Energy-Aware Real-time Tasks Processing for FPGA-Based Heterogeneous Cloud

Atanu Majumder, Sangeet Saha, Amlan Chakrabarti, *Sr. Member, IEEE* and Klaus McDonald-Maier *Sr. Member, IEEE*

Abstract—Cloud computing is becoming a popular model of computing. Due to the increasing complexity of the cloud service request, it often exploits heterogeneous architecture. Moreover, some service requests (SRs)/tasks exhibit real-time features, which are required to be handled within a specified duration. Along with the stipulated temporal management, the strategy should also be energy efficient, as energy consumption in cloud computing is challenging. In this paper, we have proposed a strategy, called "Efficient Resource Allocation of Service Request" (ERASER) for energy efficient allocation and scheduling of periodic real-time SRs on cloud platform. The cloud platform is consists of Field Programmable Gate Arrays (FPGAs) as Processing Elements (PEs) along with the General Purpose Processors (GPP). We have further proposed, an SR migration technique to reduce the tasks rejection by serving maximum SRs. Simulation based experimental results demonstrate that the proposed methodology is capable to achieve upto 90% resource utilization with only 26% SR rejection rate over different experimental scenarios. Comparison results with other state-of-the-art techniques reveal that the proposed strategy outperforms the existing technique with 17% reduction in SR rejection rate and 21% reduction in energy consumption. Further, the simulation outcomes have been validated on real FPGA test-bed based on Xilinx Zynq SoC with standard benchmark tasks.

Index Terms—Field Programmable Gate Arrays (FPGAs), service request, real-time scheduling, resource management, energy, heterogeneous cloud.

1 Introduction

Cloud computing nowadays has became a popular processing paradigm for distributed computing devices, which are inter-connected through the public or private networks [1]. A user can choose particular hardware, storage and processing platforms in a cloud, based on the performance requirements through a service request (SR). Such SRs may often exhibit real-time characteristic. Real-time cloud applications can be visible in Internet of Things (IoT) where large scale of sensing and control activities are combined with the real-time data analytics [2]. In another example, real-time cloud service requests are widely used in "intelligent transport systems" [3]. In this scenario, data centers collect data from the road side cameras of fixed objects or obstacles and conduct real time analysis, transport of information is then relayed to the drivers. In such scenario, each SR has a predefined life-time (Lt), within which the request has to be completed. These SRs may be periodic/aperiodic in nature [4]. Periodic SRs appear periodically, after a specific time interval. On the other hand, aperiodic SRs appear at an arbitrary instance of time. It is the responsibility of the cloud service architecture to assign the request to an appropriate processing element, so that the service request can be completed within the specified life-time [5].

Due to the wide variety of performance demands from the users, cloud service providers often need to be very flexible in terms of service delivery. Hence, the primary focus of recent cloud computing architecture is to exploit the heterogeneous processing architecture in order to achieve higher computing performances [6], [7]. In recent past, existing scheduling strategies for real-time service requests on cloud mainly focused on the optimization of computation time to meet the deadlines and to enhance the throughput [8]. However, with the increase in complexity level of the cloud infrastructure, such cloud computing environment consumes high energy [9]. Thus, the recent heterogeneous cloud architectures are employing FPGAs along with CPUs and GPUs to overcome the existing limitations [10]. Acceleration in execution of a service request on such heterogeneous architecture is a challenging issue in cloud. The performance and efficiency of the homogeneous CPU based cloud is insufficient to match the requirements of modern cloud servers [11]. Hence, cloud infrastructure with reconfigurable hardware, FPGA, is an emerging choice.

In cloud computing, users initiated SRs are physically mapped into the processing elements via a virtualization technique [12]. Virtualization helps to partition the physical resources into multiple virtual machines (VMs), where each VM works independently. At the core level, a "server" contains the physical resources/machines and the service requests are allocated to an appropriate server through VMs. Recently, real-time energy efficient scheduling strategies are developed for heterogeneous cloud platforms [13]. Due to the variety of processing requirements, best task-to-resource

• *Atanu Majumder and Amlan Chakrabarti are associated with the Department of AK Choudhury School of Information Technology, University of Calcutta, India. Sangeet Saha and Klaus McDonald-Maier are with School of Computer Science & Electronic Engineering, University of Essex, UK.*

E-mail: Atanu Majumder: a.majumder007@gmail.com, Sangeet Saha: sangeet.saha@essex.ac.uk, Amlan Chakrabarti: acakcs@caluniv.ac.in, Klaus McDonald-Maier: kdm@essex.ac.uk

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

mapping at runtime ensures the high performance gain in terms of throughput and helps to optimize the energy consumption [14]. To reduce the energy consumption of the servers, such strategy may allow the migration of the VMs from one server to another [15] server. Migrations not only help to reduce the number of SR's rejection by proper load balancing but also manage the processing resources efficiently. Thus, migrations enhance the overall throughput. In a cloud platform, real-time schedulers can be incorporated by devising a real-time hypervisor, for example, RT-Xen [16].

In this paper, we have proposed a novel real-time tasks allocation and scheduling strategy for FPGA-based heterogeneous cloud platform. Specifically, we answer the following question: *Given a set of real-time periodic SRs/tasks and heterogeneous processing elements, how do we ensure that those SRs will be energy-efficiently scheduled on physical platform by choosing the appropriate VMs, while satisfying the life-time of each SR and resource constraints.* In order to achieve the aforementioned objectives, we have coined the idea of "time-partitioned" task scheduling where we have partitioned the execution time into multiple time windows based on the life-time of SRs. The main technical contributions of this paper are:

- We have proposed a "time-partitioned" based scheduling strategy, "Efficient Resource Allocation of Service Request" (ERASER), which efficiently executes periodic real-time cloud SRs within the given deadline, with minimum energy consumption. (Discussed in "Section 3").
- ERASER employs an Integer Linear Programming (ILP) based technique for SRs allocation on an FPGA and CPU based heterogeneous cloud platform. In order to increase the throughput, ERASER also exploits the migration functionality of VMs between the servers. (Discussed in "Section 3.4" and "Section 4").
- Simulation based experiments reveal the efficiency of the proposed approach. ERASER can achieve up to 90% resource utilization with only 26% SR rejection rate over different experimental scenarios. Comparison results reveal that ERASER consumes 21% less energy and reduces the SR rejection rate by 17% than the existing techniques. Further, the simulation outcomes have been validated on a real FPGA platform, Xilinx Zynq SoC, with benchmark tasks (Discussed in "Section 5" and "Section 6").

The organization of the paper is as follows: the related work is described in Section 2. Adopted system model and the proposed scheduling strategy are described in Section 3. The Migration scheme is proposed and illustrated with an example in Section 4. Experiments and results are discussed in Section 5. Physical implementation of the ERASER on ZYNQ FPGA SoC is demonstrated in Section 6. Finally, Section 7 concludes the paper with a discussion on the future work.

2 Related Work

Real-time based SR execution has drawn considerable research interest in recent past. Handling of real-time SRs in cloud computing either i. employs homogeneous PEs i.e. each PE has similar processing characteristics like speed, power rating etc or, ii. employs heterogeneous PEs i.e. processing elements may have different computation capabilities. Majority of the homogeneous PE based cloud scheduling strategies focus on energy and deadline management. For example, Tian et al. [18] proposed an energy efficient scheduling mechanism for homogeneous cloud data center. The authors argued that the Longest Load Interval First (LLIF) strategy outperforms popular First-Fit Decreasing (FFD) strategy. Results were verified on Amazon EC2 platform by intensive simulations, using trace-driven and synthetically generated data sets. Similarly, an attempt for energy efficient real-time cloud SR scheduling based on time partitioning algorithm (TPA) has been proposed by Hu et al. [25]. Here, based on the task dependency and real-time requirements, SR mapping and resource allocation are re-adjusted (start time adjustment (STA)) to make a trade-off between energy consumption and makespan time.

Dynamic server provisioning technique is widely used for reducing energy consumption by turning off the idle servers of the cloud data centers. However, this strategy can lead to server queues instability. To overcome this situation, Safavi et al. [19], proposed an integer programming (IP) based algorithm, which makes a trade-off in between the server queue stability (S) and energy (E) optimization using a proper load (L) balancing technique (known as SEL). But, in some typical cases, the dynamic load (overload) of the cloud data center can affect the servers queue stability. Thus, energy consumption of the cloud data servers can be affected, which may lead to poor performances of the system. Xu et al. [21] presented a brownout technique to handle the overload situation. The technique ensures the minimization of energy consumption by deactivating the unnecessary applications. Three types of brownout-based scheduling policies are described known as Lowest Utilization Container First (LUCF), Minimum Number of Components First Policy (MNCF) and Random Selection Container Policy (RSC), where LUCF exhibits the better performance than others. Nowadays, cloudlets[1] techniques are deployed to manage the cloud based infrastructure and provide a powerful computing resource platform to the mobile devices with lower latency. Gai et al. [22] proposed an advanced energy-aware cloudlet-based mobile cloud computing model (DECM) for achieving green computing by avoiding the energy wastage in an unstable networking environment.

