.	Output: X // a set of edges
5.	Matrix $T_{n' \times m}$ // adjacency matrix where n′and m are the numbers of migratory tasks and processors
6.	If $\sum_{l=0}^{k} t'_{ijl} > 0$ then T(i, j)=1
7.	Else T(i, j)=0
8.	// create a bipartite graph
9.	VT = {V_{T_i} | $T_i$isurgenttaskoratask thatadjacenttoatleastonefullmachi
10.	VP = {V_{P_j} | $p_j$isfullmachineoraprocessorthatadjacenttoatleastoneur
11.	V = VT ∪ VP // a set of vertex
12.	E=∅ ; // a set of edge
13.	For $T_i \in U$
13.1.	For all P_j Processors
13.1.1.	IfT(i, j)=1 then $E = E \cup \{(V_{T_i}, V_{P_j})\}$ and Capacity $((V_{T_i}, V_{P_j}))=1$
14.	For $p_j \in F$
14.1.	For all T_i tasks
14.1.1.	If T(i, j)=1 then $E = E \cup \{(V_{T_i}, V_{P_j})\}$ and Capacity$((V_{T_i}, V_{P_j}))=1$
15.	// convert bipartite graph to network flow
16.	$V = V \cup \{V_S\} \cup \{V_d\}$; // add source and sink nodes to graph
17.	For all V_{T_i} : $E = E \cup \{(V_S, V_{T_i})\}$
17.1.	If $(V_{T_i} \notin U)$ Capacity$(V_S, V_{T_i}) = 1$
18.	For all V_{P_i} : $E = E \cup \{(V_{P_i}, V_d)\}$
18.1.	If $(V_{P_i} \notin F)$ Capacity$(V_{P_i}, V_d) = 1$
19.	// Edge cost definition
20.	Assign edge cost // cost function in Algorithm 3
21.	IF$(V_{T_i} \in U)$ Flow$[V_{T_i}]$=1
22.	IF$(V_{P_i} \in F)$ Flow$[V_{P_i}]$=-1
23.	IF(|U|> |F|) Flow$[V_d]$=|F|-|U|
24.	Else Flow$[V_S]$=|F|-|U|
25.	$E' = \emptyset$; // a set of edge
26.	while (true)
26.1.	$E' = Solution(Flow, V, E, C)$ // Solution function returns set of edges of the minimum cost network flow
26.2.	If ($\forall t_i \in U$, exist one edge $e=(V_{T_i}, V_{P_j}) \in E'$) And ($\forall p_j \in F$, exist one edge $e=(V_{T_i}, V_{P_j}) \in E'$)
26.2.1.	X={ (V_{T_i}, V_{P_j}) | $(V_{T_i}, V_{P_j}) \in E'$}
26.2.2	Return X
26.3	Else
26.3.1.	Flow$[V_S]$++
26.3.2.	Flow$[V_d]$--
27.	}

Algorithm 3. Cost Function

1.	Cost Function :
2.	Input : E, U, F, prevX
3.	Output: C[ ] // The matrix C indicates the cost of edges
4.	For each $(V_{T_i}, V_{P_j}) \in E$
4.1.	If $(V_{T_i}, V_{P_j}) \in$ prevX then $C(V_{T_i}, V_{P_j}) = a$
4.2.	Else if $(T_i \in U)$ and $(P_j \in F)$ then $C(V_{T_i}, V_{P_j}) = b$
4.3.	Else if $(T_i \in U)$ then $C(V_{T_i}, V_{P_j}) = c$
4.4	Else if $(P_j \in F)$ then $C(V_{T_i}, V_{P_j}) = c$

As we know, a matching which covers all critical vertices can be obtained through the following steps:

- Find a matching from all urgent tasks to machines, namely a bijection $\chi_T: U \to R_U, R_U \subseteq M$, using standard

matching algorithms for bipartite graphs,

- Find a matching from all full machines to tasks, namely a bijection $\chi_M: F \rightarrow R_F, R_F \subseteq T$, using standard matching algorithms for bipartite graphs,
- Mixing χ_T and χ_M using the algorithm shown in Algorithm 2 (known as Algorithm 2) to obtain a matching that covers all critical vertices.

Algorithm 4. Finding χ Using χ_T and χ_M

1.	$\chi = \phi$; // Domain(χ) = Range(χ) = ϕ
2.	For each $T_i \in U - Domain(\chi) - Range(\chi_M)$
2.1.	Add $(T_i, \chi_T(T_i))$ to χ
2.2.	Remove $(\chi_T(T_i), X)$ from χ_M, if there exists
3.	For each $(X,Y) \in \chi_M$
3.1.	Add (Y, X) to χ

It can be shown that for each arbitrary $S \subseteq U$, $|N(S)| \geq |S|$, where N(S) is the set of machines, which their corresponding vertices are neighbors of vertices of S in the bipartite graph. Thus, Hall's condition [45] holds and therefore χ_T and χ_M exist. In order to prove that $|N(S)| \geq |S|$, we consider the set of all the remaining parts of task segments TS_{ijl}, i.e. t'_{ijl}, so that $T_i \in S$. Since each $T_i \in S$ is urgent, we can conclude:

$$|S| \times \gamma = \sum_{T_i \in S} (\sum_{M_j \in N(S)} \sum_{l=1}^{k_j} t'_{ijl}) \quad (8)$$

Also, it is trivial that

$$\sum_{T_i \in S} (\sum_{M_j \in N(S)} \sum_{l=1}^{k_j} t'_{ijl}) \leq \sum_{M_j \in N(S)} (\sum_{T_i \in T} \sum_{l=1}^{k_j} t'_{ijl}) \quad (9)$$

Considering that no machine can execute tasks with total execution times greater than γ, we can conclude:

$$\sum_{M_j \in N(S)} (\sum_{T_i \in N(N(S))} \sum_{l=1}^{k_j} t'_{ijl}) \leq |N(S)| \times \gamma \quad (10)$$

From (8), (9), and (10), we find that $|N(S)| \geq |S|$.

To mix χ_T and χ_M, in each iteration of Line 2 of Algorithm 4, an urgent task and its paired machine in χ_T, namely $(T_i, \chi_T(T_i))$, is added to χ, and therefore, T_i is added to Domain(χ) and a task will be removed from U–Domain(χ) –Range(χ_M) for the next iteration of the algorithm. Moreover, considering a task X, removing $(\chi_T(T_i), X)$ from χ_M, if it exists, removes X from Range(χ_M). If X is an urgent task, anew member is added to U–Domain(χ)–Range(χ_M) for the next iteration. This ensures that we do not ignore the urgent task X and thus, this task will be checked in further iterations.

In Line 3 of Algorithm 4, full machine vertices are covered either in χ or in χ_M. It can be described as follows: considering a machine X, if pair (X, Y), where Y is a task, is removed from χ_M, a pair (Z, X), where Z is also a task, is added to χ. Furthermore, no urgent task vertex remains uncovered in χ unless it has been covered in χ_M owing to the condition of $T_i \in U - Domain(\chi) - Range(\chi_M)$ in line 1 of Algorithm 2. Therefore, through adding task-machine pairs of χ_M to χ, we can claim that χ covers all full machine as well as all urgent task vertices.

For clarifying the matching algorithm, figure 4 displays snapshots of execution of matching for Example 1 at the end of fifth and sixth iterations. Red nodes represent critical nodes. Each edge is labeled 'X/Y' where X and Y denote the capacity and cost of the edge, respectively. Edges which are chosen to carry flow are drawn in red. The set of red edges between machines and tasks, which are passed across an ellipse as well, compose our desired matching.

As it has shown in figure 4, after selecting the edges (T_1, P_2), (T_3, P_4), (T_6, P_1) as a final solution in iteration 5, the cost of edges (T_3, P_4), (T_6, P_1) has decreased to increase the probability of their selection in the current iteration. It should be noted that selecting the edges which were selected in the previous iteration leads to lower number of preemptions, migrations and voltage-level switches.

5.2.2. Finding the Value of δ

The remaining part of the scheduling for an iteration of Algorithm 1 is to specify the length of time in which this part of scheduling is done. Based on the work in [32], value of σ is the largest value satisfying the following three conditions:

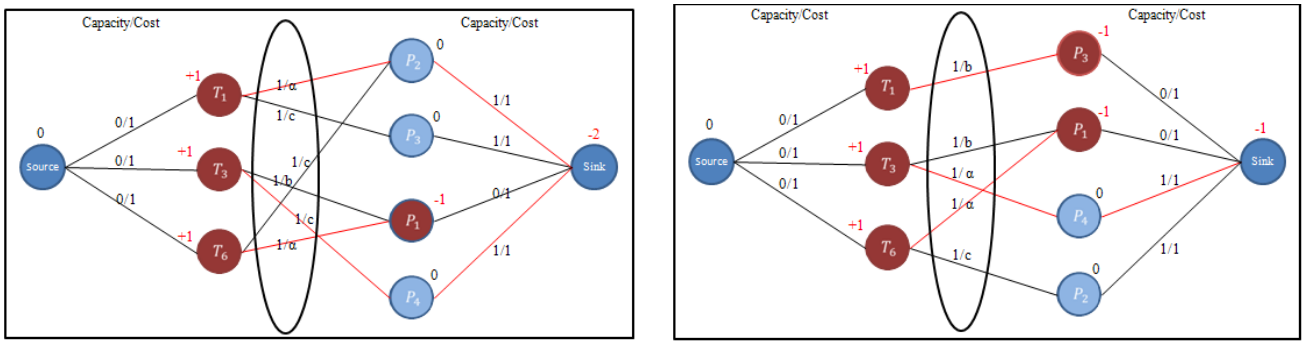
1. For each assigned task segment TS_{ijl}, we need to have:

$$\sigma \leq t'_{ijl} \quad (11)$$

This ensures that T_i has enough work to be done on M_j at voltage-level V_{jl} for σ amounts of time in $(\gamma - \sigma, \gamma)$ and

$$t'_{ijl} \geq 0 \quad (12)$$

holds for all task segments in the next iteration.



(a) Iteration 5

(b) Iteration 6

Figure 4. Snapshots of matching algorithm (iterations 5 and 6 from example 1)

2. For each migratory task T_i that have not been mapped, we need

$$\sigma \le \gamma - \sum_{j=1}^{m} \sum_{l=1}^{k_j} t'_{ijl} \quad (13)$$

This condition ensures that T_i remains non-urgent throughout interval $(\gamma - \sigma, \gamma)$ and

$$\sum_{j=1}^{m} \sum_{l=1}^{k_j} t'_{ijl} \le \gamma \quad (14)$$

holds for T_i in the next iteration, i.e., the required amounts of time for executing the remaining parts of task segments of T_i are available.

3. For each machine M_j which have not been allotted, we need

$$\sigma \le \gamma - \sum_{i=1}^{n} \sum_{l=1}^{k_j} t'_{ijl} \quad (15)$$

This ensures that M_j remains non-full throughout interval $(\gamma - \sigma, \gamma)$ and

$$\sum_{i=1}^{n} \sum_{l=1}^{k_j} t'_{ijl} \le \gamma \quad (16)$$

holds for M_j in the next iteration, i.e., in the iteration that the required amounts of time for executing the remaining parts of segments of migratory tasks on M_j are available.

In other word, scaled tasks can be scheduled on the allocated machines unless one of the following Types of Changes occurs in the status of the system:

- Type A, shown as A(TS_{ijl}): A task segment TS_{ijl} is assigned to the proper machine at the proper voltage-level for the length of t'_{ijl},
- Type B, shown as B(M_j): A machine M_j becomes full, which is equivalent to adding a critical vertex corresponding to the machine in the bipartite graph,
- Type C, depicted as C(T_i): A task T_i becomes urgent, which is equivalent to adding a critical vertex corresponding to the task in the bipartite graph.

At the end of each iteration of Algorithm 1, at least one type of the changes occurs. Thus, we need to find a new set of task segments to be scheduled for the next iteration.

It should be noted that when a change of Type A (e.g., A (TS_{ijl})) occurs as the sole change at the end of an iteration, if there is a task segment $TS_{ijl'}$ the previous matching can be used with only a voltage-level switching, e.g. from V_{jl} to $V_{jl'}$, for the new iteration. Doing so, we can effectively reduce the number of preemptions. However, if there is no other task segment $TS_{ijl'}$, an edge is removed from the bipartite graph and we need to find a new matching.

When t'_{ijl} is decremented by the amount of σ for each assigned task segment TS_{ijl}, it can be concluded for each migratory task T_i that

$$\sum_{j=1}^{m} \sum_{l=1}^{k_j} S_{ijl} \times t'_{ijl} = e'_i \quad (17)$$

where e'_i is the remaining execution requirements of task T_i for the next iteration. In addition, with this reduction in the values of t'_{ijl}s, it is trivial that (14) also holds for urgent tasks and (16) holds for full machines. In fact, (12), (14), (16), and (17) are corresponding to Constraints (5.4), (5.3), (5.2) and (5.1), respectively. This suggests that, in each iteration, it is ensured that all corresponding constraints to primitive constrains on migratory tasks and machines are held. Since at least a Change of Type A, B or C occurs at the end of each iteration, Algorithm-I terminates because of finiteness of the number of thesechanges. Next,in the last iteration, the remaining execution requirements of each migratory tasks gets zero. The reason is that, according to (12), (14) and (16), all t'_{ijl}s eventually turn into zero. Thus, it is verified that Algorithm 1 schedules the segments of migratory tasks in a proper manner.

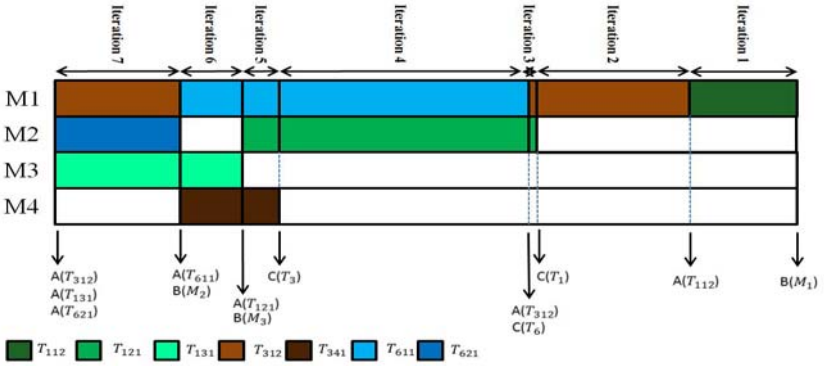


Figure 5. Assigning migratory tasks in the boundaries of a SP

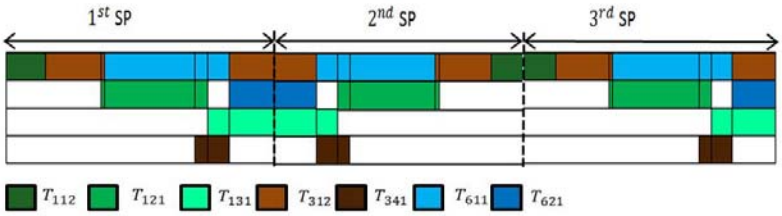


Figure 6. Migratory task segment placement

Example 2. After obtaining the task segments in Example 1, here figure 5 shows how segments of migratory tasks are assigned to the parallel machines in the boundaries of a SP. The change types occurring at the end of iterations are also shown.

5.2.3. Schedule Period Placement

In the next step, we schedule migratory tasks in entire hyper period. For further improvement in the number of migration and preemption and voltage-level switches, algorithm 5 as shown in Algorithm 5 is proposed.

Algorithm 5. Schedule Period Placement

```
1.      SP_U: Schedule set for a SP with unit length
2.      for i = 1 to N  // N=number of all SPs in hyperperiod
2.1.    if (i is Odd)
2.1.1.  SP'_U : mirror(SP_U)
2.1.2.      Scale SP_U to SP_i
2.2     else
2.2.1.      Scale SP_U to SP_i
```

In algorithm 5, SP_U is a data structure that holds the scheduling of migratory tasks. Each SP_U object specifies a processor, an execution interval, and a voltage-level for a single task. For any schedule period, the proportionally scaled unit SP is repeated. Function mirror (SP) will reverse the scheduling order for the given SP and is applied to schedule periods of odd index which merges free spaces of adjacent SPs while decreases the number of preemptions and voltage-level switches for non-migratory tasks. It also makes the migratory tasks at the end of SP to repeat in the start of next SP which causes reducing the number of migrations, preemptions and voltage-level switches. A similar technique is used in [46] to decrease the number of preemptions.

Example 3: Figure 6 shows the placement of migratory task segments to the parallel machines after applying the mirror procedure on the schedule of Example 2.

Briefly, we use the scaled version of the obtained schedule of a SP of unit length by the factor of $|SP_k|$ beside its mirror to schedule migratory tasks in SP_k, and thus, achieve a feasible scheduling for the migratory tasks in a hyper period. In this regard, each machine will have some idle intervals in which some non-migratory tasks can be scheduled. In the next subsection, we describe the method of scheduling segments of non-migratory tasks within these idle intervals. As a matter of fact, it is not vital to schedule migratory and non-migratory tasks through different algorithms, or even in different phases like the approach discussed in [32]. We could use Algorithm 1 to schedule both migratory and non-migratory tasks simultaneously. Such possible scheduling suffers from high overhead; hence, in order to reduce the number of preemptions as well as voltage-level switches, we schedule segments of non-migratory tasks separately after all migratory task segments have been scheduled. More details are presented in the following section.

5.3. Scheduling Non-Migratory Tasks

The main point about scheduling non-migratory tasks is that we are not concerned about the parallel execution of job slices of any such task. In addition, each task segment TS_{ijl} of non-migratory tasks can be executed in SP_k during the machine idle periods for $t_{ijl} \times |SP_k|$ after scheduling migratory tasks, as the LP constrains (Constraint (5.2)). By scheduling in this way, each task token would meet its fluid path at the end of each SP, and therefore, no non-migratory task misses its deadline, i.e. there is a feasible scheduling for them in idle times of each machine. However, in order to reduce the number of preemptions and voltage-level switches, we use a different algorithm for this part. As a result from [53], algorithms with work-conserving property have the advantage to avoid unnecessary task preemptions. Hence, we consider this property for scheduling of non–migratory tasks that significantly decreases the number of preemptions and voltage-level switches. Corresponding to each segment TS_{ijl} of non-migratory tasks, we define a release-based-task-segment ($RBTS_{ijl}$) with size τ_{ijl} which indicates the amount of time that task T_i should be executed on machine M_j at voltage-level V_{jl} in the task period of length p_i, where $\tau_{ijl} = t_{ijl} \times p_i$. We schedule RBTSs per task period instead of TSs per SP. Also, we use EDF to schedule the RBTSs (i.e., for non-migratory tasks) in the idle times throughout each hyper period. Of course, with employing scheduling algorithms with lower preemption overheads than EDF (e.g., EDZL), it is possible to decrease the number of preemptions even more. The sum of execution requirements of all RBTSs of a task T_i ($RBTS_{ijl}$) in a period p_i is equal to the exact execution requirements of task T_i in the period, i.e. e_i. According to Constraint (5.1), we have

$$\sum_{j=1}^{m} \sum_{l=1}^{k_j} S_{ijl} \times \tau_{ijl} = e_i \quad (18)$$

Thus, even if the scheduling strategy is changed, it will not cause any deadline to be missed while the job slices of a certain task instance are executed within their respective period (i.e., their relative deadline). Furthermore, according to the optimality of the EDF algorithm [23], it is straightforward to show that EDF gives a feasible scheduling in the idle times of each machine, if such a schedule exists. In this regard, as the existence of a feasible schedule for the task set is verified and confirmed in Step 1 of the scheduling (i.e., through the LP), it would be sufficient to use the EDF algorithm while considering each $RBTS_{ijl}$ as a task with execution requirement τ_{ijl} and period p_i. In this way, all the RBTSs, and equivalently, all the non-migratory tasks are scheduled. It is worthwhile to note how this method of scheduling results in fewer preemptions and voltage-level switches: in the case where two RBTSs, namely $RBTS_{ijl}$ and $RBTS_{ijl'}$ exist for a certain task T_i and no new task with an earlier deadline arrives before the RBTSs deadline, EDF selects $RBTS_{ijl'}$ (which has the same deadline as $RBTS_{ijl}$) for execution instead of random selection among the other admissible tasks. In this regard, one voltage-level switch occurs instead of a preemption.

Example 4. As shown in figure 7, RBTSs corresponding to segments of non-migratory tasks of Example 1 which listed in table 1, are scheduled after scheduling segments of migratory tasks in Example 2. Migratory tasks are shown in gray color, while RBTSs are represented in other colors.

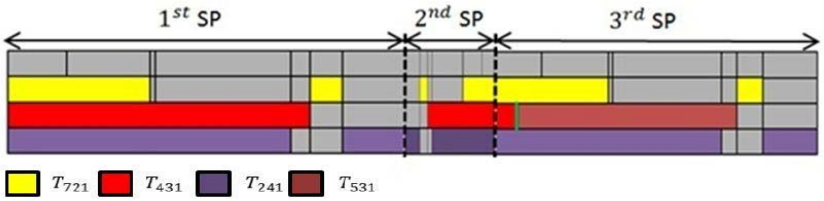


Figure 7. Complete scheduling schema over SPs

Theorem 2. The Linear Programming (LP) problem mentioned in Step 1 has a solution with minimal value of E if and only if the set of unrelated parallel machines has a feasible scheduling with minimal average energy usage E.

Proof: It is trivial to show that if there is a scheduling for a task set, the constraints of the LP would be met; therefore, the LP has a solution with minimal average energy usage E if there is a feasible scheduling. By Corollary 1, it is inferred that minimal average energy usage for scheduling the task set in a hyper period ($\overline{E(H)}_{\min}$), and therefore the system lifetime, is also the minimal average energy usage for scheduling the scaled task set in a SP of unit length, which is the minimum value of $\overline{E}$ in the LP.

To prove the other direction, consider a solution to the LP. We described how we can use this solution to construct a schedule for the tasks upon the machines, such that all deadlines are met. Specifically, for segments of migratory tasks, we will construct a schedule over the interval $[0,1)$. By Theorem 1, a schedule in which all task instance deadlines are met may be obtained by repeating the scaled versions of this schedule over all SPs. Then we have scheduled non-migratory tasks in the remained idle times. It is mentioned that non-migratory tasks can be scheduled by the EDF algorithm without missing a deadline.

Scheduling in this manner results in executing each task segment TS_{ijl} (segments of both non-migratory and migratory tasks) obtained from the LP, for $t_{ijl} \times H$ amounts of time during a hyper period. Therefore, energy usage of the scheduling is $\overline{E} \times H$ in a hyper period of length H, which means the average energy usage of the scheduling in that period, is equal to $\overline{E}$. Furthermore, based on Corollary 1, average energy usage of the system could not be less than the minimum average energy usage of the scaled task set in a SP of unit length. Thus, this algorithm gives a scheduling with the minimum energy usage if the system has a feasible solution.

6. Discussion

In this section, we first determine the number of migratory tasks based on some properties of the linear programming, and then, we specify an upper bound on the number of preemptions and voltage-level switches.

6.1. Linear Programming and the Degree of Migration

The cost of migration might be inevitable in some distributed systems since the whole task state must be migrated as well. However, the proposed algorithm schedules most tasks on only one processor, and bounds the number of migratory tasks. The total number of edges in the bipartite graph at the first iteration of Algorithm 1 is a measure to find out how much the schedule is global rather than partitioned (degree of migration); a schedule in which the number of segments of migratory tasks is equal to zero is a purely partitioned schedule [32]. This measure can determine the degree of migration as discussed in the following.

As shown in [47] and [48], there are polynomial-time algorithms for solving LPs as well as polynomial-time algorithms (e.g., see [49]) for obtaining a basic solution, given a non-basic optimal solution to a LP problem. Therefore, a basic solution can be obtained in polynomial-time. Considering a LP with n variables, I inequality constraints and E equality constraints, the basic solution to the LP problem lies at a vertex point in which n–E inequalities satisfy as equalities.

Returning back to our problem, we have

$$N = (n+1) \times \sum_{i=1}^{m} k_i \tag{19}$$

variables and inequalities (Constraint (5.4)) in the LP. Also, we have n and m equalities for Constraints (5.1) and (5.2), respectively, and, n inequalities for Constraint (5.3). However, according to the m equalities for Constraint (5.2), we can conclude

$$\sum_{j=1}^{m} (\sum_{i=1}^{n} \sum_{l=1}^{k_i} t_{ijl}) \le m \tag{20}$$

and thus,

$$\sum_{i=1}^{n} (\sum_{j=1}^{m} \sum_{l=1}^{k_i} t_{ijl}) \le m \tag{21}$$

Consequently, at most m number of the tasks can be urgent (satisfy inequalities of Constraint (5.3) with equality). According to the mentioned property of LP, N–n–m inequalities would be marginally satisfied. As shown above, at most m inequalities for Constraint (5.3) can satisfy as equalities. Thus at least N–n–2m of inequalities of Constraint (5.4) is held in the form of equalities.

This means that there are at most n+2m non-zero variables at the optimal point. On the other hand, because each task T_i ($i > 0$) must have at least one non-zero variable t_{ijl} for some values of j and l, we can conclude that n variables of n+2m non-zero variables belong to different tasks; therefore, at most 2m extra task segments exists. These extra segments can result in at most 2m migratory tasks. Also, there are at most 2m+2m=4m segments of migratory tasks.

6.2. Number of Preemptions, Migrations and Voltage-Level Switches

Regarding the issue that there is no need for voltage-level switches when a machine is idle (as described in Section 5), the upper bound on the number of migrations, preemptions and voltage-level switches can be obtained as follows. The number of migrations and voltage-level switches for migratory tasks will be at most equal to the number of iterations of Algorithm 1 in each machine; because there are no interruptions in the interval of iteration and the task can run continuously. The number of iterations of Algorithm 1 is at most equal to the number of times that one of the change types A, B, or C occur. There are at most 6m changes, namely 4m changes of Type A, which is less than the number of task segments belonging to the migratory tasks, m changes of Type B, which is less than the number of full machines, and m changes of Type C, which is less than the number of urgent tasks.

Therefore, 6m is an upper bound on the number of migrations and voltage-level switches in a SP for migratory tasks on each machine, and thus, $6m^2$ is an upper bound for the entire system model. On the other hand, using the EDF algorithm, non-migratory tasks will be preempted only when a change of Types A, B or C occurs or when a new RBTS arrives or its execution finishes, considering RBTSs as tasks. We refer to these two latter cases as Change Type D (times in which changes of Type D occur, are shown in red color in Algorithm 4). The number of SPs in a hyper period, which is at most equal to the number of task releases, is

$$\sum_{i=1}^{n} \frac{H}{p_i} \tag{22}$$

Therefore, the total number of preemptions for all machines will be equal to the number of Type A, B and C changes, which is

$$(6m^2) \times \sum_{i=1}^{n} \frac{H}{p_i} \tag{23}$$

plus the number of Type D changes that is

$$2 \times \sum_{RBTS_{ijl}} \frac{H}{p_i} \tag{24}$$

Furthermore, since there are at most 2m extra task segments (as mentioned above), the number of RBTSs is not more than n + 2m, and at least one RBTS for each non-migratory task exists. This means that:

$$\sum_{RBTS_{ijl}} \frac{H}{p_i} \le \sum_{i=1}^{n} \frac{H}{p_i} + 2m \frac{H}{\min(p_i)} \le (2m+1) \sum_{i=1}^{n} \frac{H}{p_i} \tag{25}$$

Thus, the number of preemptions and voltage-level switches in a SP using the proposed algorithm is $2 \times (2m+1) + 6m^2 = O(m^2)$ in the average case. This is independent of n and much fewer than the preemptions in the available online algorithms, even for identical platforms where $n \gg m$ (e.g. [14]).

7. Experimental Results

This section presents the evaluation results of the proposed algorithm for synthetic task sets on different configurations of parallel machines. Since, to the best of our knowledge, there is no optimal scheduling algorithm for unrelated parallel machines that support task migration, we compare EORTSA algorithm with PCG [25] which is introduced for real-time scheduling on uniform parallel machines.

Most of the optimal scheduling algorithms for heterogeneous multiprocessors are based on fairness [25, 32, 50], so the order of the number of preemptions and migrations of these algorithms is almost the same. These algorithms proportionally assign processors to the tasks according to the utilization rates. Among them, PCG is an optimal scheduling algorithm which is insensitive to the energy usage. PCG has the least number of migrations and preemptions in the class of optimal scheduling algorithms in uniform parallel machines which make it a good choice to compare with preemptions and migrations of EORTSA. On the other hand, since EORTSA scheduling algorithm is optimal regarding the energy usage, we have omitted the comparison of EORTSA with other existing algorithms which intend to reduce the energy.

Although the proposed algorithm supports all types of machines, it was restricted to uniform parallel machines in experiments. We have considered four uniform parallel machines with 2, 4, 8 and 16 XScale PX270 processor cores. Since this processor supports multiple DVS voltage-levels, it is possible to create a uniform parallel machine consisting of cores with different execution speed by assigning different voltage-levels to cores accordingly. We have used data from table 6 to create such parallel machine in which every core works at a fixed random voltage and corresponding frequency. Also, it can be observed from table 6, there is a linear dependence between the speed and the power, meaning that by doubling the speed, power consumption will increase approximately twice.

With such restriction, it is hard to reveal the potentials of our algorithm in reducing power consumption during the evaluations. Our algorithm achieves much more energy saving for unrelated platforms rather than uniform ones, as power consumption relates to execution speed through a nonlinear function and is task-dependant. To reflect the nonlinear relation between power consumption and execution speed, which is most common in unrelated platforms, a custom uniform platform with nonlinear power-speed relation is modeled using the following assumptions:

- Tasks are assumed to have the same functionality but with different input sets,
- Task execution times are calculated for different processors and are normalized based on the execution speed of the slowest processor,
- Task execution time on different processors is based on E3S benchmark suite [51].

Since we only have one type of the task on the system, this unrelated platform can be considered as a uniform platform. Now, PCG algorithm can schedule the tasks in this new platform. Tables 6 and 7 report the average powers and relative speeds of some processor cores for different operating frequencies. Although actual power and relative

speed values vary depending on the tasks, these values can be used as a guideline for our constructed platform.

Linear program and minimum weighted bipartite matching problem (via minimum cost network flow) are modelled and solved in ILOG Cplex solver [52].

Table 5. Specifications of the Intel XScale PXA270 processor [7] for different speeds

Speed (MHz)	104	208	312	416	520	624
Normalized speed	0.5	1	1.5	2	2.5	3
Active power (watts)	0.116	0.279	0.390	0.570	0.747	0.925
Idle power (watts)	0.064	0.129	0.154	0.186	0.222	0.260

Table 6. Speed and power of executing text processing benchmark on different processors according to E3S benchmark suites [47]

Processor model	Speed	Active power (watts)	Idle power (watts)
AMD ElanSC520-133 MHz	3.0	1.6	0.16
AMD K6-2E 400MHz/ACR	1	10	1
AMD K6-2E+ 500MHz/ACR	1.75	14	1.4
AMD K6-IIIE+ 550MHz/ACR	2	16	1.6
IBM PowerPC 405GP - 266 MHz	1.75	2	0.2
NEC VR5432 - 167 MHz	0.16	2.5	0.25

We conducted several different experiments to evaluate the average power, number of preemptions and number of migrations of the above-mentioned scheduling algorithms.

As stated before, the evaluation has been performed on synthetic task sets. These tasks have been generated randomly in the following manner. First the utilization value of each task has been generated randomly such that the system utilization reaches to a target value determined in the experiment setup. In order to avoid unaffordable hyper period length in simulation, we use the following approach for generating of task periods. First an integer number which has at least 150 divisors is randomly selected as the hyper period value. After that, the task period will be chosen from the set of those divisors. In this way, we have restricted the periods to integer numbers. And finally the execution time is calculated using $e_i = p_i \times u_i$.

In each experiment, system utilization has been considered in the range of 10% to 100% (with step 10%). Each experiment setup has been repeated 100 times. Then, the impact of utilization on the performance metrics has been observed. These performance measures are Average Number of Migrations and Preemptions per job (ANMP) and the average energy usage as reported in the other researches in the literature.