In order to address the diverse requirements of users, heterogeneous computation resources are introduced. Recently, heterogeneous cloud incorporates Graphics Processing Unit (GPU) and Field programmable Gate Array (FPGA) as processing elements. A few existing scheduling strategies focuses either on load balancing to distribute the computing workloads uniformly across various processing units or on makespan minimization, by considering the power consumption of the heterogeneous cloud servers. A power-aware task scheduling (PATS) strategy for heterogeneous CPU based cloud platform has been proposed in [24]. The

1. Small-scale cloud datacenters, which provide cloud computing services to mobile devices.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

TABLE 1: Comparison of focal points of existing works with ERASER

Approach	Scheduling mechanism		Scheduling objective		Processing elements (PEs)		Evaluation platform	
	Workload Balancing	Migration incorporated	Energy Minimization	Success ratio	Homogeneous PEs	Heterogeneous PEs	Simulation setup	Real testbed
Proposed	✓	✓	✓	✓	×	✓	✓	✓
Zhou et al. [17]	×	×	×	✓	✓	×	✓	×
Tian et al. [18]	×	✓	✓	✓	×	✓	✓	×
Safavi et al. [19]	✓	×	✓	✓	×	✓	✓	×
Liu et al. [20]	×	×	✓	×	×	✓	✓	×
Xu et al. [21]	✓	×	✓	✓	✓	×	×	✓
Gai et al. [22]	×	✓	✓	×	✓	×	✓	×
Auluck et al. [23]	×	×	✓	✓	×	✓	✓	×
Zhao et al. [24]	✓	×	✓	×	×	✓	✓	×
Hu et al. [25]	×	✓	✓	✓	×	✓	✓	×

main objective of the work remains to minimize the power consumption. The proposed strategy considered the types of PE and makespan time as the main scheduling parameters and it was also capable to predict the busy and idle power consumption of the physical machines (PMs) by analyzing the state of VMs in real-time. Integer Linear Programming (ILP) based real-time task mapping and scheduling are discussed in [23]. Here, the authors employed a hybrid "embedded-fog-cloud" based heterogeneous architecture. Fog nodes were used for executing tasks that have early deadlines. On the other hand, soft real-time tasks were executed on cloud. Mobile Embedded Systems (MES) with Cyber-Enabled Applications (CEA) has become an emerging technology in cloud domain for mobile computing [26]. A novel approach "Energy-Aware Heterogeneous Cloud Management (EA-HCM)" is proposed by the authors to reduce the energy consumption for such heterogeneous MES with higher performances.

As the target applications require fast responses upon arrival, low latency is one of the most important features for real-time cloud. Acceleration of task processing in cloud plays an important role for faster response. FPGAs provide the support of accelerator for modern high performance cloud computing platforms. FPGA-based heterogeneous architecture has grabbed an considerable attention on real-time cloud computing. With the increasing complexity of cloud architecture, energy consumption also increases. Hence, the authors in [27], proposed an energy-aware scheduling technique for CPU-FPGA based heterogeneous cloud platform. Proposed strategy first maps the VM applications to the custom hardware accelerators through a fine-grained hardware aware scheduler, which reduces the power consumption by optimal usages of FPGAs. While reducing the power consumption, the algorithm ensures the fulfillment of targeted performance within their assigned deadlines by CPU frequency scaling and dynamic VM allocation. Similarly, the authors in [20], implemented a real-time energy optimization strategy to minimize the energy cost while satisfying the QoS and time constraint of DVFS capable CPU/GPU/FPGA heterogeneous cloud platform.

However, majority of the existing works have focused on real-time energy efficient scheduling for general purpose processors (GPPs) but energy-aware real-time scheduling for FPGAs is still in its infancy. Thus, in this work, We propose a "time-partitioned" based energy-aware scheduling strategy ERASER, to execute periodic real-time SRs on FPGA-based heterogeneous cloud. Table 1 categorise and compares the features of the existing work with the ERASER.

3 System Model & ERASER

The adopted system model of this paper resembles a modern cloud architecture [28], [29]. In [29], the authors have shown how the FPGAs can be utilised on cloud computing architecture. A VM can exclusively access the FPGA devices through a series of OpenStack based virtualization techniques. In such architecture, a particular SR is physically mapped into a PE via VM. VMs are responsible for executing SRs on physical PEs of a server, by providing an abstraction to the users. On the arrival of τ number of SRs, the ERASER attempts to allocate them based on the energy consumption and temporal parameters. Within a server, a particular type of PE will be responsible to carry out the execution of an SR. Hypervisor will assume the responsibility for efficient resource sharing and integrates the VMs into the scheduling framework. Without loss of generality, we are assuming that,

- Each VM is capable of running on a particular server and the initial distributions of the possible upcoming SRs are known at design time.
- A VM cannot simultaneously execute on two distinct PEs of a server at the same instant of time.

An energy efficient SRs allocation scheme, (ERASER), is defined to allocate the VMs systemically on two different types of PEs. Typically, FPGAs and CPUs have different energy and performance characteristics for the same service request. Hence, ERASER aggregates all the arriving requests and schedules these SRs in such a way that each SR completes its execution within its life-time with minimal energy consumption. A pictorial representation of our system model is shown in Figure 1.

3.1 ERASER Strategy

In this section, we will discuss about the proposed ERASER strategy. ERASER will attempt to satisfy the timing requirements of all SRs by completing them within their respective life-times. Along with this, ERASER will also ensure that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

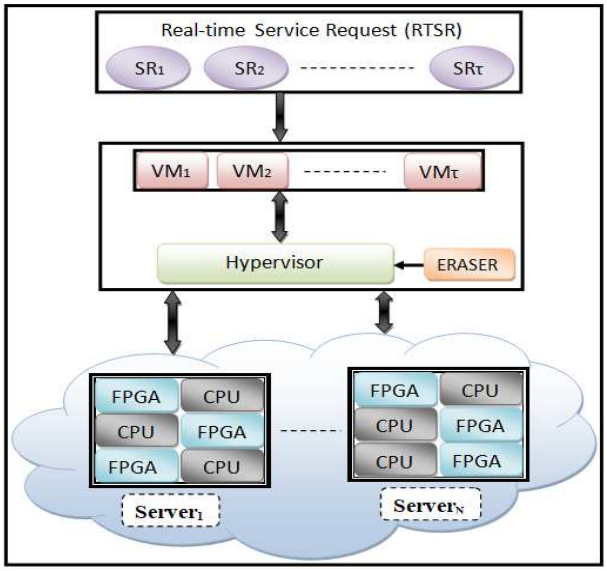


Fig. 1: Heterogeneous Cloud Service Architecture

TABLE 2: Symbols and their significance

Symbols	Explanations
$\mathcal{M}$	Number of PEs per server
$\mathcal{N}$	Number of servers
SD_i	Service Duration of the i^{th} SR
Lt_i	Life-time of the i^{th} SR
TG_j	Time-gap
WL_i^j	Workload of the i^{th} SR for j^{th} time-gap
TC_j	Total system processing capacity for j^{th} time-gap
τ	Number of periodic real-time SR
$PWF_{n,max}$	Maximum power consumption by FPGAs
$PWC_{m,max}$	Maximum power consumption by CPU cores
$PW_{k,max}$	Maximum power consumption limit by k^{th} server
$PW_{k,idle}$	Minimum power consumption at server idle mode
HPF_{k1}	Power required for hosting a VM by FPGA
HPC_{k2}	Power required for hosting a VM by CPU
UPF_i	Power/Unit of workload for serving an SR by FPGA
UPC_i	Power/unit of workload for serving an SR by CPU
FXP_i	Power consumption for an SR execution on FPGA
CXP_i	Power consumption for an SR execution on CPU
$PW_{i,execution}$	Power required to serve an SR for a time-gap
$PW_{i,migrate}$	Power consumption for an SR migration
$PW_{kl,DT}$	Data transferring power from server k to l
$MG_{T,pow}$	Total energy required to migrate the SRs
DT_i	Data transferring delay

the entire scheduling consumes minimum energy. Scheduling and mapping of SRs to appropriate PEs are obtained through an ILP based technique. Table 2 represents all the mathematical notations and their significance used in this paper.

3.2 Temporal Management

ERASER will attempt to execute the SRs by employing a "time-partitioned" based approach [30]. Our scheduling approach will partition the scheduling duration (based on the SRs life-time) into some time-gaps (TGs). It has to be ensured that each SR completes the assigned workload (portion of work to be executed) within that time-gap. SRs will be allocated to appropriate PEs using Higher Workload First (HWF) policy. Each SR has different service duration and life-time hence, workload will be different for a particular time-gap. According to the HWF policy, SR with the highest amount of workload will be allocated first in the server for a time-gap. Let us assume, service duration or computation time of i^{th} SR is represented as SD_i and its life-time by Lt_i. Let us also assume that there are $\mathcal{N}$ number of servers and each server has $\mathcal{M} = (M_1 + M_2)$ number of PEs, working in parallel. Here, M_1 represents number of FPGAs and M_2 represents number of CPU cores.

In each time-gap, each SR will be assigned a certain amount of workload to execute. The workload WL_i^j of i^{th} SR in j^{th} time-gap is defined as:

$$WL_i^j = TG_j \times \frac{SD_i}{Lt_i} \tag{1}$$

We termed $\frac{SD_i}{Lt_i}$ as the individual weight of an i^{th} SR. The time-gap is defined by the intermediate difference between two consecutive SR's life-times(Lt). TG_j is the j^{th} time-gap, which denotes the difference between i^{th} and $(i-1)^{th}$ SR's life-times. The total system level capacity TC_j for the time-gap TG_j among all servers can be defined as:

$$TC_j = TG_j \times \mathcal{M} \times \mathcal{N} \tag{2}$$

All SRs will be able to complete their workload within the particular time-gap, if the following equation 3 is satisfied.

$$\sum_{i=1}^{\tau} WL_i^j \leq TC_j \tag{3}$$

3.2.1 Time-partitioned based approach versus Greedy approach

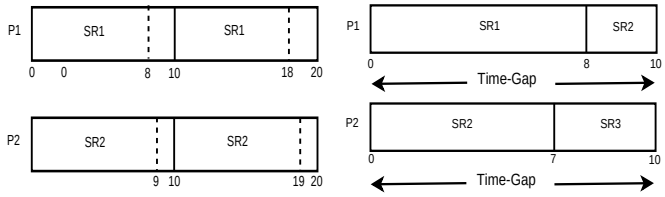
The "time-partitioned" approach actually partitions the time into small slices. Each SR is assigned with a "workload" for each time-gap. Greedy scheduling techniques are efficient but not suitable for many complex cases. SRs scheduling on multi-processor systems within a specific life-time has a higher scheduling overhead and can become infeasible. We have shown an example where greedy scheduling approaches like Earliest Deadline First (EDF) and Least Laxity First (LLF) fail to schedule SRs within its life-time. We have considered an SR set whose details are as follows:

SR(Service duration, Life-time)=$SR_1(8, 10)$, $SR_2(9, 10)$, $SR_3(6, 20)$

We have two processors and need to schedule the given SRs. The LLF strategy is not sufficient to handle this type of scenario. SR_1 and SR_2 have laxity of 1, 2, respectively. Hence, service request SR_1 and SR_2 have the priority to execute first. In case of EDF, SR_1 and SR_2 are prioritized to run for completion as it has earliest life-time compare to SR_3. SR_1 and will finish at time t = 8. SR_2 will finish at t = 9. After completing SR_1 and SR_2, both the processors will remain idle until t = 10 and SR_3 still remains to be scheduled in a processor. After t = 10, SR_1 and SR_2 will start its execution as they are periodic in nature and have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

earlier life-time and lower laxity. Thus, it is not possible for the greedy scheduler to start SR_3 before t = 8. Hence, SR_3 cannot be completed within the life-time and this scenario is pictorially shown in Figure 2(a).



(a) Greedy Scheduling Failure (b) Time-Partitioned Solution

Fig. 2: Advantages of Time-Partitioned Approach

This limitation of the greedy strategy can be overcome by the "time partitioned" strategy as shown in Figure 2(b). In the proposed technique, time is partitioned into time-gaps. Here we have illustrated for only one time-gap i.e. is TG = 10. Each SR's workload needs to be completed within the time-gap. Now for each service request, SR_1, SR_2 and SR_3 the workload[2] will be 8, 9 and 3, respectively. SR_1 executes on processor P_1 and finishes by t = 8. Thus, P_1 has a slack time of 2. SR_2 executes on P_1 for 2 time units and the rest of the workload (9-2) = 7 will be completed in P_2. SR_3 has a workload of 3, which can be scheduled on P_2, as it has 3 units of remaining slack. Hence, it can be observed that all the SRs have completed within their respective lifetimes.

3.3 Power Management (PMG)

The VMs will be allocated to the PEs within a time-gap according to the allocation strategy discussed in next section. Each SR has different requirements of power for execution on a server. Servers need a constant power to operate in the idle state.

Let us consider that within a server, $PWF_{n,max}$ and $PWC_{m,max}$ represent the maximum power consumption for M_1 number of FPGAs and M_2 number of CPU cores, respectively. $PW_{k,max}$ is the maximum limit of power consumption by the k^{th} server, which is constant for a particular time-gap, shown in equation 4.

$$PW_{k,max} = \sum_{n=1}^{M_1} PWF_{n,max} + \sum_{m=1}^{M_2} PWC_{m,max} \quad (4)$$

Let us assume, $PW_{k,idle}$ is the minimum power consumption, required for the k^{th} server in the idle mode. HPF_{K1} and HPC_{K2} are the power requirements for hosting VMs by FPGA and CPU processing cores, respectively. It has to be noted that when no SR is mapped on these VMs, a certain amount of power is still required for accommodating idle VMs. The required power by the k^{th} server before serving any request is measured as:

$$PW_{k,const} = PW_{k,idle} + \sum_{i=1}^{K_1} HPF_i + \sum_{j=1}^{K_2} HPC_j \quad (5)$$

where $PW_{k,const} \neq 0$ and K_1 and K_2 denotes the number of VMs.

2. ($\frac{8}{10} \times 10$) = 8 is the workload of SR_1 for time slice TG = 10

UPF_i and UPC_i are the unit amount of power required to serve the i^{th} SR by the FPGA and CPU, respectively. Unit power varies with the size or workload of each requested SR. Equation 6 exhibits the energy consumption by an FPGA to execute a requested SR for a particular time-gap as:

$$FXP_i = \{WL_i^j\} \times UPF_i \quad (6)$$

Similarly, equation 7 reflects the energy consumption by a CPU to execute the same SR as:

$$CXP_i = \{WL_i^j\} \times UPC_i \quad (7)$$

The energy consumption for executing SRs for a particular time-gap can be defined as:

$$\sum_{i=1}^{\tau} PW_{i,execution} = \sum_{i=1}^{\tau} (FXP_i \times XF_i + CXP_i \times XC_i) \quad (8)$$

where, XF_i and XC_i are the binary variables and indicate the assignment of an SR into an FPGA or a CPU core and denoted as follow:

$$XF_i = \left\{ \begin{array}{ll} 1, & \text{when } i^{th} \text{ SR is assigned to any of the FPGA} \\ 0, & \text{otherwise} \end{array} \right\}$$

$$XC_i = \left\{ \begin{array}{ll} 1, & \text{when } i^{th} \text{ SR is assigned to any of the CPU cores} \\ 0, & \text{otherwise} \end{array} \right\}$$

An SR can not run simultaneously on multiple PEs. Hence, if $XF_i = 1$ then $XC_i = 0$, or vice-versa.

3.4 ILP-based SR Mapping and Scheduling

Let us assume that at the time instance t, τ be the number of real-time SRs arrive for possible allotment on the cloud servers. There are M_1 number of FPGAs and M_2 number of CPU cores[3] available on each of $\mathcal{N}$ number of servers. Hence, at a time instance t, we can accommodate at most $(M_1 + M_2)$ number of SRs for parallel execution on a server.

In each time-gap TG, our objective will remain to minimize the overall energy consumption of the cloud servers by allocating the SRs to the appropriate PEs.

Objective Function:

$$minimize(\sum_{i=1}^{\tau} PW_{i,execution}) \quad (9)$$

Constraints:

1. **Parallelism Constraint**:

This constraint enforces that parallel execution of an SR on two distinct PEs (FPGA & CPU) at a same time is restricted. An SR will be executed on a single PE i.e. .

$$XF_i + XC_i = 1 \quad (10)$$

2. **Real-time Constraint**

All SRs will be able to complete its workload within each particular time-gap, if the following equation is fulfilled.

$$\sum_{i=1}^{\tau} WL_i^j \leq TC_j \qquad \forall i \in \tau \quad (11)$$

3. **Power Constraint**:

3. "core" refers the individual programming units for each type of PE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

Equation 12 refers that at each time-gap power required for hosting VMs and serving SRs cannot exceed the maximum power limit.

$$\sum_{i=1}^{\tau} PW_{i,execution} + \sum_{k=1}^{\mathcal{N}} PW_{k,const} \leq \sum_{k=1}^{\mathcal{N}} PW_{k,max} \quad (12)$$

3.5 Circular Linked-list Framework (CLF)

ERASER stores the scheduling information (obtained from Algorithm 1) with the corresponding PEs for each time-gap (upto the Hyper-period) in a Circular Linked-list Framework (CLF). A node of the linked-list will denote a time-gap. The r^{th} node is denoted as GTG_r.