Figure 8 shows ANMP as the vertical axis according to different system utilizations (horizontal axis). Also in this figure, the comparison of the number of preemptions and migrations for different task set sizes (10, 20, 30 and 40) has been reported. In this figure, when there are 10 tasks in the system and the utilization is higher than or equal to 60%, PCG algorithm has less preemptions and migrations than EORTSA (it is about 33% of EORTSA). In fact, since EORTSA aims at minimizing the energy, it may lead to higher ANMP in comparison with non-energy minimizing and general scheduling algorithms. Consequently, in the loads near 100% in which the opportunity of saving energy is limited, EORTSA is less effective than non-energy-minimizing scheduling algorithms such as PCG. However, when the number of tasks becomes larger (e.g., 20, 30 or 40), ANMP of EORTSA decreases. The reason is that with larger number of tasks, each task most likely has smaller portion in the total utilization of the system so the chance that it is preempted or migrated becomes slighter. Consequently, the performance of EORTSA improves when there are a large number of tasks in the system.

Another important feature of EORTSA is that it keeps the number of preemptions and migrations under a fixed bound, i.e. $O(m^2)$, that only depends on the number of processors in the system. Figure 8 reveals this fact especially when the number of tasks increases. In a similar fashion, PCG has considerable amounts of ANMP (see figure 8(d)). Because it makes proportional assignments on each schedule period, in the higher task counts, it encounters many task switches. Besides, EORTSA schedules RBTSs per task period instead of task segments per schedule period for non-migratory tasks and always preserves the number of migratory tasks under twice the number of processors (2m). Figure 9 shows comparison of power consumption of scheduled task set for PCG and our algorithm.

Figure 9 (a) presents power consumption for 4 processors for different system utilizations. EORTSA consumes less power at different utilizations than PCG. However, when the utilization is 100%, there will be no difference between the power consumption of PCG and EORTSA (because all processors are always busy at that utilization). For EORTSA, the maximum power saving in comparison with PCG is achieved when the utilization is around 30% to 70% (see figure 9(a)) which was about 60% for utilization 30% and 40%.

In the next experiment (Figure 10), we have considered different number of processors (2, 8 and 16), and for each setup, two task set sizes have been considered. Similar to our discussion for the 4-processor setup, by increasing the number of tasks, EORTSA outperforms PCG in ANMP. Figure 9(b) shows the average power consumption of each experiment for 2, 4, 8 and 16 processors. With the increase of the number of processors, the gap between EORTSA and PCG also increase because when EORTSA finds more opportunities to distribute tasks over the processors, it might find better options to enhance power consumption of the tasks.

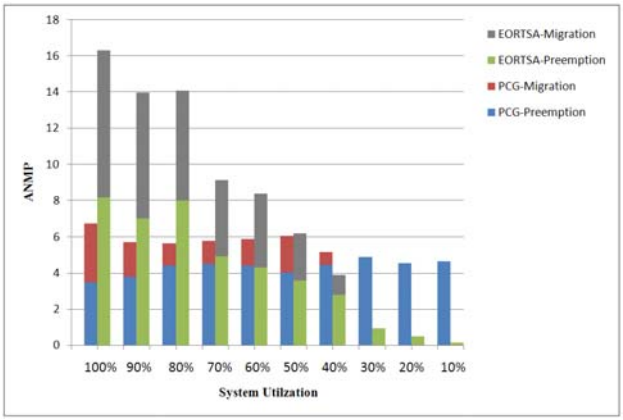
Further, a version of PCG algorithm for unrelated platform which is aware of different execution speed of tasks on different processors has been implemented. The modified PCG has lower schedule ability bound which is depending on the underlying platform configuration. The reason of non-optimal assignment of tasks to processors in this extension of PCG is its greedy assignment which is based on local view of the system.

8. Conclusions and Future Works

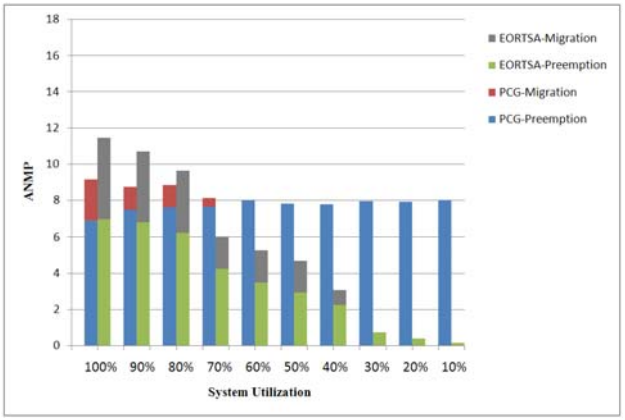
In this paper, we have studied the problem of scheduling a set of periodic real-time tasks on unrelated parallel machines among which task migration is permitted. Each processor (machine) in the system has a few discrete speed levels and

can employ DVS for voltage and speed scaling. At each voltage-level, the processor has a maximum speed as well as a maximum power usage. Further, there exist task-processor specific speeds which are fractions of the maximum speeds as well as task-processor specific power usages which are fractions of the mentioned maximum power usages. A polynomial-time optimal scheduling algorithm is proposed which feasibly schedules the set of tasks on the machines while minimizes the overall system energy usage. We introduce the first part of the algorithm as a schedule ability test for unrelated platforms. Also, with multiple improvements the number of task migrations and preemptions and voltage-level switches are reduced to avoid extra overheads. It is mentioned that the number of migratory tasks is at most 2m, where m is the number of processors. Also, it is shown that the total number of migrations, preemptions and voltage-level switches in each schedule period is in the order of $O(m^2)$.

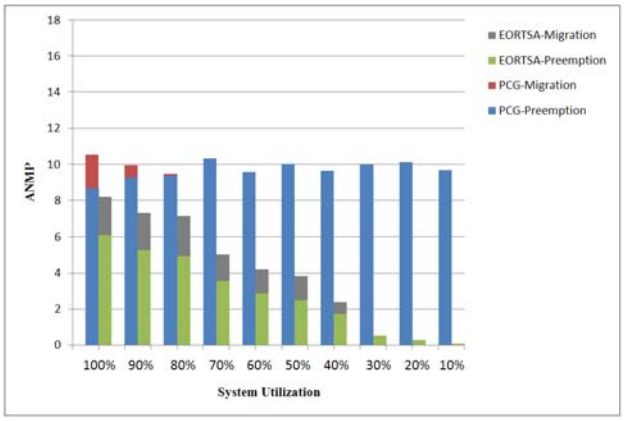
As an idea for the further work, we intend to extend our energy model to include leakage power and consider overheads of migration, preemption, and voltage level switches in terms of energy.



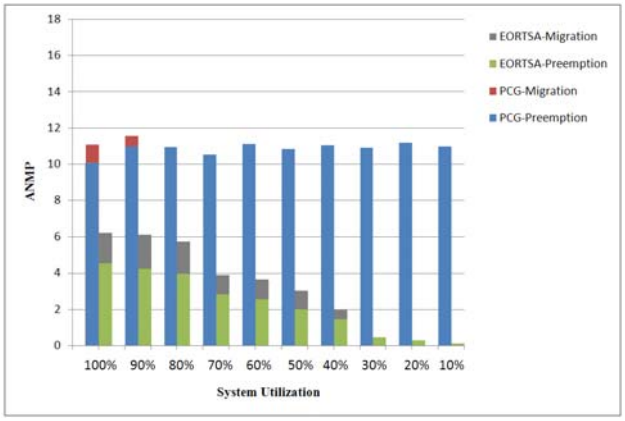
(a) n=10, m=4



(b) n=20, m=4

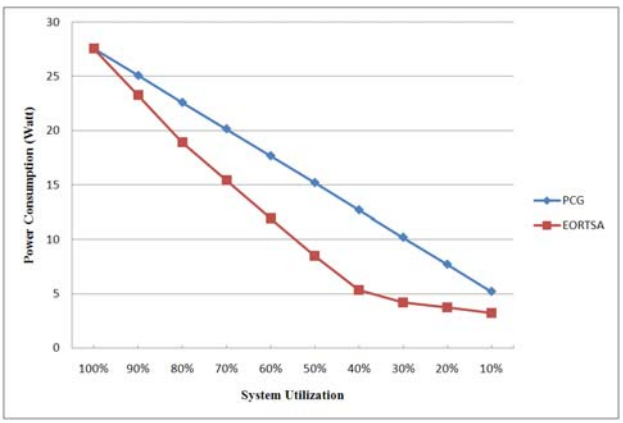


(c) n=30, m=4

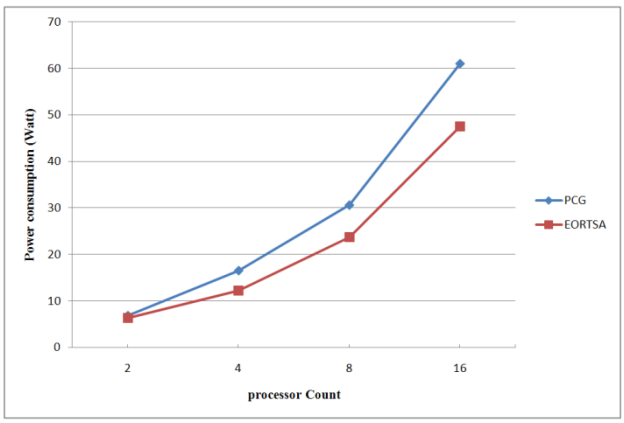


(d) n=40, m=4

Figure 8. Average Number of preemptions and migrate onsper job for 4 processors

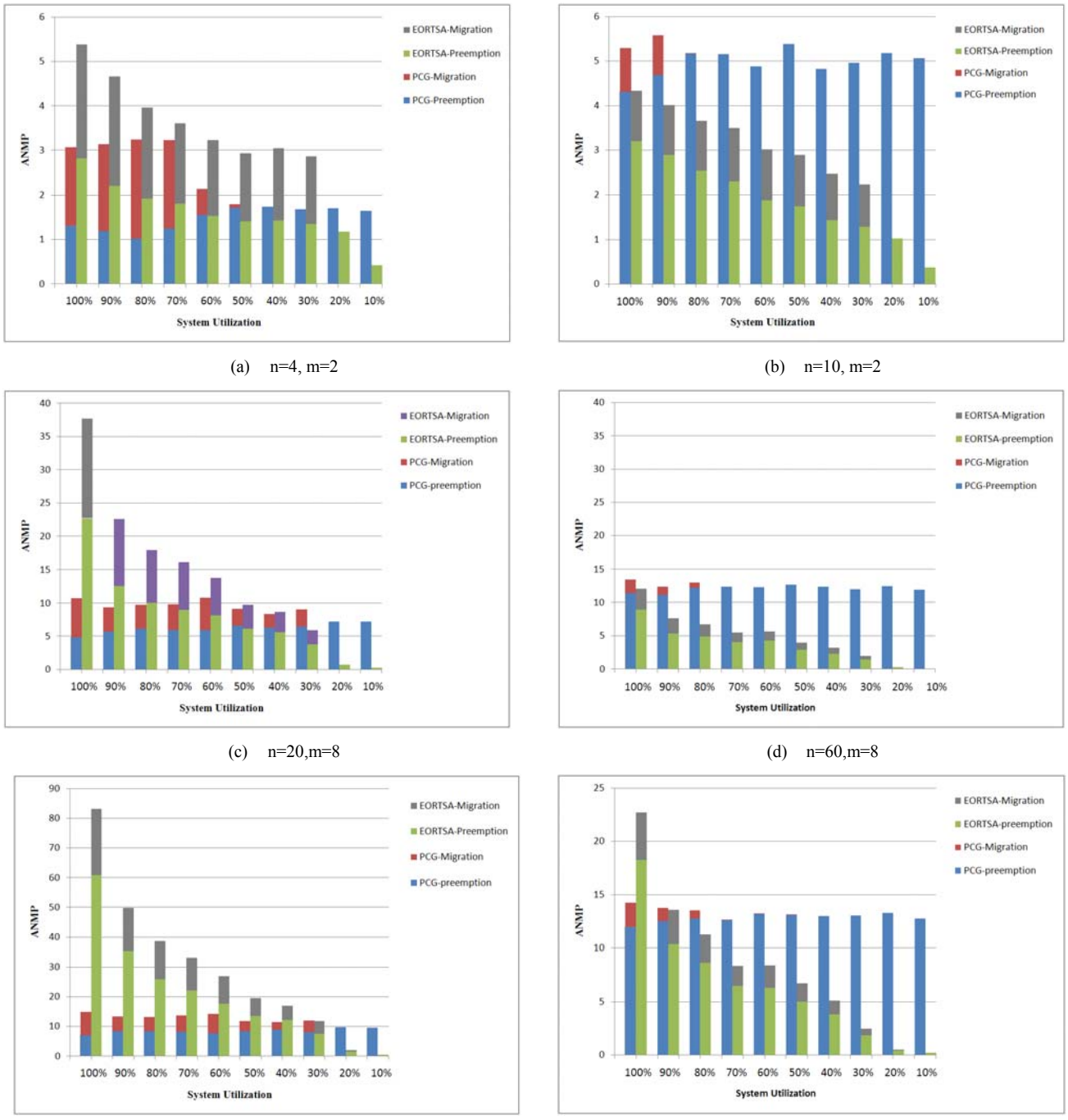


(a) For 4 Processors



(b) Average of all of utilization

Figure 9. Power consumption



(a) n=4, m=2

(b) n=10, m=2

(c) n=20,m=8

(d) n=60,m=8

(e) n=40,m=16

(f) n=120,m=16

Figure 10. Number of preemptions and migrations per job for different processor and task count

References

[1] K. K. Rangan, M. D. Powell, G.-Y. Wei, and D. Brooks, Achieving uniform performance and maximizing throughput in the presence of heterogeneity, in: High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on, pp. 3-14, 2011.

[2] U. R. Karpuzcu, B. Greskamp, and J. Torrellas, The BubbleWrap many-core: popping cores for sequential acceleration, in: 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 447-458, 2009.

[3] E. Bini, G. Buttazzo, and G. Lipari, Minimizing CPU energy in real-time systems with discrete speed management, ACM Transactions on Embedded Computing Systems (TECS), vol. 8, no. 4, pp. 31, 2009.

[4] M. A. Haque, H. Aydin, and D. Zhu, Energy-aware Standby-Sparing Technique for periodic real-time applications, in: 29th International Conference on Computer Design (ICCD), IEEE, pp. 190-197, 2011.

[5] M. Kargahi, and A. Movaghar, Performance optimization based on analytical modeling in a real-time system withconstrained time/utility functions, IEEE Transactions on Computers, vol. 60, no. 8, pp. 1169-1181, 2011.

[6] J.-J. Chen, A. Schranzhofer, and L. Thiele, Energy minimization for periodic real-time tasks on heterogeneous processing units, in: International Symposium on Parallel & Distributed Processing (IPDPS), pp. 1-12, 2009.

[7] C.-Y. Yang, J.-J. Chen, T.-W. Kuo, and L. Thiele, An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems, in: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 694-699, 2009.

[8] A. M. A. Athlon, processor model 6 CPGA data sheet, On the World Wide Web at http://www.amd. com/products/cpg/athlon/techdocs/pdf/24319. pdf, 2001.

[9] Intel Crop, Intel PXA270 processor electrical mechanical, and thermal specification data sheet, 2004, Available from: <http://www.phytec.com/pdf/datasheets/PXA270_DS.pdf>.

[10] R. R. Muntz, and E. Coffman, Optimal preemptive scheduling on two-processor systems, IEEE Transactions on Computers, vol. 100, pp. 1014-1020, 1969.

[11] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, Proportionate progress: A notion of fairness in resource allocation, Algorithmica, vol. 15, no. 6, pp. 600-625, 1996.

[12] P. Holman, and J. H. Anderson, Adapting Pfair scheduling for symmetric multiprocessors, Journal of Embedded Computing, vol. 1, no. 4, pp. 543-564, 2005.

[13] M. L. Dertouzos, and A. K. Mok, Multiprocessor online scheduling of hard-real-time tasks, IEEE Transactions on Software Engineering, vol. 15, no. 12, pp. 1497-1506, 1989.

[14] H. Cho, B. Ravindran, and E. D. Jensen, An optimal real-time scheduling algorithm for multiprocessors, in: 27th IEEE International Real-Time Systems Symposium (RTSS), pp. 101-110, 2006.

[15] W. Y. Lee, Energy-Saving DVFS Scheduling of Multiple Periodic Real-Time Tasks on Multi-core Processors, in: Proceedings of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, pp. 216-223, 2009.

[16] J.-J. Chen, H.-R. Hsu, K.-H. Chuang, C.-L. Yang, A.-C. Pang, and T.-W. Kuo, Multiprocessor energy-efficient scheduling with task migration considerations, in: Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS), pp. 101-108, 2004.

[17] J.-J. Chen, and T.-W. Kuo, Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics, in: International Conference on Parallel Processing (ICPP), pp. 13-20, 2005.

[18] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems, in: Proceedings of the 12th Real-Time and Embedded Technology and Applications Symposium, pp. 408-417, 2006.

[19] E. C. Horvath, S. Lam, and R. Sethi, A level algorithm for preemptive scheduling, Journal of the ACM (JACM), vol. 24, no. 1, pp. 32-43, 1977.

[20] T. Gonzalez, and S. Sahni, Preemptive scheduling of uniform processor systems, Journal of the ACM (JACM), vol. 25, no. 1, pp. 92-101, 1978.

[21] D. Hochbaum, and D. Shmoys, A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approximation approach, in: Foundations of Software Technology and Theoretical Computer Science, pp. 382-393, 1986.

[22] S. K. Baruah, and J. Goossens, Rate-monotonic scheduling on uniform multiprocessors, IEEE Transactions on Computers, vol. 52, no. 7, pp. 966-970, 2003.

[23] C. L. Liu, and J. W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the ACM (JACM), vol. 20, no.1, pp. 46-61, 1973.

[24] S. Funk, J. Goossens, and S. Baruah, Online scheduling on uniform multiprocessors, in: Proceedings 22nd Real-Time Systems Symposium (RTSS), pp. 183-192, 2001.

[25] S.-Y. Chen, and C.-W. Hsueh, Optimal dynamic-priority real-time scheduling algorithms for uniform multiprocessors, in: Real-Time Systems Symposium (RTSS), pp. 147-156, 2008.

[26] Y. Yu, and V. K. Prasanna, Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization, Journal of Information Science and Engineering, vol. 19, no.3, pp. 433-450, 2003.

[27] J. D. Ullman, NP-complete scheduling problems, Journal of Computer and System sciences, vol. 10, no. 3, pp. 384-393, 1975 .

[28] J. K. Lenstra, D. B. Shmoys, and É. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Mathematical programming, vol. 46, no. 1, pp. 259-271, 1990.

[29] T. Gonzalez, E. L. Lawler, and S. Sahni, Optimal preemptive scheduling of two unrelated processors, ORSA Journal on Computing, vol. 2, no. 3, pp. 219-224, 1990.

[30] K. Jansen, and L. Porkolab, Improved approximation schemes for scheduling unrelated parallel machines, Mathematics of Operations Research, vol. 26, no. 2, pp. 324-338, 2001.

[31] B. Srivastava, An effective heuristic for minimising makespan on unrelated parallel machines, Journal of the Operational Research Society, vol. 49, no. 8, pp. 886-894, 1998.

[32] S. Baruah, Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms, in: Proceedings of the 25th International Real-Time Systems Symposium (RTSS), pp. 37-46, 2004.

[33] H.-R. Hsu, J.-J. Chen, and T.-W. Kuo, Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint, in: Proceedings of the conference on Design, automation and test in Europe (DATE), (European Design and Automation Association, pp. 1061-1066, 2006.

[34] J.-J. Chen, and T.-W. Kuo, Allocation cost minimization for periodic hard real-time tasks in energy-constrained DVS systems, in: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design (ICCAD), ACM, pp. 255-260, 2006.

[35] T.-Y. Huang, Y.-C. Tsai, and E.-H. Chu, A near-optimal solution for the heterogeneous multi-processor single-level voltage setup problem, in: International Parallel and Distributed Processing Symposium (IPDPS), pp. 1-10, 2007.

[36] E.-H. Chu, T.-Y. Huang, and Y.-C. Tsai, An optimal solution for the heterogeneous multiprocessor single-level voltage-setup problem, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, pp. 1705-1718, 2009.

[37] J. Luo, and N. K. Jha, Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems, in: Proceedings of the 2002 Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 719, 2002.

[38] D. Ding, L. Zhang, and Z. Wei, A novel voltage scaling algorithm through ant colony optimization for embedded distributed systems, in: International Conference on Integration Technology (ICIT), pp. 547-552, 2007.

[39] J. Lin, A. M. Cheng, and R. Kumar, Real-time task assignment in heterogeneous distributed systems with rechargeable batteries, in: International Conference on Advanced Information Networking and Applications (AINA), pp. 82-89, 2009.

[40] M. Kargahi, and A. Movaghar, Stochastic DVS-based dynamic power management for soft real-time systems, Microprocessors and Microsystems, vol. 32, no. 3, pp. 121-144, 2008.

[41] M. DeVuyst, A. Venkat, and D. M. Tullsen, Execution migration in a heterogeneous-ISA chip multiprocessor, in: Proceedings of the 17th international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 261-272, 2012.

[42] T. Li, P. Brett, R. Knauerhase, D. Koufaty, D. Reddy, and S. Hahn, Operating system support for overlapping-ISA heterogeneous multi-core architectures, in: 16th International Symposium on High Performance Computer Architecture (HPCA), pp. 1-12, 2010.

[43] V. Nollet, P. Avasare, J.-Y. Mignolet, and D. Verkest, Low cost task migration initiation in a heterogeneous mp-soc, in: Proceedings of the International Conference on Design, Automation and Test in Europe (DATE), pp. 252-253, 2005.

[44] H. Shen, and F. Pétrot, Novel task migration framework on configurable heterogeneous MPSoC platforms, in: Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 733-738, 2009.

[45] J. A. Bondy, and U. S. R. Murty, Graph theory with applications, Elsevier, New York, 1976.

[46] B. Andersson, and E. Tovar, Multiprocessor scheduling with few preemptions, in: Proceedings of the 12th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), pp. 322-334, 2006.

[47] N. Karmarkar, A new polynomial-time algorithm for linear programming, in: Proceedings of the 16th annual ACM symposium on Theory of computing, pp. 302-311, 1984.

[48] J. Renegar, A polynomial-time algorithm, based on Newton's method, for linear programming, Mathematical Programming, vol. 40, no. 1, pp. 59-93, 1988.

[49] A. Schrijver, Theory of linear and integer programming, John Wiley and Sons, New York, 1986.

[50] S. Funk, and V. Nanadur, LRE-TL: An Optimal Multiprocessor Scheduling Algorithm for Sporadic Task Sets, in: 17th International Conference on Real-Time and Network Systems (RTNS), pp. 159-168, 2009.

[51] R.Dick, Embedded System Synthesis Benchmarks Suite(E3S), Available from: <http://ziyang.eecs.umich.edu /~dickrp/e3s/>.

[52] IBM ILOG CPLEX 12.1, User's Manual, Available from: <http://www.ilog.com/products/cplex>, 2010.

[53] K. Funaoka, S. Kato, and N. Yamasaki, Work-conserving optimal real-time scheduling on multiprocessors, in: Euromicro Conference on Real-Time Systems (ECRTS), pp. 13-22, 2008.

[54] K. Funaoka, S. Kato, and N. Yamasaki, Energy-efficient optimal real-time scheduling on multiprocessors, in: 11th International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp. 23-30, 2008.

[55] K. Funaoka, A. Takeda, S. Kato, and N. Yamasaki, Dynamic voltage and frequency scaling for optimal real-time scheduling on multiprocessors, in: International Symposium on Industrial Embedded Systems (SIES), pp. 27-33, 2008.

[56] L. Cucu-Grosjean, and J. Goossens, Exact schedulability tests for real-time scheduling of periodic tasks on unrelated multiprocessor platforms, Journal of Systems Architecture, vol. 57, pp. 561-569, 2011.

[57] T. Megel, R. Sirdey, and V. David, Minimizing task preemptions and migrations in multiprocessor optimal real-time schedules, in: 31th Real-Time Systems Symposium (RTSS), pp. 37-46, 2010.

[58] G. Raravi, B. Andersson, K. Bletsas, and V. Nelis, Task assignment algorithms for two-type heterogeneous multiprocessors, in: 24th Euromicro Conference on Real-Time Systems (ECRTS), pp. 34-43, 2012.

Appendix

List of Symbols

Symbols	Present
T	Set of all tasks
T_i	The i^{th} task in T
p_i	Period and relative deadline of task T_i
e_i	Execution requirement of task T_i
N	Number of tasks
M	Set of all machines
M_j	The j^{th} machine in M
m	Number of machines
k_j	Number of voltage-levels of machine M_j
V_j	Set of all voltage-levels of machine M_j
V_{jl}	The l^{th} voltage-level of machine M_j
S_{jl}	Maximum speed of machine $M_j$at voltage-level V_{jl}
Pow_{jl}	Maximum power consumption of machine M_j at voltage-level V_{jl}
T_0	Dummy task; (its execution is equivalent to be idle)
Pow_{0jl}	Power consumption of machine M_j at voltage-level V_{jl} when idle
S_{ijl}	Speed of machine M_j at voltage-level V_{jl} when it is executing task T_i (i> 0)
Pow_{ijl}	Power consumption of machine M_j at voltage-level V_{jl} when it is executing task T_i (i> 0)
$E_{ijl}(t)$	Energy consumed by machine M_j at voltage-level V_{jl} when it is executing task T_i (i> 0)for t amount of time
$(T_i, M_j, V_{jl}, t_s, t_f)$	A job slice, that means Task T_i is scheduled on machine M_j at voltage-level V_{jl} in time interval $[t_s, t_f)$
$\overline{E(a,b)}$	Minimum average energy usage of M in time interval $[a,b)$
H	Size of a hyper period
$\overline{E(H)}_{min}$	Minimum average energy usage of T on M in a hyper period
SP	Schedule Period
SP_k	The k^{th} SP (the time interval between the k^{th} and $(k + 1)^{th}$ task releases)
$\lvert SP_k \rvert$	Length of SP_k
ST	Scaled task
ST_i	Scaled task corresponding to task T_i
e_i^k	Execution time of the scaled task corresponding to task T_i in SP_k
s_k	Start time of SP_k
t_k	Finish time of SP_k
$\overline{E(SP)}_{min}$	Minimum average energy usage of scaled task set corresponding to Tin a SP, on M
t_{ijl}	The amount of time needed by task T_i to be executed on machine M_j at voltage-level V_{jl}(LP variables)
TS_{ijl}	Task segments corresponding tot_{ijl}> 0wherei> 0
t'_{ijl}	Size of non-scheduled part of task segment TS_{ijl} throughout Algorithm 1
e'_i	The remaining execution requirement of migratory task T_i in the ongoing iteration of Algorithm 1
γ	the length of available interval in which the rest of scheduling takes place in each iteration of Algorithm 1
σ	The length of available interval in which scheduling of current iteration is going to be done in Algorithm 1
F	Set of full machines in each iteration of Algorithm 1
U	Set of urgent tasks in each iteration of Algorithm 1
X	A matching between machines and tasks that covers all urgent task and full machines
χ_T	A matching from all urgent tasks to machines
χ_M	A matching from all full machines to tasks
R_U	The range of χ_T
R_F	The range of χ_M

Change of Type A	A task segment TS_{ijl} is assigned to the proper machine at the proper voltage-level for the length of t'_{ijl}
Change of Type B	A machine M_j becomes full
Change of Type C	A task T_i becomes urgent
Change of Type D	A new RBTS arrives or its execution finishes
$A(TS_{ijl})$	A Change of Type A on task segment TS_{ijl}
$B(M_j)$	A Change of Type B on machine M_j
$C(T_i)$	A Change of Type C on task T_i
$RBTS_{ijl}$	Released-based task-segment corresponding to task segment TS_{ijl}
τ_{ijl}	The length of $RBTS_{ijl}$
S	A set of machines that is used in Algorithm 4
N(S)	Set of all neighbours of node S in a graph
prevX	Set of selected edges from the previous iteration
nextX	Set of edges in current edges
U	Set of urgent tasks
F	Set of full machines
$T_{n'\times m}$	adjacency matrix where $n'$and m are the numbers of migratory tasks and processors
VT	Set of nodes corresponding to urgent tasks or each task that adjacent to at least one full machine
VP	Set of nodes corresponding to full machines or each machine that adjacent at least one urgent task
V_S	Source node
V_d	Sink node

Mahmood Gholipour received his Master's degree in Computer Engineering from the School of Electrical and Computer Engineering, University of Tehran. In 2010, he received hisBachelor's degree in Computer Engineering from Arak University, Arak, Iran.

E-mail: m.gholipour@ut.ac.ir

Mehdi Kargahi received his B.S. degree in computer engineering from Amir-Kabir University in 1998 and the M.S. and Ph.D. degrees in computer engineering from Sharif University of Technology in 2001 and 2006, respectively. He is currently an associate professor in the Department of Electrical and Computer Engineering at University of Tehran, Iran. He has been a researcher at the Institute for Studies in Theoretical Physics and Mathematics (IPM) from 2003. His research interests include distributed systems, performance modeling and dependable real-time systems

E-mail: kargahi@ut.ac.ir.

Heshaam Faili received his B.S. and M.S. degrees in Software Engineering and his PhD in Artifical Intelligence from Sharif University of Technology on 1997, 1999 and 2006 respectively. He joined to the Artificial Intelligence and Robotic group of the School of Electrical and Computer Engineering, University of Tehran at 2008. He is now an associate professor, and his main research interests include approximation and randomized algorithms, AI and statistical approaches, text processing and mining methods.

E-mail: hfaili@ut.ac.ir

Shahbaz Youssefi received his B.S. degree in Computer Engineering from the School of Electrical and Computer Engineering, University of Tehran, Iran. He has worked on real-time scheduling theory in the Dependable Real-Time Systems (DRTS) Research Laboratory. He is now a researcher in at the University of Genova, Italy.

E-mail: shahbaz.youssefi@unige.it

Hadi Ravanbakhsh received his B.S. degree in Computer Engineering from the Department of Electrical and Computer Engineering at University of Tehran, Iran. His project for graduation has been on energy-aware scheduling of real-time tasks on unrelated parallel machines which is done in the Dependable Real-Time Systems (DRTS) Laboratory. He is now a PhD candidate at Department of Computer Science, University of Colorado-Boulder.

E-mail: hadi.ravanbakhsh@colorado.edu

Paper Handling Data:

Submitted: 10.11.2016
Received in revised form: 20.12.2016
Accepted: 02.01.2017
Corresponding author: Dr. Mehdi Kargahi,
School of Eectrical and Computer Engineering, University of Tehran, Tehran, Iran.

[1]A complete list of the symbols used throughout this paper is presented in the appendix at the end of this paper.

The CSI Journal on Computer Science & Engineering

The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 23-31
Regular Paper

Taxonomy and Overview of Distributed Malfunction Diagnosis in Networks of Intelligent Nodes

Behrooz Parhami Nan Wu Sixin Tao

Department of Electrical and Computer Engineering, University of California, Santa Barbara, California, USA

Abstract

Started 50 years ago, the field of (system-level) malfunction diagnosis has expanded immensely and continues to be a very active subfield in both parallel processing and dependable computing research communities, with much of the new research coming from China and Taiwan in recent years. This paper represents an attempt to organize the field of research in distributed malfunction diagnosis via an overarching, descriptive, and consistent taxonomy that not only covers all of the past work, but also foretells of possible future research to fill gaps left by current results and areas that are just beyond the domains already investigated. The paper is accessible to computer science and engineering specialists who are new to the field, because it uses analogies to unveil the nature of the research problems and pertinent challenges.

Keywords: Comparison-Based Diagnosis, Diagnosability, Distributed System, Fault Tolerance, Interconnection Network, Malfunction Diagnosis, MM* Model, Parallel Processing, PMC Model, Self-Diagnosis, System-Level Fault Diagnosis.

1. Introduction

In the freshman seminar "Ten Puzzling Problems in Computer Engineering," designed 10 years ago and taught since by the first author [23] [24], mathematical and logical puzzles are used to introduce advanced science/technology topics in a manner understandable to first-year college students. Two of the seminar's puzzle types are relevant to the topic of this paper. One puzzle type places you on a remote island inhabited by members of two tribes, Truth-tellers and Liars. Truth-tellers provide the correct answer to any question, while Liars always give an untruthful answer. Members of the two tribes recognize each other, but you have no basis to judge which tribe a particular person belongs to, except by analyzing answers to questions you ask. A more advanced version of these puzzles postulates the tribes Truth-tellers and Randoms (they lie or tell the truth, completely at random). It turns out that Randoms are more difficult to deal with, because the consistency of Liars in providing untruthful answers is actually helpful in making deductions. The latter version of these puzzles models diagnosis in a distributed environment: You ask each node to perform self-diagnosis and report the result to you. A healthy node gives you a truthful answer about it being healthy, whereas a malfunctioning node gives you an untrustworthy answer. Is it possible to deduce which nodes are malfunctioning based on the responses received? The short answer is no, if there is no cross-checking of results.