The information is stored corresponding to the r^{th} node GTG_r of the circular linked list is as follows: 1) Pointer PTR_r to the Linked-List of Schedulers ($LL-SCH$). $LL-SCH$ is composed for $\mathcal{N}$ number of servers. Each server of $LL-SCH$ contains the following information:

- Minimum power needed for execution: Min-PW.
- SR to PE mapping: MAP
- Link to next node: NN.

2) Pointer to next node of GTG_r: $LINK_r$.

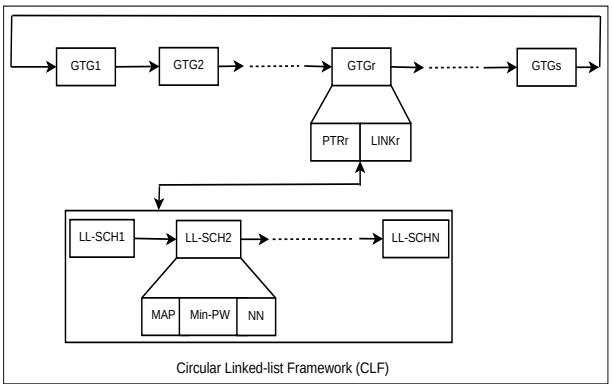


Fig. 3: ERASER working module

A pictorial representation of the linked-list structure is shown in Figure 3.

The pseudo-code for the ERASER scheduling strategy is shown in Algorithm 1 and 2. The ERASER has two parts. First part ensures the temporal requirements of the scheduling and the second part, power management (PMG) function maps the SRs to the PEs, energy efficiently. Scheduling framework for a hyper-period $\mathcal{H}$ is generated by considering the both parts of the algorithm. The ERASER scheduling strategy is depicted with an example in the next section.

3.6 ERASER : Illustration with an Example

Let us assume 12 periodic SRs $\{SR_1, SR_2, ..., SR_{12}\}$, arrive for possible allotment over $\mathcal{N} = 2$ number of servers. Further, we assume that each server contains one FPGA and one CPU core, respectively. Hence, $\mathcal{M} = 2$ for each server. SRs can run in parallel by utilizing the PEs. Each SR has a different service duration for different types of PEs and each SR should be completed within its life-time (Lt). The characteristic (service duration, life-time and unit power consumption) of each SR on these distinct types of PEs are shown in Table 3.

Algorithm 1: ERASER Scheduling Strategy

Input: τ number of SRs, $\mathcal{N}$ number of servers each containing $\mathcal{M}$ number of PEs, SR parameters SD_i, Lt_i, Hyper-period $\mathcal{H}$
Output: Circular Linked-list Framework (CLF)

```
Initialize Node = 1;
Let us assume that a H is composed of s number of time-gaps;
Calculate the Time-Gap (TG_j) up-to the hyper-period H;
for (Each TG_j ∈ H) do
    for (Each SR ∈ τ) do
        Calculate the workload WL_i^j using the equation 1;
    Calculate the TC_j by using the equation 2;
if (Σ_{i=1}^{τ} WL_i^j ≤ TC_j) then
    Execute the SRs using HWF policy;
    if (List node GTG_r NOT already created) then
        Create a new CLF node GTG_r;
        Call the function PMG();
        Go to the next node;
else
    Discard the SR set τ;
```

Algorithm 2: Function PMG()

Input: Unit power (UPF_i & UPC_i) required by each SR, Idle power $PW_{k,const}$, Maximum power consumption by the Server $PW_{k,max}$
Output: SR to PE mapping : MAP

```
Calculate the energy consumption for executing SRs PW_{i,execution} using the equation 8;
if (M > 0) && (Required PEs are available) then
    Solve the ILP ;
if (List node LL − SCH_i NOT already created) then
    Create a new GTG_r node LL − SCH_i;
    Insert Values of MAP and Min-PW into LL − SCH_i for each SR;
    Return SR to PE mapping (MAP) and Min-PW;
```

We assume that a server has a maximum power capacity of $PW_{k,max}$ = 150 watts [4]. In idle state, the power consumption of a server is assumed as 75 watts. We further assume that to host a VM on FPGA and CPU, power consumption is 3 watts and 2 watts, respectively. These are the hypothetical values used for this example only. However, the experiments have conducted with the actual values. Now, According to the Equation 5, $PW_{k,const}$ will be 80 watts. For two operational servers $PW_{k,const}$ is 160 watts.

ERASER will allocate the service requests to the PEs and executes the portion of workload for each time-gap. The length of the first time-gap, TG_1 is 45. Using equation 1, we obtain the workload (with min service duration) of each

4. In [31], authors experimentally showed such typical max power consumption

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

SR_i (WL_i) for each time-gap as shown in Table 4. The overall system capacity (TC_1) for the first time-gap (TG_1) using Equation 2 becomes (45 $\times$ 4) = 180 units. We can observe the feasibility criteria as described in Equation 3, is satisfied. Hence, all the SRs complete their respective workloads within the first time-gap and similarly, it can be observed that this condition will be true for subsequent time-gaps. Within a time-gap, in order to map an SR to an appropriate PE, the ILP based technique will be applied.

Energy consumption of SRs on FPGA and CPU according to the Equations 6 and 7, are shown in the Table 5. The ILP is formulated and solved using the IBM CPLEX tool [32] with the parameters given in Table 5. As an output, we obtain the overall energy consumption as ($\sum_{i=1}^{12} PW_{i,execution}$) = 112.3 units for the first time-gap, TG_1. The corresponding allocation of each SR for each PE (Whether FPGA or CPU) is also obtained from the ILP and shown in Figure 4.

TABLE 3: SRs PARAMETERS

SR No.	Lt	SD_F	SD_C	UPF	UPC
SR_1	45	13	14	0.6	0.7
SR_2	60	18	22	0.7	0.6
SR_3	90	26	28	0.7	0.9
SR_4	120	37	37	0.8	0.6
SR_5	90	21	25	0.8	0.6
SR_6	120	39	37	0.7	0.9
SR_7	180	60	77	1.0	0.8
SR_8	90	27	29	0.6	0.8
SR_9	45	13	16	0.6	0.5
SR_{10}	60	19	13	0.5	0.7
SR_{11}	60	19	21	0.8	0.8
SR_{12}	90	26	28	0.7	0.9
SR_{13}	45	8	10	0.9	0.8
SR_{14}	60	15	14	0.8	0.7

SD_F;SD_C: Service Duration units on FPGA & CPU, respectively
UPF;UPC: Power units of FPGA & CPU, respectively

TABLE 4: Workload Of SRs

SR No.	WL^1	WL^2	WL^3	WL^4	WL^5
SR_1	13	4	8	8	17
SR_2	16	5	11	11	22
SR_3	13	4	8	8	17
SR_4	13	4	9	9	18
SR_5	12	4	8	8	16
SR_6	14	4	9	9	19
SR_7	15	5	10	10	20
SR_8	14	4	9	9	19
SR_9	13	4	8	8	17
SR_{10}	9	3	6	6	13
SR_{11}	14	4	9	9	19
SR_{12}	14	4	9	9	18
Sum	160	49	104	104	215
SR_{13}	7	3	5	5	11
SR_{14}	9	3	6	6	14

WL^j: Workload of SRs in j^{th} time-gap

4 ERASER WITH MIGRATION SCHEME

In case of dynamically arriving tasks, the SR migration may be useful when the required PE of a server is not available to fulfil the requested SR's execution. Migration technique enhances the utilization of the resources and helps to maximize the throughput.

TABLE 5: Power Consumption Of SRs

SR No.	PW_R^1	PW_C^1	PW_R^2	PW_C^2	PW_R^3	PW_C^3	PW_R^5	PW_C^5
SR_1	7.8	9.8	2.4	3.6	5.2	6.5	10.4	13.0
SR_2	10.5	9.9	3.5	3.0	7.0	6.6	14.0	13.2
SR_3	9.1	12.6	3.0	4.2	6.0	8.4	12.1	16.8
SR_4	11.1	8.3	3.7	2.7	7.4	5.5	14.8	11.1
SR_5	8.4	7.5	2.8	2.7	5.6	5.0	11.2	10.0
SR_6	10.2	12.4	3.4	4.1	6.8	8.3	13.6	16.6
SR_7	15.0	15.4	5.0	5.1	10.0	10.2	20.0	20.5
SR_8	10.8	8.7	3.6	2.9	7.2	5.8	14.4	11.6
SR_9	7.8	8.0	2.6	2.6	5.2	5.3	10.4	10.6
SR_{10}	7.1	6.8	2.3	2.2	4.7	4.5	9.5	9.1
SR_{11}	11.4	12.6	3.8	4.2	7.6	8.4	15.2	16.8
SR_{12}	11.7	9.8	3.9	3.2	7.8	6.5	15.6	13.0
SR_{13}	6.3	6.4	2.7	3.2	4.5	4.8	9.9	10.4
SR_{14}	7.2	5.6	2.4	2.1	5.6	4.9	12.0	9.8

PW_R^j; PW_C^j: Power consumption units of SRs on FPGA and CPU, respectively for j^{th} time-gap

4.0.1 Effectiveness of the Migration Scheme

To illustrate the benefits of the migration scheme, let us consider an example. We have considered the same tasks set as discussed in the previous example however, we have also considered two additional periodic SRs i.e. SR_{13} and SR_{14}. Let us assume that both the tasks have arrived at t = 10. Detailed information of the SRs are provided in Table 3. The length of the first time-gap is 45 and SRs arrived at t = 10. Hence, the workload and energy consumption of the newly arrived SRs are calculated for both the PEs and provided in Table 4 and Table 5, respectively. Now according to the HWF policy the scheduling of SR_{14} and SR_{13} will be performed for TG_1. SR_{14} has to complete 9 units of workload and according to the Algorithm 1 it can execute on the second server within the remaining slack time. But in case of SR_{13}, it cannot be allocated to any of the servers, as required remaining slack is not sufficient. In such scenario, migration could be a promising solution. We have discussed the detailed migration strategy in the next section.

4.1 ILP Based Migration

A migration should be performed from server k (source) to server l (destination). In order to migrate a task, we need to measure three parameters: (i) The portion of WL_i^j, executed at the current server. (ii) Find the server(s) having the maximum slack times. (iii) The expected completion time of the remaining workload at the destination server, including the delay for transferring the data.

Let us assume that an SR migration is initiated after completing the Θ unit of its allocated workload. Hence, for the destination server l, the new workload will be $WL_i^j - \Theta$. This denotes the remaining portion of the SR's workload to be executed and $TG_j - \Theta$ is the updated life-time for the SR on the l^{th} server. Slack time of a server for a particular time-gap is calculated as $(TG_j \times \mathcal{M}) - \sum_{i=1}^{\tau} WL_i^j$. Transferring the SR from one server, k to another server l, consumes DT_i amount of delay. Hence, the power required for migrating an SR will be:

$$PW_{i,migrate} = PW_{kl,DT} + PW_{i,execution} \quad (13)$$

where, $PW_{kl,DT}$ represents the power consumption for the SR to transfer the data from k^{th} server to l^{th} server.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

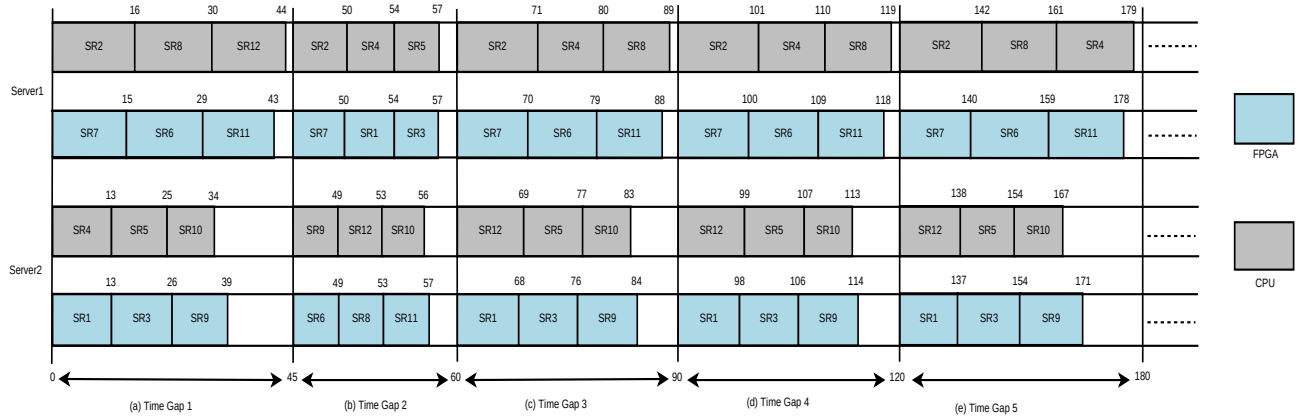


Fig. 4: Energy efficient SRs mapping to PEs by ERASER

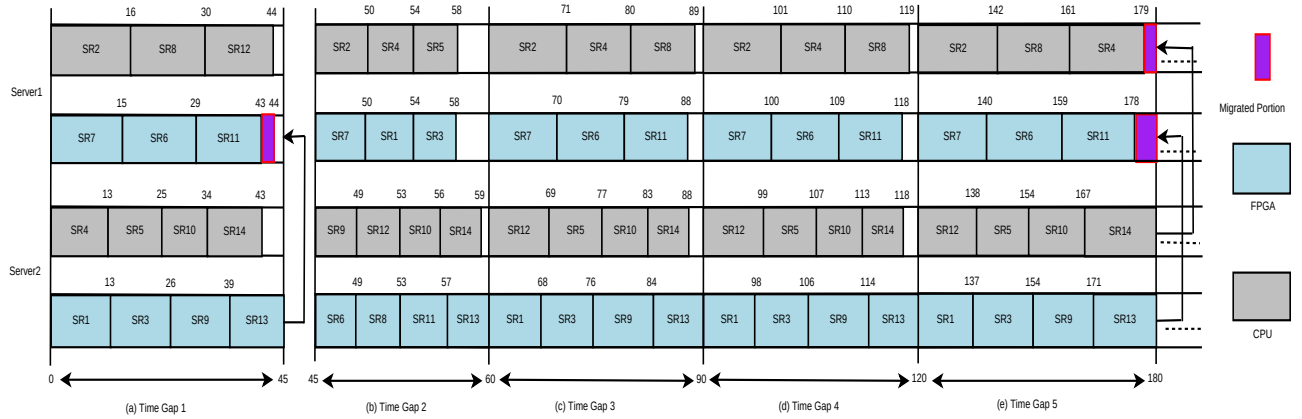


Fig. 5: ERASER with migration

The required power due to SRs migrations can be defined as:

$$MG_{T,pow} = \sum_{k=1}^{\mathcal{N}} \sum_{i=1}^{\tau} (PW_{i,migrate} \times Z_{kli}) \tag{14}$$

where, Z_{kli} is a trivalent variable expressing the migration of i^{th} SR from k^{th} server to l^{th} server.

$$Z_{kli} = \left\{ \begin{array}{l} 1, \text{ when } i^{th} \text{ SR migrated from } k^{th} \text{ to } l^{th} \text{ server.} \\ 0, \text{ otherwise} \end{array} \right\}$$

4.1.1 ILP Formulation

The objective is to optimize the power consumption due to migration by conducting minimum number of migrations of SRs.

Objective Function:

$$minimize(MG_{T,pow}) \tag{15}$$

Constraints:

1. **Migration Constraint**: When migrating an SR, we need to ensure that the SR is only migrated from the k^{th} server to the l^{th} server. Thus, it needs to be ensured that the migration should be between CPU-to-CPU or FPGA-to-FPGA. The equation which enforces the constraint can be denoted as follows:

$$Z_{kli} + Z_{kqi} \leq 1 \tag{16}$$

2. **Power Constraint**: Let us assume k^{th} server has maximum power consumption limit of $PW_{k,Max}$ and servers should not exceeding their power limits while migrating the SRs between the servers, which can be enforced as:

$$\sum_{i=1}^{\tau} PW_{i,execution} + \sum_{k=1}^{\mathcal{N}} PW_{k,const} + MG_{Tpow} \leq \sum_{k=1}^{\mathcal{N}} PW_{k,max} \tag{17}$$

ERASER with migration capability is presented in the Algorithm 3. Migration process comes into play only when the required PEs of a server are not capable to ensure the completion of dynamic SRs executions within their life-times. According to the example described above, SR_{13} has workload of 7 units, but the second server does not have enough slack to accommodate it. Hence, when a migration process is initiated, the portion of the workload which has already been completed, should be calculated in the source server and similarly, the remaining workload needs to be completed in the destination server. SR_{13} completes 6 units of workload in second server and 1 unit of workload needs to be migrated to another server. Migration of an SR should be carried out to the server which has the maximum remaining slack time. It needs to be ensured that the remaining portion of the workload (considering the transfer delay) of the migrated SR should be completed within the life-time. In some typical scenario, frequency scale up might be required to satisfy the life-time by speeding up the execution. PEs operate at frequency (default) f and clock tuning factor β

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

is set to 1 by default. We can tune the clock to regulate the frequency as per requirement. After solving the migration ILP scheduling outcome of the example, is shown in the Figure 5.