Another puzzle asks you to imagine n people, mostly medical doctors (MDs), but mixed with a small number of impostors, sitting at a round table. Each person is told to interview the person seated to his/her right and render a judgment on whether that person is an MD or an impostor. Let's assume that an MD knows how to question a person to determine with absolute certainty whether that person is indeed an MD. The n judgments are given to you and you must identify the impostors. Clearly, a judgment provided by an impostor is untrustworthy, much like answers provided by Randoms in the previous set of puzzles, not only because s/he does not have the knowledge to judge, but also because

s/he may actually want to deceive you in order to remain undetected. This puzzle models the malfunction diagnosis problem as a directed graph G = (V, E), where vertices in V are intelligent nodes capable of testing each other and edges in E define a testing relation, with the directed edge (u, v) representing node u testing node v.

Five decades ago, this notion was formalized by Preparata, Metze, and Chien [27] into what has come to be known as the PMC model of malfunction diagnosis. Subsequently, Maeng and Malek [19] [20] devised a different formal model in which diagnosis is based on a managing unit comparing responses from two other units to which it is connected, concluding that the two responding units are healthy if their responses match and at least one of them malfunctioning otherwise. The model was subsequently refined and given the name MM* or comparison-based malfunction diagnosis model. As before, if the test manager is itself malfunctioning, no reliable conclusion can be reached.

An unfortunate side effect of the rapid advances in the field of distributed malfunction diagnosis is the emergence of a rather non-descriptive, and at times misleading, terminology. To cite one example, the two terms t/k-diagnosability and t/s-diagnosability mean different things, and the distinction of k versus s is lost when the parameters are replaced with actual numbers in a specific case; e.g., is 5/6 diagnosability of the first kind (t = 5, k = 6) or of the second kind (t = 5, s = 6)? Furthermore, the qualifiers "one-step," "sequential," and "pessimistic," applied to some kinds of diagnosis strategies discussed are rather undescriptive.

In this paper, we propose a taxonomy of malfunction diagnosis methods to facilitate understanding and contributing new results to the field. As a byproduct of the taxonomy, we expose certain areas of the field that need to be studied or explored in greater depth. This is not intended to be a complete survey of the field, as there have been literally hundreds of research contributions in the area of malfunction diagnosis over the past five decades. References cited are meant to cover pioneering contributions that have defined the field as a whole or its various sub domains, or have introduced new concepts, plus a few sources that support our contention that a new, descriptive nomenclature and taxonomy is indeed required.

2. What Is Malfunction Diagnosis?

Fault testing and fault diagnosis have been with us for centuries in connection with gadgets and systems whose designs are to be verified at the outset and whose correct functioning must be ascertained in the field as they are put to use. The term "fault" is a bit overused, as it has been applied at various levels of a digital system hierarchy, from devices and circuits to sizable modules incorporating hardware and software components. Fault testing in circuit and logic entail different methods than testing of higher-level modules. In fact, in modern practice, we often don't care about diagnosing a fault (identifying its location) within a circuit, say, a chip. Rather, we perform what is known as a go/no-go test that merely indicates whether the circuit is usable, replacing the entire circuit in case of a no-go result.

At the system level, by contrast, we do want to identify which module is causing problems, so that we can isolate and eventually repair/replace it. This requires a more elaborate diagnostic testing, instead of the go/no-go variety. For this reason, the term "system-level fault diagnosis" has been used for the latter situation. In the first author's nearly completed book on dependable computing [25], the term "malfunction diagnosis" is used to refer to the context above, avoiding the overuse of the term "fault" and obviating the need for the qualifier "system-level." So, our "malfunction diagnosis" is "system-level fault diagnosis" in much of the published literature. This use of malfunction diagnosis is the first element of our nomenclature and taxonomy.

Let us begin with the basic terminology and assumptions. We consider a system composed of interconnected, intelligent modules, where by intelligent we mean modules with internal processing and decision-making abilities. This isn't a restrictive assumption, as modern digital systems are composed of interconnection of processors, memory modules, I/O units, and the like, each having hardware control for basic functions and software control for functions that are not speed-critical and/or need flexibility over time. Each module is assumed to be capable of running a sophisticated self-test routine, when prompted, and to report the result to other modules.

3. Reflective vs. Comparative Models

Throughout our discussions, each test is assumed to return a yes/no value, indicating that all is good (yes = 0) or something is wrong (no = 1). If there are q tests, then the syndrome is a q-bit vector S with S[j] holding the result of test j. The diagnosis problem is to deduce from the binary syndrome vector S[1:q] the diagnosis vector D[1:n] reflecting the health (0) or non-health (1) of each of the n modules in the system.

In the reflective mode of diagnosis, known in the literature as the PMC model [27], when a module is connected to another module, we assume that one is capable of testing the other one. Actually, not all links may be usable as testing links and a sub graph of the directed graph representing the system may be designated as the testing graph. In fact, the connectivity of the system may be completely different from the testing graph. It is possible, for example, for the n nodes to be connected via a bus, so that each node can potentially test any other one. This situation can be represented by K_n, the n-node complete graph, assuming that the single bus cannot be a source of problems in testing; that is, it is modeled either as a malfunction-free system core or a set of n(n – 1) independent directed channels.

From now on, we focus on the testing graph only and ignore the fact that there may be other links in the system besides those used for testing or that the hardware connectivity may in fact be less dense than the testing graph. The nature of the test can vary, from significant interaction of passing back and forth test patterns and test outcomes to minimal interaction, with one module initiating the test (perhaps by sending a key or seed value) and the target module carrying out a self-test routine. The key or seed value serves to ensure that the test result isn't a constant that a malfunctioning module may produce by accident or from a previously stored result in memory, thus compromising diagnostic accuracy.

The abstract reflective testing relationship is shown in figure 1a, where details of how a test is performed are suppressed and only the yes/no or 0/1 conclusion from the test is deemed relevant. The comparative testing relationship can be abstracted as in figure 1b, where a test manager u and two participants v and w are involved. The node u initiates the testing and the nodes v and w respond to it by each sending a test result to u. If the two test results are identical, u concludes that all is well, producing the decision 0, provided u itself is not malfunctioning. Non-matching results lead to the 1 decision by u. If the manager u is malfunctioning, then we make no assumption about the decision it might produce [19] [20].

In the reflective model, the tests correspond to the edges of the testing graph, one test per edge. Thus, we have |E| = q, the number of tests. In comparative testing, however, triples (u, v, w) of nodes correspond to tests, with the triples used pre-defined as part of the diagnostic scheme. Viewed in this way, we immediately see that the 2-way and 3-way relationships of reflective and comparative testing can readily be generalized to higher-degree collaborative testing, where clusters of nodes perform intra-cluster testing according to some local schema and the overall result is deduced from the collection of cluster-level tests. If clusters constitute replaceable units within our system, then it does not matter which nodes within a cluster are malfunctioning. Those can be diagnosed off-line and the requisite repairs performed in parallel with a new replacement cluster taking over.

4. One-Step vs. Multi-Step Diagnosis

Conceptually, the simplest diagnostic scheme is when a single round of q tests are performed and the resulting binary syndrome vector S[1:q] is used to deduce which nodes are healthy and which are malfunctioning. When the information in the syndrome vector is always enough to do the required diagnosis for up to t malfunctions, we say that the system is one-step t-diagnosable. Necessary and sufficient conditions are known for one-step diagnosability, that is, the mapping of the syndrome vector S[1:q] into the diagnostic vector D[1:n], which correctly identifies the health (0) or malfunctioning (1) status of each unit. Theorem 1 represents an example of theoretical results that are available for practical use.

Theorem 1. An n-unit system in which no two units test one another, is 1-step t-diagnosable if and only if each unit is tested by at least t other units.

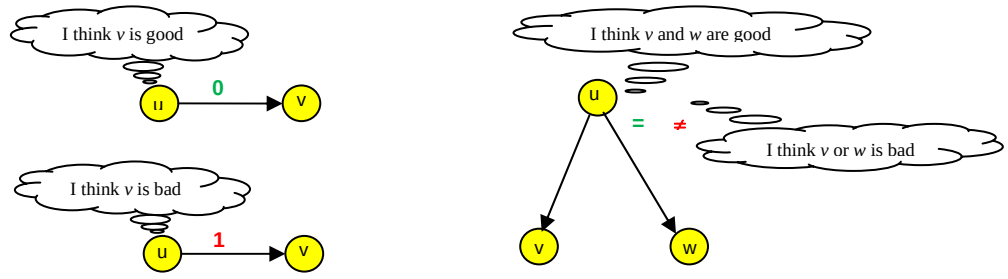
If, on the other hand, the syndrome vector isn't sufficient for full diagnosis but always leads to the identification of at least h malfunctioning units, h < m, we say that the system is multi-step t-diagnosable, because once the identified malfunctioning units have been repaired or replaced, the resulting system, which now has fewer malfunctioning units, can be subjected to the same process for identifying additional malfunctions. The extreme case where each diagnosis step identifies a single malfunctioning unit is referred to as "sequential diagnosis." A system is sequentially diagnosable if there exist a diagnosis strategy for it that guarantees the identification of at least one malfunctioning unit in each diagnosis step. Theorem 2 represents an example of theoretical results that are available with regard to sequential diagnosis, in this case a sufficient condition for sequential t-diagnosability.

Theorem 2. An n-unit system is sequentially t-diagnosable if the condition $n \geq 2t + 1$ holds. A majority of the n units being healthy is a sufficient condition, but it may not be necessary.

5. Sensitivity vs. Specificity of Diagnosis

The terms "sensitivity" and "specificity" are taken from the medical diagnosis domain. Suppose we have a population of individuals, mostly healthy but containing some who are afflicted with a particular disease. A medical test exists for the disease. The test can identify people afflicted with the disease (positive indication, or 1) and those not afflicted (negative indication, or 0), but it has some probability of yielding a false positive (identifying a healthy person as sick) and a certain probability of yielding a false negative (missing the detection of a sick person). Such a test is referred to as "sensitive" if it has a fairly small false-negative probability, that is, it detects nearly all sick individuals (Figure 2a). The test is dubbed "specific" if it has a fairly small false-positive possibility, that is, only a minute fraction of healthy individuals will be wrongly diagnosed as having the disease (Figure 2b).

In the context of studies on malfunction diagnosis, false negatives have not been allowed so far. Put another way, the diagnosis outcome can have healthy nodes marked as bad (this is a safe situation) but no malfunctioning node is allowed to be misidentified as healthy. However, there is no fundamental reason for excluding false negatives, if the system has some built-in malfunction tolerance capability that allows it to function correctly in the presence of a very small number of malfunctioning units. Such a system will use a combination of malfunction masking and malfunction diagnosis to continue correct operation in the presence of some malfunctions, aiming to remove malfunctions that put it over its tolerance capacity.



(a) Reflective testing (PMC model) (b) Comparative testing (MM* model)

Figure 1. Reflective and comparative testing abstractions

(a) A highly sensitive test (b) A highly specific test

Figure 2. Diagnostic sensitivity and specificity

6. Unrestricted vs. Conditional Malfunction Patterns

If the subset of m malfunctioning units can be arbitrary, the diagnosis scheme is unrestricted. This is the default assumption for any diagnosis scheme in which no restriction is mentioned.

The main kind of conditional diagnosis schemes studied thus far is when the m malfunctions are restricted not to include all neighbors of any node. When all neighbors of a node are malfunctioning, that node becomes isolated from healthy units and thus cannot be correctly diagnosed. This isolation poses a problem for the diagnosis algorithms, effectively restricting t to at most d – 1, where d is the minimum node degree, when no false positives are allowed. Recently, a stronger restriction, requiring each node to have at least g good neighbors, has been proposed. The g-good-neighbor diagnosability schemes requires each node to have at least g good neighbors, in which case t-diagnosability for larger values of t can be ensured. The previously-studied "conditional" diagnosability corresponds to the special 1-good-neighbor case of this more general scheme.

Unrestricted and conditional diagnosabilities can be combined in many different ways. For example, it is possible to prove that certain classes of networks are (t + a)-diagnosable, except when the pattern of malfunctions belongs to some undesirable class, in which case they become t-diagnosable. In other words, the absence of the undesirable malfunction patterns increases the diagnosability extent by a. An example of such combining is "strong diagnosability," where the level of diagnosability rises from t to t + 1 (that is, a = 1 in the formulation above) when every node possesses at least one healthy neighbor.

Again, more general conditions can be entertained. In a cluster-based hierarchical network, one may postulate that not all nodes in any given cluster be malfunctioning, that each cluster remain connected internally, or that at least one inter-cluster connection remain intact between any two clusters. The possibilities are quite varied. In general, a restriction on the malfunction pattern leads to some increase in the diagnosability extent.

7. Analysis vs. Synthesis Considerations

Diagnosability problems to be solved are of two types: analyzing diagnosabilities of known networks, and synthesizing interconnection architectures with desired diagnosability properties.

Analysis problem 1: Given a syndrome vector S[1:q], identify a set M that includes the requisite number of malfunctioning nodes (m, 1, or some other number, depending on the model used and the diagnostic strategy). Note that the suspected malfunction set M may be allowed to include false positives or prohibited from signaling false negatives.

In the simplest case, polynomial-time algorithms exist that take the vector S[1:q] and the testing graph as input and produce the set M when the set is restricted to contain all and only the m malfunctioning units. Efficient algorithms exist for certain other cases as well, though the space of possibilities has not been exhausted at this writing.

Analysis problem 2: Given a testing structure (testing graph of PMC, 3-groupings for MM*), identify the extent of diagnosability in the case of one-step, multi-step (including sequential), and other strategies for various unrestricted and conditional patterns of malfunctions.

Much work has been done in this area, including the derivation of diagnosability results for a wide array of known and newly proposed interconnection networks. The networks studied include meshes, tori [1], hypercubes [3] [11] [12] [14] [26] (or its generalizations [34] [36] [38]), k-ary n-cubes [1], numerous hypercube variants [16], cube-connected cycles, OTIS or swapped networks (including the biswapped variant), Cartesian product networks [1], and many other regular [5] [6] [18] [35] and hierarchical (multi-level) networks.

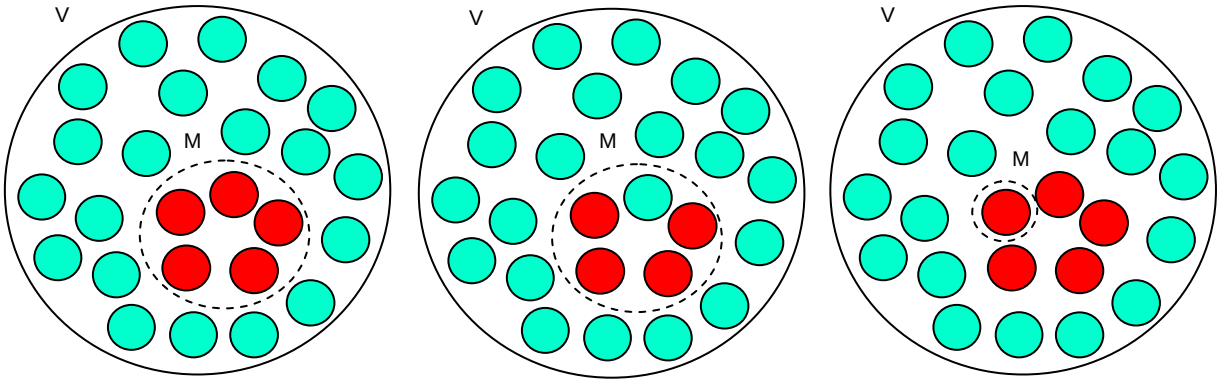
Synthesis problem: Given a desired diagnosability extent, the number of nodes, and other physical attributes, derive a testing graph that is optimal in some respect.

The synthesis problem is easy when only diagnosability is of interest, but becomes very challenging (like most combinatorial optimization problems) when other criteria are included.

8. How the Taxonomy Is Used

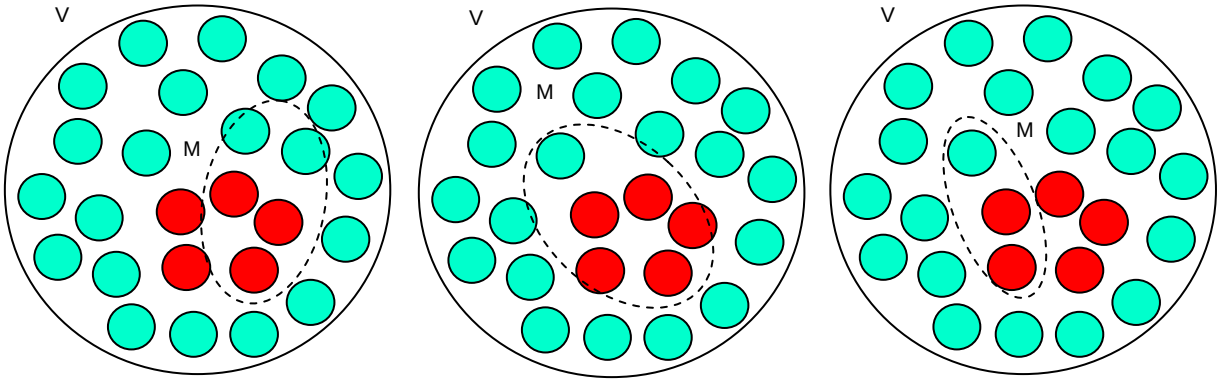
Our taxonomy essentially entails the mentioning of each of the four parameters t, T_p, F_p, and F_n in the form of $t/T_p/F_p/F_n$-diagnosability. These parameters also contain information about whether the scheme is one-step or multi-step (including sequential) and whether it is precise or pessimistic. This method of specifying a diagnostic scheme, including incorporation of the maximum number of false negatives as the last of four parameters is new. Existing diagnosis schemes do not allow false negatives (the corresponding number is 0 in our model), but, as mentioned in Section 5, there is no fundamental reason to exclude them forever. In the examples that follow, $F_n = 0$ and is thus not discussed explicitly.

The existing models correspond to the following scheme with our terminology:



(a) Precise 1-step 5-diagnosability (b) Pessimistic 1-step 5-diagnosability, |M| = t (c) Sequential 5-diagnosability

Figure 3. Some commonly studied diagnosis strategies and outcomes



(a) Imprecise 1-step 5-diagnosability (b) Pessimistic 5-diagnosability, |M| > t (c) Multi-step 5-diagnosability

Figure 4. Examples of diagnosis strategies allowing false negatives and |M| possibly going beyond t.

Precise = No false positives allowed, that is, $F_p = 0$

Pessimistic = Up to t – m or s – m (with s > t) false positives allowed

Example 1 (5/5/0/0-diangnosability): Up to 5 malfunctions are diagnosed with no false positives. This essentially specifies precise one-step 5-diagnosability with existing terminology (see figure 3a).

Example 2 (5/5/1/0-diagnosability): Up to 5 malfunctions are diagnosed, with the malfunctioning units identified to within a set of 5 units (up to 5 true positives and up to 1 false positive; see figure 3b).

Example 3 (5/1/0/0-disgnosability): One malfunction is diagnosed in each step, with no false positives. This corresponds to sequential 5-diagnosability (Figure 3c).

Example 4 (5/2/0/0-diagnosability): Up to 5 malfunctions are diagnosed, with the bad units identified in 3 steps (at least 2 true positives and no false negative at each step).

And here are a few examples, not yet studied, that entail false negatives.

Example 5 (5/5/2/2-diagnosability): Up to 5 malfunctions are allowed, with three of the malfunctioning units identified in one step to within a set of 5 units (at least 3 true positives and up to 2 false negatives; see figure 4a).

Example 6 (5/6/1/0-diagnosability): Up to 5 malfunctions are diagnosed, with the malfunctioning units identified to within a set of 6 units (up to one false positive; see figure 4b).

Example 7 (5/2/1/0-diagnosability): Up to 5 malfunctions are diagnosed in 3 steps, each step identifying 2 true positives and up to 1 false positive (Figure 4c).

9. Partial Survey of Prior Work

The references at the end of our paper contain a representative sample of work in the field of distributed malfunction diagnosis, both early work laying the foundations and more recent work developed within a mature field. It would be instructive to categorize these references with regard to our taxonomy. Tables 1 and 2 show the results of classification for reflective (PMC) and comparative (MM*) models of malfunction diagnosis.

Several patterns emerge from the survey of representative work reflected in tables 1 and 2. First, the synthesis problem has not received much attention, particularly within the comparative diagnosis model. Second, multi-step diagnosis, which is often a more difficult problem from a theoretical standpoint, has not been the focus of much work. Studies on single-step diagnosis are dominant, particularly with comparative methods. High-specificity diagnosis has received more attention than low-specificity versions.

It is also evident from tables 1 and 2 that the sensitivity of diagnosis has been completely ignored (this is why our tables do not include columns for this attribute).

Table 1. Categorization of prior work on reflective malfunction diagnosis (PMC model)

Reference Citation	**Paper's Aim** Analysis / Synthesis	**Steps** Single / Multiple	**Specificity** High / Low	**Qualification** Unrestricted / Conditional
[1] Araki & Shibata 2000	Analysis	Single	Both	Unrestricted
[2] Araki & Shibata 2003	Analysis	Multiple	High	Unrestricted
[3] Armstrong & Gray 1981	Analysis	Single	High	Unrestricted
[4] Barsi et al. 1976	Synthesis	Both	High	Unrestricted
[5] G. Y. Chang et al. 2005	Analysis	Single	Both	Unrestricted
[6] G. Y. Chang 2010	Analysis	Multiple	High	Unrestricted
[7] N. W. Chang & Hsieh 2012	Analysis	Single	High	Conditional
[8] Hakimi & Amin 1974	Analysis	Single	High	Unrestricted
[13] Karunathini & Friedman 1979	Analysis	Both	Low	Unrestricted
[14] Kavianpour & Kim 1991	Analysis	Single	Low	Unrestricted
[15] Lai et al. 2005	Analysis	Single	High	Conditional
[16] Lin et al. 2014	Analysis	Single	High	Conditional
[17] Lin et al. 2015	Analysis	Single	High	Conditional
[18] Lin et al. 2016	Analysis	Single	Low	Unrestricted
[26] Peng et al. 2012	Analysis	Single	High	Conditional
[27] Preparata et al. 1967	Analysis	Both	High	Unrestricted
[30] Somani et al. 1987	Synthesis	Single	High	Unrestricted
[31] Somani et al. 1996	Analysis	Single	Low	Unrestricted
[32] Tsai & Chen 2013	Analysis	Single	Both	Unrestricted
[34] M. Xu et al. 2009	Analysis	Single	High	Conditional
[35] L. Xu et al 2016	Analysis	Single	Both	Both
[38] Zhu 2008	Analysis	Single	High	Conditional
[39] Zhu et al. 2014	Analysis	Single	High	Both

Table 2. Categorization of prior work on comparative malfunction diagnosis (MM* model)

Reference Citation	**Paper's Aim** Analysis / Synthesis	**Steps** Single / Multiple	**Specificity** High / Low	**Qualification** Unrestricted / Conditional
[5] G. Y. Chang et al. 2005	Analysis	Single	Both	Unrestricted
[10] Hong & Hsieh 2012	Analysis	Single	High	Both
[11] Hsieh & Kao 2013	Analysis	Single	High	Conditional
[12] Hsu et al. 2009	Analysis	Single	High	Conditional
[17] Lin et al. 2015	Analysis	Single	High	Conditional
[19] Maeng & Malek 1981	Analysis	Single	High	Unrestricted
[20] Malek 1980	Analysis	Single	High	Unrestricted
[29] Sengupta & Dabbura 1992	Analysis	Single	High	Unrestricted
[36] Yang 2013	Analysis	Single	High	Conditional
[39] Zhu et al. 2014	Analysis	Single	High	Both
[40] Ziwich & Duarte 2016	Analysis	Single	High	Unrestricted

Other areas where there is no work yet include hierarchical or cluster-based diagnosis. Numerous hierarchical or multi-level interconnection schemes have been proposed based on hypercube [9] and its many variants [21] [28]. There are also interesting hierarchical interconnection architectures that are grown from arbitrary basis topologies. A prime example is the class of swapped or OTIS networks [22] [37], and their symmetric variants known as biswapped networks [33], which have been the subjects of very limited diagnosability studies [32].

10. Conclusion and Future Work

The nomenclature and taxonomy introduced in this paper puts the field of malfunction diagnosis into a much-needed order, allowing a uniform formulation of the problems already explored and the exposure of additional possibilities not yet investigated. The various diagnostic strategies are expressed in terms of the four parameters t, T_p, F_p, and F_n that collectively specify not only the extent of diagnosability but also whether the scheme is 1-step, multi-step, precise, or pessimistic in the prevailing terminology.

The idea of allowing false positives in the diagnostic scheme isn't new, but the explication of the number of false positives allowed as a model parameter is helpful and removes some of the ambiguities in the current nomenclature. False positives aren't as undesirable as they once were, given that the steep reduction in hardware cost makes system down time considerations much more important than the loss of a healthy unit. In fact, a unit falsely identified as malfunctioning may only be lost temporarily, because off-line testing can verify that the unit is in fact good, allowing it to return to the spare supply. False negatives, on the other hand are new to our model. The presence of some malfunctioning units may be tolerated by a system's built-in malfunction tolerance, which may include replicated computation with voting or data replication with primary and back-up nodes.

We plan to work on further refining this taxonomy as we discover diagnostic schemes that it does not properly cover or see the need for additional expressive power as system complexity and diagnostic strategies evolve.

Appendix

List of Symbols

a Additional diagnosability beyond t under special circumstances
D Diagnosis binary vector of length n
d Minimum node degree in G
E Set of edges of the testing graph, with $|E| \geq q$
F_n Number of false negatives allowed by the testing strategy
F_p Number of false positives allowed by the testing strategy
G The testing directed graph, G = (V, E)
g Minimum number of good neighbors for each node assumed in some conditional models
h Minimum number of malfunctioning units (true positives) included in M
K_n Complete graph of n nodes
k Bound on the number of false positives in previous terminology (our F_p)
M Set of purportedly malfunctioning units returned by the diagnosis algorithm; $|M| = T_p + F_p$
m Actual number of malfunctioning units, $m \leq t$
n Number of nodes in the network or testing graph (length of the binary diagnosis vector D[1:n])
q Number of tests performed in one step (length of the binary syndrome vector S[1:q])
S Syndrome binary vector
s Bound on the size of the returned set M, with $s > t$
t Upper bound on the number of malfunctioning nodes
T_p Number of true positives (correctly diagnosed malfunctioning units) by the testing strategy
u Graph node doing the testing or coordination
V Set of system nodes, with $|V| = n$
v Graph node under test by u
w Second graph node under test by u in the comparative model

References

[1] T. Araki, and Y. Shibata, "Diagnosability of Networks Represented by the Cartesian Product," *IEICE Trans. Fundamentals*, vol. E83-A, no. 3, pp. 465-470, 2000.

[2] T. Araki, and Y. Shibata, "(*t, k*)-Diagnosable System: A Generalization of the PMC Models," *IEEE Trans. Computers*, vol. 52, no. 7, pp. 971-975, 2003.

[3] J. R. Armstrong, and F. G. Gray, "Fault Diagnosis in a Boolean *n*-Cube Array of Microprocessors," *IEEE Trans. Computers*, vol. 30, no. 8, pp. 587-590, 1981.

[4] F. Barsi, N. Grandoni, and P. Maestrini, "A Theory of Diagnosability of Digital Systems," *IEEE Trans. Computers*, vol. 25, no. 6, pp. 585-593, 1976.

[5] G.-Y. Chang, G. J. Chang, and G.-H. Chen, "Diagnosabilities of Regular Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 4, pp. 314-323, 2005.

[6] G. Y. Chang, "(*t, k*)-Diagnosability for Regular Networks," *IEEE Trans. Computers*, vol. 59, no. 9, pp. 1153-1157, 2010.

[7] N. W. Chang, and S. Y. Hsieh, "Conditional Diagnosability of Augmented Cubes under the PMC Model," *IEEE Trans. Dependable and Secure Computing*, vol. 9, no. 1, pp. 46-60, 2012.

[8] S. L. Hakimi, and A. T. Amin, "Characterization of Connection Assignment of Diagnosable Systems," *IEEE Trans. Computers*, vol. 23, no. 1, pp. 86-88, 1974.

[9] J. P. Hayes, and T. Mudge, "Hypercube Supercomputers," *Proc. IEEE*, vol. 77, no. 12, pp. 1829-1841, 1989.

[10] W. S. Hong, and S. Y. Hsieh, "Strong Diagnosability and Conditional Diagnosability of Augmented Cubes under the Comparison Diagnosis Model," *IEEE Trans. Reliability*, vol. 61, no. 1, pp. 140-148, 2012.

[11] S. Y. Hsieh, and C. Y. Kao, "The Conditional Diagnosability of *k*-ary *n*-cubes under the Comparison Diagnosis Model," *IEEE Trans. Computers*, vol. 62, no. 4, pp. 839-843, 2013.

[12] G. H. Hsu, C. F. Chiang, L. M. Shih, L. H. Hsu, and J. J. Tan, "Conditional Diagnosability of Hypercubes under the Comparison Diagnosis Model," *J. Systems Architecture*, vol. 55, no. 2, pp. 140-146, 2009.

[13] S. Karunanithi, and A. D. Friedman, "Analysis of Digital Systems Using a New Measure of System Diagnosis," *IEEE Trans. Computers*, vol. 28, no. 2, pp. 121-133, 1979.

[14] A. Kavianpour, and K. H. Kim, "Diagnosabilities of Hypercubes under the Pessimistic One-Step Diagnosis Strategy" *IEEE Trans. Computers*, vol. 40, no. 2, pp. 232-237, 1991.

[15] P. L. Lai, J. J. Tan, C. P. Chang, and L. H. Hsu, "Conditional Diagnosability Measures for Large Multiprocessor System," *IEEE Trans. Computers*, vol. 54, no. 2, pp. 165-175, 2005.

[16] L. Lin, S. Zhou, L. Xu, and D. Wang, "Conditional Diagnosability of Arrangement Graphs under the PMC Model," *Theoretical Computer Science*, vol. 548, pp. 79-97, 2014.

[17] L. Lin, L. Xu, D. Wang, and S. Zhou, "The *g*-Good-Neighbor Conditional Diagnosability of Arrangement Graphs," *IEEE Trans. Dependable and Secure Computing*, December 2015.

[18] L. Lin, L. Xu, S. Zhou, and S. Y. Hsieh, "The *t*/*k*-Diagnosability for Regular Networks," *IEEE Trans. Computers*, vol. 65, no. 10, pp. 3157-3170, 2016.

[19] J. Maeng, and M. Malek, "A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems," *Proc. 11th Int'l Symp. Fault-Tolerant Computing*, 1981, pp. 173-175.

[20] M. Malek, "A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems," *Proc. 7th Symp. Computer Architecture*, pp. 31-36, 1980.

[21] B. Parhami, *Introduction to Parallel Processing: Algorithms and Architectures*, Plenum, 1999.

[22] B. Parhami, "Swapped Interconnection Networks: Topological, Performance, and Robustness Attributes," *J. Parallel and Distributed Computing*, vol. 65, no. 11, pp. 1443-1452, 2005.

[23] B. Parhami, "A Puzzle-Based Seminar for Computer Engineering Freshmen," *Computer Science Education*, vol. 18, no. 4, pp. 261-277, 2008.

[24] B. Parhami, "Motivating Computer Engineering Freshmen Through Mathematical and Logical Puzzles," *IEEE Trans. Education*, vol. 52, no. 3, pp. 360-364, 2009.

[25] B. Parhami, *Dependable Computing: A Multi-Level Approach*, graduate-level textbook in development, available on-line at: http://www.ece.ucsb.edu/~parhami/text_dep_comp.htm.

[26] S.-L. Peng, C.-K. Lin, J. J. M. Tan, and L.-H. Hsu, "The g-Good-Neighbor Conditional Diagnosability of Hypercube under PMC Model," *Applied Mathematics and Computation*, vol. 218, pp. 10406-10412, 2012.

[27] F. P. Preparata, G. Metze, and R. T. Chien, R. T., "On the Connection Assignment Problem of Diagnosable Systems," *IEEE Trans. Electronic Computers*, vol. 16, no. 6, pp. 848-854, 1967.

[28] F. P. Preparata, and J. Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation," *Communications of the ACM*, vol. 24, no. 5, pp. 300-309, 1981.

[29] A. Sengupta, and A. T. Dahbura, "On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach," *IEEE Trans. Computers*, vol. 41, no. 11, pp. 1386-1396, 1992.

[30] A. K. Somani, V. K. Agarwal, and D. Avis, "A Generalized Theory for System Level Diagnosis," *IEEE Trans. Computers*, vol. 36, pp. 538-546, 1987.

[31] A. K. Somani, and O. Peleg, "On Diagnosability of Large Fault Sets in Regular Topology-Based Computer Systems," *IEEE Trans. Computers*, vol. 45, no. 8, pp. 892-903, 1996.

[32] C.-H. Tsai, and J.-C. Chen, "Fault Isolation and Identification in General Biswapped Networks under the PMC Diagnostic Model," *Theoretical Computer Science*, vol. 501, pp. 62-71, 2013.

[33] W. J. Xiao, B. Parhami, W. D. Chen, M. X. He, and W. H. Wei "Fully Symmetric Swapped Networks Based on Bipartite Cluster Connectivity," *Information Processing Letters*, vol. 110, no. 6, pp. 211-215, 2010.

[34] M. Xu, K. Thulasiraman, and X. D. Hu, "Conditional Diagnosability of Matching Composition Networks under the PMC Model," *IEEE Trans. Circuits and Systems II*, vol. 56, no. 11, pp. 875-879, 2009.

[35] L. Xu, L. Lin, S. Zhu, and S.-Y. Hsieh, "The Extra Connectivity, Extra Conditional Diagnosability, and *t*/*m*-Diagnosability of Arrangement Graphs," *IEEE Trans. Reliability*, vol. 65, no. 3, pp. 1248-1262, September 2016.

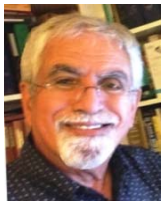
[36] M. C. Yang, "Conditional Diagnosability of Matching Composition Networks under the MM∗ Model," *Information Sciences*, vol. 233, pp. 230-243, 2013.

[37] F. Zane, P. Marchand, R. Paturi, and S. Esener, "Scalable Network Architectures Using the Optical Transpose Interconnection System (OTIS)," *J. Parallel and Distributed Computing*, vol. 60, no. 5, pp. 521-538, 2000.

[38] Q. Zhu, "On Conditional Diagnosability and Reliability of the BC Networks," *J. Supercomputing*, vol. 45, no. 2, pp. 173-184, 2008.

[39] Q. Zhu, G. Guo, and D. Wang, "Relating Diagnosability, Strong Diagnosability and Conditional Diagnosability of Strong Networks," *IEEE Trans. Computers*, vol. 63, no. 7, pp. 1847-1851, 2014.

[40] R. P. Ziwich, and E. P. Duarte, "A Nearly Optimal Comparison-Based Diagnosis Algorithm for Systems of Arbitrary Topology," *IEEE Trans. Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3131-3143, November 2016.



Behrooz Parhami (PhD, UCLA 1973) is Professor of Electrical and Computer Engineering, and former Associate Dean for Academic Personnel, College of Engineering, at University of California, Santa Barbara, where he teaches and does research in computer arithmetic, parallel processing, and dependable computing. A Life Fellow of IEEE, a Fellow of IET and British Computer Society, and recipient of several other awards (including a most-cited paper award from J. Parallel & Distributed Computing), he has written six textbooks and more than 290 peer-reviewed technical papers. Professionally, he serves on journal editorial boards and conference program committees and is also active in technical consulting.
E-mail: parhami@ece.ucsb.edu

Nan Wu is a doctoral student in Computer Engineering at Department of Electrical and Computer Engineering, University of California, Santa Barbara. Born in 1994, she received a BE degree in electronics engineering from Tsinghua University, Beijing, China, in 2016. Her research interests include applications of emerging technologies and spiking neural networks.
E-mail: nanwu@umail.ucsb.edu

Sixin Tao is a master's student in Computer Engineering at Department of Electrical and Computer Engineering, University of California, Santa Barbara. Born in 1992 in Jiangsu Province, she received a Bachelor of Engineering degree from College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, China, in 2015.
E-mail: sixin@umail.ucsb.edu

Paper Handling Data:

Submitted: 28.11.2016
Received in revised form: 16.12.2016
Accepted: 02.01.2017
Corresponding author: Prof. Behrooz Parhami, Department of Electrical and Computer Engineering, University of California, Santa Barbara, California, USA.

The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 32-41
Regular Paper

Exploring Reconfigurability Options Among Decimal Adders

Samaneh Emami Mehdi Sedighi

Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

Abstract

Decimal arithmetic has become a hot research topic in recent years. Many hardware units have been designed and proposed to perform high performance and accurate decimal arithmetic operations. Traditionally, decimal arithmetic units have been designed as application-specific specialized hardware modules. But there is an emerging trend towards the design and implementation of reconfigurable structures to perform decimal arithmetic. This paper contributes to this trend by exploring different reconfigurability options in decimal adders, proposing new reconfigurable parallel prefix trees (PPTs), and presenting a reconfigurable combined binary/decimal adder with a variable input width. Our analysis shows that it is possible to combine two conventional PPTs to reach a reconfigurable version with a reasonable overhead. Furthermore, we will suggest two criteria for choosing which PPTs to combine and will compare these two criteria. Experimental results demonstrate that the reconfigurability in the proposed designs comes at the cost of at most 5% overhead in area.

Keywords: Decimal Arithmetic, Parallel Prefix Tree, Decimal Adder, Reconfigurable Hardware, Granularity.

1. Introduction

Human beings have traditionally used decimal arithmetic probably because of having ten fingers. As such, using decimal arithmetic appeared to be the natural choice for the first generation of computers such as ENIAC [1], UNIVAC [2], and IBM 650 [3]. Over time, however, the decimal arithmetic modules were gradually replaced by their easier-to-implement binary counterparts to the extent that binary arithmetic became the prevailing choice for designers of computer systems. But there is growing evidence that decimal arithmetic is emerging again in the modern computers. For instance, modern processors like IBM Power and System z processors [4-7] have specialized decimal hardware units.

One of the main reasons for the resurgence of decimal arithmetic is its accurate representation of decimal floating point (DFP) numbers. In contrast, a major problem of binary arithmetic is its inaccuracy in representing some non-integer numbers such as 0.1 that can cause unacceptable errors in financial and commercial applications. For instance, it has been reported that in a large telephone billing system, using binary arithmetic instead of decimal can result in an estimated annual loss of up to five million dollars [8]. It is safe to say that the lower precision of binary arithmetic in certain applications has become an inhibiting factor in modern computers that handle those applications.

A good evidence for the increasing importance of decimal arithmetic in recent years is that the IEEE 754-2008 standard for floating-point arithmetic [9] includes specifications for handling decimal floating-point numbers. In this standard, two decimal number formats for both software and hardware implementations of decimal arithmetic are presented:

Binary Integer Decimal (BID) is proposed for software implementations of decimal arithmetic and is used in IBM dec Floats modules [10], Intel DFP Math Library [11], and built-in GCC DFP types [12]. These implementations can eliminate the inaccurate representation errors, but they are usually slow and inefficient [13].

Densely Packed Decimal (DPD) is recommended for hardware implementations of decimal arithmetic and is used

in machines that have dedicated decimal hardware units, like IBM Power and System z processors [4-7]. These implementations usually provide both accuracy and performance [14]. Additionally, the dramatic increase in chip density has lowered the overall cost of hardware implementations. Consequently, hardware solutions are gaining prominence in industry [15].

There is also a recent trend toward the design and implementation of reconfigurable arithmetic hardware units. Traditionally, the reconfigurable platforms have been fine-grain modules found in commercial Field Programmable Gate Arrays (FPGA). Fine-grain modules can address bit-level granularity, but generally suffer from high reconfiguration overheads. The ultimate design goal would be to achieve an ASIC-like performance and FPGA-like flexibility, design time and cost. Therefore, the coarse-grain reconfigurable units appear to be a promising compromise between ASIC and FPGA.

While there are numerous studies on various hardware implementations of decimal arithmetic operations [16-21], there are only a few works that focus on the reconfigurebility aspects of designing such hardware. This paper explores different reconfigurability options and possibilities in decimal adders. To do so, the idea of combining existing conventional Parallel Prefix Trees (PPTs) will be considered first. As will be elaborated in detail, it is possible to combine two conventional PPTs to reach a reconfigurable version with a reasonable overhead. In doing so, one has to choose the original PPTs carefully in order to achieve low latency and area overheads. As such, we will suggest two criteria for choosing which PPTs to combine and will compare these two criteria. Some new reconfigurable PPTs will be introduced based on these criteria. Another aspect of reconfigurability can be in the form of varying input size and type. So a reconfigurable combined binary/decimal adder with a variable input width will be justified and presented subsequently.

The remainder of this paper is organized as follows: Section 2 provides a review of the literature in this field. Section 3 discusses the proposed reconfigurable architectures in detail. Synthesis results and their analysis are provided in Section 4. The paper is concluded in Section 5.

2. Prior Works

There are numerous studies on various implementations of basic arithmetic operations such as addition, subtraction, multiplication, and division. Among these studies, some focus on optimized but inflexible decimal hardware implementations [19-21] and some present decimal arithmetic units with some degree of flexibility [22-27]. For instance, some researchers have focused on efficient implementations of decimal arithmetic operations on reconfigurable architectures such as FPGAs. This is because the FPGA implementation can provide an added flexibility in terms of compliance with various standards and the desired objective that the implemented design is trying to reach. Nannarelli [22], for example, has studied FPGA-based acceleration of decimal arithmetic operations.

This study shows that applications requiring decimal operations can be expedited by an arithmetic processor implemented on an FPGA board that connects to a computer. This processor ran a telephone billing application and achieved a speed-up of around 10 over its execution on the CPU of the host computer. This achievement is mainly due to more flexibility in the FPGA implementation with respect to ASIC design. Vazquez and Dine chin [23] have also presented a new method for fast implementation of multi-operand decimal addition in current FPGAs. This method is based on pre- and post-corrections of the binary sum. With the pre-corrections, the hexadecimal carries correctly serve as decimal carries, which brings about the opportunity to utilize built-in carry chain in FPGAs. As a result, the authors have reported that their implementation on a Virtex-6 FPGA device halves the area and latency of previous reconfigurable implementations [24].

There are other works that propose reconfigurable architectures for efficient implementation of decimal arithmetic. Such architectures may have an inherent flexibility in terms of the input format and width, or operands' radices and implementation. Combined binary/decimal arithmetic circuits can also be included in this category. For example, Calderon et al. [25] have proposed a new adder/sub tractor unit. Their implementation can operate on sign-magnitude, unsigned, and different complement representations. Another similar work [26] presented a novel combined adder/sub tractor arithmetic unit for binary, BCD (Binary Coded Decimal), and single precision BFP (Binary Floating Point) representations.

Another example for a fully reconfigurable architecture specialized for efficient implementation of binary arithmetic can be found in [27]. This work employs Coarse Grain Reconfigurable Architectures (CGRAs) which are a compromise between ASICs and FPGAs since they provide better computational efficiency compared to FPGAs and better engineering efficiency compared to ASICs. The CGRA fabric introduced, called Dynamically Reconfigurable Resource Array (DRRA), is a parallel digital signal processing fabric with distributed arithmetic, logic, interconnect and control resources.

Overall, one can safely say that reconfigurability in arithmetic circuits may take different forms and shapes. The previous works have considered some of them. But there is still room to explore other aspects of reconfigurability such as hybrid adders that implement more than one Parallel Prefix Tree (PPT) depending on their configuration, or a combination of binary and decimal addition with variable width. These ideas will be discussed in the next section.

3. Proposed Architectures

Addition is a very basic arithmetic operation. Fast adders are needed in virtually any digital system. They are also necessary in other arithmetic operations like multiplication. The speed of an adder is usually governed by its ability to quickly handle the carry chain. A common structure for fast computation of the carry signal is called a Parallel Prefix Tree (PPT). Six conventional PPTs have been introduced in the past [28-33]. These PPTs differ in terms of critical path delay, area, fan-out, wiring tracks, and number of levels (or tree depth). That is why each one may be appropriate in some applications and inappropriate in others. However, there are applications in which two or more of these

conventional PPTs might be helpful or necessary. In such applications it will be beneficial to have a hybrid reconfigurable PPT that provides the advantages of more than one PPT but at the cost of one single hardware. This idea will be discussed later in this paper.

In all binary PPTs, the inputs are calculated from $p_i(= x_i + y_i)$ and $g_i(= x_i . y_i)$, where $x_i$and y_i are two corresponding bits of addend and augend. Also, every node of the tree presents carry propagation (P) and carry generation (G) signals. The implementation of each node in the PPTs is shown in figure 1. At the lowest level of trees, carry of each position is ready to be added to the partial sum in order to produce the final result.

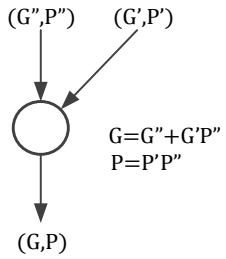


Figure 1. Implementation of a node in decimal PPTs

The implementation of a decimal PPT is similar to its binary counterpart, but its inputs, called P_i and G_i, are calculated using the following equations, where p_i^j and g_i^j are the corresponding binary propagate and generate signals:

$$P_i = p_i^3 p_i^2 p_i^1 p_i^0$$
$$G_i = g_i^3 + g_i^2 p_i^3 + g_i^1 p_i^2 p_i^3 + g_i^0 p_i^1 p_i^2 p_i^3 \quad (1)$$

The notion of reconfigurability in adders can take many different forms. In the next subsections, two of these various forms and possibilities will be explored: the prospect of combining two existing PPTs to form a new hybrid PPT that can be reconfigured into either of its two parent PPTs whereby providing their advantages albeit with some overhead; and the possibility of having a binary/decimal adder with a configurable input width.

3.1. Proposed Reconfigurable PPTs

The proposed reconfigurable PPT in this section can implement two different types of conventional PPTs such as Brent-Kung and Han-Carlson. The first question in designing such a hybrid PPT, is which two PPTs may be combined efficiently well to compensate for the inevitable reconfiguration cost. The preliminary answer to this question might be that trees with equal number of nodes appear to be suitable candidates. To explore this point, Ladner-Fischer (L-F) and Han-Carlson (H-C) trees were considered.

The implementations of L-F and H-C PPTs for 16-bit inputs with four and five levels are shown in figures 2 and 3, respectively. The gray nodes in both figures have the same inputs. These nodes can be shared easily in the hybrid PPT. The white nodes (in figure 2) and black nodes (in figure 3) have differing nodes in the two PPTs.

As the first step, a combined PPT with the same number of nodes as the original L-F and H-C trees was formed. This tree is depicted in figure 4. As this figure shows, the gray nodes require no changes in the data path. However, the white and black nodes need multiplexers to configure their varying data path according to the desired configuration. Needless to say, the multiplexers' selector signals determine which PPT is selected but for simplicity, they are not shown in the figure. Another important point in this figure is that the number of levels in the combined PPT is the same as the PPT with the higher number of levels. In this case, since H-C tree has 5 levels, the combined version also has 5 levels.

Multiplexers usually have considerable area and delay overheads. If they are implemented using complex gates (as opposed to transmission gates), their area and delay can even be more than a cell in the tree. So it seems plausible to assume that if redundant nodes are used instead of a shared node and multiplexer(s), the overall delay and area overhead would be smaller. As such, in the second step, the common nodes with same inputs (gray) remain intact. But the differing nodes are placed twice in the reconfigurable design. The resulting structure is shown in figure 5. In this figure, the gray nodes are shared between two PPTs whereas white and black nodes correspond to L-F and H-C PPT's, respectively. Notice that the redundant nodes are marked by the same numbers and only one of them is activated in each configuration.

The usage of redundant nodes in the PPT shown in figure 5 eliminates the need for the multiplexers whereby improving the area and delay of the reconfigurable PPT, as will be shown in Section 4. However, to explore the effect of structural matching between the original PPTs on the characteristics of the reconfigurable PPT, the first criterion used in selecting the basic trees (same number of nodes) was changed to structural similarity in the structures of chosen PPTs. The structural similarity was defined as the number of common (gray) nodes that can be found in the two PPTs. A careful examination of Han-Carlson and Brent-Kung (B-K) reveals that for 16-bit inputs, H-C has 32 nodes in five levels and B-K has 26 nodes in six levels. Figure 6 illustrates the implementation of B-K PPT.

In order to remain consistent with H-C node numbers, the nodes are not enumerated sequentially in this figure. Even though the total number of nodes in the PPTs are different, but there are 18 common nodes between H-C and B-K. So, these two were identified as similar structures. The common nodes were shared in the combined PPT. As for the nodes with different inputs, if only one input of a node is different (e.g., node 19), a MUX is used to steer the logic properly. But if both inputs of corresponding nodes are different (e.g., node 26), then instead of sharing the node and using MUX is, both nodes are kept. Figure 7 shows the resulting reconfigurable PPT that can implement H-C PPT (with gray and black nodes) or B-K PPT (with gray and white nodes).

As will be discussed in Section 4, the experimental results show that this approach provides superior results compared to the other approaches depicted in figures 4 and 5. Therefore, one may conclude that if a hybrid reconfigurable PPT is desired, then choosing the PPTs based on their structural similarity leads to better designs with smaller area and delay overheads compared to just considering equal number of nodes in them. To explore another kind of reconfigurability, a reconfigurable adder that takes operands of different types and sizes is introduced next.

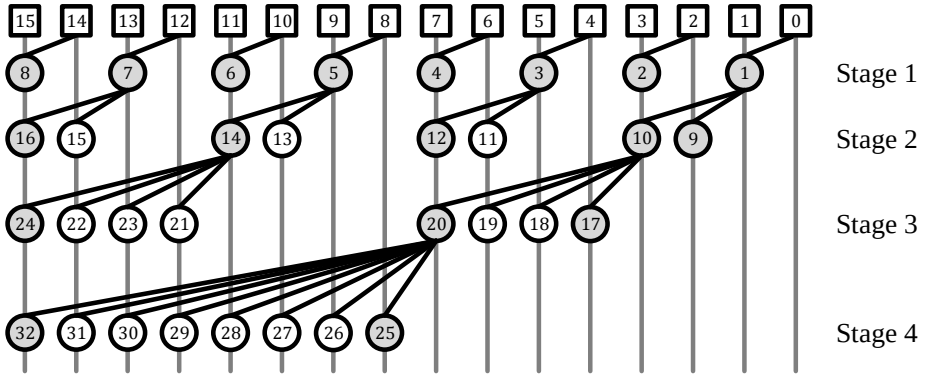


Figure 2. L-F Parallel Prefix Tree (reproduced from [33])

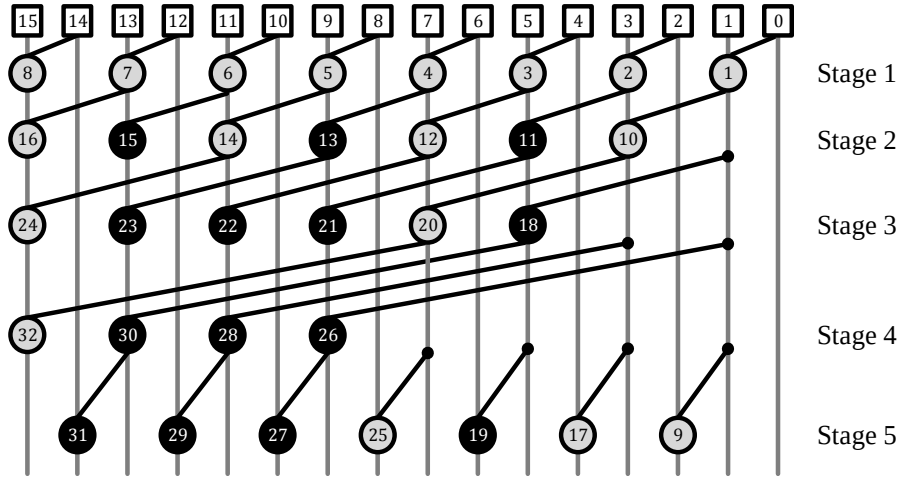


Figure 3. H-C Parallel Prefix Tree (reproduced from [31])

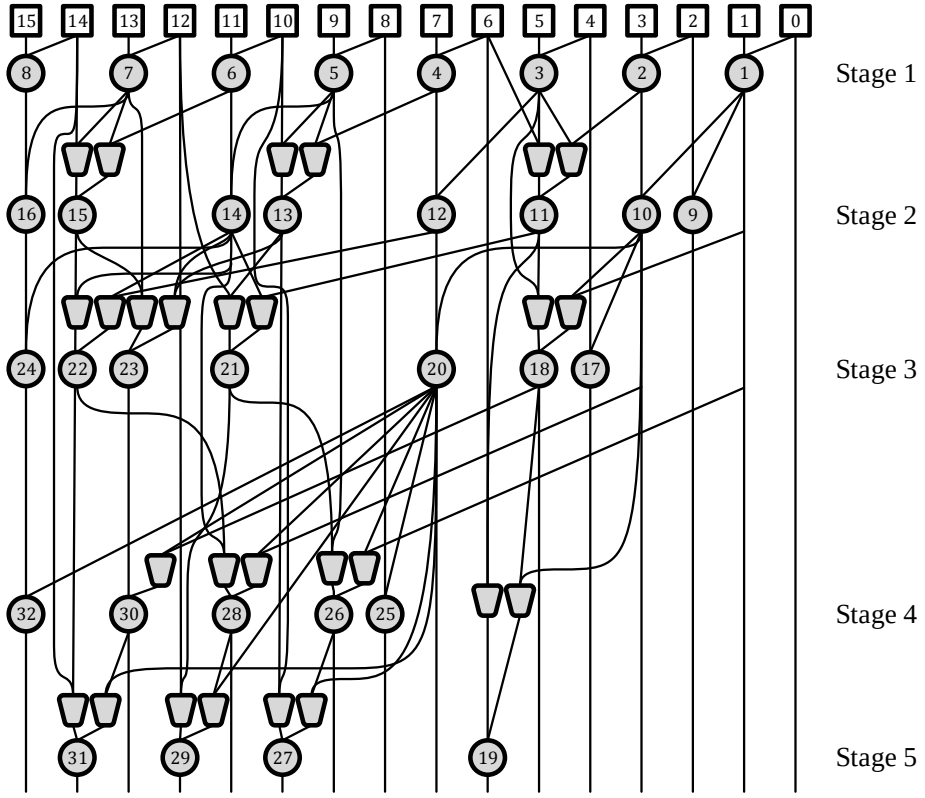


Figure 4. Reconfigurable PPT (Combined L-F/H-C), Step 1

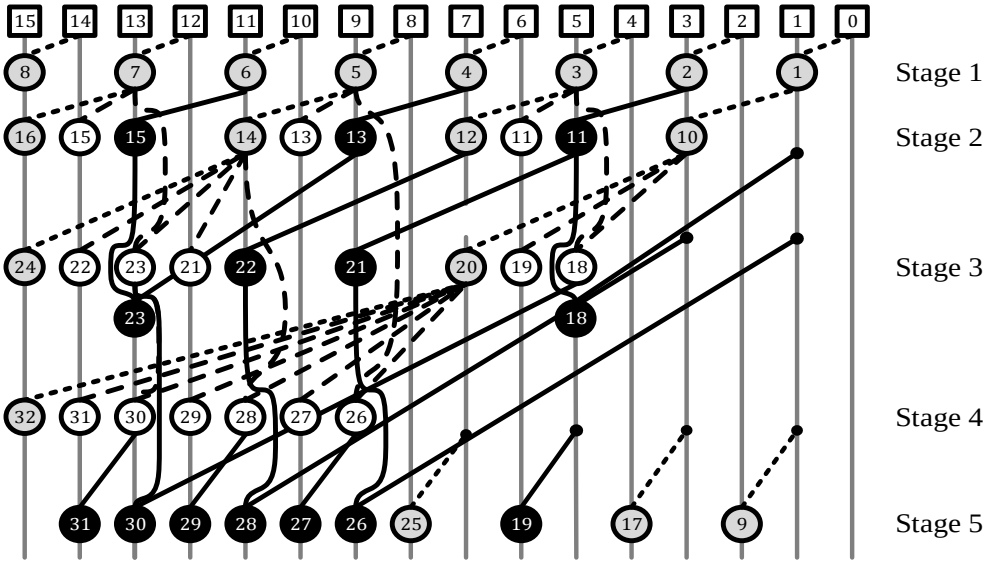


Figure 5. Reconfigurable PPT (Combined L-F/H-C), Step 2

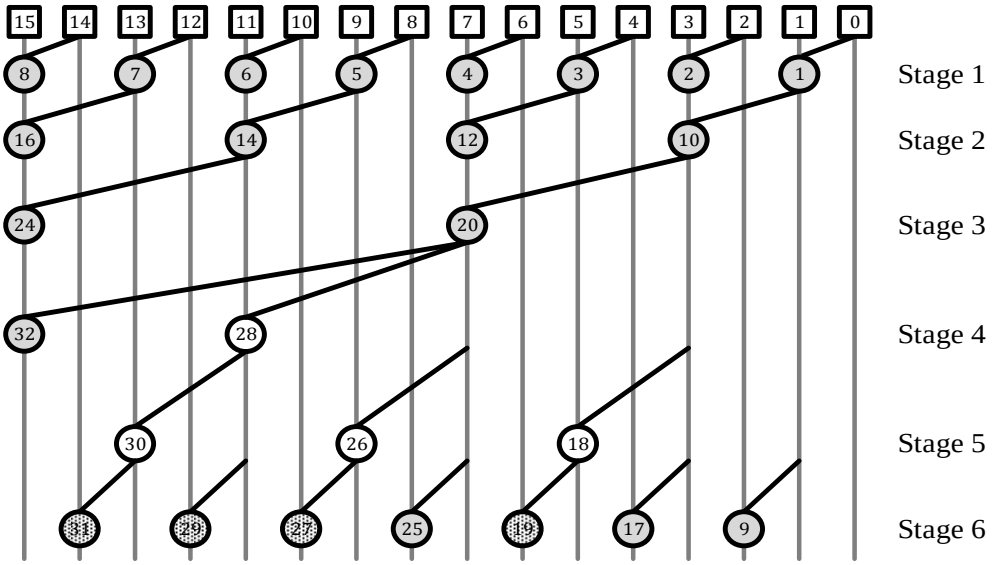


Figure 6. B-K Parallel Prefix Tree (reproduced from [28])

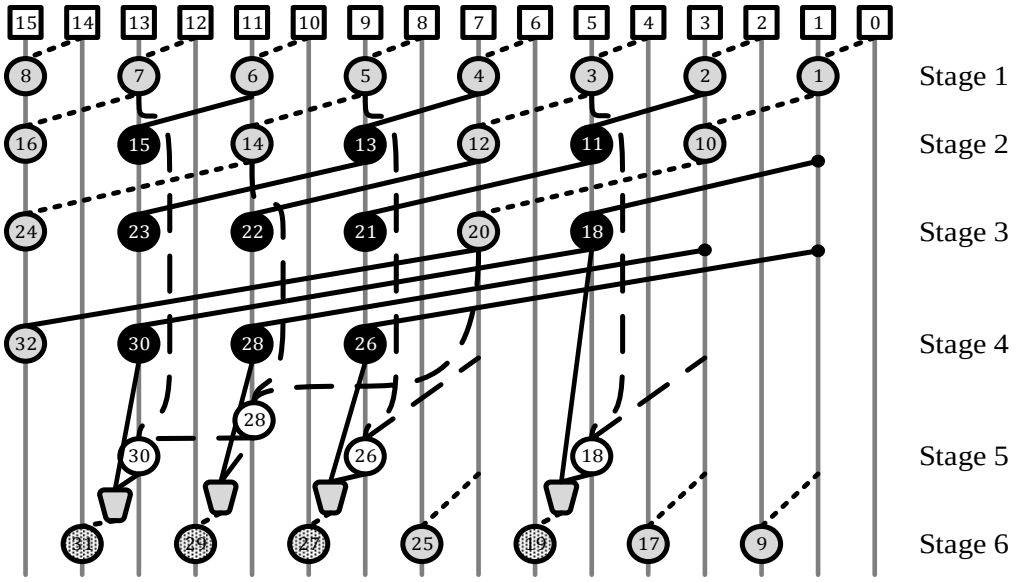


Figure 7. Reconfigurable PPT (Combined H-C/B-K), Step 3

3.2. Proposed Reconfigurable Adder

Modern processors are generally designed with a wide (like 64 bits) data path. A wide data path usually necessitates a wide ALU as well. For instance, a 64-bit processor requires a 64-bit adder. Research has shown that there are many applications in which a sizeable portion of operations are executed on shorter data structures (such as Smallint) as compared to longer ones (like Long) supported by the hardware [34]. The statistics found in literature indicates that 60% of binary operations contained in various applications such as airline systems, banking, financial analysis, insurance, etc. use 16-bit Smallint variables and only the remaining 40% of the operations need 32-bit Integers even though the processors still have to provide support for 32-bit operations [34].

In those circumstances, a wide adder is not always needed and having the option to configure the wide adder as two narrower adders that can work in parallel can offer a potentially considerable improvement in terms of throughput and/or performance. Furthermore, if equipped with power gating mechanism, the upper half of a wide adder can be switched off in case only the lower half is used whereby providing a power saving potential.

There are also many reported scenarios in which the input of an arithmetic unit might alternate between binary and decimal. For instance, in applications that provide scientific and financial services to their customers, the service providers often use binary arithmetic for the former and decimal for the latter group of services using the same (cloud) hardware infrastructure [35]. If only single-type arithmetic units are available, designers will have to use both binary and decimal hardware modules simultaneously in their implementations. A reconfigurable binary/decimal adder in these cases will certainly be beneficial.

Given these conditions, a new reconfigurable adder is proposed in this paper. The proposed adder, shown in figure 8, is a combined binary/decimal adder that can operate on different sizes of operands. In the proposed design, X_i and Y_i represent the binary or BCD digits of two input operands. A signal "$\mathcal{D}$" indicates that inputs are binary ($\mathcal{D} = 0$) or decimal ($\mathcal{D} = 1$). The design has two distinct parts that can operate independently or together. Each part can be configured as either a 4n-bit binary adder or an n-digit decimal adder. If configured to perform the addition in a cascaded form, the proposed adder will take the shape of either an 8n-bit binary adder or a 2n-digit decimal adder. Each part of the design is n comprised of "dual sum generation" units. The internal circuitry of these units is illustrated in figure 9 (reproduced from [36]). In this figure, the squares produce the bit generate, propagate, and half-sum ($\hbar_0$-$\hbar_3$) signals and the circles are ordinary parallel prefix nodes. The dual sum generation units compute P_i and G_i for each decimal digit that is fed into a quaternary PPT.

Each dual sum generation unit, also provides the result of its corresponding digit addition in two conditions: If the carry-in to that position is zero, the result is called S_i^0 and if the carry-in is one, the result is called S_i^1. In the lower half adder (the right hand side one), when quaternary PPT prepares correct decimal carries, the proper sum digit is selected. However, in the left hand side adder, the problem is slightly different. An "integrated" signal is defined that indicates the size of operation. If this signal is equal to one, two adders are cascaded to perform a 2n-digit decimal or 8n-bit binary addition. Otherwise, they implement two independent additions. The "carry selection" units receive the carry-out of right hand side adder and produce the correct control signal for selecting the appropriate sum. figure 10 shows the block diagram of the carry selection unit. In this figure, $P_i' = P_{i-1}P_{i-2} \dots P_n$, where $n < i < 2n$.

The proposed reconfigurable adder uses a Carry Select Adder (CSA) as its basis. This basic adder was chosen because it has fast implementations in literature. Depending on the design objectives that one pursues, other types of adder (such as Carry Skip Adder, Carry Look ahead Adder, etc.) could also be used in the proposed architecture.

The next section provides the experimental results and their analysis.