Algorithm 3: ERASER with Migration

Input: Data transferring delay DT, Workload WL_i^j, Number of servers $\mathcal{N}$, Completed workload Θ units, Operational frequency f
Output: Migrated SRs mapping to PEs with minimum power

```
Default operational frequency of PE = f;
Clock tuning factor denoted by β, where β range varies from 1,2,...,R;
Initialize β = 1;
for (Each Migrated SR ∈ τ) do
    Calculate the remaining execution requirement workload WL_i^j of SR;
    Calculate the slack time of each server;
    Calculate the remaining time-gap TG_j for destination server to complete the execution;
    if (TG_j ≥ WL_i^j + DT_i) then
        SR migrated from server k to server l;
        Solve the migration ILP;
    else
        if (β < R) then
            Increment β value by 1;
            WL_i^j = 1/(β×f) × WL_i^j;
            Go to the step 7;
        else
            Insufficient time for migration;
```

5 Experiments and Results

We have evaluated the proposed ERASER through software simulations followed by a physical FPGA-based implementation. In order to incorporate heterogeneity, VM contains either a CPU or an FPGA, where the CPU performs at a rate of 1,500 MIPS and FPGA at 2,000 MIPS, respectively. The peak power consumption rate of these two types of VMs are considered as 250 and 400 watts, respectively [33]. Similarly, we have considered idle power consumption as 85 and 95 watts, respectively [18].

Service Rejection Rate (SRR) and Energy Consumption are the principal metrics based on which the evaluation has been performed. SRR can be defined as the percentage of the total number of SRs rejected by the system[5] over the entire schedule length out of total number of appeared services. Mathematically, SRR can be formulated as:

$$SRR = \frac{Total\,number\,of\,SRs\,rejected}{Total\,number\,of\,SRs\,arrived} \times 100\% \qquad (18)$$

Normalized energy consumption (NEC): The NEC is measured for different set of accepted SRs.

Experimental Setup: Each data set consists of randomly generated hypothetical SRs obtained from distinct distributions. In order to make, our simulation realistic, we have considered the example SR sets given in [33]. The weights ($wt_i = \frac{SD_i}{Lt_i}$) of the SRs have been taken from normal distribution with standard deviation $\sigma_{wt} = 0.1$ and varying different values of mean (μ_{wt}) from 0.1 to 0.4. [6] Similarly, SRs deadlines have also been generated from a normal distribution with a standard deviation. Given the service weights, we can obtain the total utilization of the system (U) by summing up the weights of all the SRs. Given the system utilization the total system load (L) can be derived by:

$$L = \frac{U}{\mathcal{M}} \times 100\% \qquad (19)$$

where M denotes the number of PEs.

It may be noted that for a given the system load (L), the average number of SRs (ρ) in the system can be achieved as:

$$\rho = \frac{L \times \mathcal{M}}{100 \times \mu_{wt}} \qquad (20)$$

The total schedule length is 100000 time-slots and all the requested SRs follow a poisson arrival process.

After considering real-life parameters (in order to make our simulation studies more fruitful and realistic), we have generated various types of data sets by setting different values for the following parameters:

1) *Average individual SR weight:* The average individual SR weight is given by the mean of the distribution from which task weights have been generated. The values of μ_{wt}, 0.1 to 0.4 have been considered.
2) System Load L: We have varied the system load L value from 50% to 90% .
3) *Number of PEs $\mathcal{M}$:* A total of 50 cores of two different types of PEs have been used for the simulation. We have considered equal number of FPGAs and CPU cores (25 of each type) in the server.

All results are generated by running 40 different instances of each data set type and then taking the average over these 40 distinct runs.

5.1 Implementation and Outcomes

5.1.1 Performance of ERASER

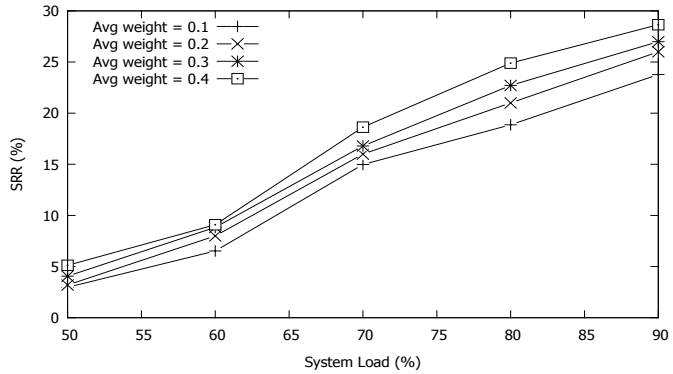
Figure 6(a) exhibits the SRR suffered by ERASER, while varying the system load. It is evident from the figure that obtained SRR for SR sets with average individual weight $\mu_{wt} = 0.1$ is comparable to that for SR sets having $\mu_{wt} = 0.2$. It may be observed that for a given U, the average number of SRs (ρ, refer equation 20) with $\mu_{wt} = 0.1$ will be nearly double compared to those having $\mu_{wt} = 0.2$. This result therefore indicates the fact that the system performs robustly against variations in the number of SRs.

From Figure 6(b) it can be observed that SRR increases with the increase in system load L. This is because higher values of L result in a correspondingly larger number of SRs (ρ), resulting in the LHS of equation 3 to become larger. Due to this, the probability of failure of the condition

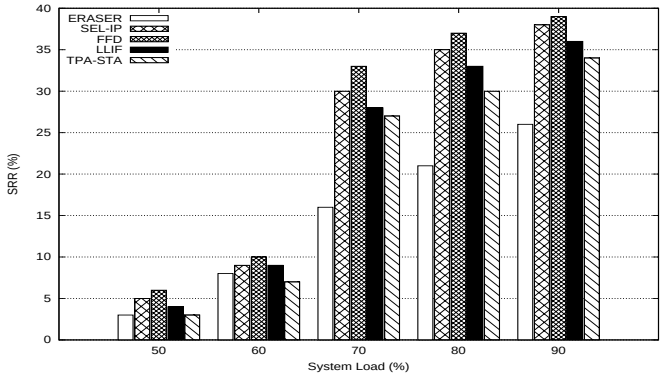
5. i.e.by the admission controller of ERASER. ERASER successfully schedules all the SRs it accepts

6. The reconfiguration overhead has been included in the SR execution time.

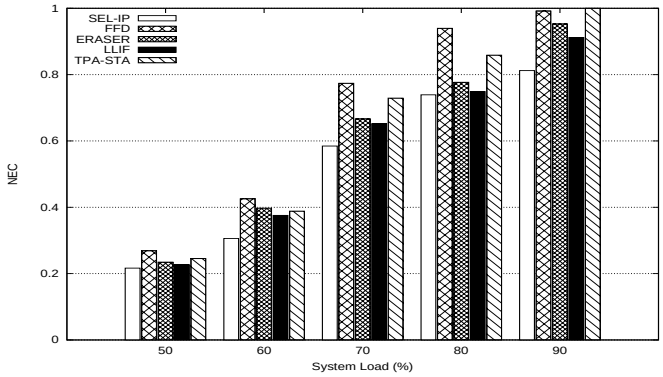
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing



(a) SR rejection rate of ERASER with different μ_{wt}



(b) SR rejection rate by different Algorithms



(c) Energy consumption for SRs execution by different Algorithms

Fig. 6: Performance of ERASER

(equation 3) increases. Increase in system load results in high rejection rate however, it also increases the energy consumption. This can be attributed to the fact that higher be the load, host will remain activated for longer duration (as the length of accepted SR increases) in order to satisfy the huge workloads of SRs and thus, energy consumption will also be high. Energy consumption of the system at different system load is shown in the Figure 6(c).

5.1.2 Comparison Results

We have compared the proposed ERASER strategy with state-of-the-art techniques like TPA-STA [25], FFD, LLIF [18] and SEL-IP [19] with respect to SRR(%), energy consumption. The comparative results are shown in Figure 6(b) and Figure 6(c). We have measured the performance of these algorithms at different levels of L, where μ_{wt} remains fixed as (0.2).

It can be observed from Figure 6(c) that SEL-IP suffers higher rejections than ERASER as the system load increases. This is mainly due to the fact that SEL-IP attempts to minimize the energy comparisons by turning off some set of servers and thus, if the load increases (i.e. number of arriving SR increases) SEL-IP rejects higher number of SRs due to resource unavailability. However, this approach become beneficial in terms of energy consumption. Hence, we can argue that SEL-IP may achieve fair energy consumption but only at the cost of higher rejection rate. On the other hand, being an "time-partitioned" based approach, ERASER can achieve 70% resource utilisation with 14% less rejection. However, the energy consumption only increases by less than 5%.

We also compared ERASER with TPA-STA strategy [25]. It can be observed that TPA-STA performance is poor than ERASER in terms of rejection rate as well as the energy consumption, as the L increases. It can be attributed to the fact that TPA-STA focuses to optimize the SR execution time by keeping a good balance in energy consumption. However, as SRs counts go high, schedule a large number of SRs based on dependency and real-time constraint becomes challenging and it causes higher rejection rate and energy consumption. ERASER depicts better performance in terms of SRR by 11% and also consumes 10.2% less energy.