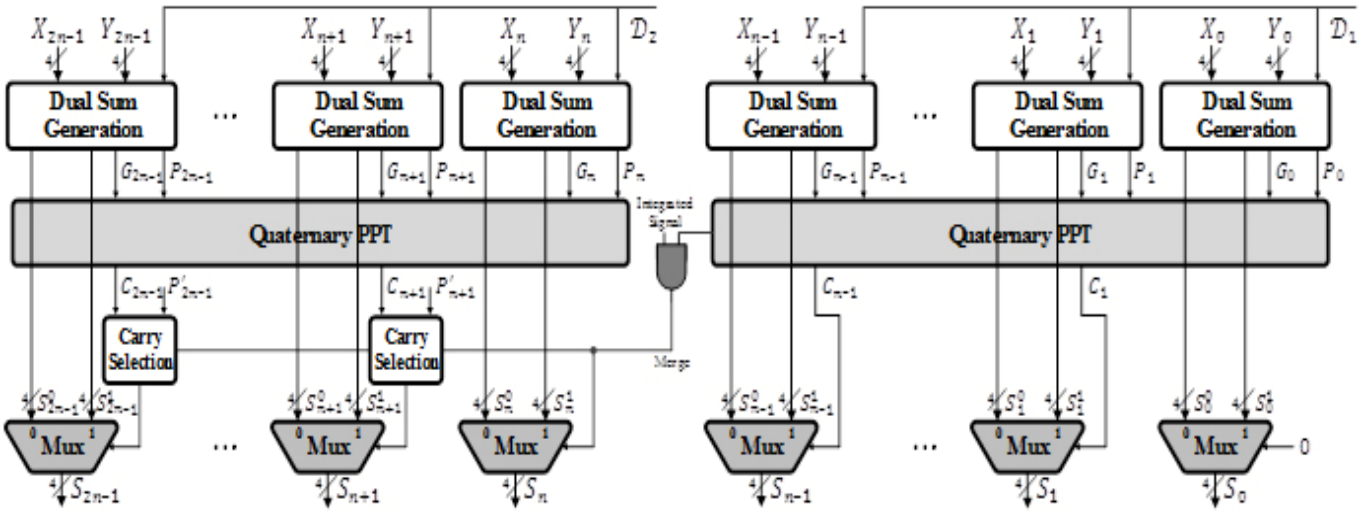


Figure 8. A reconfigurable combined binary/decimal adder

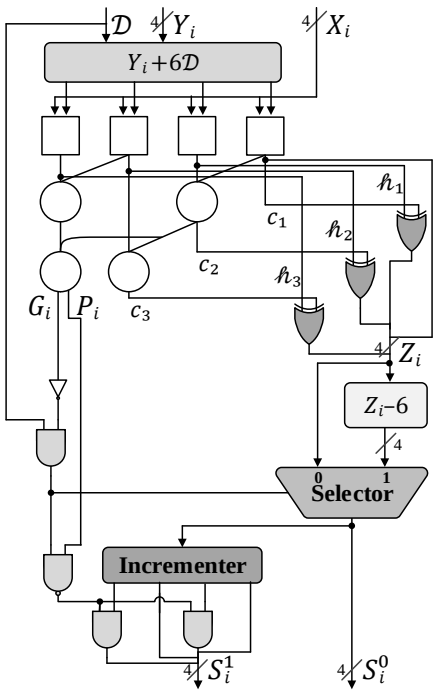


Figure 9. Dual sum generation unit structure (reproduced from [36])

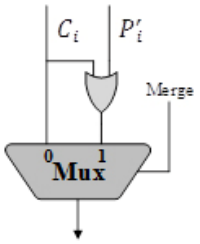


Figure 10. Carry selection unit structure

4. Comparison

To analyze the characteristics of the proposed reconfigurable designs, they were implemented at RT level using VHDL. The codes were validated using Modelsim. The area and critical path delay (both in terms of number of gates) of each design are listed in table 1. The column "Reconfigurability Overhead (Area)" shows the ratio of the corresponding reconfigurable PPT divided by the average area of its basic PPTs. Likewise, the column "Reconfigurability Overhead (Delay)" contains the ratio of the corresponding reconfigurable PPT delay divided by the average of its parent PPTs. As can be seen in the table, considering the structural similarity (step 3) produces the circuit with the smallest area with 28% average area overhead. But the minimum critical path delay and average delay overhead of 10% is obtained when equal number of nodes in the basic PPTs is considered (step 2).

Table 1. Comparison between basic PPTs and the proposed reconfigurable PPTs (for 16-bit wide inputs)

Parallel Prefix Tree		Area (Number of gates)	Reconfigurability Overhead (Area)	Critical Path Delay (Number of gates)	Reconfigurability Overhead (Delay)
Basic	Ladner-Fischer (L-F)	128	N/A	9	N/A
	Han-Carlson (H-C)	128	N/A	11	N/A
	Brent-Kung (B-K)	110	N/A	13	N/A
Reconfigurable	Combined L-F/H-C (Step 1)	209	1.63	19	1.9
	Combined L-F/H-C (Step 2)	170	1.33	11	1.1
	Combined H-C/B-K (Step 3)	152	1.28	15	1.25

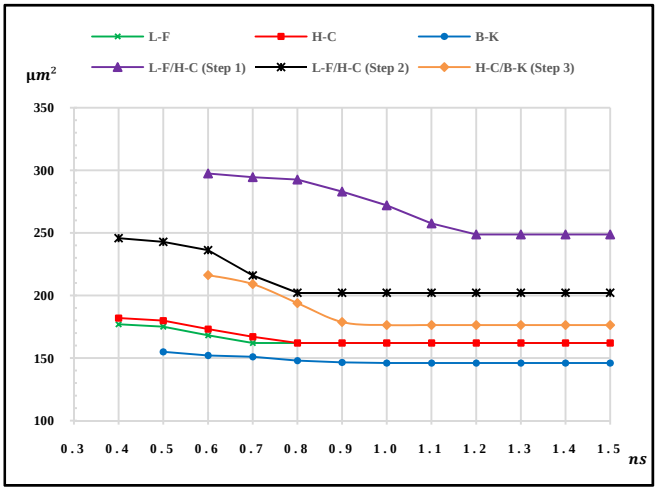


Figure 11. Area of 16-bit binary adders

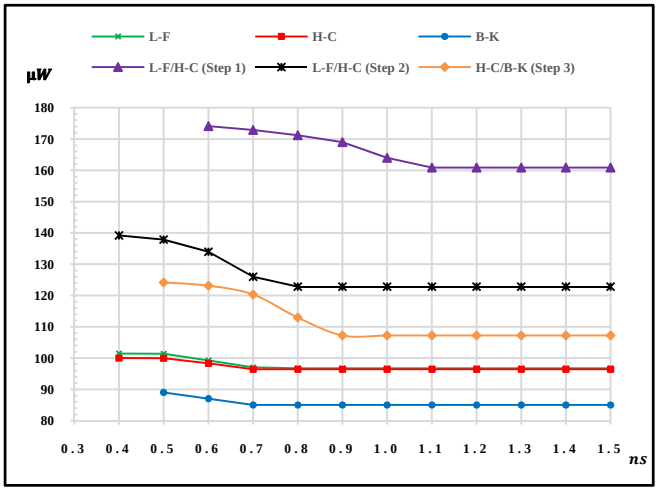


Figure 12. Power of 16-bit binary adders

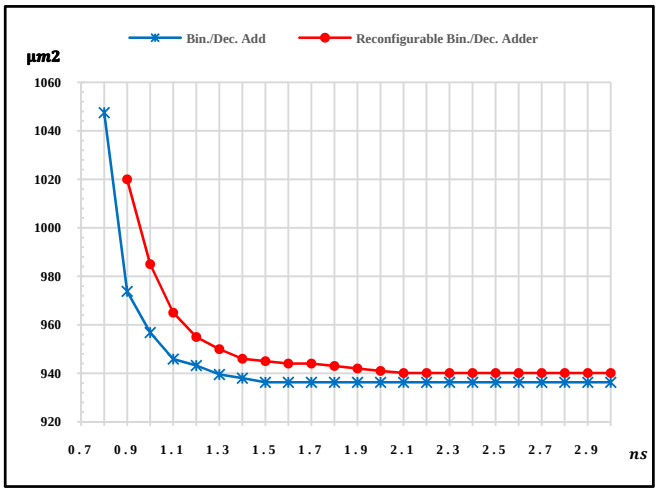


Figure 13. Area of 16-digit/64-bit binary/decimal adders

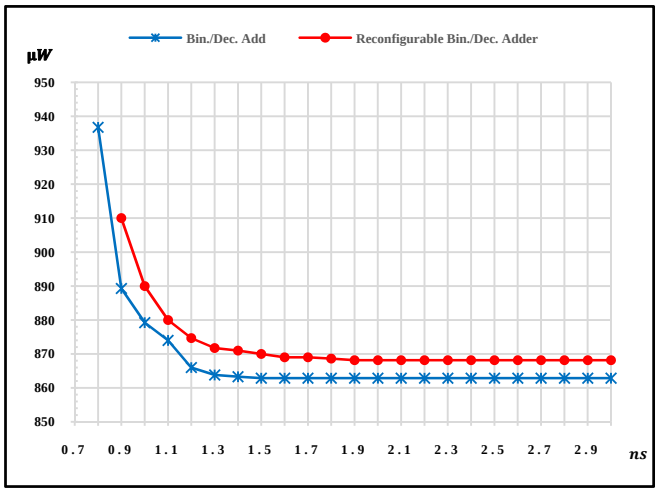


Figure 14. Power of 16-digit/64-bit binary/decimal adders

For more accurate analysis, the designs were also synthesized based on Nan Gate FreePDK45 Open Cell Library under typical process and normal operating conditions using Synopsys Design Compiler tool. The reconfigurable PPTs were synthesized with multiple target delays as shown on the horizontal axes of figures 11 and 12. These vertical axes in these figures represent the resulting area and power, respectively. The target delays are in the range of 0.3ns to 1.5ns with 0.1ns increments. If there is no point in the curves for a specific target delay, it means that no circuit could be found to satisfy the prescribed target delay. The curves demonstrate a general congruence with table 1.

Figures 13 and 14 depict the synthesis results of combined binary/decimal reconfigurable adder for area and power, respectively. To comply with IEEE 754-2008 standard [9], the adders are assumed to be 16-digit wide (when decimal) and 64-bit wide (when binary). The horizontal and vertical axes are similar to the ones in figures 11 and 12 except that the target delay range has been modified to 0.7-3ns in order to accommodate for the larger design in this case. The results are compared with the best combined binary/decimal adder found in literature [36]. As figure 13 shows, the area overhead of the proposed reconfigurable adder is at most 5%. Notwithstanding the worst case scenario, the average area overhead is far less than that.

5. Conclusion

In this paper, different options for reconfigurability in decimal adders were discussed. To this end, the possibility of combining two conventional PPTs in order to reach a hybrid reconfigurable PPT was explored. A few different criteria for choosing which two PPTs should be combined were considered and their effect on the resulting reconfigurable PPT was discussed.

In a different perspective, the concept of reconfigurability was extended to having an adder with flexible input size and type. The proposed reconfigurable adder can implement one 2n-digit or two n-digit binary/decimal additions concurrently. The synthesis results showed that for the same target delay, the advantages of reconfigurability in the proposed adder were reached with a maximum area overhead of 5% and average area overhead of 1%.

References

[1] H. Goldstine, and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," *IEEE Ann. Hist. Comput.*, vol. 18, no. 1, pp. 10–16, 1996.

[2] G. Gray, "UNIVAC I Instruction Set," *Unisys History Newslett.*, vol. 5, no. 3, 2001.

[3] F. E. Hamilton, and E. C. Kubie, "The IBM magnetic drum calculator type 650," *J. ACM*, vol. 1, no. 1, pp. 13–20, Jan. 1954.

[4] L. Eisen, J. W. Ward III, H. W. Tast, N. Mading, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough, "IBM POWER6 Accelerators: VMX and DFU," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 663–684, 2007.

[5] A. Y. Duale, M. H. Decker, H. G. Zipperer, M. Aharoni, and T. J. Bohizic, "Decimal Floating-point in z9: An Implementation and Testing Perspective," *IBM Journal of Research and Development*, vol. 51, no. 1/2, pp. 217–228, 2007.

[6] E. M. Schwarz, J. S. Kapernick, and M. F. Cowlishaw, "Decimal Floating-point Support on the IBM System z10 Processor," *IBM Journal of Research and Development*, vol. 53, no. 1, pp. 4:1–4:10, 2009.

[7] S. Carlough, A. Collura, S. Mueller, and M. Kroener, "The IBM zEnterprise-196 Decimal Floating-Point Accelerator," *Proceedings of the 20th IEEE Symposium on Computer Arithmetic*, pp. 139-146, 2011.

[8] IBM Corporation, The "telco" Benchmark, http://speleotrove.com/decimal/telcoSpec.html,October 2016.

[9] Standards Committee, "754-2008 IEEE Standard for Floating-Point Arithmetic," IEEE Computer Society Standard, pp. 1–58, 2008.

[10] N. Chainani, "Decfloat: The Data Type of the Future," http://www.ibm.com/developerworks/data/library/techarticle/dm-0801chainani, January 2008.

[11] M. Cornea, "Intel decimal floating-point math library," https://software.intel.com/en-us/articles/intel-decimal-floating-point-math-library, October 2016.

[12] GCC, the GNU Compiler Collection, https://gcc.gnu.org, October 2016.

[13] M. Anderson, C. Tsen, L. K. Wang, K. Compton, and M. J. Schulte, "Performance Analysis of Decimal Floating-Point Libraries and its Impact on Decimal Hardware and Software Solutions," *Proceedings of the 27th IEEE International Conference on Computer Design*, pp. 465–471, 2009.

[14] H. Fahmy, R. Raafat, A. M. Abdel-Majeed, R. Samy, T. ElDeeb, and A. Farouk, "Energy and Delay Improvement via Decimal Units," *Panel on Decimal Arithmetic in Industry, Proceedings of the 19th IEEE Symposium on Computer Arithmetic*, pp. 221–224, 2009.

[15] L. K. Wang, M. A. Erle, C. Tsen, E. M. Schwarz, and M.J. Schulte, "A Survey of Hardware Desgins for Decimal Arithemetic," *IBM Journal of Research and Development*, vol. 54, no. 2, pp. 8:1–8:15, 2010.

[16] Á. Vázquez, and E. Antelo, "A High-performance Significand BCD Adder with IEEE 754-2008 Decimal Rounding," *proceedings of the 19th IEEE Symposium on Computer Arithmetic*, pp. 135-144, 2009.

[17] Á. Vázquez, *High-performance Decimal Floating-point Units*, Ph.D. dissertation, University of Santiago de Compostela, Santiago de Compostela, Spain, 2009.

[18] R. D. Kenney, and M. J. Schulte, "High-speed Multioperand Decimal Adders," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 953–963, 2005.

[19] Á. Vázquez, E. Antelo, and P. Montuschi, "A New Family of High-performance Parallel Decimal Multipliers," *Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, pp. 195–204, 2007.

[20] D. Chen, L. Han, Y. Choi, and S. B. Ko, "ImprovedDecimal Floating-point Logarithmic Converter based on Selection by Rounding," *IEEE Transactions on Computers*, vol. 61, no. 5, pp. 607–621, 2012.

[21] Á. Vázquez, E. Antelo, and P. Montuschi, "Improved Design of High-performance Parallel Decimal Multipliers," *IEEE Transactions on Computers*, vol. 59, no. 5, pp. 679–693, 2010.

[22] A. Nannarelli, "FPGA based Acceleration of Decimal Operations," *Proceedings ofInternational Conference on Reconfigurable Computing and FPGAs*, pp. 146–151, 2011.

[23] A. Vazquez, and F. de Dinechin, "Multi-operand Decimal Tree Adders for FPGAs," *INRIA, Research Report*, 2010.

[24] G. Bioul, M. Vazquez, G. P. Deschamps, and G. Sutter, "Decimal Addition in FPGA," *Proceedings of the 5th Southern Conference on Programmable Logic*, pp. 101–108, 2009.

[25] H. Calderon, G. Gaydadjiev, and S. Vassiliadis, "Reconfigurable Universal Adder," *Proceedings of International Conference on Application-specific Systems, Architectures and Processors*, pp. 186–191, 2007.

[26] T. Mohit, V. Apurva, and K. Kavita, "A Novel Hardware Efficient Reconfigurable 32-bit Arithmetic Unit for Binary, BCD and Floating-point Operands," *International Journal of Engineering Science & Technology*, vol. 3, no. 5, pp. 4449–4464, 2011.

[27] M. A. Shami, "Dynamically Reconfigurable Resource Array," *Ph.D. Dissertation, KTH Royal Institute of Technology, Sweden*, 2012.

[28] R. Brent, and H. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, 1982.

[29] J. Sklansky, "Conditional-sum Addition Logic," *IRE Transaction Electronic Computers*, vol. EC-9, pp. 226–231, 1960.

[30] P. Kogge, and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,"*IEEE Transactions on Computers*, vol. C-22, pp. 786–793, 1973.

[31] T. Han, and D. Carlson, "Fast Area-efficient VLSI Adders," *Proceedings of the 8th IEEE Symposium on Computer Arithmetic*, pp. 49–56, 1987.

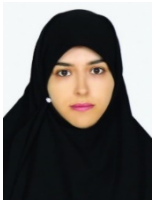
[32] S. Knowles, "A Family of Adders," *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pp. 277–281, 2001.

[33] R. Ladner, and M. Fischer, "Parallel Prefix Computation," *Journal of ACM*, vol. 27, no. 4, pp. 831–838, 1980.

[34] M.Cowlishaw,"Doapplicationsactuallyusedecimaldata?, "http://speleotrove.com/decimal/decifaq1.html#dbstats, October 2016.

[35] NCR Corporation, http://www.ncr.com/products-and-services/cloud-services, October 2016.

[36] M. Dorrigiv, and G. Jaberipur, "Low Area/power Decimal Addition with Carry-select Correction and Carry-select Sum-digits," *Integration, the VLSI Journal*, vol. 47, no. 4, pp. 443–451, 2014.



Samaneh Emami received her B.S. and M.S. degrees in Computer Engineering from Shahid Beheshti University in 2008 and 2011, respectively. Since 2011, she has been pursuing her Ph.D. in Computer Engineering at the Department of Computer Engineering and Information Technology (CEIT), Amirkabir University of Technology. Her research interests include computer arithmetic, high-level synthesis and reconfigurable design.

E-mail: s.emami@aut.ac.ir

Mehdi Sedighi received his B.S. in Electrical and Computer Engineering from Sharif University of Technology in 1990 and his M.S. and Ph.D. in the same field from University of Colorado at Boulder in 1994 and 1998, respectively. Since late 2001, he has been with the Department of Computer Engineering and Information Technology at Amirkabir University of Technology where he is currently an associate professor. His research interests include VLSI design, synthesis of arithmetic circuits, embedded systems and quantum computing.

E-mail: msedighi@aut.ac.ir

Paper Handling Data:

Submitted: 07.10.2016
Received in revised form: 21.11.2016
Accepted: 01.12.2016
Corresponding author: Dr. Mehdi Sedighi,
Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran.

The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 42-47
Regular Paper

A Deep Learning Method to Estimate 3D Point of Regard by Joint Head and Eye Information

Rahim Entezari Mohammad Mahdi Arzani Mahmood Fathy
Amir Hossein Bayat

Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

Abstract

The development of systems that can characterize the state of the human is now important for many applications. In particular, as an indicator of attention and interest, the human gaze is an important cue in people behaviors, personality, intentions, and activities. Gaze also play a crucial role in the communication process. However, in spite of great advances during last three decades, current gaze estimation methods cannot addresses required conditions in this field, e.g. user head movements and minimum user calibration. There have been some works to resolve such problems but those methods lack good precision. In this work, we have used a method for appearance-based gaze estimation using convolutional neural networks, which is multimodal. This method in our implemented setting significantly outperforms state-of-the-art methods.

Keywords: Gaze Estimation, Convolutional Neural Networks, Head Movement, Attantion, Calibration.

1. Introduction

The most important ways of non-verbal communication consist of facial expressions, hands position, gestures, and gaze. The last one plays a crucial role when people interact, as it is used to regulate the flow of communication, monitor feedback, reflect cognitive activity, express emotions, and communicate the nature of the interpersonal relationship [1]. In general, gaze is a very strong indicator for subject attention process.

In the recent years, as there is much rich information in non-verbal cues, there has been a growing interest from diverse domain on tools, which are able to retrieve the state of human.

Appearance-based gaze estimation which does not need specific tools is a hot topic research in computer vision as it can be used for several application domains, including gaze-based human-computer interaction and visual behavior analysis [2].

We now know that the desired gaze consists of two factors [3] i.e. the head pose and the eye locations. The estimation of these mentioned factors is often achieved using expensive or limiting hardware.Therefore, the problem is often simplified by either considering the head pose or the eye center locations as the only feature to understand the interest of a subject [4].

Here, Appearance-based 3D gaze estimation is a supervised regression problem in which 3D gaze direction is predicted from input features, i.e. a set of an eye image and a 3D head pose. In general, the performance of appearance-based methods depend on the quality and diversity of training data and also generalization ability of the regression algorithm [5].

In most of the previous studies, evaluation has been conducted using the test and training data of the same person, and this leads to less generalization for the different conditions [6].

This paper is based on previous work [6] done by Xucong Zhang, et al .They have employed the LeNet for the deep

network. They have used cropped eyes and head pose information directly integrated with EYEDIAP dataset for the features, i.e. there is no explicit feature extraction phase.

We have deployed the same method except for the following differences in the setting:

- Extracted head pose information
- Cropped eye images
- Number of training data
- Training parameters

Zhang et al. at Max Plank Institute (MPI) have used vectors of 2D angle for the head pose, while we have used vectors of 12, consisting of 9 float numbers of head pose rotation matrix and 3 of translation matrix.

They also cropped both eyes in one image (Figure 1(a)) whereas we have used both eyes in separate images (Figure 1(b)).

They have only used the screen target sequences and have not covered floating target data, which are more challenging and contain many extreme gaze directions, but we have used all dataset videos, divided for the train, test and validation. Figure 2. Shows some frames which are not covered by [6].

(a) Both eyes in one image (b) Left and Right eyes

Figure 1. Example eye images (a) MPI, (b) ours

Figure 2. Example frames not covered by [6]

In this work, we also have used different training parameters for the deep network, which obviously helped us to converge better and get better results. The goal here is to predict the point of regard. i.e. where the participant is looking at (Figure 3).

Below, we review the related works on gaze estimation (Section 2) and then introduce used method (Section 3). In section 4, we talk about details of used method and experiments. Section 5 consists of conclusion and future works.

2. Related Work

The survey by Hansen [1] provides a comprehensive overview of computer vision methods for gaze estimation. In general, gaze estimation methods can be further divided to model-based or appearance-based [1]. Model-based methods use a geometric eye model and are divided into corneal-reflection and shape-based methods, depending on whether they require external light sources to detect eye features.

Some related works on corneal reflection-based methods were focused on stationary setting and some more complicated with arbitrary head poses [6].

On the other hand, the focus of shape-based methods is on pupil center and iris edges [1]. Zhu et al. [7] used thres holding for pupil center estimation. Yamazoe et al. [17] also proposed to fit a geometric model from segmentations of the eye images. These segments were obtained through simple thres holding. During the test phase, they used the iris center derived from a fitted ellipse to infer gaze. Voting-based methods are also used in edge detection process [8]. The more complex shape models [9] were proposed to compensate problems in simple shape-based models, but this leads to more computation and most of these models need high-quality images [1]. Some later works in shape-based methods [10] used the ensemble of Random Regression Trees for pupil center localization. Strupczewski et al. [11] proposed different geometric model for webcam eye gaze tracking.

The most important advantage of appearance-based gaze estimation methods, which are also known as holistic methods [1], is that they use eye images as input, therefore there is potential to work with lower resolution eye images. Some early works assumed a fixed head pose while newer works deal with 3D head pose estimation [6].

It is worth mentioning that there is another category for hybrid models [1]. Some works [12] use methods of combining shape and appearance or shape and color [13].

Appearance-based methods require larger amounts of user-specific training data than model-based methods [1] but it can be compensated with last released datasets, e.g. EYEDIAP which consists many frames.

Williams et al. relied on semi-supervised Gaussian Process Regression (GPR) for visual mapping [14]. More recently, Lu et al. [15] proposed adaptive linear regression which is based on sparse image reconstruction. Their method required a fixed head pose. To remove fixed head pose constraint Lu et al. proposed a Gaussian Process Regression based pose correcting scheme on top of fixed head pose model [16]. Funes et al. [17] used RGBD cameras to directly handle eye appearance variation by generating frontal looking eye images used as input to adaptive linear regression.

Some recent works [18]uses egocentric videos and based on activity, head and hand locations, eye gaze is estimated. In [19] authors used conditional random field as a graphical model to map relation between head, human body pose and face for eye gaze estimation. Mansouryar et al. directly maps 2D pupil positions to 3D gaze directions in scene camera coordinate space [20]. Feng Lu et al. have employed the advantages of recent sparse auto-encoding techniques. They partition any eye image into small patches .Using these patches they learn a codebook comprising a set of bases,

which can reconstruct any eye image patch with sparse coefficients. By examining these coefficients, they can analyze the eye shape more effectively [21]. Krafkaet. al. [22] released the dataset of gaze tracking on the mobile device, called Gaze Tracker. They also have used the convolutional neural network to estimate gaze of mobile device user on the screen. They employed eyes, face and face grid, that is a binary mask used to indicate the location and size of the head within the frame. They reported a prediction error of 1.71cm on mobile phone screen according to the location of the camera.

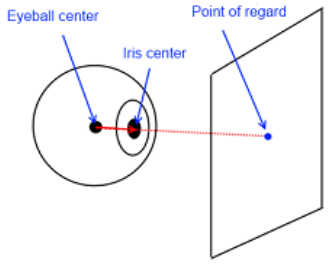


Figure 3. Illustration of point of regard

3. The Proposed Method

We first convert each video to frames, then eyes are extracted from frames. Extracted eyes are fed to a convolutional network. This multimodal convolutional network is the same one used in [6]. They have used LeNet network architecture that consists of one convolutional layer followed by a max-pooling layer. The third layer is a second convolution layer, which is followed by another max-pooling layer, and a final fully connected layer.

Similar to [6] we train a linear regression layer on top of the fully connected layer to predict point of regard. This CNN has two inputs, cropped eyes and head pose information. The head pose information here is nine values of the rotation matrix and three values for translation. A rotation of α radians about the x-axis, β radians about y-axis and γ radians about z-axis are defined as following, respectively [23]:

$$R_x(\alpha)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha\\ 0 & \sin\alpha & \cos\alpha\end{bmatrix}$$

$$R_y(\beta)=\begin{bmatrix}\cos\beta & 0 & \sin\beta\\ 0 & 1 & 0\\ -\sin\beta & 0 & \cos\beta\end{bmatrix}$$

$$R_z(\gamma)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0\\ \sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}$$

So if we rotate first about the x-axis, then the y-axis and finally the z-axis, this can be represented as the matrix product of$R= R_x(\alpha)\ R_y(\beta)\ R_z(\gamma)$ which is equivalent to following Rotation matrix:

$$\begin{bmatrix}\cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma-\cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma\\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma-\sin\alpha\cos\gamma\\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\alpha\end{bmatrix}$$

We tried to extract three values of angles in addition to the translation matrix. As it has not been mentioned in the dataset that what are the right orders of rotations, i.e. x-axis, y-axis then z-axis, x-axis, z-axis then y-axis, etc. we have deployed these six possible permutations to find the corresponding order. We have found that mentioned order "A rotation of α radians about the x-axis, β radians about the y-axis, and γ radians about the z-axis" is the right one. So we tested two scenarios, first the extracted three rotation angles were concatenated to translation matrix; secondly, we have used integrated twelve values. Our results showed the later scenario gives better results. This information used to learn mapping from eye images and head pose vectors to point of regard.

3.1. Preparing Data

We have used EYEDIAP dataset[24]. In total, there are 94 recorded sessions in EYEDIAP. Each session will be denoted by the string "P-C-T-H" which refers to the participant id (1-16), the recording conditions C= (A or B), the used target T= (DS, CS or FT), i.e. Discrete Screen, Continuous Screen and floating target moving in the space, respectively. The head pose also consists of H=(S or M), static or moving. Each session in conditions "A" correspond to 2.5 minutes of recording time, whereas the sessions recorded in conditions "B" last approximately 3 minutes each. This corresponds to more than 4 hours of data. Table 1 summarize all recorded data.

Table 1. Summary of the recorded sessions

Participants	Recorded sessions
1-11	A-DS-S; A-DS-M; A-CS-S; A-CS-M; A-FT-S; A-FT-M
12-13	B-FT-S; B-FT-M
14-16	A-DS-S; A-DS-M; A-CS-S; A-CS-M; A-FT-S; A-FT-M; B-FT-S; B-FT-M

To standardize the definition of all 3D variables in the data, Funes et al. have defined a common world coordinate system (WCS), in which the variables refers to meters. In this definition, if $p_k \epsilon R^3$ is a point defined w.r.t. the coordinate system of Kinect RGB camera, then defined WCS is the equivalent p_w, w.r.t. the WCS is given by $p_w = R_w p_k + t_w$ where:

$$R_w=\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & -1\end{bmatrix},\ t_w=\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}$$

In which R_w and t_w are rotation and translation matrices respectively.

The very first stage is to convert videos into frames using OPENCV. We tried to use MATLAB for framing, but the problem was that framed images were not synchronized with the corresponding frame in the video. It seems MATLAB has some problems with framing videos, after framing, we have about 500K frames. It is mentioned in EYEDIAP dataset that some frames are not valid. This is due to two factors, the participant is blinking at the given frame or the participant is not looking at the visual target at the given frame, e.g. whenever the participant is distracted. So non-OK frames are removed from extracted frames using given data within

dataset. For cropping eyes we used the integrated data within the dataset, i.e. Kinect RGB eyeball center left and right (x_l,y_l,x_r,y_r).

3.2. POR Estimation Using CNN

Here we have used Multimodal CNN [6] to learn how to map inputs to gaze coordinates in world coordinate space. Inputs are eye images and 12 value head vectors. This model is multimodal as it uses both eye images and head pose float numbers.

In our model, we have used the Le Net network architecture that consists of one convolutional layer followed by a max-pooling layer, a second convolution layer followed by a max-pooling layer, and a final fully connected layer. We train a linear regression layer on top of the fully connected layer to predict gaze angle vectors. The input to the network is RGB eye images with a fixed size of 40 x 40 pixels, as available eyeball center is at the exact center of this window (Figure 4).

The output of the network is a 3D gaze point where the observer is looking at. The point of regard, which is given within dataset, is compared to this output coordinate. As a loss function, we used the sum of individual L_2 losses that measures the difference between predicted coordinate and the actual point of regard.

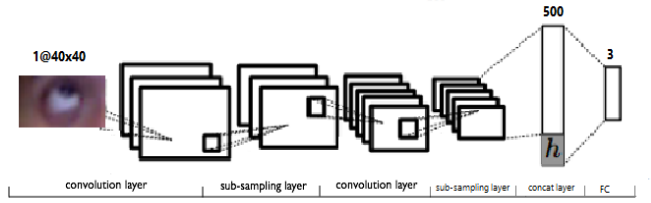


Figure 4. Architecture of used multimodal CNN. Head vectors are added to the output of fully connected layer.Here we have used [6] but with 3D gaze point

4. Experiments

In this section, we discuss gaze estimation task and validate the effectiveness of used multimodal CNN approach.

We compare our method with state-of-the-art methods on the EYEDIAP dataset. We have divided the dataset into training, validation, and test sets.

4.1. Training

We have used the first 10 people of EYEDIAP for training. This means training set consists of about 200K frames. We trained deep network using caffe [25]. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. We have used the batch size of 500, so every epoch is 400. We have trained this network from scratch, with the base learning rate of 0.001, the momentum of 0.95 and weight decay of 0.0005. We also have used ad a delta to get better convergence instead of SGD. Delta parameter is set to 1e-6. With respect to validation loss, we have trained our CNN for 80K iterations.

4.2. Validation

The next 2 person are used for validation, and this was about 40K frames. With batch size of 500, we had 80 test iterations with interval of 10K. This last value means we have carried out validating every 10K training iterations.

4.3. Test

We have used remaining dataset videos for the phase of the test, i.e. we have 70K frames. In this phase we have fed each frame of test to the trained model, in addition to the head pose data. The output which is a 3D float number refers to the point the person is looking at. For getting Mean Error Degree, for each testing frame, we have connected the center of eyes to the predicted and actual gaze point which results in 2 vectors. The degree between these two vectors are calculated with the following formula:

$$\Theta = \cos^{-1}\left(\frac{\vec{x}.\vec{y}}{||x||||y||}\right)$$

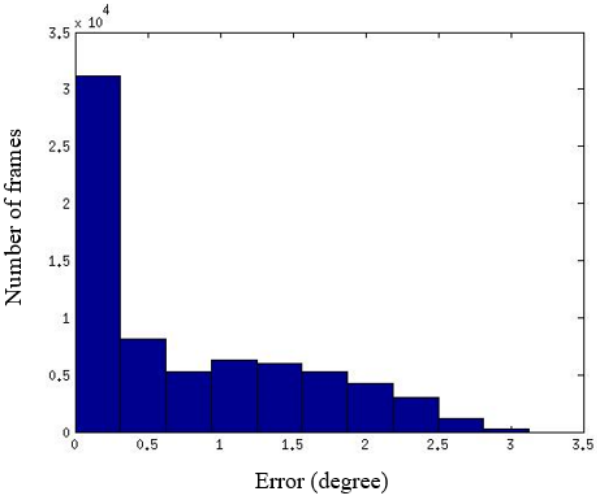


Figure 5. histogram of error for test frames

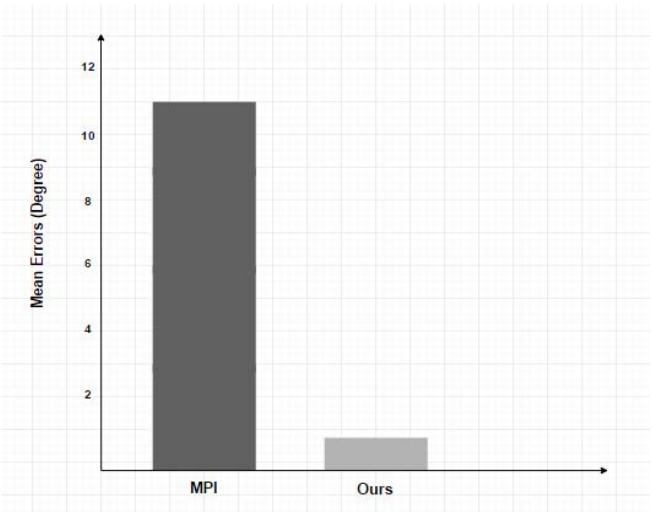


Figure 6. performance of our method compared to [6]

Figure 5 show the histogram of error for all test frames. As you can see in figure 6. We have much better result than state-of-the-art [6] method for EYEDIAP dataset. The mean

error degree is less than one degree, meaning the difference between actual gaze vector and estimated one is about 0.79 degree.

Figure 7. Also shows the output of some layers in this CNN.

In this figure, (a) corresponds to an example of input RGB frame. This input is cropped eye images. (b) Shows the filters of first convolution layer, (c) shows the output of first convolution layer. Another layer shown in this figure is max-polling, which is a form of non-linear down-sampling. Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value:

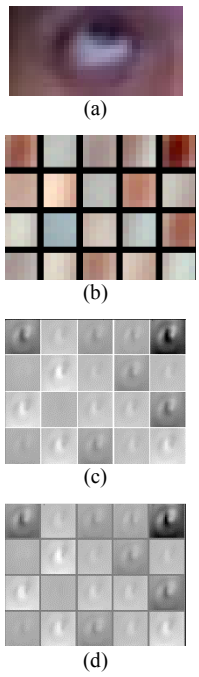


Figure 7. Figure 7 (a) Cropped eye, (b) conv1 filters, (c) conv1 output (rectified responses of the filters), (d) pool1 output

5. Conclusion and Future Work

Despite lots of works mentioned in literature, there have been few works for appearance-based gaze estimation with low-quality images. Those with this condition evaluated exclusively under controlled conditions with low range head poses. In this work we have employed [6] method with different setting for EYEDIAP dataset which varies in different head pose and illumination conditions. This dataset is very challenging as floating target sequences contain many extreme gaze directions. Our setting for CNN-based estimation model significantly outperforms [6] which is state-of-the-art. Our future work will consist in using different deep network. We also will try to get eye images and head pose automatically.

References

[1] Hansen, D. Witzner, and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* pp. 478-500, 2010.

[2] C. H. Morimoto, and M. R. M. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding,* vol. 98, pp. 4-24, 2005.

[3] S. R. Langton, H. Honeyman, and E. Tessler, "The influence of head contour and nose angle on the perception of eye-gaze direction," *Perception & psychophysics,* vol. 66, pp. 752-771, 2004.

[4] N. Robertson, and I. Reid, "Estimating gaze direction from low-resolution faces in video," in *European Conference on Computer Vision*, pp. 402-415, 2006.

[5] Y. Sugano, Y. Matsushita, and Y. Sato, "Learning-by-synthesis for appearance-based 3d gaze estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1821-1828, 2014.

[6] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4511-4520, 2015.

[7] Z. Zhu, K. Fujimura, and Q. Ji, "Real-time eye detection and tracking under various light conditions," in *Proceedings of the 2002 symposium on Eye tracking research & applications*, pp. 139-144, 2002.

[8] R. Valenti, and T. Gevers, "Accurate eye center location and tracking using isophote curvature," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1-8, 2008.

[9] I. F. Ince, and J. W. Kim, "A 2D eye gaze estimation system with low-resolution webcam images," *EURASIP Journal on Advances in Signal Processing,* vol. 2011, pp. 1-11, 2011.

[10] N. Markuš, M. Frljak, I. S. Pandžić, J. Ahlberg, and R. Forchheimer, "Eye pupil localization with an ensemble of randomized trees," *Pattern recognition,* vol. 47, pp. 578-587, 2014.

[11] A. Strupczewski, B. Czuprynski, J. Naruniec, and K. Mucha, "Geometric Eye Gaze Tracking," in *International Conference on Computer Vision Theory and Applications*, pp. 446-457, 2016.

[12] D. W. Hansen, J. P. Hansen, M. Nielsen, A. S. Johansen, and M. B. Stegmann, "Eye typing using Markov and active appearance models," in *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, pp. 132-136, 2002.

[13] D. W. Hansen, "Using colors for eye tracking," *Color Image Processing: Methods and Applications,* pp. 309-327, CRC Press, 2006.

[14] O. Williams, A. Blake, and R. Cipolla, "Sparse and Semi-supervised Visual Mapping with the S^ 3GP," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06),* pp. 230-237, 2006.

[15] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Adaptive linear regression for appearance-based gaze estimation," *IEEE transactions on pattern analysis and machine intelligence,* vol. 36, pp. 2033-2046, 2014.

[16] F. Lu, T. Okabe, Y. Sugano, and Y. Sato, "A Head Pose-free Approach for Appearance-based Gaze Estimation," in *BMVC*, pp. 1-11, 2011.

[17] K. A. F. Mora, "3D Gaze Estimation from Remote RGB-D Sensors," PhD Thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2015.

[18] Y. Li, A. Fathi, and J. M. Rehg, "Learning to predict gaze in egocentric video," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3216-3223, 2013.

[19] B. Benfold, and I. Reid, "Unsupervised learning of a scene-specific coarse gaze estimator," in *2011 International Conference on Computer Vision*, pp. 2344-2351, 2011.

[20] M. Mansouryar, J. Steil, Y. Sugano, and A. Bulling, "3D gaze estimation from 2D pupil positions on monocular head-mounted eye trackers," in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 197-200, 2016.

[21] F. Lu, and X. Chen, "Person-independent eye gaze prediction from eye images using patch-based features," *Neurocomputing,* vol. 182, pp. 10-17, 2016.

[22] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and et al., "Eye tracking for everyone," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2176-2184, 2016.

[23] G. G. Slabaugh, "Computing Euler angles from a rotation matrix," accessed on June 2016, at http://thomasbeatty.com/MATH%20PAGES/ARCHIVES%20-%20NOTES/Applied%20Math/euler%20angles.pdf.

[24] K. A. F. Mora, F. Monay, and J.-M. Odobez, "EYEDIAP: a database for the development and evaluation of gaze estimation algorithms from RGB and RGB-D cameras," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 255-258, 2014.

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, and et al., "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675-678, 2014.

Rahim Entezarireceived the B.S. degree in Computer Engineering from Amirkabir University (Tehran Polytechnic), Iran, 2013. He is currently pursuing the MSc. program of Artificial Intelligence at Iran University of Science and Technology. His current research interests include deep learning, probabilistic graphical models, and computational neuroscience.
E-mail: r_entezari@comp.iust.ac.ir

Mohammad Mahdi Arzani received the B.S. degree in Computer Engineering from Shahed University, Iran ,2008, and the M.S. degree in Artificial Intelligence from Sharif University of Technology, Iran , 2010. He is currently pursuing the Ph.D. program at Iran University of Science and Technology. His current research interests include deep learning, probabilistic graphical models and human behavior analysis.
E-mail: marzani@iust.ac.ir

Amir Hossein Bayat recievied his BSc in Computer Engineering from K.N.Toosi University of Technology in 2014 He is currently pursuing the MSc. program of Artificial Intelligence at Iran University of Science and Technology. His research interests include computer vision, machine learning, probabilistic graphical models, deep learning and big data mining.
E-mail: a_bayat@comp.iust.ac.ir

Mahmood Fathyreceived his BSc in electronics from Iran University of Science and Technology in 1984, MSc in computer architecture in 1987 from Bradford University, United Kingdom, and PhD in image processing computer architecture in 1991 from University of Manchester, United Kingdom. Since 1991, he has been an academic member in the Computer Engineering School of IUST. His research interests include image and video processing, parallel and distributed processing machine learning, vehicular social network, and big data mining.
E-mail: mahfathy@iust.ac.ir

Paper Handling Data:

Submitted: 11.11.2016
Received in revised form: 15.12.2016
Accepted: 27.12.2016
Corresponding author: Dr. Mahmood Fathy,
Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.

The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 48-53
Regular Paper

The Impact of Excess-Modulo Representation of Residues on Modulo-$(2^n - 5)$ Parallel Prefix Addition

Ghassem Jaberipur Hassan Ghasemi Motlagh

Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran

Abstract

It is desirable to realize latency-balanced computation channels in residue number systems (RNS). In particular modulo-$(2^n - \delta)$ addition with selected values of δ (e.g., $\delta \in \{1, 3, 5\}$) is of interest. Modulo-$(2^n - 1)$ adders are realized via one's complement addition, where $(2^n - 1)$ is also valid as a second representation for zero. Fast parallel prefix realization of such adders exist with $(3 + 2\lceil \log n \rceil)$ ΔG latency, where ΔG denotes the delay of a simple 2-input gate. Similarly, modulo-$(2^n - 3)$ adders with excess-modulo representations of residues in $\{0, 1, 2\}$ has been recently proposed with $(4 + 2\lceil \log n \rceil)$ ΔG latency. It has been shown that such double representation of residues does not jeopardize the subsequent RNS operations. To approach larger dynamic ranges, while keeping the channel widths (i.e., n) as low as possible, fast modulo-$(2^n - 5)$ adders could be quite useful. That's how we are motivated to explore the design and implementation of such adders with the goal of achieving the same latency of $(4 + 2\lceil \log n \rceil)$ ΔG. The proposed scheme is basically the same as that of the aforementioned modulo-$(2^n - 3)$ adders. However, there are particular non-trivial challenges to be overcome. The proposed scheme is analyzed for latency and area measures, which are confirmed via circuit synthesis by Synopsis Design Compiler.

Keywords: Modulo-$(2^n - 5)$ Adder, Excess-Modulo Representation, Parallel Prefix Modular Adders, Digital Signal Processing.

1. Introduction

The latency balance within the parallel computation channels of a residue number system (RNS) arithmetic architecture is desirable [1]. On the other hand, limited dynamic range of the classical moduli set $\tau = \{2^n - 1, 2^n, 2^n + 1\}$ calls for additional balanced moduli, where modular addition and multiplication can be performed with the same speed as in the original three moduli. Parallel prefix fast adders [2-5] for the aforementioned τ system with $(3 + 2\lceil \log n \rceil)$ ΔG latency has been proposed [6], where ΔG refers to the delay of a simple 2-input AND or OR gate. The corresponding modulo-$(2^n - 1)$ adder takes advantage of a second representation for zero, as $2^n - 1$, where a regular parallel prefix (RPP) and the so called totally parallel prefix (TPP) architectures [7] lead to $(5 + 2\lceil \log n \rceil)$ and $(3 + 2\lceil \log n \rceil)$ ΔG latencies, respectively.

Moreover, similar modulo-$(2^n - 3)$ RPP and TPP adders with $(6 + 2\lceil \log n \rceil)$ and $(4 + 2\lceil \log n \rceil)$ ΔG latencies have also been reported [8], where 0, 1 and 2 have alternative representations, as $2^n - 3$, $2^n - 2$, and $2^n - 1$, respectively. However, we have not encountered any special modulo-$(2^n - 5)$ adder design in the literature, thus motivated to present one with the same performance as that of the aforementioned modulo-$(2^n - 3)$ adder. The rest of this paper deals with a background on RNS arithmetic in Section 2, also with a brief description of previous relevant modular adders, offers our proposed modulo- $(2^n - 5)$ adder designs in Section 3, evaluations and comparisons in Section 4, and finally concluding remarks in Section 5.

2. Background

A residue number system is mainly defined by a k-moduli set $\{m_1, \ldots m_k\}$, where the modulus are commonly pairwise prime to allow for maximal range (aka dynamic range) of presentable numbers [9]. The dynamic range, as such, is

equal to $M = \prod_{i=1}^{k} m_i$. The RNS representation of a binary number X is composed of k residues $x_i = |X|_{m_i}$ (i.e., the remainder of integer division $\left\lfloor \frac{X}{m_i} \right\rfloor$, for $1 \le i \le k$), where addition and multiplication on binary operands X and Y are performed on their corresponding residues x_i and y_i in k parallel data paths.

Modulo-2^n addition is exactly the same as conventional n-bit addition, while multiplication is even simpler and less costly. However, given the mutually prime property, only one m_i can be a power of two. The next popular moduli are of the form $2^{n\pm p} - 1$ ($n \gg p$), since the required adder is exactly the same as a one's complement adder, and the main component of the corresponding multiplier is an $(n \pm p)$-operand modular adder. The $n \gg p$ restriction is to keep the moduli set balanced in terms of arithmetic speed. However, the number of such mutually prime moduli are limited. Therefore, moduli of the form $2^{n\pm p} + 1$, and more recently modulo-$(2^n - 3)$ have been employed in order to set up high dynamic range balanced moduli sets [8].

Parallel prefix modular addition is popular, since it provides for more balanced arithmetic circuits. For example, $(3 + 2\lceil \log n \rceil)$ ΔG TPP adders have been proposed for modulo-$(2^n - 1)$, where figure 1 (borrowed from [8]) depicts the required architecture. Also the fastest modulo-$(2^n - 3)$ adder that we have encountered is due to [8], where the latency of its TPP architecture (see figure 2 that is borrowed from [8]) amounts to $(4 + 2\lceil \log n \rceil)$ ΔG. The legend for these figures and the proposed designs are compiled in table 1.

3. The Proposed Modulo-$(2^n - 5)$ Adder

The modulo-$(2^n - 1)$, and $-(2^n - 3)$ adders are fairly balanced, since for instance in case of n = 8 (i.e., a typical bit length in RNS applications such as image processing [10]), 9, and 10 ΔG latencies are achieved, respectively. Nevertheless, in this section, we propose an RPP modulo-$(2^n - 5)$ adder, with the same $(6 + 2\lceil \log n \rceil)$ latency as the fastest previous modulo-$(2^n - 3)$ RPP adder [8], which consumes slightly more area and dissipates marginally more power. We also provide a TPP version of the proposed adder that performs as fast as the previous modulo-$(2^n - 3)$ TPP adder [8], while the additional area and power measures are negligible.

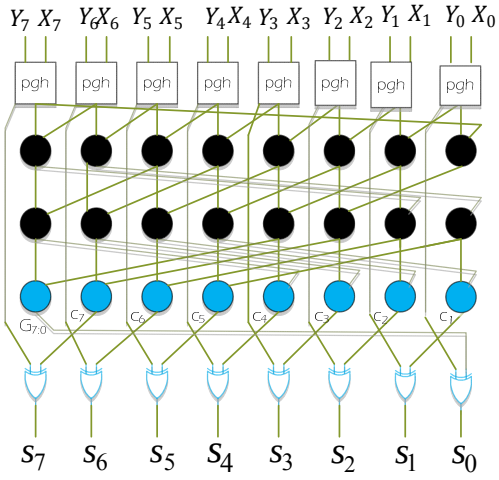


Figure 1. The TPP modulo- $(2^n - 1)$ adderarchitecture

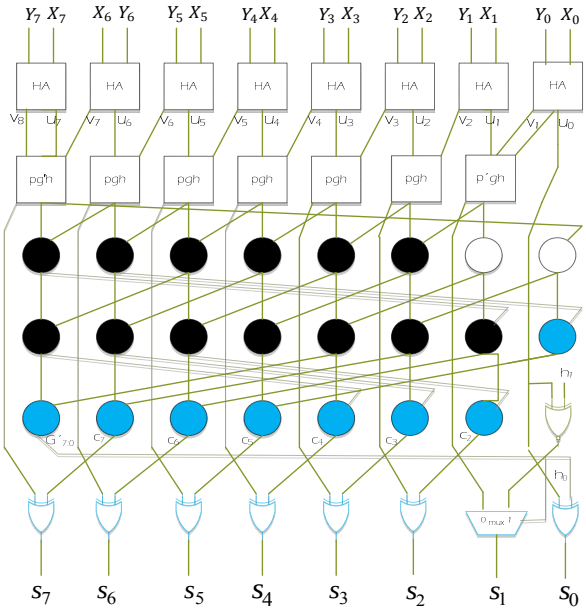


Figure 2. The TPP modulo-$(2^n - 3)$ adder architecture

Table 1. Logical cells used in the respected figures

Legend	Description	Figures
(G, P)  (G, P)	Buffer node	2, 3
$(G_\ell, P_\ell)\ (G_r, P_r)$  $G_\ell \vee P_\ell G_r$	G node	1, 2, 3, 4
$(G_\ell, P_\ell)\ (G_r, P_r)$  $(G_\ell \vee P_\ell G_r, P_\ell P_r)$	(G, P) node	1, 2, 3, 4
$(G_l, P_l)\ (G_r, P_r)$ P_{r-1}  $(G_l v P_l G_r, P_l v G_r v P_r P_{r-1})$	(G, P′) node	3, 4
$x\ \ y$ HA $xy\ \ x \oplus y$	Half adder	2, 3, 4
$v\ \ u$ pgh $p = u \vee v$ $g = u \wedge v$ $h = u \oplus v$	p = u ∨ v g = u ∧ v h = p$\bar{g}$	1, 2, 3, 4
$u_n\ \ u_{n-1} v_{n-1}$ pg'h $u_{n-1} \vee v_{n-1}$ $u_{n-1} \oplus v_{n-1}$ $(u_{n-1} \wedge v_{n-1}) \vee v_n$	$g' = g \vee v_n$	2, 3, 4
$u_1\ \ v_1\ u_0$ P'gh $u_0 \vee v_1 \vee u_1$ $u_1 \oplus v_1$ $u_1 v_1$	$p' = p \vee u_0$	2

3.1. The Proposed Modulo-$(2^n - 5)$ RPP Adder

We follow the same approach as in [8] and come up with table 2 that describes the steps of modulo-$(2^n - 5)$ addition $S = |X + Y|_{2^n-5}$, where EAC stands for end-around carry. This table is quite similar to the figure 4 of [8], except that position 2 (instead of 1) is occupied by $G'_{n-1:0} = v_n + G_{n-1:0}$, where $G_{n-1:0}$ denotes the generated carry-out of $U + V'$ [11]. Such difference, however, leads to different equations for carry signals c_1 to c_3 with respect to the corresponding ones in [8]. These equations are derived as Eqns. 1-3, where $c_3 \leq 1$ is the generated carry for $u_2 + v_2 + G'_{n-1:0} + c_2 \leq 3$, since v_2 and c_2 cannot be 1 at once. The reason is that$c_2 = u_1v_1 + (u_1 + v_1)c_1 = u_1(v_1u_0G'_{n-1:0}) + v_1u_0G'_{n-1:0}$, since $v_2u_1 = 0$ and $v_1u_0 = 0$. Therefore, c_3 is in fact the carry of summation $u_2 + (v_2 \vee c_2) + G'_{n-1:0}$. Note that Eqn. 3 includes the modified group propagate signal $P'_{2:0} = p_2 \vee G_{1:0} \vee P_{1:0}$. Also, the fact that $g_0 = 0$, and $p_0 = u_0$, leads to $G_{2:0} = G_{2:1}$, and simpler equation for $P'_{2:0}$ as in Eqn. 4.

Figure 3 depicts an RPP realization of such carry signals and the corresponding sum signals for $n = 8$, where the preprocessing stage consists of an array of half adders, and thus leads to $(7 + 2\lceil \log n \rceil)\,\Delta G$ latency. Note that derivation of $s_2 = h_2 \oplus G'_{n-1:0} \oplus c_2$ requires an extra XOR operation, which does not lengthen the critical delay path (CDP). The rest of carry equations are described by Eqn. 5, where the group propagation signals $P_{i-1:0}$ (for $i > 3$) are based on $P'_{2:0}$ (i.e., $P_{i-1:0} = P_{i-1:3}P'_{2:0}$).

Table 2. Modulo-$(2^n - 5)$ EAC addition with carry-save preprocessing

X Y		x_{n-1} ···· x_2 x_1 x_0 y_{n-1} ···· y_2 y_1 y_0			
U V		u_{n-1} ···· u_2 u_1 u_0 v_n v_{n-1} ···· v_2 v_1 0			
V′ 5EAC		u_{n-1} ···· u_2 u_1 u_0 v_{n-1} ···· v_2 v_1 0 $G'_{n-1;0}$ $G'_{n-1;0}$			
		h_{n-1} ···· h_2 h_1 h_0 c_{n-1} ···· c_2 c_1 c_0			
S		s_{n-1} ···· s_2 s_1 s_0			

$$c_1 = u_0G'_{n-1:0} \tag{1}$$

$$c_2 = G_{1:0} \vee P_{1:0}G'_{n-1:0} \tag{2}$$

$$\begin{aligned} c_3 &= u_2v_2 \vee u_2c_2 \vee (u_2 \vee v_2 \vee c_2)G'_{n-1:0} \\ &= u_2v_2 \vee (u_2 \vee v_2)G'_{n-1:0} \vee (u_2 \vee v_2 \vee G'_{n-1:0})c_2 \\ &= g_2 \vee p_2G'_{n-1:0} \vee p_2(G_{1:0} \vee P_{1:0}G'_{n-1:0}) \vee c_2G'_{n-1:0} \\ &= g_2 \vee p_2G_{1:0} \vee (p_2 \vee c_2)G'_{n-1:0} \\ &= G_{2:0} \vee (p_2 \vee G_{1:0} \vee P_{1:0}G'_{n-1:0})G'_{n-1:0} \\ &= G_{2:0} \vee (p_2 \vee G_{1:0} \vee P_{1:0})G'_{n-1:0} \\ &= G_{2:0} \vee P'_{2:0}G'_{n-1:0} \end{aligned} \tag{3}$$

$$\begin{aligned} P'_{2:0} &= p_2 \vee g_1 \vee p_1p_0 = p_2 \vee u_1v_1 \vee (u_1 \vee v_1)u_0 \\ &= p_2 \vee u_1(v_1 \vee u_0) = p_2 \vee u_1(x_0y_0 \vee (x_0 \vee y_0)\overline{x_0y_0}) \\ &= p_2 \vee u_1(x_0 \vee y_0) \end{aligned} \tag{4}$$

$$c_i = G_{i-1:0} \vee P_{i-1:0}G'_{n-1:0} \tag{5}$$

1) Further Speedup of RPP Architecture

Recalling the $(6 + 2\lceil \log n \rceil)\,\Delta G$ latency of the modulo-$(2^n - 3)$ adder of [8], we can use the technique, therein, to achieve the same $1\Delta G$ gain. Eqn. 6 is reproduced from [8], where $u'_{i-1} = x_{i-1} \vee y_{i-1}$, $p'_i = u_i$, and $g'_{i-1} = u'_{i-1}v_{i-1}$. Therefore, $\Delta G_{i:i-1}$ is reduced to $4\Delta G$, since p'_i and g'_{i-1} are delivered in $2\Delta G$, and thus the total delay for the proposed RPP adder is also $(6 + 2\lceil \log n \rceil)\,\Delta G$.

$$G_{i:i-1} = g_i \vee p_ig_{i-1} = g_i \vee p'_ig'_{i-1} \tag{6}$$

3.2. Corresponding TPP Architecture

The TPP architectures are expected to lead to $2\,\Delta G$ less overall delay with respect to the corresponding RPP architectures, as is the case in the TPP modulo-$(2^n - 3)$ adder of [8]. Therefore, we elaborate here on developing a TPP modulo-$(2^n - 5)$ adder that yields all the sum bits in $(4 + 2\lceil \log n \rceil)\,\Delta G$, where we use Eqn. 7 that describes the general TPP carry formula, with the understanding that $P_{i-1:0} = P_{i-1:3}P'_{2:0}$. As such the required TPP equations for $n = 8$ are described below.

$$\begin{aligned} c_i &= G_{i-1:0} \vee P_{i-1:0}G'_{n-1:i} \\ c_1 &= g_0 \vee p_0G'_{7:1} \\ &= g_0 \vee p_0\left(g'_7 \vee p_7(G_{6:5} \vee p_{6:5}(G_{4:3} \vee p_{4:3}G_{2:1}))\right) \\ c_2 &= G_{1:0} \vee p'_{1:0}G'_{7:2} \\ &= G_{1:0} \vee p'_{1:0}(G'_{7:6} \vee p_{7:6}(G_{5:4} \vee p_{5:4}G_{3:2})) \\ c_3 &= G_{2:0} \vee p'_{2:0}G'_{7:3} \\ &= G_{2:1} \vee p'_{2:1}\left(g_0 \vee p_0(g'_7 \vee p_7(G_{6:5} \vee p_{6:5}G_{4:3}))\right) \\ c_4 &= G_{3:0} \vee p_{3:0}G'_{7:4} \\ &= G_{3:2} \vee p'_{3:2}(G_{1:0} \vee p'_{1:0}(G'_{7:6} \vee p_{7:6}G_{5:4})) \\ c_5 &= G_{4:0} \vee p_{4:0}G'_{7:5} \\ &= G_{4:3} \vee p_{4:3}\left(G_{2:1} \vee p'_{2:1}(g_0 \vee p_0(g'_7 \vee p_7G_{6:5}))\right) \\ c_6 &= G_{5:0} \vee p_{5:0}G'_{7:6} \\ &= G_{5:4} \vee p_{5:4}\left(G_{3:2} \vee p'_{3:2}(G_{1:0} \vee p'_{1:0}G'_{7:6})\right) \\ c_7 &= G_{6:0} \vee p_{6:0}g'_7 \\ &= G_{6:5} \vee p_{6:5}\left(G_{4:3} \vee p_{4:3}(G_{2:1} \vee p'_{2:1}(g_0 \vee p_0g'_7))\right) \end{aligned} \tag{7}$$

The corresponding circuitry is depicted by figure 4, where the delay figures are indicated along the data paths. In particular, s_1 and s_5 are delivered in $11\Delta G$ and s_2 in $12\Delta G$, while the other sum signals are available at $10\Delta G$. The reason for the extra delays for s_1 and s_5 is $1\Delta G$ delay of g'_7 with respect to other g_i signals that leads to $5\Delta G$ delay of $G'_{0:7}$, and that of s_2 is due to the extra XOR. The former can be fixed via implementing Eqn. set 8 and the latter by Eqn. 9. Regarding s_2, one of the two possible sum values fo $G'_{n-1:0} =$ 0r or 1 (i.e., $h_2 \oplus G_{1:0}$ or $h_2 \oplus \overline{G_{1:0} + P_{1:0}}$) is selected by a multiplexer, and thus $\Delta s_2 = 2\Delta G + \Delta G'_{n-1:0}$, as the other s_i outputs (see figure 5). As for g'_7, the modified equation (i.e., Eqn. set 8) delivers it in $3\Delta G$, and thu $G'_{0:7}$ s is available in $4\Delta G$, as other $G_{i:i-1}$ signals.

$$\begin{aligned} g'_7 &= g_7 \vee v_8 = u_7v_7 \vee v_8 = (x_7 \oplus y_7)v_7 \vee x_7y_7 \\ &= (x_7 \vee y_7)v_7 \vee x_7y_7 = v_8 \vee u'_7v_8 \\ G'_{0:7} &= g_0 \vee p_0g'_7 \end{aligned} \tag{8}$$

$$\begin{aligned} s_2 &= h_2 \oplus G'_{n-1:0} \oplus c_2 \\ &= h_2 \oplus G'_{n-1:0} \oplus (G_{1:0} \vee P_{1:0}G'_{n-1:0}) \\ &= h_2 \oplus (G'_{n-1:0}\overline{G_{1:0} + P_{1:0}} + G_{1:0}\overline{G'_{n-1:0}}) \end{aligned}$$

$$= (h_2 \oplus \overline{G_{1:0} + P_{1:0}})G'_{n-1:0} + (h_2 \oplus G_{1:0})\overline{G'_{n-1:0}} \quad (9)$$

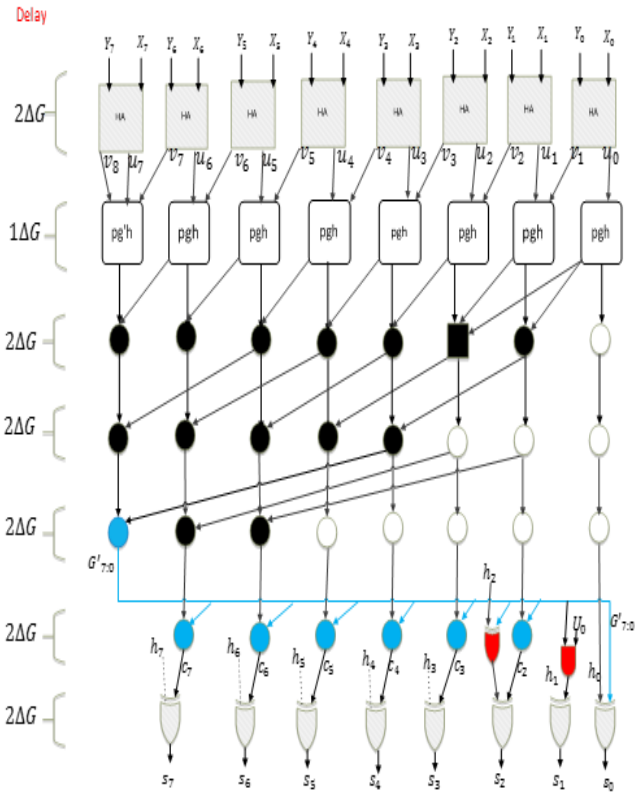


Figure 3. The proposed RPP modulo-$(2^n - 5)$ adder

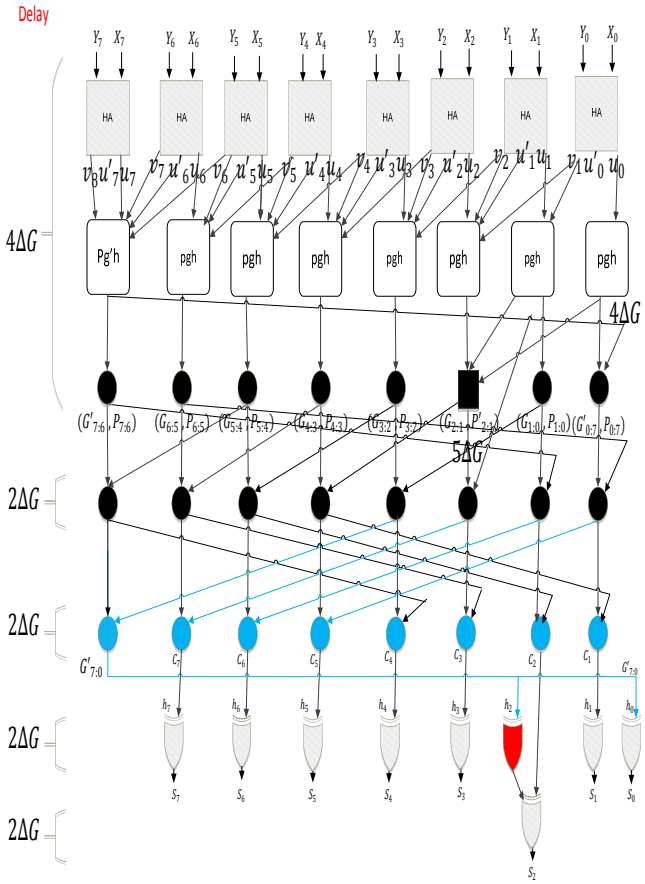


Figure 4. The proposed TPP modulo-$(2^n - 5)$ adder

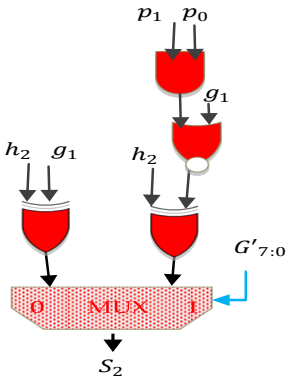


Figure 5. Modification for the terminal part of the s_2 path in figure 4

3.3. Addition of Two Excess-Modulo Residues

Most of RNS applications require modular addition and multiplications, where at least one operand is furnished via input data, which is therefore normally represented (i.e., single representation of residues). In such cases, Eqn. set 10 is meant to show that excess-modulo representation of the other operand does not jeopardize the resulted residue, where $A \in [0, 2^n - 5]$, $B \in [2^n - 5, 2^n - 1]$, and $B' = B - (2^n - 5)$, represent the normal representation of B. However, the same analysis fails for the case of both operands in excess-modulo representations, which can be handled as follows, in hypothetical cases of occurrence.