A scheduling strategy called LLIF was proposed in [18]. However, being a greedy based strategy, LLIF schedules SR with longest load interval first and hence, suffers higher rejection than ERASER as evident from Figure 6(c). But, as this strategy in offline decides the minimum number (lower bound) of machines to execute a set of SRs. Such usages of machines with limited number of migration help to reduce the energy consumption. LLIF consumes 7% less energy but exhibits the increase in SRR by 12%. We have also compared ERASER with FFD [18]. FFD arranges SRs in descending order of processing time or energy consumption and allocates to the PEs. Without proper load balancing support FFD exhibits poor SRR and high energy consumption even at low system utilization. ERASER suffers 17% less SRR than FFD and ERASE is energy efficient by 21%.

6 Physical Implementation on ZYNQ SoC Testbed

6.1 Testbed development

We validated the performance of ERASER on a heterogeneous system i.e. Xilinx Zynq-7000 (ZC-702) All-Programmable SoC [34]. Figure 8 shows the diagrammatic representation of the proposed architecture. The architecture contains two types of PEs i.e. Processing System (PS), which consists of Dual ARM core for software based execution and Programmable logic (PL) utilised for hardware based execution. The region inside the PL termed as hardware accelerator is the place holder of the hardware version of SR. AXI bus creates an interface in between the DDR memory, PS and PL with other modules. In addition, Mailbox and Mutex are coordinating communication and signalling between PS and PL. Resource utilization for configuration of each hardware accelerator has been shown in Table 6.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2021.3082189, IEEE Transactions on Sustainable Computing

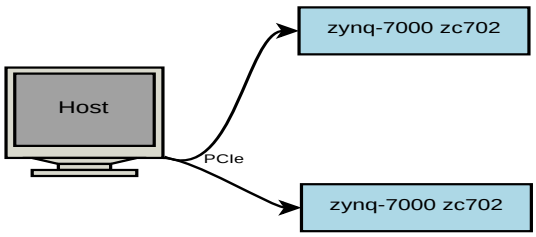


Fig. 7: PCIe Communication Architecture

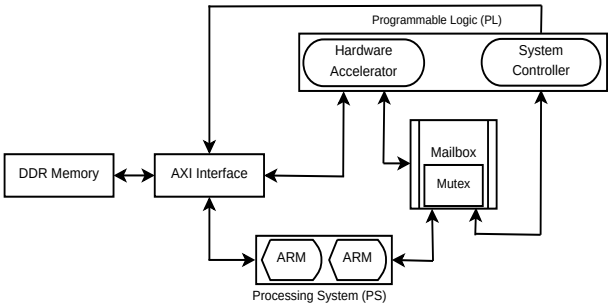


Fig. 8: ZYNQ PS-PL Architecture

TABLE 6: Resource Utilization of PL Components

	Resources		Utilization (%)	
Component	Flip-Flops	LUTS	Flip-Flops	LUTs
Hardware Accelerator	9646	14884	9.01	29.17
Mutex	90	78	0.09	0.16
Mailbox	220	274	0.21	0.54
Total	9956	15240	9.31	29.87

We have connected two Zynq boards with a PC (acting as host computer) through the PCI-express bus (3.0 standard) [35]. The pictorial representation of the connection is shown in Figure 7. The proposed algorithm is running at the host PC. Open computing language (OpenCL) based framework has been used with Xilinx software development kit (SDK). OpenCL helps to manage the data transfer between host and Zynq, allocating memory and invoking parallel code executed on Zynq etc. In PL section, the PL region is marked as "hardware accelerator" for SR execution, which is ensured by maintaining the UCF constraints [34]. SRs are allocated to the FPGAs through ICAP ports. The FPGAs are operated with PL clock (FCCLK) frequency of 50 MHz. Operating clock frequency of the ARMs (PS) is 667 MHz. Zynq SDK supports system C language, which is used to implement software version fo SR and similarly, VHDL is used for creating the hardware version. We have computed the power consumption for executing the proposed strategy through Xpower tool [34].

6.2 SR Creation Framework

We have constructed and profiled two types of SRs for the validation purpose. A well-known 'EPFL' benchmark [36] is considered. This benchmark consists of numerous applications such as *"Router"*, *"Priority"*, *"Integer to Float conversion"* etc. The power consumption of these benchmarks on PL part is measured by X-power tool and Intel platform power estimation tool (IPPET) logic is used to measure the power consumption of the SR's executed on PS. Calculated data are stored in Table 7, which is known as power generation table (PGT). The pictorial representation of the Power Measurement Framework (PMF) is shown in the Figure 9. Based on the power report generated by the Xilinx 14.4 tool, idle power for both PS and PL regions are depicted in Table 8.

TABLE 7: Benchmark SRs power (Watt)

PEs	Router	Priority	Dec	Add	I2F	Lg2	Sqrt	Calvc
FPGA (PL)	0.691	0.874	1.257	0.818	0.949	1.342	0.828	1.245
ARM (PS)	0.873	1.237	0.981	1.170	0.603	1.491	0.653	1.284

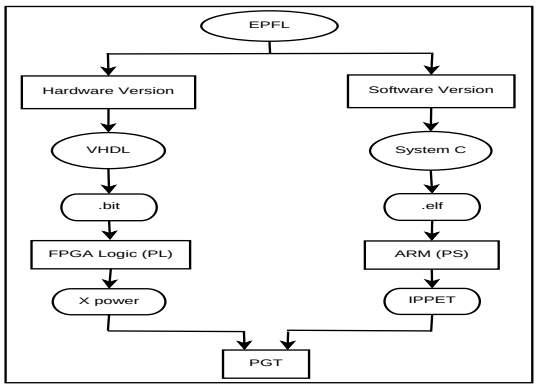


Fig. 9: PMF model

TABLE 8: Parameters of Zynq ZC702 Platform

ARM	FPGA	
PW_{idle}(W)	PW_{idle}(W)	$PW_{reconf.}$(W)
0.673	0.053	0.072

TABLE 9: Performance of ERASER with Various μ_{wt} at different utilization level

		μ = 0.2		μ = 0.1	
PEs	$L(\%)$	$Energy(J)$	$SRR(\%)$	$Energy(J)$	$SRR(\%)$
6	50	486.3	3.98	344.5	3.07
	60	563.7	7.99	413.3	5.49
	70	610.2	16.51	481.4	15.61
	80	688.4	22.39	551.6	19.79
	90	744.7	28.41	620.1	26.07

6.3 Performance of ERASER on ZYNQ SoC

We have shown the performance of the ERASER strategy in terms of SRR% and energy consumption with migration technique for heterogeneous platform where system load L and average SR weight (μ) are the variable parameters. Performance results are depicted in Table 9. We executed SRs, (i) by keeping μ constant and L is varied (ii) L is constant, while μ is varied. Table 9 shows the power consumption (measured in Joule) and rejection rate of ERASER allocation strategy over a different number of parameter combinations. Software simulation result and real implementation result on Zynq platform concludes that ERASER strategy is efficient in performances and able to fulfill our objectives.

7 Conclusions

This paper proposes, ERASER, an energy-aware real-time task allocation and scheduling strategy for heterogeneous cloud platforms. Heterogeneous cloud servers are equipped with two distinct types of PEs i.e. FPGAs and GPPs. Firstly, an ILP based technique is employed with timing constraint, to map the real-time SRs on the appropriate PEs such that energy is minimised. Further, To improve the resource utilization we have also incorporated an SR migration technique, which allows to serve maximum number of SRs by considering the dynamic request arrival scenario. We have evaluated our proposed strategy via simulation based experiments followed by the physical implementation on ZYNQ FPGA testbed with benchmark taskset. Experimental results reveal that ERASER achieves upto 90% resource utilization with only 26% SR rejection rate over different experimental scenarios. Comparison results exhibit that the ERASER outperforms state-of-the-art techniques.

Recently green cloud computing has attracted increasing attention from both the academia and the industry. The major challenge for such cloud computing is to deliver a high level of Quality of Service (QoS) by considering the factors like i. cost effectiveness for the end users and ii. energy efficiency for the cloud providers. Towards this end, in future, we would like to work on developing an energy-efficient, QoS-aware and cost-effective real-time scheduling strategy for heterogeneous cloud computing systems.

Acknowledgments

This work is supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) through grants EP/R02572X/1, EP/P017487/1, EP/V000462/1 and Rajiv Gandhi National Fellowship (RGNF) from University Grants Commission, Government of India (Fellowship Number RGNF-2017-18-SC-WES-38849) awarded to Mr. Atanu Majumder.

References

[1] A. Chauhan and P. Downing, "Systems and methods for cloud bridging between public and private clouds," Feb. 2 2016, uS Patent 9,253,159.

[2] T. Wang, Y. Li, G. Wang, J. Cao, M. Z. A. Bhuiyan, and W. Jia, "Sustainable and efficient data collection from wsns to cloud," *IEEE Transactions on Sustainable Computing*, 2017.

[3] A. Majumder, S. Saha, and A. Chakrabarti, "Eaam: Energy-aware application management strategy for fpga-based iot-cloud environments," *The Journal of Supercomputing*, pp. 1–30, 2020.

[4] H. Alhussian, N. Zakaria, A. Patel, A. Jaradat, S. J. Abdulkadir, A. Y. Ahmed, H. T. Bahbouh, S. O. Fageeri, A. A. Elsheikh, and J. Watada, "Investigating the schedulability of periodic real-time tasks in virtualized cloud environment," *IEEE Access*, vol. 7, pp. 29 533–29 542, 2019.

[5] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, "Energy-efficient scheduling for real-time systems based on deep q-learning model," *IEEE transactions on sustainable computing*, vol. 4, no. 1, pp. 132–141, 2017.

[6] M. S. Hasan, F. Alvares, T. Ledoux, and J.-L. Pazat, "Investigating energy consumption and performance trade-off for interactive cloud application," *IEEE Transactions on Sustainable computing*, vol. 2, no. 2, pp. 113–126, 2017.

[7] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 2, pp. 1–25, 2018.

[8] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft," *Future Generation Computer Systems*, vol. 93, pp. 278–289, 2019.

[9] Z. Li, S. Tesfatsion, S. Bastani, A. Ali-Eldin, E. Elmroth, M. Kihl, and R. Ranjan, "A survey on modeling energy consumption of cloud applications: deconstruction, state of the art, and trade-off debates," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 3, pp. 255–274, 2017.

[10] Y. Yamato, "Server selection, configuration and reconfiguration technology for iaas cloud with multiple server types," *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 339–360, 2018.

[11] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J.-Y. Kim *et al.*, "A cloud-scale acceleration architecture," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–13.

[12] Z. Li, "An adaptive overload threshold selection process using markov decision processes of virtual machine in cloud data center," *Cluster Computing*, vol. 22, no. 2, pp. 3821–3833, 2019.

[13] A. Majumder, K. Guha, S. Saha, and A. Chakrabarti, "Auction based power aware real-time scheduler for heterogeneous fpga cloud platform," in *2019 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS)*. IEEE, 2019, pp. 81–86.

[14] K. Gai, X. Qin, and L. Zhu, "An energy-aware high performance task allocation strategy in heterogeneous fog computing environments," *IEEE Transactions on Computers*, 2020.

[15] J. A. Jeba, S. Roy, M. O. Rashid, S. T. Atik, and M. Whaiduzzaman, "Towards green cloud computing an algorithmic approach for energy minimization in cloud data centers," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 9, no. 1, pp. 59–81, 2019.

[16] S. Xi, C. Li, C. Lu, C. D. Gill, M. Xu, L. T. Phan, I. Lee, and O. Sokolsky, "Rt-open stack: Cpu resource management for real-time cloud computing," in *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 2015, pp. 179–186.

[17] L. Zhou, L. Zhang, L. Ren, and J. Wang, "Real-time scheduling of cloud manufacturing services based on dynamic data-driven simulation," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5042–5051, 2019.

[18] W. Tian, M. He, W. Guo, W. Huang, X. Shi, M. Shang, A. N. Toosi, and R. Buyya, "On minimizing total energy consumption in the scheduling of virtual machine reservations," *Journal of Network and Computer Applications*, vol. 113, pp. 64–74, 2018.

[19] M. Safavi and B. Landfeldt, "Energy-efficient stable and balanced task scheduling in data centers," *IEEE Transactions on Sustainable Computing*, 2020.

[20] X. Liu, P. Liu, H. Li, Z. Li, C. Zou, H. Zhou, X. Yan, and R. Xia, "Energy-aware task scheduling strategies with qos constraint for green computing in cloud data centers," in *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, 2018, pp. 260–267.

[21] M. Xu, A. N. Toosi, and R. Buyya, "Ibrownout: an integrated approach for managing energy and brownout in container-based clouds," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 53–66, 2018.

[22] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[23] N. Auluck, A. Azim, and K. Fizza, "Improving the schedulability of real-time tasks using fog computing," *IEEE Transactions on Services Computing*, 2019.

[24] H. Zhao, G. Qi, Q. Wang, J. Wang, P. Yang, and L. Qiao, "Energy-efficient task scheduling for heterogeneous cloud computing systems," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2019, pp. 952–959.

[25] B. Hu, Z. Cao, and M. Zhou, "Scheduling real-time parallel applications in cloud to minimize energy consumption," *IEEE Transactions on Cloud Computing*, 2019.

[26] K. Gai, M. Qiu, and H. Zhao, "Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 126–135, 2018.

[27] S. K. Tesfatsion, J. Proaño, L. Tomás, B. Caminero, C. Carrión, and J. Tordsson, "Power and performance optimization in fpga-accelerated clouds," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 18, p. e4526, 2018.
[28] F. Xhafa, G. Mastorakis, C. X. Mavromoustakis, and C. Dobre, "Guest editorial: Special issue on algorithms and computational models for sustainable computing in cloud and data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 47–48, 2017.
[29] X. Wang, Y. Niu, F. Liu, and Z. Xu, "When fpga meets cloud: A first look at performance," *IEEE Transactions on Cloud Computing*, 2020.
[30] S. Saha, A. Sarkar, and A. Chakrabarti, "Scheduling dynamic hard real-time task sets on fully and partially reconfigurable platforms," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 23–26, 2015.
[31] C.-H. Hsu, K. D. Slagter, S.-C. Chen, and Y.-C. Chung, "Optimizing energy consumption with task consolidation in clouds," *Information Sciences*, vol. 258, pp. 452–462, 2014.
[32] R. Lima and E. Seminar, "Ibm ilog cplex-what is inside of the box?" in *Proc. 2010 EWO Seminar*, 2010, pp. 1–72.
[33] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, 2014.
[34] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014.
[35] A. Rjabov, A. Sudnitson, V. Sklyarov, and I. Skliarova, "Interactions of zynq-7000 devices with general purpose computers through pci-express: A case study," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*. IEEE, 2016, pp. 1–4.
[36] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," in *Proceedings of the 24th International Workshop on Logic & Synthesis (IWLS)*, no. CONF, 2015.

Atanu Majumder is a Ph.D student in A. K. Choudhury School of Information Technology (AKCSIT), University of Calcutta. He has completed his M.Tech from University of Calcutta in 2016 and is also the recipient of Rajiv Gandhi National Fellowship award from University Grants Commission, Government of India. His research area deals with energy and performance efficient real-time resource management in FPGA-based heterogeneous cloud platform.

Sangeet Saha Sangeet Saha received his B.Tech degree in Information Technology in 2011 and M.Tech degree in Computer Science and Engineering in 2013 from University of Calcutta, Kolkata. He completed his PhD from the same institute as a Tata Consultancy Services (TCS) research fellow in the year 2018. After completing the PhD, he is currently appointed as a Senior Research Officer in the Embedded and Intelligent Systems (EIS) Research Group at the University of Essex in Colchester as a Senior Research Officer from May 2018. His current research interests include real-time scheduling, Scheduling for reconfigurable computers, real-time and fault-tolerant embedded systems, cloud computing. He published several of his research contributions in conferences like CODES+ISSS, ISCAS, IOLTS, etc. and in journals like ACM TODAES, IEEE TMSCS, IEEE TSMC:Systems, Journal of Supercomputing (Springer) and IEEE Sensor Journal.

Amlan Chakrabarti (Senior Member, IEEE) is presently Professor and Director of AKCSIT, University of Calcutta. He was also the Dean of Faculty for Engineering and Technology in University of Calcutta. Prior to this, he completed his post doctoral research in Princeton University after completing his Ph.D. from University of Calcutta in association with ISI, Kolkata. He is the recipient of DST BOYSCAST fellowship award in Engineering Science in 2011, Indian National Science Academy (INSA) Visiting Faculty Fellowship in 2014, JSPS Invitation Research Award in 2016 and Erasmus Mundus Leaders Award from EU in 2017 and Hamied Visiting Fellowship from Cambridge University in 2018. He has been associated with reputed international and national institutes of repute as a Visiting Professor like University of Cambridge (UK), City University of London (UK), University of Oradea (Romania), SUNY Buffalo (USA), GSI Helmholtz Research Laboratory (Germany), University of Bremen (Germany), CERN (Geneva), Kyushu Institute of Technology (Japan). His present research interests include VLSI Design, Quantum Computing and Embedded System Design

Klaus McDonald-Maier (Senior Member, IEEE) is currently the Head of the Embedded and Intelligent Systems Laboratory, University of Essex, Colchester, U.K. He is also the Chief Scientist with UltraSoC Technologies Ltd., the CEO of Metrarc Ltd., and a Visiting Professor with the University of Kent. His current research interests include embedded systems and system-on-chip design, security, development support and technology, parallel and energy-efficient architectures, computer vision, data analytics, and the application of soft computing and image processing techniques for real-world problems. He is a member of VDE and a Fellow of the BCS and the IET.