$$|A + B|_{2^n-5} = |A + B' + 2^n - 5|_{2^n-5} = |A + B'|_{2^n-5},$$
$$|A \times B|_{2^n-5} = |A \times (B' + 2^n - 5)|_{2^n-5} = |A \times B'|_{2^n-5} \quad (10)$$

The excess-modulo operand pairs $2^n - \delta_1$ and $2^n - \delta_2$, where $1 \le \delta_1, \delta_2 \le 5$, should be examined for possible wrong sums. Let $A + B = G_{n-1:0}w_{n-1} \dots w_0$, and $S = s_n s_{n-1} \dots s_0 = w_{n-1} \dots w_0 + 5G_{n-1:0}$. Therefore, $s_n = w_{n-1}c_{n-1} = (h_{n-1} \oplus G_{n-2:0})c_{n-1}$, where all the required three operands are available in the RPP and TPP realizations of figure 3 and 4. On the other hand, $s_{n-1} \dots s_0 = |A + B|_{2^n-5} = |2^n - \delta_1 + 2^n - \delta_2|_{2^n-5}$ denotes the sum that is produced by the proposed adders, where the EAC $= G'_{n-1:0} = 1$. Therefore, Eqn. set 11 holds, where the cases of $S \ge 2^n$ (i.e., $S \in \{2^n, 2^n + 1, 2^n + 2, 2^n + 3\}$), denoted by $s_n = 1$, are not valid, and should be corrected by another subtraction by $2^n - 5$, or actually addition by 5. However, this correction act affects only the four least significant bits of the sum (i.e., $s_3 s_2 s_1 s_0$), as is justified by the content of table 3.

$$S = 2^n - \delta_1 + 2^n - \delta_2 - (2^n - 5), 2^n - 5 \le S \le 2^n + 3 \quad (11)$$

Table 3. Two excess-$(2^n - 5)$ operands

A, B	$s_3 s_2 s_1 s_0$	$s'_3 s'_2 s'_1 s'_0$
$2^n - 4, 2^n - 1$	0000	0101
$2^n - 3, 2^n - 2$	0000	0101
$2^n - 3, 2^n - 1$	0001	0110
$2^n - 2, 2^n - 2$	0001	0110
$2^n - 2, 2^n - 1$	0010	0111
$2^n - 1, 2^n - 1$	0011	1000

Eqn. set 12 describes the corrected sum bits $s'_i (0 \le i \le 3)$ in terms of the original sum bits s_i. Note that $s_n = 0$ in other

cases that are not listed in table 3. Therefore, additional latency of the correction logic is 4ΔG.

$$s'_0 = s_0 \oplus s_n, s'_1 = s_1\overline{s_n} \vee (s_1 \oplus s_0)s_n, s'_2 = s_2\overline{s_n} \vee \overline{s_1 s_0}s_n, s'_3 = s_3\overline{s_n} \vee s_1 s_0 s_n \quad (12)$$

4. Evaluations and Comparisons

Recalling the modifications to figure 4 that was explained in Section 3.2, the overall latency of the proposed TPP modulo-$(2^n - 5)$ adder is equal to the desired $(4 + 2\lceil \log n \rceil)\Delta G$ in the practical cases, where at most one of the operands is excess-modulo. Also that of the RPP realization of figure 3 is $(6 + 2\lceil \log n \rceil)\Delta G$. Therefore, the proposed adders are latency balanced with the corresponding previous modulo-$(2^n - 3)$ adders of [8]. The gate level delay and area measures of the reference and proposed adders are listed in table 4. To confirm the results, therein, we have coded function of the four adders, perform correction tests, and simulated the corresponding circuits by TSMC .9 μm Synopsys Design Compiler. Figures 6-8 depict the area and power curves in terms of time constraints of the synthesis tool, for $n \in \{8, 16, 32\}$, respectively. These results confirm the latency balance between modulo-$(2^n - 3)$ and -$(2^n - 5)$ channels, where the area consumption and power dissipation are also compatible. The corresponding measures for the least met time constraint are compiled in tables 5-7.

Table 4. Analytical gate-level performance measures

Design	Delay (ΔG)	Area (ΔA)
RPP $(2^n - 5)$	$6 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 7n + 2}$
TPP $(2^n - 5)$	$4 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 7n + 7}$
RPP $(2^n - 3)$	$6 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 7n - 6}$
TPP $(2^n - 3)$	$4 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 8n - 10}$

Table 5. Synthesis results for n = 8

Design	Delay (ns)	Area (μm²)	Power (μw)
RPP $(2^n - 5)$	0.5897	19695.16	650.87
TPP $(2^n - 5)$	0.5499	19150.68	616.61
RPP $(2^n - 3)$	0.5499	17220.08	585.71
TPP $(2^n - 3)$	0.5498	20376.62	624.14

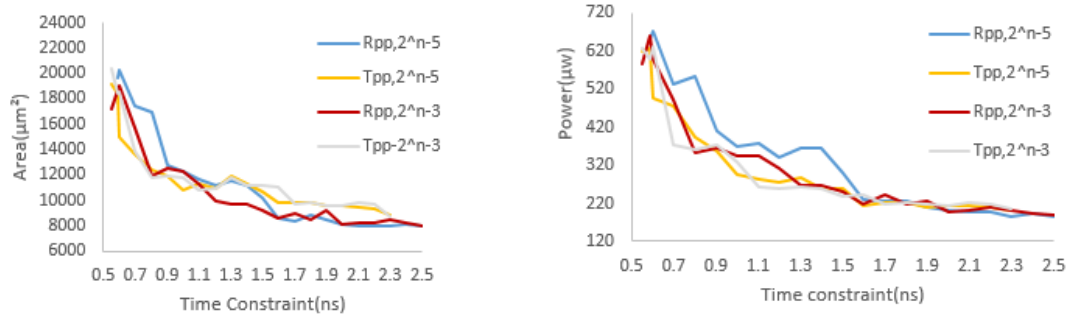


Figure 6. Area and power comparisons for n = 8

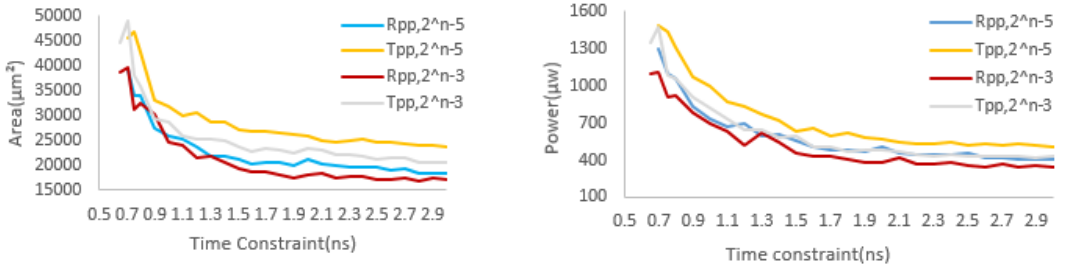


Figure 7. Area and power comparisons for n = 16

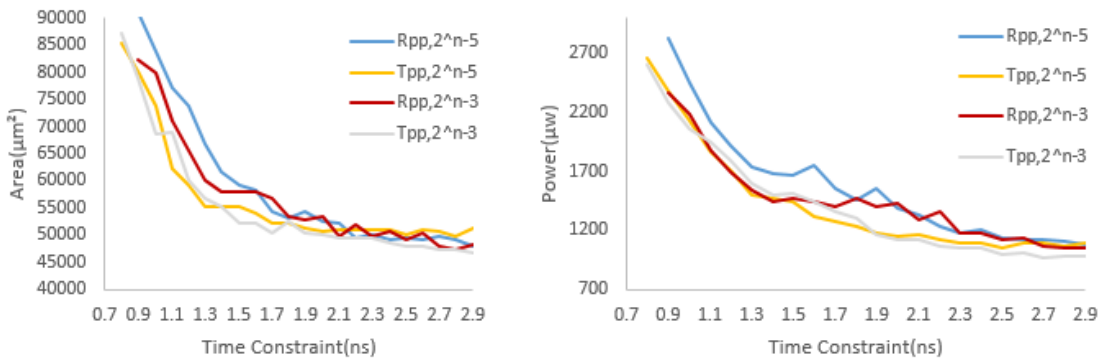


Figure 8. Area and power comparisons for n = 32

Table 6. Synthesis results for n = 16

Design	Delay (ns)	Area (μm^2)	Power (µw)
RPP (2^n-5)	0.7000	39286.79	1301
TPP (2^n-5)	0.6999	45391.73	1486.8
RPP (2^n-3)	0.6500	38553.01	1098.4
TPP (2^n-3)	0.6499	44467.15	1350.8

Table 7. Synthesis results for n = 32

Design	Delay (ns)	Area (μm^2)	Power (µw)
RPP (2^n-5)	0.89	91145.98	2822.4
TPP (2^n-5)	0.85	84412.00	2460.2
RPP (2^n-3)	0.90	82295.13	2363.9
TPP (2^n-3)	0.79	87038.07	2602.3

5. Conclusions

RNS applications can be realized to show better figures of merit (especially performance) in case of balanced moduli set, where the latency of all the computation channels are almost equal (i.e., at most $1\Delta G$ difference). Modular addition is in particular of interest, where parallel prefix $(3+2\lceil\log n\rceil)\,\Delta G$ adders have been presented for moduli set $\{2^n-1, 2^n, 2^n+1\}$ [2], and also modulo-(2^n-3) adders with $(4+2\lceil\log n\rceil)\,\Delta G$ performance [8]. Additional moduli of the form $(2^n-\delta)$ are desirable to accommodate larger dynamic ranges, provided that $(4+2\lceil\log n\rceil)\,\Delta G$ adders are feasible. We presented a modulo-(2^n-5) parallel prefix adder that meets the latter performance constraint. This was shown via gate-level analysis that was confirmed by synthesis results. Such performance was achieved at marginal penalty in area and power measures.

As for the relevant future work, for extending the dynamic range while keeping the whole moduli set balanced, we plan to study the design and implementation of compatible modulo-(2^n+3) and -(2^n+5) adders.

References

[1] H. Ahmadifar, and G. Jaberipur, "A New Residue Number System with 5-Moduli Set: $\{2^{2q}, 2^q\pm3, 2^q\pm1\}$," *The Computer Journal*, to appear.

[2] R. P. Brent, and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. On Computers, vol. C-31, no. 3, pp. 260-264, Mar. 1982.

[3] P. M. Kogge, and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," in IEEE Transactions on Computers, vol. C-22, no. 8, pp. 786-793, Aug. 1973.

[4] S. Knowles, "A family of adders," *Proceedings 14th IEEE Symposium on Computer Arithmetic*, pp. 277-281, June 2001.

[5] R. Ladner, and M. Fischer, "Parallel prefix Computation," *J. ACM*, vol. 27, no. 4, pp.831.838, Oct. 1980.

[6] G. Jaberipur, and S. Nejati, "Balanced Minimal Latency RNS Addition for Moduli Set $\{2^n-1, 2^n, 2^n+1\}$," 18th International Conference on Systems Signals and Image Processing (IWSSIP), pp. 1-7, 16-18 June 2011.

[7] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-Speed Parallel-Prefix Modulo 2^n-1 Adders," *IEEE Trans. Comput.* , vol. 49, no. 7, pp. 673-680, July 2000.

[8] G. Jaberipur, and S. H. F. Langroudi," $(4+2\log n)\Delta G$Parallel Prefix Modulo-2^n-3 Adder via Double Representation of Residues in [0, 2]," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 62, no. 6, pp. 583-587, June 2015.

[9] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, Oxford Univ. Press, 2000.

[10] R. Chokshi, K. S. Berezowski, A. Shrivastava, and S. J. Piestrak, "Exploiting residue number system for power-efficient digital signal processing in embedded processors," in Proc. of the international conference on Compilers, architecture, and synthesis for embedded systems (CASES), pp. 19-28, Oct 2009.

[11] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Fast Parallel-Prefix Modulo 2^n+1 Adder," *IEEE Trans. Comput.*, vol. 53, no. 9, pp. 1211-1216, Sept 2004.

Ghassem Jaberipur is an Associate Professor of Computer Engineering in the Department of Computer Science and Engineering of Shahid Beheshti University, Tehran, Iran. He received his BS in electrical engineering and PhD in computer engineering from Sharif University of Technology in 1974 and 2004, respectively, (where he is recognized as one of the 50 distinguished graduates for years 1966-2016), MS in engineering from UCLA in 1976, and MS in computer science from University of Wisconsin, Madison, in 1979. His main research interest is in computer arithmetic. Dr. Jaberipur is also affiliated with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), in Tehran, Iran.
E-mail: jaberipur@sbu.ac.ir

Hassan Ghasemi Motlagh received his B.Sc. degree in Computer Engineering from Mazandaran University of Science and Technology, Mazandaran, Iran, in 2014. Now, he is working on his M.Sc. in Computer Architecture at Shahid Beheshti University. His research interests include Computer Arithmetic, RNS.
E-mail: h.ghasemimotlagh@sbu.ac.ir

Paper Handling Data:

Submitted: 07.11.2016
Received in revised form: 09.12.2016
Accepted: 28.12.2016
Corresponding author: Dr. Ghassem Jaberipur,
Department of Computer Science and Engineering,
Shahid Beheshti University, Tehran, Iran.

The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 54-60
Regular Paper

A Low-Power Hierarchical FinFET-Based SRAM

Somayeh Maabi[1] Sina Sayyah Ensan[2] Mohammad Hossein Moaiyeri[3]
Shaahin Hessabi[2]

[1]Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran
[2]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[3]Faculty of Electrical Engineering, Shahid Beheshti University, Tehran, Iran

Abstract

In this paper, a low-power energy-efficient hierarchical SRAM design capable of working in near-threshold region is proposed. The proposed method enhances the noise margin using an extra circuitry, while restricting the hardware redundancy by sharing the additional circuitry between each two SRAM cells in a hierarchical style. The results of simulating the FinFET-based SRAM cells using Synopsys HSPICE at 10nm technology node indicate that the proposed design reduces, on average, the power-delay product, read and write delays by 14.34%, 2.37% and 8.54%, respectively, and significantly improves the static noise margins even in the presence of major process variations.

Keywords: Static Random Access Memory (SRAM), Multiport Memories, Low-Power Design, Static Noise Margin (SNM), FinFET, Nanoelectronics.

1. Introduction

Nowadays, by scaling the technology node, the variability concerns have become very important. This is due to (i) very small geometrics where even small variations can cause big changes, (ii) reduced power supply voltage, where V_{DD} level is very close to transistors threshold voltage [1]. Accordingly, the SRAM cell stability depending on transistors strength ratios has become one of the biggest concerns in VLSI design [2]. With the ever-increasing short channel effects, parametric variations and cache capacity, SRAMs have become very sensitive to variability. As a result, a new trade-off between the variability, power consumption and performance parameters has come in VLSI design [2].

Ignoring redundancy, each cell must work under worst-case variations. In the conventional 6T SRAM, strength ratios of devices must be adopted such that cell static noise margin and write margin are both upheld, while they are in conflict with each other. During the read state, it is desirable to have stronger storage inverters and weaker pass transistors, while during the write state it is desirable to have stronger access transistors and weaker storage inverters. This fine balance of transistor strength ratios can be easily affected by process variations, which will specifically degrade cell stability and read margin in nanoscale technologies [2].

In order to consider variability problems, many design techniques have been proposed to enable low-voltage operation for SRAM cells. A higher supply voltage can be used for SRAM cells such that the SRAM voltage does not decrease with the technology to meet the desirable margins [3]. However, this approach does not provide the demanded low power consumption. Another approach is to use dynamic voltage sources such that SRAM voltage source can dynamically change during different states [4]-[5]. This technique increases the design complexity, while it improves the cell stability and reduces the standby leakage.

Scaling the technology node has also led to a major power consumption challenge. The importance of power consumptions is more prominent in the systems with restricted energy sources. Reducing the supply voltage is considered as one of the most effective ways to decrease the power dissipation. This is achieved by sub-threshold or near threshold circuits design which enhances energy efficiency [6]. In this way, leakage power is also considerably reduced due to less voltage difference between drain and source of a transistor, which suppresses the DIBL effect, and consequently reduces the OFF current [7].

On the other hand, reducing the supply voltage results in higher propagation delay and slower circuits. At the present time, power saving is becoming very crucial in many circuits and applications that do not require very high speed, such as memories, nano-sensors, radio frequency identification and implantable medical devices. These applications may be in hold or stand-by modes most of the time, and need very low energy consumption and long battery lifetimes [8]. In addition, investigations have demonstrated that the minimum energy can be achieved in the sub-threshold region. In other words, appropriate power-delay product (PDP) can be found in the near-threshold region [9]. In the sub-threshold region, the drain current is defined by Eq. 1 [10].

$$I_{sub} = \mu c_{ox} \frac{W}{L} e^{1.8} \phi_T^2 \, e^{\frac{V_{GS}-V_{th}}{n\phi_T}} \left(1 - e^{\frac{-V_{DS}}{\phi_T}}\right) \quad (1)$$

where, $n = 1 + \frac{C_{dep}}{C_{ox}}$ is the sub-threshold factor and $\emptyset_T$is the thermal voltage.

A specific and important state of a transistor in a nanoscale digital circuit is the OFF state, where V_{GS}=0V and V_{DS}=V_{DD}. The drain current in this state, which is denoted as I_{OFF}, is given by Eq. 2. It can be concluded from Eq. 3 that the I_{OFF} leakage current is reduced exponentially by decreasing the supply voltage.

$$I_{OFF} \simeq \mu c_{ox} \frac{W}{L} e^{1.8} \phi_T^2 \, e^{\frac{-V_{th}}{n\phi_T}} \quad (2)$$

On the other hand, reducing the supply voltage to near-threshold region considerably increases the circuit delay. The propagation delay in the sub-threshold region is almost given by Eq. 3, where K is a suitable fitting coefficient.

$$t_d \simeq \frac{KCV_{DD}}{\mu c_{ox} \frac{W}{L} e^{1.8} \phi_T^2 \, e^{(V_{GS}-V_{th})/n\phi_T}} \quad (3)$$

Due to a significant speed reduction, sub-threshold circuits are often unsuitable for high-performance applications. However, they are practical for some applications, which do not require high-frequency operation but demand ultra-low power consumption, such as wireless sensor networks. However, increase in variations in sub-threshold region can degrade circuit stability [17].

SRAMs occupy a considerable part of the area and dissipate a large amount of power in VLSI chips [16]. Therefore, reducing the power consumption in SRAM cells is a vital issue in low-power VLSI design. As SRAM cells are often in the hold state, decreasing the static power dissipation of SRAM cells considerably contributes in reducing the total power consumption of a chip. Operating of SRAM in the sub-threshold region can dramatically reduce the leakage power, but it degrades the speed and static noise margin (SNM) of the cell, which should be considered in the design methodology [8].

In order to meet the challenges imposed by transistor scaling, different device structures have been explored as alternatives to the conventional bulk MOSFET. However, considering these technologies, FinFET is more appropriate for scaling the MOSFET to near 10nm feature size [11]. FinFETs demonstrate superior gate control on the channel, lower sub-threshold swing, lower short channel effect and higher scalability. However, width quantization is a challenge in FinFETs, which restricts the design possibilities of FinFET-based circuits such as SRAMs that need transistor sizing for correct operation and robustness. This constraint should be carefully considered and compensated in the design procedure [12].

In this paper, a new hierarchical SRAM cell design, denoted as HSRAM, is proposed with improved SNM and static and dynamic power consumption metrics. The number of transistors in every cell is decreased compared to its counterpart presented in [14], by multiplexing the additional transistors between each two cells, which reduces the power and energy consumptions of the proposed design.

The rest of this paper is organized as follows: In the next Section related work is reviewed. The proposed SRAM cells are introduced and described in the Proposed Design Section. The simulation results and comparisons are given in Simulation Result Section, and finally Summery Section concludes the paper.

2. Related Work

Designing low-voltage memories is demanded to reach lower power consumption. The classic 6T (6-transistor) SRAM cell is shown in figure 1. Assume that the X and Y nodes have stored 0 and 1 values, respectively, as illustrated in figure 1. In this case, M1 and M4 are ON and M2 and M3 are OFF. During the hold time, when WL is 0, M5 and M6 are OFF. In this state, the leakage current through M3 can proportionally lead to failure, since in the idle mode the leakage current through M2 and M5 causes a small increment in the voltage of node X, which is not acceptable due to small SNM in sub-threshold region.

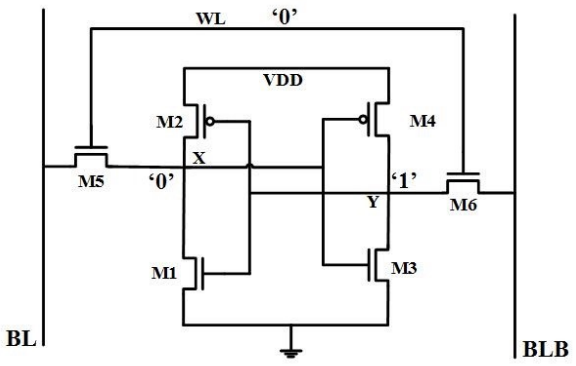


Figure 1. 6T SRAM cell

During the read cycle, when WL=1 and BL=BLB=V_{DD}, M5 and M6 are ON and consequently, BL will be discharged through M1 and M5, and read operation is accomplished. In this situation, due to existence of some leakage through M3 there is a drop in Y, which results in reading speed degradation and possible data flips on storage nodes. In addition, transistor sizing is not adequate to prevent failures, especially in near- and sub-threshold regions [13].

The 11T SRAM cell [14], illustrated in figure 2, improves the whole energy consumption, the ability of working in sub threshold supply voltages the SNM of the 6T SRAM cell but with the penalty of higher number of transistors and larger area. However, the power consumption specially static power, is the critical challenge in designing digital systems especially in SRAMs.

As the 11T and 13T SRAM cells [14] focus on reducing the static power consumption and improving the performance at sub-threshold region, we discuss these cells in more detail. In the 11T cell, M2, M4, M5 and M6 transistors keep the aforementioned characteristics of the 6T cell. However, the size of the M1 and M3 transistors are scaled down equal to the size of the p MOS transistors. In addition, separate BL, WL and RDWL (read word line) lines are considered in this cell, which leads to separated read and write ports.

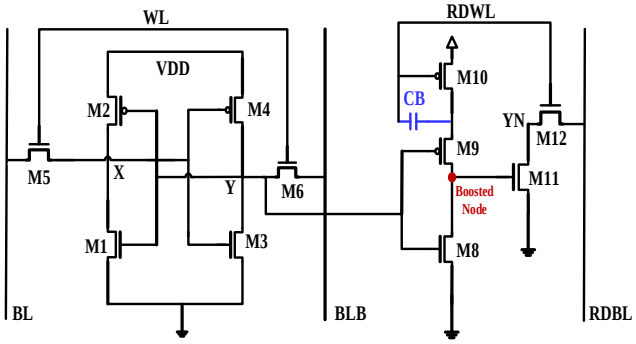


Figure 2. The 11T SRAM cell [12]

During the hold state, the WL and RDWL lines are not activated. Suppose the situation when Y=0 and X=1, as described before for the 6T cell. In the additional 5T circuitry, M12 is OFF and the state of M11 depends on the voltage at node Y. When Y=0, M8 is OFF and M9 is ON, and consequently the gate of M11 is charged. Therefore, a path is created through M11 that connects the node YN to zero. The transistors of the added 5T circuit have minimum size, except the access transistor (M12) that has a larger size.

The boost capacitor (CB) that connects the source of M9 to RDWL is a substantial part in this cell. Given that Y=0 (during the hold and write times) and RDWL is connected to ground, CB starts to charge up to V_{DD}. Though, the amount of charging is limited by M9 and finally the maximum voltage reaches $V_{DD}/2$. Thus, the gate of M11 is connected to a voltage of approximately less than $V_{DD}/2$ and YN discharges to the ground slowly.

When RDWL is selected, the source of M9 rises to 1.5 V_{DD} and the gate of M11 connects to a voltage higher than V_{DD}. This enhances the read current by an order of magnitude as compared to the classical 6T SRAM cell, which leads to faster read operation. In the other case when Y=1, M8 is on and M11 is off, which makes the reading path isolated from the ground. Although the leakage through M11 causes some reduction in the RDBL voltage, this leakage has no effect on YN or RDBL because of connecting M9 to ground via M8.

To suppress the channel leakage path through M8, a modified topology denoted as 13T cell is suggested in [14]. This cell which has 13 transistors is shown in figure 3. When Y is high the source of M8 is connected to ground through the added inverter, but when Y stores “0”, the source of M8 is connected to V_{DD}, which suppresses the leakage path in M8 during the read cycle.

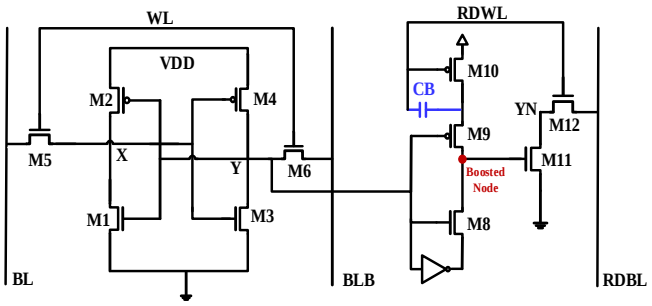


Figure 3. An improved 11T SRAM cell (13T SRAM) [12]

3. Proposed Design

Design of low-voltage digital circuits operating in near- or sub-threshold regions has emerged as a low-power solution for applications with energy constraints. As SRAM memory constitutes a significant percentage of the whole power and area of most digital chips, reducing its power consumption, especially at the stand-by mode, is quite important.

As stated before, the 6T SRAM cell is not suitable for low-voltage applications as it has a small SNM. Accordingly, some effective solutions for improving its SNM in low voltages have been presented in the literature [14]. Despite the improvements in SNM, adding 5 to 7 transistors to the 6T SRAM cell increases the hardware overhead, and consequently the delay and power consumption in the memory system. As a result, for designing an SRAM that maintains the aforementioned SNM improvements, while restricting the hardware redundancy, delay and power, a new hierarchical SRAM (HSRAM) cell is proposed in this paper. The proposed SRAM cell is shown in figure 4. In the proposed design, the hierarchical style also reduces the cell power consumption by use of a multiplexer in the reading path. In this design, due to separating the reading part from the cell, no current is drawn for reading from the cell. Unlike the 6T cell, the read SNM of the proposed cell is not degraded, and is similar to its hold SNM.

As illustrated in figure 4, for reducing the hardware overhead in the 11T SRAM, instead of adding five transistors for each cell, these 5 additional transistors can be shared between each two cells. In this design, the read lines are connected to each other, and the write lines for each cell are separate, as the write ability should be provided for every cell in an SRAM. In order to read from these SRAM cells, a transmission gate-based 2:1 multiplexer is utilized, which connects the SRAM core cells to the read stage. It is worth mentioning that by connecting the read lines, the data stored in each core cell does not become faulty and it is not corrupted during the read procedure.

In the read state, the read line is activated for each of the two cells, and the multiplexer connects the corresponding core cell to the 5-transistor read stage based on the selector signal. If the selector signal becomes '0', the first cell is chosen; otherwise, the second cell is selected. It is noteworthy that the transmission gate structure of the multiplexer provides low-resistive paths and full-swing signals for the read stage.

In addition, the additional inverter existing in the original circuit [14] is omitted in the proposed design as it does not have any considerable effect on the static power, while omitting these two transistors reduces the overall switching power of the SRAM. It is worth pointing out that sharing the read stage also leads to a considerable reduction in the read bit line (RDBL) and read word line (RDWL) capacitances, and consequently results in lower dynamic power consumption and higher speed.

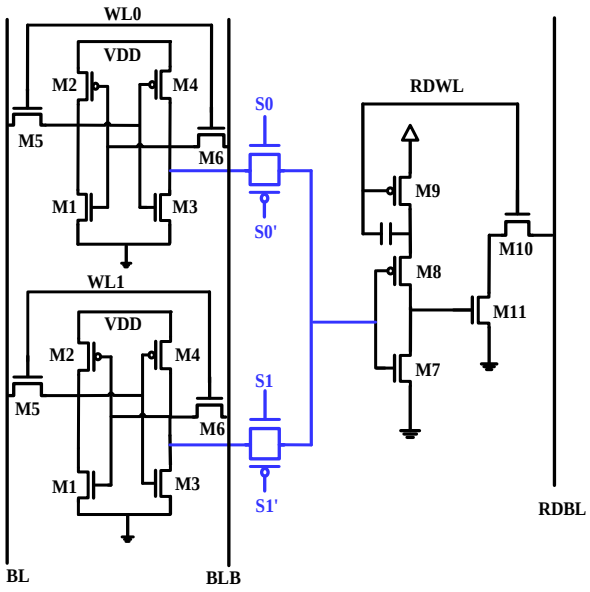


Figure 4. The proposed SRAM (HSRAM)

By categorizing the memory cells and integrating the read line, changing the addressing model of the SRAM cells is necessary. As there is one read line activation for each two cells, according to the conventional models, some clusters of cells may be activated and put on the output simultaneously, which leads to data collision. For solving this problem, the address is divided into two 2-bit parts.

The least significant bit is used as the selectors of the clusters. This bit can be 0 or 1, which will select one of the two cells of each cluster, and the remaining bits will choose the corresponding cluster, and accordingly the corresponding cell is selected to put its data on the output line. These two addressing models are illustrated in figure 5.

4. Simulation Result

The circuits are simulated using the HSPICE simulator tool with 10nm LSTP FinFET technology [15] at 0.45 V supply voltage and at 60°C temperature. Some of the important parameters of the utilized model are given in table 1. In the simulations, all the SRAM's transistors have the minimum size except the ones in the 6T conventional SRAM cell, which have multiple sizes in order to have a good Read SNM. The capacitors of the BL, BLB, RDBL and RDWL lines are assumed as 10fF for a basic design. The power consumption, read and write delays, average case power delay product (PDP), best case PDP, worst case PDP and SNM of the proposed SRAM cell are calculated and compared with those of the other low-power SRAM cells.

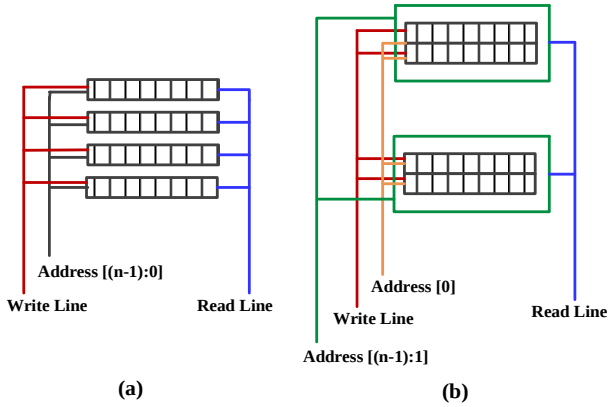


Figure 5. Two addressing models: (a) Conventional model (b) HSRAM Model

Table 1. The FinFET parameters

Parameter	Description	Value
L_g	Physical Gate Length	14 nm
H_{Fin}	Fin Height	21 nm
T_{Fin}	Fin Thickness	8 nm
EOT	Equivalent oxide thickness	0.68 nm
N_{body}	Doping of the body	2.5×10^{16} cm^{-3}
N_{sd}	Doping of the source-drain regions	3×10^{20} cm^{-3}
Φ_{gn}	Work Function of the gate for N-type Transistor	4.42 eV
Φ_{gp}	Work Function of the gate for P-type Transistor	4.75 eV

The simulation results for all SRAM cells are given in table 2 Average PDP is the multiplication of average power and average delay. Best case PDP stands for the product of the average power and best delay, while worst case PDP stands for the product of the worst dynamic power and the worst delay. According to the results, the proposed cell has lower power, write delay and PDP than the other designs. This is because of different structure and operation as compared to 6T, and lower number of transistors and less line capacitance in comparison with 11T and 13T cells.

Table 2. The simulation results at V_{DD}=0.45V

SRAM	Average Power (nW)	Static Power (pW)	Read Delay (ps)	Write Delay (ps)	Average Case PDP (aj)	Best Case PDP (aj)	Worst Case PDP (aJ)
HSRAM	24.81	286.04	311.85	173.11	5.3080	3.8930	19.72
6T	25.60	307.30	319.46	212.64	6.6243	4.7376	20.86
11T	52.91	274.98	244.31	173.86	10.216	8.4497	33.16
13T	36.81	298.11	245.08	177.27	7.1120	5.8346	23.08

Read access delay, which is the time required to discharge RDBL to "0.8×V_{DD}" after RDWL reaches to $V_{DD}/2$, is one of the metrics in SRAM cells. During read operation, RDWL is activated and the access transistor is turned on. If the data which is going to be read is '0', then RDBL will be discharged to ground; and if the data is '1', no change happens in RDBL. According to the results, the read delay access of the proposed HSRAM is 27.24% and 21.65% longer than those of the 13T and 11T SRAM cells, respectively, due to the multiplexer stage, but 2.37% lower than the 6T SRAM since it has a separate read circuitry. However, the proposed method considerably improves the power consumption and PDP metrics as compared to other cells.

Another important metric in evaluation of SRAM cells is the write access delay, which is defined as the time required to charge the node that has stored 0 to $V_{DD}/2$ after WL reaches to $V_{DD}/2$. During write operation, WL is activated and access transistors are turned on. Then, new data is transferred by these transistors to cell. Write delay access of the proposed HSRAM is 2.4%, 0.43% and 22.8% lower than the 13T, 11T and 6T cells, respectively. In the 6T SRAM, in order to have read ability, the size of transistors are different, and therefore, they do not have the same strength. On the other hand, as the pull down transistors have higher strengths than the access transistors, the write delay gets longer.

The delay, power consumption and PDP of the cells versus supply voltage are plotted in figure 7 and figure 8, respectively. According to the results, the proposed cell has lower power consumption and PDP as compared to the other cells in a wide range of supply voltages, especially at lower voltages.

Static noise margin (SNM) is an important metric in evaluating the robustness of an SRAM cell. The SNM parameter is calculated as the diameter of the biggest square found in the butterfly graph of a SRAM cell. For analyzing SNM, Monte Carlo analysis with Gaussian distribution with 10% variation at the 3-sigma is conducted. Process variations are considered in the Fin height, Fin width and channel length as the important FinFET device parameters. Finally, the diameter is measured in the butterfly graph. The butterfly graphs indicating the hold and read SNMs of the propose SRAM cell are illustrated in figure 7.

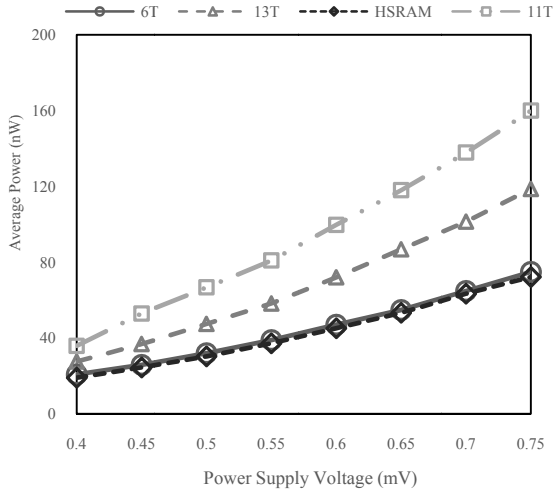


Figure 6. Average Power of the FinFET SRAM cells (T = 60 °C)

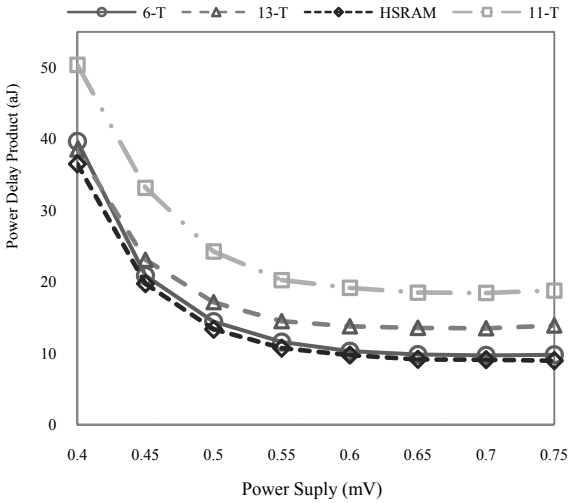
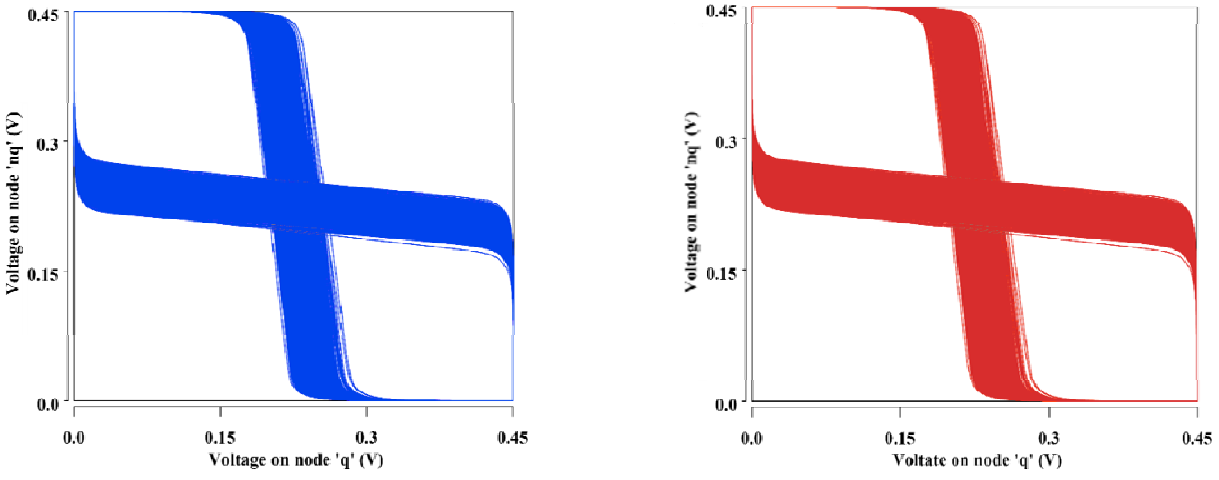


Figure 7. PDP of the FinFET SRAM cells (T = 60 °C)



(a) (b)

Figure 8. The butterfly curves of the proposed cell (a) Hold (b) Read

Table 3 gives the hold and read SNMs, as well as their variations for the SRAM cells. According to table 3, the hold SNMs for all cells are almost comparable. In the read state, the proposed cell as well as the 11T and 13T cells has SNMs equal to their hold state as there is no charge sharing between the separated read bit line and the main cell. However, in the 6T SRAM cell, due to connecting the cell to BL and BLB, the pulling current (charge sharing) from the cell decreases the read SNM.

Table 3. SNM of the SRAM Cells (V_{DD}= 0.45 V)

SRAM Cell	SNM (mV)	Variation (mV)
Hold		
HSRAM	183	34
6T SRAM	182	34
11T SRAM	182	35
13T SRAM	182	35
Read		
HSRAM	183	35
6T SRAM	78	41
11T SRAM	182	35
13T SRAM	182	35

5. Conclusion

In this paper, a hierarchical FinFET SRAM cell (HSRAM) has been proposed. In this design, each two 6T SRAM cells share an additional five-transistor circuitry to improve design complexity and power consumption, and achieve higher SNM and execution speed in sub-threshold region. For reading from these SRAM cells, each two cells are connected to the shared read circuitry through a 2:1 multiplexer and finally to the SRAM output. It is worth mentioning that by connecting the read lines, the data stored in every cell does not become faulty, and is not corrupted during the reading procedure. The SRAM cells have been simulated using 10nm LSTP FinFET HSPICE model. It was shown that the proposed structure reduces the number of transistors, line capacitance, power consumption and PDP as compared to its counterpart cells.

References

[1] A. Shafaei, and M. Pedram, Energy-efficient cache memories using a dual-Vt 4T SRAM cell with read-assist techniques. In*2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 457-462, 2016.

[2] L. Chang, R. K. Montoye, Y. Nakamura, K. A. Batson, R. J. Eickemeyer, R. H. Dennard, W. Haensch, and D. Jamsek, "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches," IEEE Journal of Solid-State Circuits, vol. 43, no. 4, 2008.

[3] J. Davis, D. Plass, P. Bunce, Y. Chan, A. Pelella, R. Joshi, A. Chen, W. Huott, T. Knips, P. Patel, K. Lo, and E. Fluhr, "A 5.6 GHz 64kb dual-read data cache for the power6tm processor," IEEE International Solid State Circuits Conference (ISSCC), Digest of Technical Papers, 2006.

[4] A. J. Bhavnagarwala, S. V. Kosonocky, S. P. Kowalczyk, R. V. Joshi, Y. H. Chan, U. Srinivasan, and J. K. Wadhwa, "A transregional CMOS SRAM with single logic VDD and dynamic power rails," Symposium on VLSI Circuits, Digest of Technical Papers, 2004.

[5] K. Zhang, U. Bhattacharya, Z. Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Y. Wang, B. Zheng, and M. Bohr, "A 3-GHz 70-Mb SRAM in 65-nm CMOS technology with integrated column-based dynamic power supply," International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers, pp. 474-611, 2005.

[6] Y. Yang, H. Jeong, S. C. Song, J. Wang, G. Yeap, and S. O. Jung, "Single Bit-Line 7T SRAM Cell for Near-Threshold Voltage Operation With Enhanced Performance and Energy in 14 nm FinFET Technology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 7, pp. 1023-1032, 2016.

[7] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-sub micrometer CMOS circuits," Proceedings of the IEEE, vol. 91, no. 2, pp. 305-327, 2003.

[8] M. Moghaddam, S. Timarchi, M. H. Moaiyeri, and M. Eshghi, "An Ultra-Low-Power 9T SRAM Cell Based on Threshold Voltage Techniques," Circuits, Systems and Signal Processing, Springer, vol. 35, no. 5, pp. 1437-1455, 2016.

[9] M. Alioto "Ultra-Low Power VLSI Circuit Design Demystified and Explained, a Tutorial," IEEE Trans. Circuits Syst. I, Regular papers, vol. 59, pp. 3–29, 2012.

[10] J. Rabaey, Low Power Design Essentials, first ed., New York, Springer US, 2009.

[11] J. P. Colign, FinFETs and Other Multi-Gate Transistors, first edition, Springer, New York, 2008.

[12] S. Kumar Gupta, and K. Roy, Low Power Robust FinFET-Based SRAM Design in Scaled Technologies, Circuit Design for Reliability, Springer E-Publishing Inc., New York, pp. 223-253, 2015.

[13] O. Thomas, M. Reyboz, and M. Belleville, "Sub-1V, Robust and Compact 6T SRAM cell in Double Gate MOS technology," Proc. IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2778 – 2781, 2007.

[14] F. Moradi, D. T. Wisland, S. Aunet, H. Mahmoodi, and Tuan Vu Cao1, "65nm Sub-Threshold 11T-SRAM for Ultra Low Voltage Applications," IEEE international SoC conference, pp. 113-118, 2008.

[15] Predictive Technology Model, [Online]. Available at http://ptm. asu.edu/.

[16] S. M. Salahuddin, and M. Chan, "Eight-FinFET Fully Differential SRAM Cell with Enhanced Read and Write Voltage Margins," *IEEE Transactions on Electron Devices*, vol. 62, no. 6, pp. 2014-2021, 2015.

[17] Y. Yang, J. Park, S. C. Song, J. Wang, G. Yeap, and S. Q. Jung, "Single-Ended 9T SRAM Cell for Near-Threshold Voltage Operation with Enhanced Read Performance in 22-nm FinFET Technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2748-2752, 2015.

Somayeh Maabi received the B.S. and M.S. degrees in Computer Engineering from Islamic Azad university of Qazvin branch, in 2008 and 2013, respectively. She is currently a PhD candidate in Computer Engineering in Shahid Beheshti University, Tehran. Her research interests include SoC, NoC, fault tolerance, embedded systems, and processor design specially in end nodes of IoT and WSN
E-mail: s_maabi@sbu.ac.ir

Sina Sayyah Ensan Received his B.Sc. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, in 2014. He is currently pursuing M.Sc. degree in computer engineering at Sharif University of Technology, Tehran, Iran. His research interests include Low Power Circuits and VLSI design
E-mail: sayyah@ce.sharif.edu

Mohammad Hossein Moaiyeri received the Ph.D. in Computer Engineering from Shahid Beheshti University, Tehran, Iran in 2012. He is currently an Assistant Professor in the Faculty of Electrical Engineering of Shahid Beheshti University. His research interests mainly focus on nanoelectronic circuit design, Low-power VLSI design, VLSI implementation of MVL and fuzzy logic, and mixed-signal circuit design.
E-mail: h_moaiyeri@sbu.ac.ir

Shaahin Hessabi received the B.Sc. and M.Sc. degrees in Electrical Engineering from Sharif University of Technology, Tehran, Iran in 1986 and 1990, respectively. He received his Ph.D. degree in Electrical and Computer Engineering from University of Waterloo, Waterloo, Ontario, Canada in 1995. He joined Sharif University of Technology in 1996, and is currently an associate professor at the Department of Computer Engineering. His current research interests include System-on-Chip and Network-on-Chip, optical interconnects, and VLSI design and test. He has published more than 100 refereed papers in the related areas. Dr. Hessabi has served as the program chair, general chair, and program committee member of various conferences
E-mail: hessabi@sharif.edu

Paper Handling Data:

Submitted: 20.09.2016
Received in revised form: 02.11.2016
Accepted: 12.12.2016
Corresponding author: Dr. Shaahin Hessabi,
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

The CSI Journal on Computer Science & Engineering

INFORMATION FOR AUTHORS

The final manuscript of the English papers accepted for publication in the CSI JCSE should be prepared in accordance with this guide. It is, however, strongly recommended that preparation of the initial manuscript also follows the instructions given here. Manuscript may be in English or Farsi.

1. Paper Organization

- Title: The title should be short (at most 15 words) and indicative of the paper contents.
- Authors: the authors' names (initial and last name) and their affiliations should appear next.
- Abstract: the abstract should include the problem explanation, methods used for solution, and the significant results; and should not be longer than 150 words.
- Keywords: The keywords should be relatively independent and together optimally characterize the paper. Include 5 to 10 keywords.
- Text: the main body of the paper should start with Introduction and end with concluding remarks. All sections and sub-sections should be numbered. The number for introduction is 1.
- Acknowledgement: If required, Acknowledgements appear after the concluding remarks.
- References: All publications cited in the text should be presented in the order they are cited in a numbered list of References following Acknowledgements. In the text refer to references by the reference number in square brackets on the line.
- Appendices: If appendices are necessary, they are placed after the list of References.
- Farsi Section: the title, authors' names and affiliations, abstract, and keywords should also be given in Farsi a separate sheets (for papers written in English). For the non-Farsi speaking authors, the journal will supply this section.

2. Figures, Tables, Photographs and Equations

- Figures and Tables: Each figure or table must have a number and a caption. In figures, the number and the caption appear under the figure while in tables, they appear over it. The size of text and numbers in tables and figures must be suitable to allow high legibility. Do not use any type of shading in computer generated illustrations.
- Photographs: High quality glossy black and white photographs must be supplied as they are to be reproduced.
- Equations: Equations are to be numbered consecutively. The number of each equation should appear in parentheses in the right-most end of the equation line. Sufficient space should be allowed above and below each equation.

3. References

The format for various types of references should be as follows:

[1] M. A. Ahadi, and M. H. Rahimi, *Fuzzy Set Theory*, New Jersey, Prentice-Hall, 1995.

[2] M. A. Ahadi, M. H. Rahimi, and A. Fatemi, "Evidence-Based Recognition of 3D Objects," *IEEE Trans. Pattern Analysis and Mach. Intell.*, vol. 12, no. 10, pp. 18-25, 1994.

[3] A. Taheri, "On-line Fingerprint verification," *Proc, IEEE Int'l Conf. Pattern Recognition*, pp. 752-759, 1992.

[4] M. A. Ahadi, *On-line Fingerprint verification*, Ph.D. Dissertation, University of Tehran, Tehran, Iran, 1994.

[5] M. A. Washington, "The Fingerprint of Malcom X," http://www.dermatologyphicisc.com, June 2004.

[6] International Biometrics Group, http://www.Biometrics.com, May 2003.

4. Manuscript preparation

Typing and Printout: The manuscript should be prepared using Microsoft Word and should be printed on A4 size paper using a laser printer.

Fonts: Use Times New Roman font type. The font size must be 9 for the Abstract and 10 for the main text. All the titles should be made bold. The paper title must have a font size of 18, first-level sub-titles a size of 14, second-and third-level sub-titles a size of 12.

Layout: The paper should be typed in 2-column single space format. The top margin for the first page should be equal to 85 mm, each column length should be 87mm, and the spacing between columns 6mm. Only the abstract is to be typed in a single column format. Two space line must be allowed between the paper title and authors' names, and one space line should be allowed above each section title or sub-title.

5. Paper Submission

Exclusive Submission: Submission of a paper to JCSE implies that it has not been published previously, that it is not under consideration for publication elsewhere, and that if accepted, it will not be published elsewhere in any language. Explicit announcement of this matter must be made in a letter to the editor.

Paper Length: The paper is expected to be no longer than 30 pages.

Submission Process: Authors are requested to submit their papers electronically in PDF format at the journal homepage (www.jcse.ir). All relevant correspondence should be addressed to csi-jcse@ipm.ir.

Review Process: Each manuscript will be reviewed by experts and their constructive criticism will be forwarded to the authors.

Final Manuscript: After the acceptance of a paper, the authors should provide the final manuscript in PDF and Word formats. In addition, the authors should sign and submit the form for the transfer of copyright to the Computer Society of Iran.

علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴

یک سلول SRAM سلسله مراتبی کم توان مبتنی بر FinFET

سمیه مابی[۱] سینا سیاح انسان[۲] محمد حسین معیری[۳] شاهین حسابی[۲]

[۱]دانشکده علوم و مهندسی کامپیوتر، دانشگاه شهید بهشتی، تهران، ایران
[۲]دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران
[۳]دانشکده مهندسی برق، دانشگاه شهید بهشتی، تهران، ایران

چکیده

در این مقاله، طراحی یک سلول SRAM سلسله مراتبی، که قابلیت کار در ناحیه زیرآستانه را دارا می‌باشد، ارائه شده است. روش‌های ارائه شده با استفاده از مدارهای اضافی، حاشیه نویز ایستا را افزایش می‌دهند، در حالی که با اشتراک‌گذاری بعضی از قسمت‌های مدار بین دو سلول در شیوه سلسله مراتبی، افزونگی سخت‌افزاری را محدود می‌کنند. نتایج شبیه‌سازی سلول SRAM مبتنی بر FinFET با استفاده از نرم‌افزار Synopsis H Spice با تکنولوژی 10nm نشان می‌دهد که سلول ارائه شده انرژی، تاخیر خواندن و تاخیر نوشتن را به طور میانگین به ترتیب ۱۴.۳۴٪، ۲.۳۷٪ و ۸.۵۴٪ کاهش می‌دهد. به علاوه، این سلول حاشیه نویز ایستای خواندن را حتی در حضور تغییرات فرایند به شدت افزایش می‌دهد.

کلمات کلیدی: حافظه دسترسی تصادفی ایستا (SRAM)، طراحی با توان مصرفی کم، حاشیه نویز ایستا، حافظه سلسله مراتبی.

علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴

تاثیر نمایش افزون بر پیمانه مانده‌ها روی جمع پیشوندموازی به پیمانه 2^n-5

قاسم جابری‌پور حسن قاسمی مطلق

دانشکده علوم و مهندسی کامپیوتر، دانشگاه شهید بهشتی، تهران، ایران

چکیده

در سیستم اعداد مانده‌ای به دست آوردن کانال‌های محاسباتی که تاخیر متعادل داشته باشند، بسیار مطلوب است. در حالت خاص می‌توان به جمع‌کننده به پیمانه $(2^n-\delta)$ با مقادیر انتخابی δ (بصورت $\delta\in\{1, 3, 5\}$) اشاره کرد. از آن جمله جمع‌کننده به پیمانه (2^n-1) تنها با یک جمع‌کننده مکمل یک ساخته می‌شود که در آن خروجی برابر با پیمانه بعنوان نمایش دوم صفر محسوب می‌گردد. جمع‌کننده پیشوند موازی سریع برای این حالت با تاخیر $\Delta G\ ([\log n]+2)$وجود دارد که ΔG تاخیر یک گیت دو ورودی ساده را مشخص می‌کند. بطور مشابه، جمع‌کننده به پیمانه (2^n-3) با تاخیر $\Delta G\ ([\log n]+2)$ پیشنهاد شده است که در آن مانده‌های $\{0, 1, 2\}$ دو نمایش دارند. ولی دو نمایشی بودن این مانده‌ها عمل بعدی RNS را خراب نمی‌کند. برای رسیدن به دامنه پویا گسترده‌تر، و در عین حال حفظ عرض کانال (به عنوان مثال n) در کمترین مقدار ممکن، جمع‌کننده سریع به پیمانه (2^n-5)، می‌تواند کاملا مفید باشد. این موضوع انگیزه‌ای شد تا به طراحی و پیاده‌سازی چنین جمع‌کننده‌هایی با هدف به دست آوردن تاخیر برابر با $\Delta G\ ([\log n]+2)$ بپردازیم. اگر چه طرح پیشنهادی با جمع‌کننده به پیمانه (2^n-3) که قبلاً ذکر شد، اساساً یکسان می‌باشد، ولی در عین حال چالش‌های خاص بسیاری وجود دارد که باید بر آن‌ها غلبه کرد. نتایج آنالیزهای به دست آمده از اندازه‌گیری‌های تاخیر و مساحت در طرح پیشنهادی از طریق سنتز مدار که با نرم‌افزار Design Compiler برای شرکت Synopsis شبیه‌سازی شده نیز مورد تایید قرار گرفته است.

کلمات کلیدی: جمع‌کننده به پیمانه 2^n-5، نمایش افزون بر پیمانه، جمع‌کننده‌های پیمانه‌ای پیشوندموازی، پردازش علائم رقمی.

علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴

ارائه روشی مبتنی بر یادگیری عمیق برای تخمین جهت نگاه انسان با استفاده از اطلاعات چشم و جهت سر

رحیم انتظاری محمد مهدی ارزنی محمود فتحی امیرحسین بیات

دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران

چکیده

امروزه توسعه سامانه‌هایی که توانایی تحلیل حالات انسان را دارند از اهمیت ویژه‌ای برخوردار هستند. به‌طور خاص، جهت نگاه انسان نقش مهمی در بیان خواسته‌های فرد، نیازها، فرآیندهای شناختی، حالات احساسی و عاطفی و ارتباطات بین افراد ایفا می‌کند. تخمین جهت نگاه انسان کاربردهای فراوانی ازجمله تحلیل توجه، رابط‌های کاربری مبتنی بر چشم و تشخیص فعالیت‌های انسان دارد. علیرغم پیشرفت‌های اخیر در این زمینه، بسیاری از روش‌های کنونی نمی‌توانند شرایط مختلفی همچون حرکت سر را به‌خوبی مدل کنند. اخیراً مقالاتی برای حل این مشکل ارائه شده‌اند اما دقت خوبی را به دست نداده‌اند. در این مقاله، از روشی مبتنی بر شبکه‌های عصبی عمیق برای تحلیل ویژگی‌های ظاهری استفاده شده است که از اطلاعات جهت سر انسان نیز بهره می‌برد. روش مذکور به همراه تنظیمات استفاده شده به‌طور چشمگیری دقت را بهبود داده است.

کلمات کلیدی: تخمین جهت نگاه، تحلیل توجه، جهت سر انسان، شبکه‌های عصبی عمیق.

علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴

بررسی ایجاد قابلیت بازپیکربندی در جمع‌کننده‌های دهدهی

سمانه امامی مهدی صدیقی

دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران

چکیده

در سال‌های اخیر حساب دهدهی به موضوع داغی در تحقیقات تبدیل شده است. واحدهای سخت‌افزاری زیادی برای انجام عملیات حساب دهدهی کارا و دقیق، طراحی و پیشنهاد شده‌اند. به طور معمول، واحدهای حساب دهدهی به صورت ماژول‌های سخت‌افزاری خاص‌منظوره برای یک کاربرد مشخص طراحی می‌شدند. اما امروزه گرایش به سمت طراحی و پیاده‌سازی ساختارهای قابل بازپیکربندی برای انجام حساب دهدهی افزایش یافته است. این مقاله، با بررسی انتخاب‌های مختلف قابلیت بازپیکربندی در جمع‌کننده‌های دهدهی، با این جریان همراه می‌شود و درخت‌های پیشوندی موازی جدیدی با قابلیت بازپیکربندی پیشنهاد کرده و یک جمع‌کننده‌ی ترکیبی دودویی/دهدهی قابل بازپیکربندی برای ورودی‌های با اندازه‌ی مختلف ارائه می‌کند. تحلیل‌ها نشان می‌دهد که با تحمل یک هزینه‌ی سربار قابل قبول می‌توان دو درخت پیشوندی موازی رایج را ترکیب کرده و یک درخت قابل بازپیکربندی به دست آورد. به علاوه، دو معیار متفاوت برای انتخاب درخت‌های پیشوندی پایه پیشنهاد شده و با یکدیگر مقایسه خواهند شد. نتایج تجربی نشان می‌دهد که افزودن قابلیت بازپیکربندی در طراحی‌های پیشنهاد شده، به حداکثر ٪۵ سربار مساحت منجر خواهد شد.

کلمات کلیدی: حساب دهدهی، درخت پیشوندی موازی، جمع‌کننده دهدهی، سخت‌افزار قابل بازپیکربندی، ریزدانگی.

علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴

طبقه‌بندی و مرور روش‌های خرابی‌یابی توزیعی در شبکه‌های متشکل از گره‌های هوشمند

بهروز پرهامی نان وو سیزین تائو

دانشکده مهندسی برق و کامپیوتر، دانشگاه کالیفرنیا، سانتاباربارا، امریکا

چکیده

رشته‌ی خرابی‌یابی در سطح سیستم از زمان پیدایشش در پنجاه سال پیش گسترش چشمگیری داشته و هنوز هم از تخصص‌های پژوهشی فعال در هر دو زمینه‌ی پردازش موازی و سیستم‌های اعتمادپذیر است. بخش چشمگیری از نتایج سال‌های اخیر در این رشته از سوی پژوهشگران کشورهای چین و تایوان عرضه شده‌اند. در این مقاله رشته‌ی پژوهشی خرابی‌یابی توزیع یا نامتمرکز از راه ارائه یک طبقه‌بندی فراگیر، که هم تمام فعالیت‌های گذشته را در بر می‌گیرد و هم زمینه‌های تازه را برای پیگیری به ما می‌نمایاند، سازماندهی می‌شود. پیشنهاد واژگانی یکدست و روشن برای ویژگی‌ها و روابط مهم یکی از دستاوردهای این طبقه‌بندی است. مقاله برای متخصصان کامپیوتر که قبلا در این رشته کار نکرده‌اند هم قابل استفاده است، چرا که مفاهیم تازه و چالش‌های پژوهشی را از راه تمثیل به زبان ساده ارائه می‌کند.

کلمات کلیدی: خرابی‌یابی قیاسی، سیستم توزیعی، تحمل‌پذیری خرابی، شبکه میان ارتباطی، مدل *MM، پردازش موازی، مدل PMC، خرابی‌یابی در سطح سیستم.

علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴

یک الگوریتم زمان‌بندی بی‌درنگ انرژی- بهینه برای ماشین‌های موازی نامتجانس با قابلیت DVS

محمود قلی‌پور[۱] مهدی کارگهی[۱و۲] هشام فیلی[۱] شهباز یوسفی[۳] هادی روانبخش[۱]

[۱]دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران، تهران، ایران
[۲]پژوهشکده علوم کامپیوتر، پژوهشگاه دانش‌های بنیادی (IPM)، تهران، ایران
[۳]دانشکده انفورماتیک، مهندسی پزشکی، رباتیک و مهندسی سیستم، دانشگاه جنوا، ایتالیا

چکیده

ماشین‌های موازی نامتجانس از واقعی‌ترین مدل‌های سیستم‌های چندپردازنده‌ای غیرهمگن محسوب می‌شوند. در این مدل، زوج وظیفه- ماشین زمان اجرا و انرژی مصرفی مورد نیاز برای اجرای یک وظیفه را مشخص می‌نماید. این مقاله به ارائه یک الگوریتم زمان‌بندی بر روی چنین بستری با پردازنده‌های دارای قابلیت DVS است. یک الگوریتم بهینه با زمان چندجمله‌ای به نام EORTSA ارائه شده است که در ابتدا یک تخصیص انرژی- بهینه وظیفه-به- ماشین و ولتاژ-به-وظیفه را در قالب یک برنامه‌ریزی خطی انجام می‌دهد. از خروجی این مرحله به عنوان آزمون زمان‌بندی‌پذیری وظایف دوره‌ای استفاده می‌گردد. سپس وظایفی که این آزمون را با موفقیت پشت سر بگذارند توسط یک الگوریتم تطابق زمان‌بندی می‌شوند. در این راستا تعداد حدکثر مهاجرت، حداکثر انقطاع وظیفه، و حداکثر تعداد تغییر سطح ولتاژ مورد بحث قرار گرفته و به عنوان تابعی از تعداد ماشین‌ها استخراج می‌گردند. مقایسه با الگوریتم PCG تا ۶۰٪ بهبود در انرژی مصرفی را گزارش می‌نماید.

کلمات کلیدی: مقیاس‌پذیری پویای ولتاژ، بهینه‌سازی انرژی، مهاجرت، سیستم‌های بی‌درنگ، زمان‌بندی، ماشین‌های موازی نامتجانس.

علوم و مهندسی کامپیوتر نشریه علمی پژوهشی انجمن کامپیوتر ایران

مجلد ۱۳	شماره ۲	۱۳۹۴

فهرست چکیده فارسی مقالات

علوم و مهندسی کامپیوتر

نشریه علمی پژوهشی انجمن کامپیوتر ایران

صاحب امتیاز: انجمن کامپیوتر ایران
مدیر مسئول: دکتر جعفر حبیبی
سردبیر: دکتر حمید سربازی‌آزاد

نشانی

تهران، خیابان آزادی، ضلع غربی دانشگاه صنعتی شریف، کوچه شهید ولی‌ا... صادقی، پلاک ۲۶، طبقه ۴، واحد ۱۶، دفتر انجمن کامپیوتر ایران، نشـریه علوم و مهندسی کامپیوتر
تلفن: ۶۶۰۳۲۰۰۰-۶۶۰۸۷۲۲۴
دورنگار: ۶۶۰۲۱۱۴۹
پست الکترونیکی: csi-jcse@ipm.ir
تور جهان گستر: http://www.jcse.ir

لیتوگرافی، چاپ و صحافی: