Real-Power Computing

Rishad Shafik†, Alex Yakovlev† and Shidhartha Das‡

***Abstract*—The traditional hallmark in embedded systems is to minimize energy consumption considering hard or soft real-time deadlines. The basic principle is to transfigure the uncertainties of task execution times in the *real* world into energy saving opportunities. The energy saving is achieved by suitably controlling the reliable power supply at circuit or system-level with the aim of minimizing the slack times, while meeting the specified performance requirements.**

Computing paradigm for emerging ubiquitous systems, particularly for the energy-harvested ones, has clearly shifted from the traditional systems. The energy supply of these systems can vary temporally and spatially within a dynamic range, essentially making computation extremely challenging. Such a paradigm shift requires disruptive approaches to design computing systems that can provide continued functionality under unreliable supply power envelope and operate with autonomous survivability (i.e. the ability to automatically guarantee retention and/or completion of a given computation task). In this paper, we introduce *Real-Power Computing*, inspired by the above trends and tenets. We show how computation systems must be designed with power-proportionality to achieve sustained computation and survivability when operating at extreme power conditions. We present extensive analysis of the need for this new computing approach using definitions, where necessary, coupled with detailed taxonomies, empirical observations, a review of relevant research works and example scenarios using three case studies representing the proposed paradigm.

I. Computing is Changing

Over the years, computing systems have found their usage in a large number of domains. Considering their typical power consumptions, these domains can be roughly categorized into six major applications, such as high-end many-core server systems, desktop computing, portable computing, mobile systems, embedded systems and low-end ubiquitous systems. Fig. 1 depicts four different trends of these applications: design and optimization requirements, expected/current population of these systems, power supply variations and energy efficiency requirements by them. These trends show how design considerations have evolved with power and/or performance constraints for different application domains, highlighting their degree of usage in terms of device populations. For line supply powered computing applications, such as high-end server and desktop computing systems, performance is typically constrained by power consumption (which ranges from tens of watts to several mega-watts) to control the operational costs and system overheating [4]. For battery-powered systems, such as portable computing devices and embedded systems (with typical power from a few milliwatts

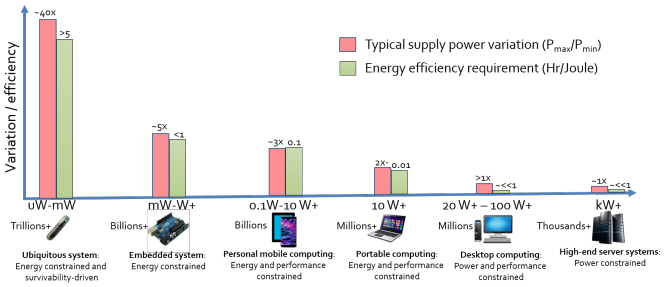


Fig. 1. Major computing applications and their typical system design and optimization requirements, expected current population of these systems, normal supply power variations and energy efficiency requirements [1]–[3].

to several watts), performance is often compromised in favor of extended operating lifetime [5]. In many embedded systems it is common to have real-time constraints, which can either be hard (i.e. time constraint cannot be violated) or soft (i.e. time constraint can be occasionally violated) [6]. The energy saving in these systems is achieved by suitably controlling the power supply at circuit- or system-level with the aim of minimizing the slack time (i.e. the time between task execution time and its deadline). Fig. 2 shows a demonstration of performance-driven energy minimization approaches for real-time systems. A common denominator for all these applications is the capability of operating under reliable power supplies, while providing with certainty in computational performance.

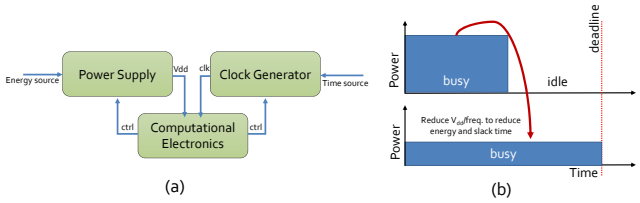


Fig. 2. Real-time performance-driven power minimization in embedded systems. Dynamic voltage/frequency scaling (DVFS) is a key aspect of these systems; using DVFS the slack time (difference between deadline and execution time) can be minimized for power/energy efficiency.

A point worth noting from Fig. 1 is the energy efficiency requirements of these applications. As the operating power level becomes smaller, particularly for the battery-powered systems, they are being challenged with longer operating lifetimes. This has led to research and innovation in the general area of Low-Power Computing with the basic premise of "making the most of available energy" [7]. A key aspect of achieving such energy efficiency is the ability to operate with multiple supply voltages (*i.e.* Vdd), from sub-threshold to super-threshold [8], [9]. As energy efficiency needs become more prominent, the Vdd range between minimum (V_{min}) to maximum (V_{max}) point also tends to be higher (see Fig. 1).

The dramatic spread of computing, at the scale of trillions of emerging ubiquitous systems, is delivering on the pervasive penetration into the real world in the form of data-driven Internet of Things (IoT) [10], [11]. Examples

†R. Shafik and A. Yakovlev are with the School of Electrical and Electronic Engineering, Newcastle University, Newcastle upon Tyne, UK
e-mail: {rishad.shafik,alex.yakovlev}@ncl.ac.uk.
‡Shidhartha Das is with ARM, Cambridge, UK
e-mail: {shidhartha.das}@arm.com
Manuscript received March 29, 2018.

include implantable or wearable devices, cybernetics and *fire-and-forget* sensing systems in smart cities and offices. For ubiquity, typical design requirements of these devices are to be small, low-cost and light-weight by harvesting energy from the real-world through vibration [12] or thermal [13] energy, or from environment through solar or kinetic energy [13], [14]. As harvesting sources have natural fluctuations in their physical properties, the available energy of these devices also varies significantly, both in spatial and temporal dimensions, often by two orders of magnitude or more [15]. This makes the operation of these devices challenging, particularly when the available energy is unreliable but the device needs to complete useful computations [16]. Hence, a highly desirable property of these devices is to have natural survival instincts defined by the available energy levels. In other words, they should continue to provide a required computation capacity at limited energy levels, even if it requires gracefully degrading the computation quality or retaining the computation states for resumption when more energy is available. Biological organisms and systems, such as microbes, work with similar principles as they morph and adapt for carrying out useful synthesis and regenerative processes for their survival under varying sunlight [17].

Traditional approaches are agnostic of such survival instincts under varying supply energy levels [16]. These approaches react to low energy situations by scaling operating voltage/frequency to extend lifetime, which does not guarantee retention or completion of computation tasks before the system is depleted of power [18]. In fact due to lack of survival instincts the direct application of existing approach can cause loss of computation in such an event, as shown in Fig. 3. Indeed, a change in the computing paradigm is needed to design computing systems with natural survivability and adaptability instincts that go beyond traditional approaches for dealing with unreliable power supply. This paper introduces one such computing paradigm, named Real-Power Computing, with the following key ***objectives and contributions***:

1) a definition of the new paradigm underpinning rationale and an extensive review of related works,
2) a detailed taxonomy of the paradigm, showing different design and run-time optimization approaches,
3) three case studies and exemplars demonstrating the effectiveness of the proposed paradigm applied in different taxonomy scenarios, and
4) a brief outline of the open research challenges and opportunities surfacing this paradigm.

The rest of the paper is organized as follows. Section II argues the rationale of real-power computing, together with its definition, manifestations and taxonomies. Section III outlines design methods for the envisioned new paradigm, while Section IV gives insights into run-time adaptation needs for power proportionality and survivability. Section V provides three different case studies as exemplars of different real-power computing aspects. Section VII and VI summarize challenges, opportunities underpinning existing research works. Finally, Section VIII concludes the paper.

Throughout the paper we will use energy and power terms as follows. From the supply side, the *energy* term will be used to refer to harvesters with built-in storage, while the *power* term will indicate to the rate of energy dispensation over time. For the computing logic side, the energy term will define the total *power* consumed over a given time interval.

II. Real-Power Computing

In his visionary article [19](p. 438), Oliver Heaviside wrote: *"Now, in Maxwell's theory there is the potential energy..., and there is the kinetic or magnetic energy.... They are supposed to be set up by the current in the wire. We reverse this; the current in the wire is set up by the energy transmitted through the medium around it. The sum of the electric and magnetic energies ....is definite in amount, and the rate of transmission of energy (total) is also definite in amount."*

In computing systems the situation is analogous; the energy consumed by the electronic devices (e.g., transistor switches, parasitic capacitors, current mirrors and interconnects) allows for the information transformation from one form to another. If we reverse this angle of thinking, we can see that the information transformation is in fact the product of the energy input to the underlying circuits and this energy as well as its rate are definite in amount. The consequence of this reversal is remarkable. Traditionally, our view would be to consider given computation task as something definitely known, determined by the algorithm, the hardware underneath and the data [20]. Hence, definitely known is the list of actions this system will go through. Then, we can estimate, although approximately, the amount of energy consumed by this definite computation.

With the reversed view, the key question is: *can we guarantee reliable computation under unreliable power supplies, mitigating frequent computational uncertainties*? One particular form of computational uncertainty is performance uncertainty in terms of the time it takes to perform the computation. While we have the definite power level what we can also have is a definite computation (hardware, algorithm, data and sequence of actions) but with uncertain performance [21]. Another form would be to have both definite power or energy budget and time deadlines, but then accept the possibility of the temporary termination of the computation when either energy or time limits has been reached.

However unusual the computational uncertainty might appear to us, raised to traditional approaches of computing, our pervasive electronic systems will increasingly follow the second and non-traditional view. This is because, today's widely used paradigms such as those of Real-Time (compute by deadlines) and Low-Power (prolonging battery life or throttling for power densities) cannot address the strict computation requirements imposed by the above question. The new generation of devices and applications in the computing swarm, many of which are expected to be confronted with challenges of autonomy in the absence of batteries, will need a power-centric design and run-time adaptation. This leads us to define the new envisioned paradigm as *Real-Power Computing*. The engineering definition and taxonomies of real-power computing follow.

Real-power computing can be defined as follows:
Real-power computing (RPC), or energy-driven computing,

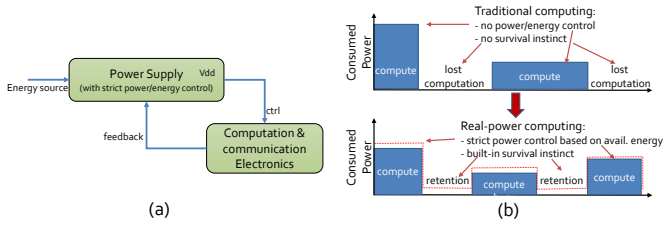


Fig. 3. (a) High-level block diagram of the proposed real-power computing paradigm consisting of feedback based power control unit and the computation/communication electronics, and (b) traditional computing compared to real-power computing, showing two key features of the proposed paradigm: the ability to control power budgets based on the available energy, and autonomous deeply-embedded survival instincts.

describes hardware and software systems subject to a "real-power constraint", for example availability of energy from a power source, or restriction on power dissipation. Real-power applications must guarantee performance within specified power constraints, referred to as "power bands" Systems of this type are linked to the notion of survivability, which depends on their power aspects as well as their ability to morph functional aspects to ensure continued computation. Real-power systems are not simply low-power systems which are optimized to the criterion of minimum power consumption.

Based on the above definition, we have a range of possible formulations of the problems leading to various scenarios in real-power computing. Considering the control derivatives of the input power, we categorize these as hard or soft real-power computing. Later, we also discuss the implications of introducing the real-time constraints alongside power/energy.

A. Hard Real-Power Computing

Hard real-power computing systems have no energy storage capacity, i.e. the scavenged power is delivered directly into the circuits and systems. As such, the input power will need to be strictly budgeted for guaranteeing a certain set of computations; if the available power does not allow this, no computation is carried out. The failure to meet power budget will eventually lead to incomplete computations, which can be carried out when more power is available. Examples of hard real-power systems include autonomous cybernetic or signal processing systems that have to carry out periodic and non-critical data sensing and computations.

A key requirement for establishing hard real-power computation is to have maximum predictability of supply power so that power scheduling policies can be derived accordingly. Moreover, it is equally important to have a high level of transparency of the computing units in terms of worst case power consumption (WCPC), similar to worst case execution time (WCET) in hard real-time computing systems. The evaluation of WCPC would need extensive off-line characterisation of deterministic computational loads (*eg.* ASICs, microcontrollers, memories and interconnects) against different power supply situations. We term this process of scheduling computational tasks based on power availability as power-compute co-design. Figure 4 shows demonstration of a hard real-power computing system that periodically processes sensed data in four phases: wake up, sense, process, communicate and retain/sleep.

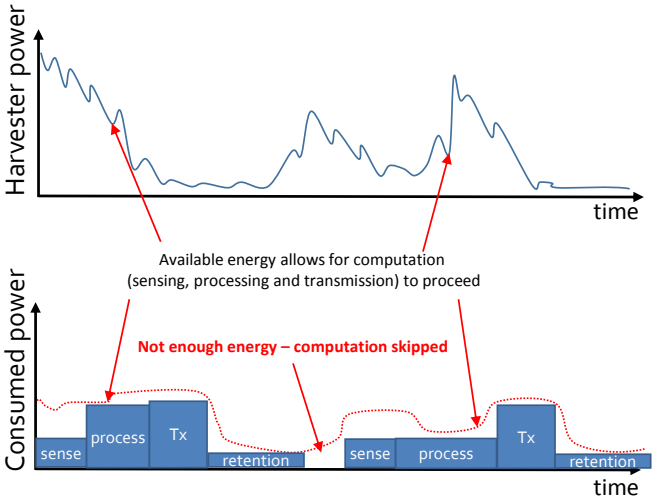


Fig. 4. An example demonstration of hard real-power computing showing four key tasks: initiate wake up, sense data, process data, transmit data and retain data/initiate sleep mode. When energy is available, the system formulates a strategy to control power with the aim of allowing these tasks to be carried out, strictly meeting the power budget. However, when the available energy is not sufficient, the system does not allow these tasks to be carried out.

B. Soft Real-Power Computing

Unlike hard real-power computing, soft real-power computing has built-in energy buffers (e.g. supercapacitors) with limited capacity. As such, it does not strictly control computations against a power budget derived from the scavenged power. Instead, power budgets are formulated based on the currently available energy. When the available energy is lower than expected, soft real-power systems can allow for partial computation to be carried out, even if these violate the power budget. Examples of soft real-power applications include sporadic data ingestion or dynamic sensing systems, which can tolerate loss of periodic computations in full or in part.

Soft real-power computing systems can use a combination of power-compute co-design and run-time optimisation. Power-compute co-design simplifies the control problem with estimation of the expected power consumption (EPC) of computational loads. Computational loads, such as microprocessors, dynamically reconfigurable systems and storage subsystems are typical candidates for soft real-power controls as EPC models can be instrumented for dynamic feedback based run-time control. Based on the feedback, the control decisions can be adapted to enforce high energy frugality [22]. Figure 5 demonstrates soft real-power computing systems using similar example tasks as in Fig. 4.

C. Managing Performance Uncertainty

Some applications inherently require certainty in performance, which could be imposed through either hard or soft real-time deadlines. Within the remits of real-power computing the delivery of such performance expectations can be explained as follows. With an additional real-time deadline, the problem of devising power budgets in real-power computing is reduced to identifying the least energy (product of average power budget and time deadline) that can be frugally utilized to deliver the best quality of computation (which can be application-dependent), which can be modulated in favor

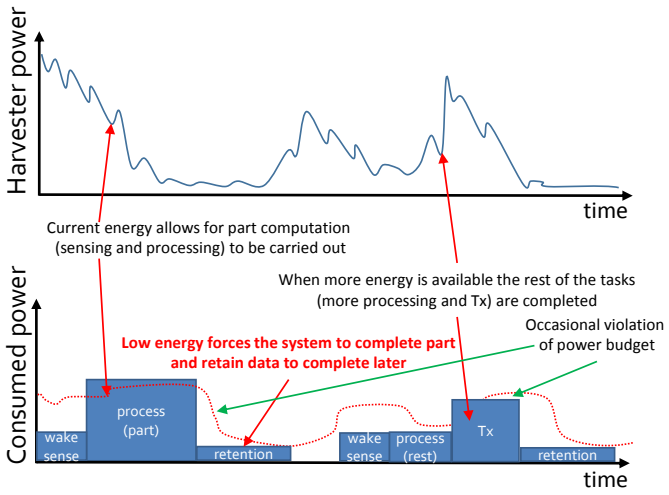


Fig. 5. An example demonstration of soft real-power computing showing similar tasks. When energy is available, the system proceeds with power budgeting to allow for part of the tasks to be carried out, often with some violation of the power budget. The remaining tasks are completed when more energy becomes available.

of energy efficiency through approximate or heterogeneous computing or a combination of both.

Considering different performance uncertainly scenarios, a number of different optimization problems (i.e. taxonomies) are given below.

Given:

T_d	Expected time deadline
T	Expected computation time
P_t	Power consumption at time t
P_t^{budget}	Power budget at time t
P_{avg}	Average power budget over time T
E	Energy consumption over time $T (E = P_{avg} \times T)$
E_{avail}	Energy available over time T
C	Computation/communication task functionalities
c	Chosen task functionality/implementation
Q	Quality of computation carried out

---

1. Hard real-power computing:

$\forall_t \exists_{P_t \leq P_t^{budget}:c \in C} \quad \max : Q(c)$

Given no storage for scavenged energy, there exists a variant of computation functionality c which will always meet the power budget ($P_t \leq P_t^{budget}$) at time t. The choice of functionality c will ensure the best possible computation quality within the power budget. A case study of hard real-power computing is shown in Section VI.B.

2. Soft real-power computing:

$\forall_t \exists_{P_t \approx P_t^{budget}:c \in C}$

$min : E(P_t, T, c) \leq E_{avail}, \ \& \max : Q(c)$

Given some storage for scavenged energy, there exists a variant of computation functionality c which will approximately meet the power budget ($P_t \approx P_t^{budget}$) at time t. The choice of functionality c will ensure that energy consumption is always less than the available stored energy, and provide the best possible computation quality. A case study of soft real-power computing is presented in Section VI.A.

3. Hard real-power hard real-time computing:

$\forall_t \exists_{P_t \leq P_t^{budget}:c \in C} \quad \max : Q(c) \text{ st. } T \leq T_d$

Given no storage for scavenged energy, there exists a variant of computation functionality c which will always meet the power budget ($P_t \leq P_t^{budget}$) at time t. The choice of functionality c will ensure the best possible computation quality within the power budget and also strictly meet the real-time deadline.

4. Hard real-power soft real-time computing:

$\forall_t \exists_{P_t \leq P_t^{budget}:c \in C} \quad \max : Q(c) \text{ st. } T \approx T_d$

Given no storage for scavenged energy, there exists a variant of computation functionality c which will always meet the power budget ($P_t \leq P_t^{budget}$) at time t. The choice of functionality c will ensure the best possible computation quality within the power budget and also approximately meet the real-time deadline.

5. Soft real-power hard real-time computing:

$\forall_t \exists_{P_t \approx P_t^{budget}:c \in C}$

$min : E(P_t, T, c) \leq E_{avail}, \ \& \max : Q(c) \text{ st. } T \leq T_d$

Given some storage for scavenged energy, there exists a variant of computation functionality c which will approximately meet the power budget ($P_t \approx P_t^{budget}$) at time t. The choice of functionality c will ensure that energy consumption is always less than the available stored energy, and strictly meet the given real-time deadline, while also providing with the best possible computation quality.

6. Soft real-power soft real-time computing:

$\forall_t \exists_{P_t \approx P_t^{budget}:c \in C}$

$min : E(P_t, T, c) \leq E_{avail}, \ \& \max : Q(c) \text{ st. } T \approx T_d$

Given some storage for scavenged energy, there exists a variant of computation functionality c which will approximately meet the power budget ($P_t \approx P_t^{budget}$) at time t. The choice of functionality c will ensure that energy consumption is always less than the available stored energy, and approximately meet the given real-time deadline, while also providing with the best possible computation quality.

Ensuring energy efficiency, performance and quality requirements are met in real-power computing can be challenging due to large system space during optimization. Hence, it requires a systematic and cross-layer approach for design-time power-compute co-design, together with run-time adaptation as described in Fig. 6. The details of power-compute co-design are described next, which is then followed by run-time adaptation for energy efficiency and survivability (Section IV).

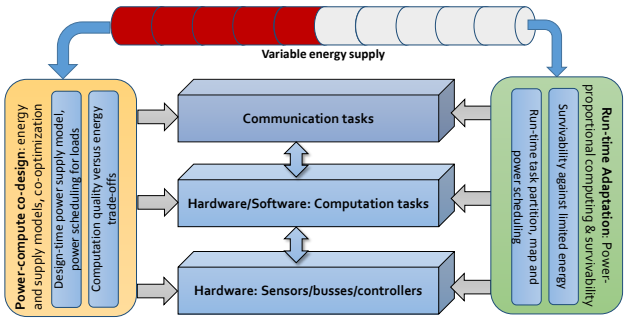


Fig. 6. A cross-layer block diagram of the envisioned real-power computing paradigm. Power-centric control for resources, including communication/computation tasks and input/output subsystems, is fundamental to the concept of this new paradigm. To ensure optimal controls, the system requires early stage power-compute co-design, which extensively analyzes the power supply variations against computation/communication energy consumption. The aim is to determine the optimal power scheduling for resources. Power-compute co-design is often coupled with continuous run-time adaptation to ensure application-aware survivability in the event of limited energy availability using hardware/software knobs and feedback through monitors.

III. Power-Compute Co-Design

Power-compute co-design is a design-time optimization approach of real-power computing. It can be defined as *a set of design automation tools and techniques that models the relationships between power sources and computational loads (hardware, software and communication subsystems), thereby formulating efficient power scheduling policies for*

computational tasks. In the following the different aspects of power-compute co-design are briefly discussed.

A. Power Supply Models

The power supply in ubiquitous systems typically have large spatial and temporal variations [23]. Understanding and modeling these variations is core to real-power computing. Spatial variation models characterize the power supply voltages and their variations, and determine the maximum and minimum operating points [24]. Since harvested power is typically a function of the operating environment, realistic assumptions must be made to derive accurate spatial variation models. Additionally, temporal variations also need to be modeled to establish high predictably of the available energy over a given time. Hard real-power systems can use more pessimistic assumptions of the available energy, while soft real-power computing can leverage deviations in assumptions to a run-time adaptation problem. Power supply models can then be used to design power controllers with survivability instincts and enable appropriate power scheduling for computational loads at design-time [25]. Fig. 7 depicts harvested power of four sources highlighting the temporal and spatial variations.

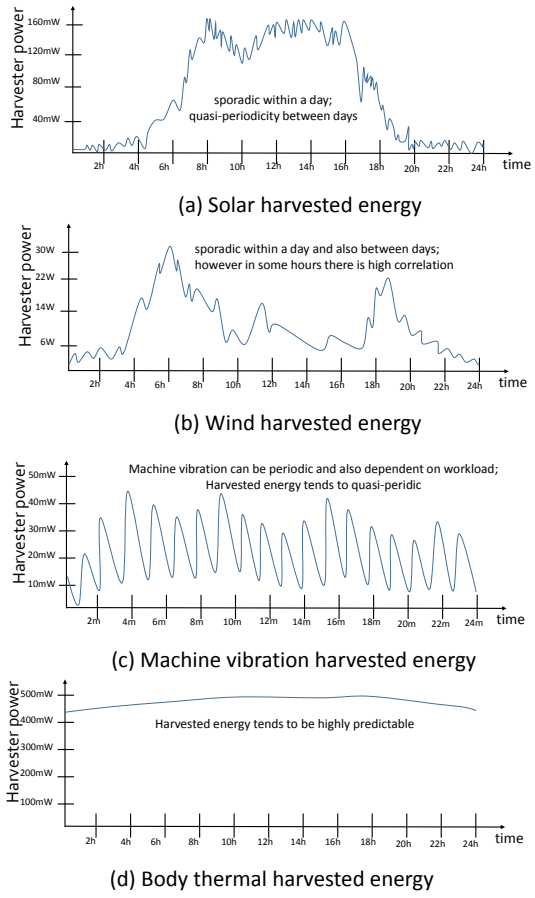


Fig. 7. Power supply behavior of different energy harvesting sources: (a) solar harvested energy [16], [26] tends to have a sporadic pattern within a day; however, between days the energy be quasi-periodic depending on the weather (not shown), (b) wind harvested energy [27], [28] shows similar behaviour as solar energy with typically larger spatial variation, (c) machine vibration harvested energy [29], [30] can be quasi-periodic depending on the machine workload, and (d) human body thermal energy harvested using a body area network with 1700 nodes [31] tends to be highly predictable as the temporal variations are generally slow.

B. Energy Transparency Models

Energy transparency models study the impact of performing a set of computation or communication tasks in terms of their resulting energy consumption, carried out at design-time. In the following these models are briefly outlined for computation and communication tasks:

1) Computation Tasks

Energy transparency models of computation tasks study the energy consumption variations of hardware/software resources [32]. Depending on the intended real-power computing paradigm (hard or soft), the energy cost estimation either needs to be accurate or approximate. For example, hard real-power systems requires accurate estimation at instruction- or micro-architectural level as under-estimation can lead to violation of the power budgets imposed by the power controller. Microcontrollers and ASICs typically have deterministic computational behavior [33], and hence these are well-suited for accurate energy transparency models using worst-case power consumption (WCPC) estimations. On the other hand, soft real-power systems can leverage approximations in energy estimations, and adapt during run-time. Microprocessors with hierarchical caches and reconfigurable logic circuits and systems tend to exhibit variations in their energy consumptions [34], and as such they are suitable for power-compute co-design using expected power consumption (EPC) models.

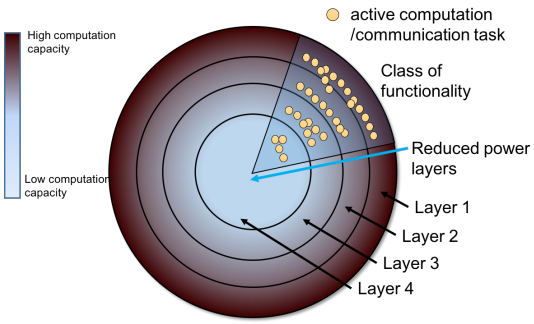


Fig. 8. Layered computational activity in response to power levels. The different layers signify power layers and their classes of functionalities: the inner layers have lower computational capacity and hence more energy-frugal, while the outer layers have higher power/computation capacity. In a typical system different classes with higher number of software tasks are carried out at outer layers, while modular and simpler hardware tasks are performed at innermost layers. The layers in the middle gradually change in their computational capacities and energies. In a real-power computing system, when energy is scarce, the tasks at outer layers are progressively shut down to retain the most essential tasks, e.g. retention, check-pointing.

2) Communication Tasks

Communication tasks are carried out in parallel or in an interleaved manner alongside computation [35]. In ubiquitous systems, these tasks are deterministic (in regular patterns of sense, process and communicate data). However, the energy consumption of these tasks can vary due to network behavior (wireless or wired) [36]. As such, to generate energy transparency models for these tasks a key requirement is to define the detailed network characteristics (including traffic, channel or network availability and congestion scenarios) [37]. Based on such network characteristics, the expected energy consumed for each of the communication packets can be estimated. Similar to computation tasks, these estimations can

be carried out optimistically for soft real-power systems or pessimistically for hard real-power systems.

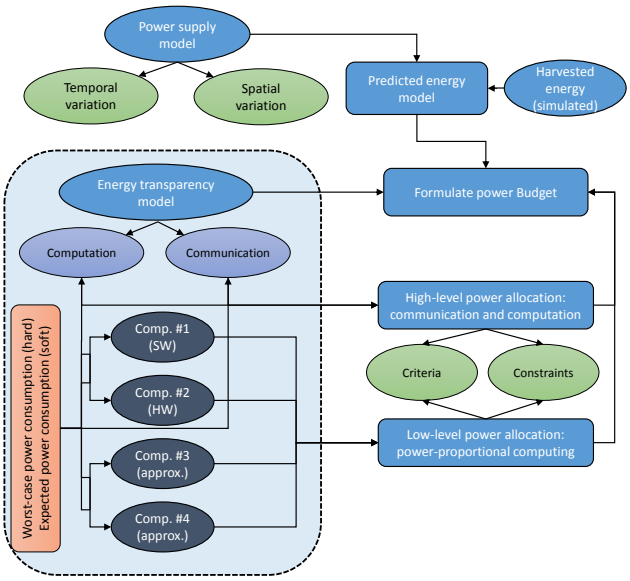


Fig. 9. A flowchart of power-compute co-design for real-power computing systems. The power supply model, together with harvested energy simulation enables a power budget formulation. For hard real-power system, a more conservative budget is formulated, while for soft real-power systems, a more pessimistic power control is applied. Based on the power budget, the power allocation to the computation and communication loads are carried out using the energy transparency model. Initially, a high-level power scheduling is done between computation and communication tasks. The higher-level power schuduling is also coupled with a low-level power scheduling algorithm that suitably identifies the best hardware/software partitioning and mapping needed to optimize for the given criteria (*eg.* soft or hard real-power) and constraints (*eg.* deadline, quality) for the optimization problem. For accurate formulation of the design-time policies, energy/power models of the individual computational loads are also fed back to the power scheduling governor.

C. Power Scheduling and Optimization Problems

Existing electronic design automation tools allow for hardware/software co-design to optimize for performance and/or reduced power consumption [38]. A common feature of these tools is the mapping and partitioning decisions between hardware and software resources that need exploration at early design phase. For real-time systems, such exploration can drastically reduce the complexity of run-time optimization to warrant task executions by a given deadline.

Within the remits of proposed real-power computing, power-compute co-design needs to identify the appropriate power scheduling and optimization policies to ensure a fixed energy budget can be effectively allocated among the computation (and communication) resources. The co-design tool will be provided with a set of supply power and load energy transparency models, and it will then generate a set of rules for power distribution among computation and communication loads. Using these rules, multiple layers of the system architecture can turn on/off at different power levels (cf. analogies with living organisms nervous systems or underwater life, or layers of expensive/cheap labour in most of the resilient economies). As power goes lower computation at deeper layers (i.e. survival layers with lower accuracy and computation capacity) stay on, while the surface layers (i.e. higher accuracy and more computation capacity) turn off; this is where instincts become more in charge! The more effectively the system manages these layers, the more energy-efficient and survivable it is. This layered view is reflected in Fig. 8, which puts it in analogy with the sea layers and ability of different forms of life to survive in different conditions of sunlight penetration. Fig. 9 shows a flowchart of power-compute co-design, demonstrating how layered power scheduling is governed to ensure the desired optimization criteria and constraints are met [39].

IV. Run-time Adaptation

Run-time survivability is a new concept for electronic systems. According to Oxford English Dictionary, "Survival is the continuing to live after some event; remaining alive, living on." Such an event could be the termination of power supply or drastic reduction in power levels. Conventional definitions of survival and survivability in ICT systems render themselves to considering this notion as a synonym to graceful degradation in the presence of faults or something abnormal, i.e. "out of order" conditions. This might be a suitable way if we treat the system to work normally under the unlimited energy resources.

When we consider real-power conditions, we are actually staying within "normal operating" conditions. Hence our system should be equipped with the capability to react to power and energy interruptions. In biology, such capability is usually called 'instincts' [40], which can be better described by the following two quotations: 1) "*the very essence of an instinct is that it is followed independently of reason*." [1], and 2) "*the operation of instinct is more sure and simple than that of reason*." [2]. Based on these observations, an equivalent of instincts in electronic systems can be associated with the notion of "Deep Survival". Deep Survival refers to maintaining the operation in several structural and behavioural layers, with mechanisms to adapt to unexpected energy situations. Electronic systems can be designed for deep survival by providing multi-modal computational capability in layers, where each layer corresponds to a given computational complexity and associated quality/energy tradeoffs [41] (see Fig. 8).

Fig. 11 depicts an illustrative approach to engineering run-time adaptability and survivability. As can be seen, power-proportional heterogeneous computing and adaptability for survival are two key issues for real-power computing as described below:

A. Power-Proportional Computing

A fundamental approach to achieving survivability is the principle of power-proportional computing [18], [42]. A given power, when applied to a computational device, can be converted into a corresponding amount of computation activity by selecting the appropriate layer of computation. Run-time adaptation must ensure this through seamless switching between layers of different computation activities at different power levels (see Section III) and Fig. 10).

Servicing a known functionality (a set of computation and communication tasks) in different modes and types is key to achieving power proportionality. One mode of this could be computing (and communicating) with heterogeneous resources. These resources can provide similar functionality

[1] C. Darwin, Descent of Man I, 1871
[2] E. Gibbon, Decline & Fall of The Roman Empire, 1804

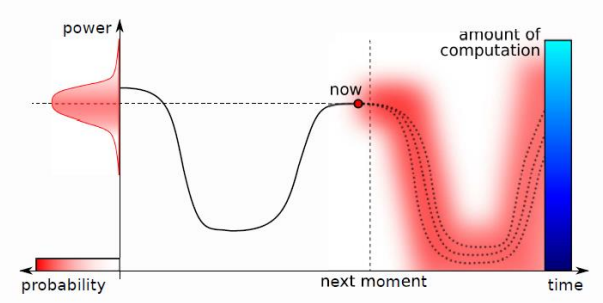


Fig. 10. Power profile in time, its uncertainty and illustration of power-modulated computing. The past observations can provide certainty in power distribution in response to computation capacities through statistical inference. Current and future observations use the past observations to manage the uncertainty in performance responding to different computation capacity and its power allocations.

but with different energy/performance trade-offs. When there is good energy availability, it may be more convenient in terms of controllability, precision and programmability to perform functionality using traditional computing resources, such as CPUs with DSPs. However, when the energy is scarce, similar functionality can be provided through more customized resources, such as FPGAs/ASICs for better quality of service at low energy. The decision of performing computation (and communication) through a resource will be strictly governed by design-time rules and run-time adaptation algorithms built in the system based on the energy availability and proportionality [43], [44].

In extreme energy conditions, computing can be challenging using these traditional computing resources. To ensure useful computation (and communication) tasks can still be carried out, the traditional definition of functionality, whereby the output data and their quality can be deterministically related to a given set of input data, will need to be relaxed. This leads to another mode of computing using power-proportional approximate computing. To enable this promising mode new computational units need be designed to meet the ultra low-energy computing requirements at gracefully degraded quality of the functionality [45]. The impact of trading quality off in favor of energy can be strictly application-specific, and hence these will need to carefully analyzed at design-time during power-compute co-design (see Section III).

B. Adaptability for Survival

When power levels become uncertain or scarce, deep survival becomes incumbent. During such an event, the system will need to "consciously" switch between a full functionality mode to a low-latency hibernating mode primarily depending on the data processing and application requirements (see Fig. 11). The consciousness requires two unique design features in the system: dynamic retention and adaptability. In the following we briefly discuss these characteristics with examples.

1) Dynamic retention

Retention is the ability to save a stable state of the computing system. Using this state, the system can continue the computation from where it left off. In traditional computing systems, retention is carried out through check-pointing

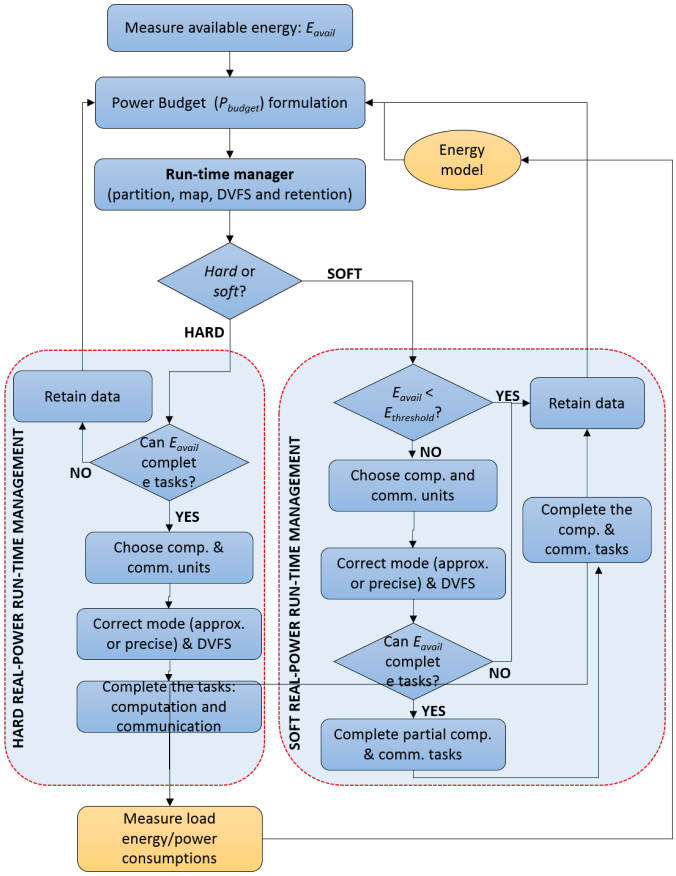


Fig. 11. A demonstrative flowchart of run-time adaptation for real-power computing systems. For run-time adaptation the available energy (E_{avail}) is directly measured from the power supply; based on this energy power budget (P_{budget}) is formulated. The run-time manager allocates power resources using P_{budget} as a guideline. For hard real-power systems, initially E_{avail} is checked against the required energy by the tasks in concern. If the required energy is less than E_{avail}, no computation/communication is carried out and data are retained; however, when E_{avail} can ensure survivability, computation and communication tasks are carried out meeting a specified quality. Unlike hard real-power systems, soft real-power systems are managed most optimistically. When E_{avail} is greater than a threshold voltage, $E_{threshold}$, computation and communication tasks are carried out in full on in part, depending on the availability of energy. For both systems, energy/power measurements are fed back to the run-time energy transparency models for more accurate power budget allocations, and run-time management.

process that requires saving the instruction and data states in special purpose registers and memory units. The check-pointing process is governed by a special software routine that is triggered on-demand when the system encounters any known hardware/software anomalies.

In real-power computing systems, traditional approaches of check-pointing can prove challenging due to the following two reasons. First, the requirement to retain data can be aperiodic and on-demand, and second, the typical latency of check-pointing can result in diminish returns in terms of energy efficiency and performance [46].

A promising approach to integrating dynamic retention is deeply embedding non-volatile logic or storage registers in the electronic system. This will require additional survivability controls and run-time adaptability features as described next.

2) Run-time Adaptability

Real-power systems are expected to operate autonomously. Hence, they must be capable of adapting to changes in the

real world. This will include learning to predict the changes in power supply conditions and thereby reviewing power budget formulations, managing around unreliable situations in harsh operating environments for safety-critical applications, and ensuring energy frugality through dynamic power scheduling policies. To incorporate these capabilities, these systems will need to have a new types of knobs and monitors. At the power supply unit monitors will need to be designed to accurately capture the available energy levels (as shown in Fig. 11). Computation/communication resources must also be designed with power and performance monitors to enable feedback based run-time controls. Additionally, these resources will need to appropriately instrumented with retention control knobs to enable deep survivability [47].

V. Case Studies

Following the definitions of real-power systems with design- and run-time aspects (Sections III and IV), in this section, we present three different case studies as exemplars to demonstrate our research in the development of real-power systems. Case Study 1 represents a soft real-power system, which aims to maximize computation under variable power supply levels ignoring the impact of delays. Case Study 2 shows an example of a hard real-power system, demonstrating how resources can be dynamically proportional to incoming power. Case Study 3 presents the example of a tunable delay element with survivable instinct with the aim of learning delay based on the incoming power and remembering the delay properties in the event of a power loss. These case studies should be considered as systematic developments in different directions of real-power systems proposed in this paper; however, the complete chain of hardware/software tools, techniques and automation remain subjects of further research (See Section VIII).

A. Case Study 1: Self-timed Soft Real-Power Micropipelines

Traditionally, concurrency has been used to improve computing performance and/or efficiency, but there had been limited studies to leverage concurrency under variable energy situations. In this case study, a self-timed micropipeline is designed, based on our work [23]. The aim is to maximize the amount of compute per energy unit through dynamically variable concurrency. The case study represents a soft real-power systems as the processing of the data tokens of a given computation functionality will be favorably completed in part or in full following a non-stringent power budget derived from the available energy.

Fig. 12 shows the architecture of a C-element based micropipeline. It is implemented using an unrolled configuration, forming a ring by connecting the last stage output back to the first stage input [48]. The latches (i.e. C elements) can be set or reset by S1 or S0 inputs as shown in Fig. 12(a). The data items, called *tokens*, are identified as "01" or "10" input to the pair of C elements shown in dashes, while "00" is considered as non-data. Due to time delay based events it is possible to have an old copy of the token, called a *bubble*. With more incoming *tokens* moving forward, the *bubbles* move backward. The maximum number of tokens that can be processed by an N-stage pipeline is (N-1) tokens in order to free one stage to hold a bubble. N and 0 tokens are deadlock states.

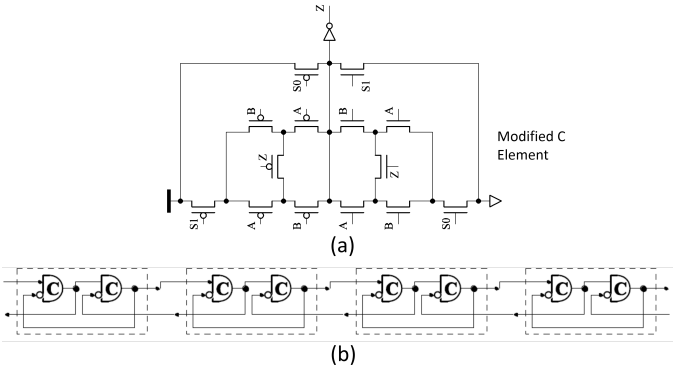


Fig. 12. (a) Modified asynchronous C element architecture with set/reset functions, and (b) four-stage micropipeline architecture using C elements.

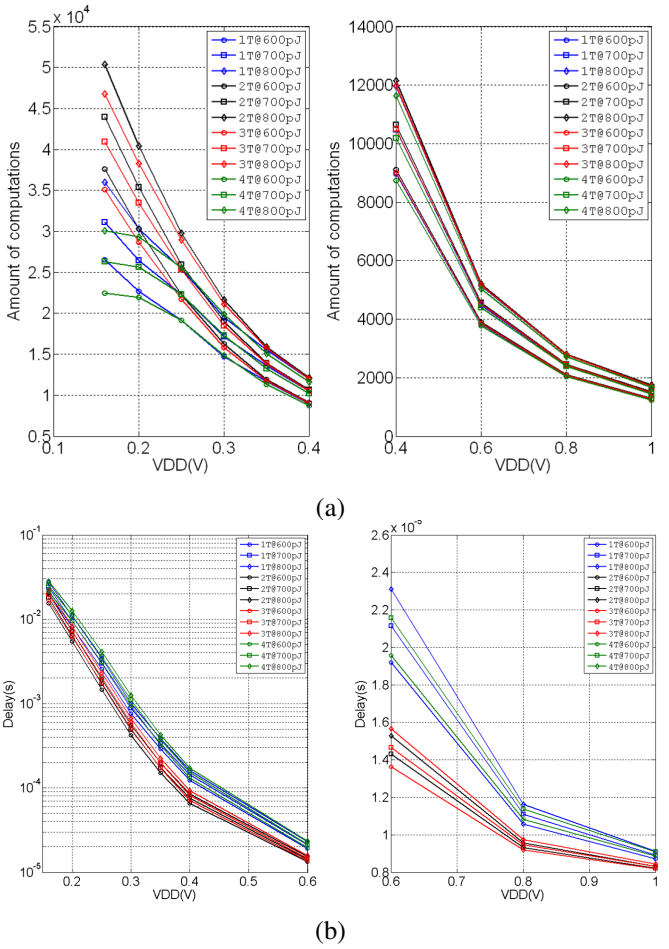


Fig. 13. (a) Units of computations carried out against different supply voltage levels (split in two levels for legibility), and (b) computation delays for these voltage levels (again split in two levels for legibility).

To validate the effectiveness of power-centric management of computation, a 5-stage ring micropipeline is used in experiments. Simulation results are obtained with different parallelism (1, 2, 3, 4 tokens), in different working voltages (1.0V, 0.8V, 0.6V, 0.4V, 0.35V, 0.25V, 0.2V, 0.16V) under various energy levels (600pJ, 700pJ, 800pJ). In each experiment, the power-compute run stops when the energy is fully consumed through power budgets (established through power-compute co-design steps prior to experiments). The resulting amount of computation is counted for each run in terms of a unit, defined as one pulse generated in the pipeline. Fig. 13 shows

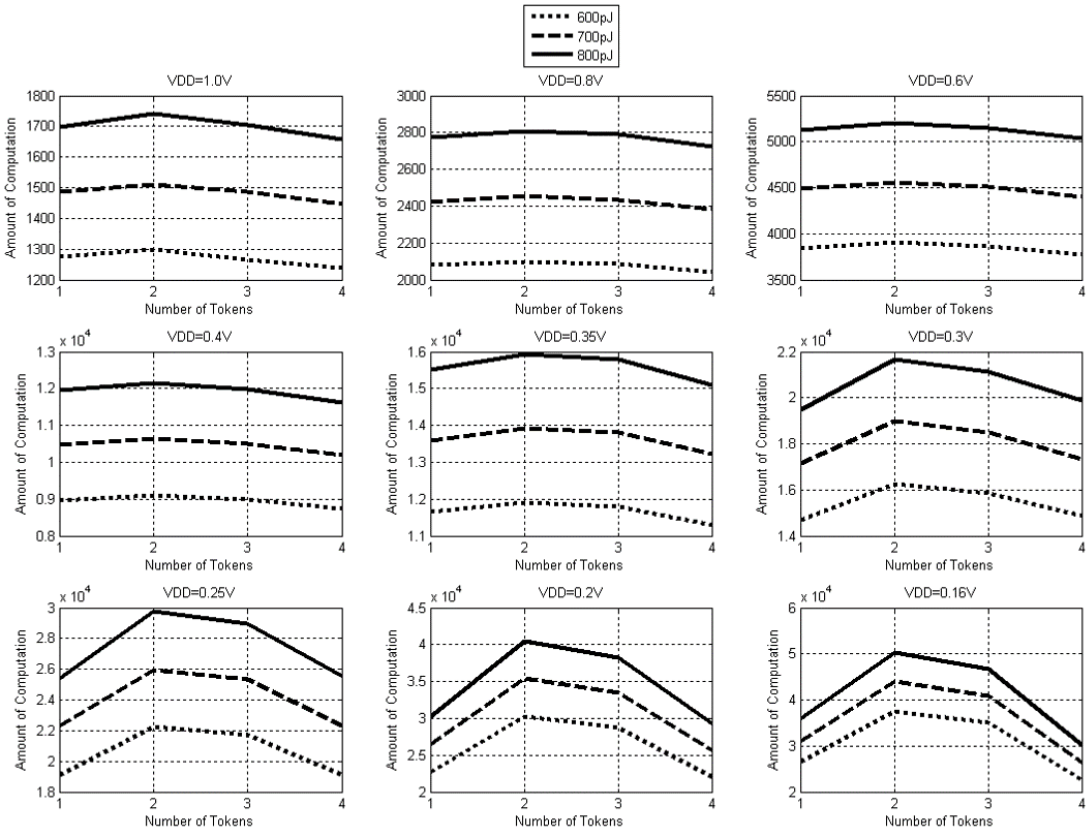


Fig. 14. Computations versus concurrency at different supply voltages

the different voltage levels and the resulting computation units and associated delays from the experiments.

The micropipeline (Fig. 12) concurrency adapts to the incoming energy levels and the data availability. For example, under 600pJ @1V with one data item, the power budget was adjusted to ensure 1276 computations (Fig. 13(a)). However, when two data become available for the same energy and voltage, the power budget and pipeline are set to deliver the highest concurrency (1299 computations), which is a small increase. However the time is more than halved. For the same amount of energy, when V_{dd} drops from 1V to 0.8V, the amount of computations for the most concurrent case is increased 61.7%. It takes longer, about 50% more for the same amount of computation. The power budget is 387uW at 0.8V compared to 968uW at 1V, about a 60% lower. When further reducing V_{dd} to 0.2V, exhausting the same amount energy, the amount of computation is increased 23.3 times and it takes about 10,000 times more time. For the same amount of computation, it takes about 380x more time compared to working at the nominal V_{dd}. However, the power figure goes from 968uW down to 110nW. Fig. 13(b) shows the corresponding delay for different computation voltages.

Fig. 14 shows computations versus the degree of concurrency at different V_{dd}s with 1-4 tokens. It can be concluded that the maximum number of computations happens at the same condition of the optimum throughput, which is at N/2 tokens when N is even or (N-1)/2 tokens when N is odd; the higher the concurrency the greater the amount of computation. At the nominal voltage, moving from the lowest extreme (four tokens) to the optimum point (two tokens) results in a 5% improvement in terms of the amount of computations per unit energy. But at a sub-threshold voltage the effect on the computation from 4 to 2 tokens is much more considerable - nearly 1.7 times. Theoretically, at a fixed V_{dd}, under the same amount energy, the optimum case will shorten the execution time to half. Across a shorter period of time the amount of leakage loss will be lower, hence the improvement. The results (Fig. 14) further suggest that above threshold voltage, the amount of computation per given amount of energy is practically insensitive to the degree of concurrency, but below threshold the dependency on the degree of concurrency and thereby also the energy efficiency goes up significantly due to the dynamic power adaptation in this case study.

B. Case Study 2: Hard Real-Power Signal Processing

This case study represents a hard real-power computing system with the aim of delivering the best possible computation functionality proportional to the input power. The design and experiments assumed no energy storage (the difference from our original definition in Section II-A is in the absence of coordinated dynamic retention circuitry, which is our ongoing research). Fig. 15 shows the simplified block diagram of the proposed computing using signal processing as an exemplar. As shown, the incoming harvested power is fed into the target logic circuit through a voltage protection and conditioning circuit (our ongoing works include using self-powered voltage sensors [49]). The logic block consists of power measurement subsystem (using shunt resistance network), followed by three different DC-DC converters to ensure variable voltage-current requirements into the main logic circuit (i.e. signal convolution

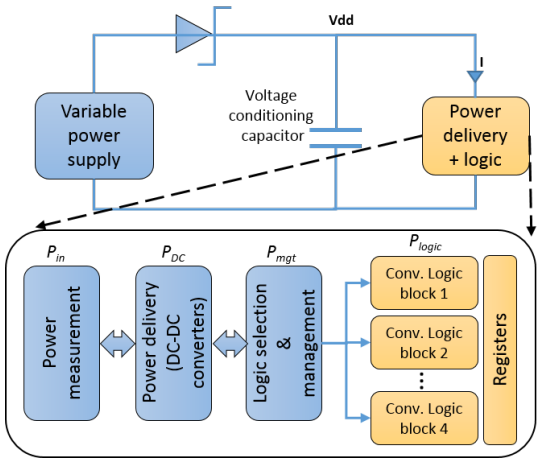


Fig. 15. Power-proportional convolution circuitry

circuit, which is commonly used for filtering information from raw signals). Based on the incoming power, one of the four convolution logic blocks with suitable approximation is chosen such that incoming power budget is not violated, while also maximizing the quality of computational functionality.

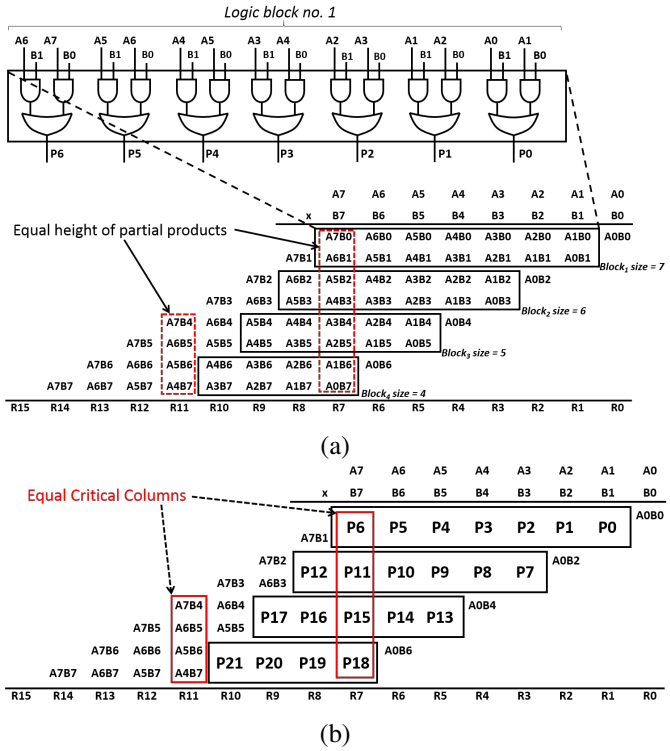


Fig. 16. (a) An 2-level SLDC multiplier showing approximate compaction of the partial product terms, and (b) the partial product terms after commutative remapping, showing reduction in the critical path.

The circuit in Fig. 15 was designed with simulated power scavenging with the assumption that the incoming power does not vary faster than the period of synchronous logic clock. Since multipliers constitute the bulk proportion of the convolution logic blocks, they were designed with 16-bit multipliers of four different approximations: precise Wallace-Tree Multiplier (WTM), approximate multiplier with 2-, 3- and 4-level significance-driven logic compression (SDLC) [50]. All four convolution configurations used precise carry-propagation adders organized in array of multiply-accumulate (MAC) units.

Fig. 16 shows the stylized block diagram of a 2-level SDLC multiplier. For demonstration purposes an 8-bit multiplier is shown. As can be seen, the basic premise of such an approximate multiplier is to reduce the number of partial product terms, and effectively shortening the critical path. For example, instead of producing two product terms $A1B0$ and $A0B1$, the multiplier directly produces an approximate sum term of the two (Fig. 16(a)). The sum term is generated by using an OR gate in place of an XOR gate, which only produces an incorrect output of 1, when both inputs are 1 (as compared to 0 in an XOR gate). The grouping of partial product terms is done using progressing bit significance, from lower to higher. The product terms are then commutatively remapped to reduce the number of partial product terms in the critical path to 4, instead of 8 (Fig. 16(b)). As a result, substantial energy reduction is achieved at the cost of low loss in accuracy. For higher level SDLC multipliers, further reductions in the critical path can drastically cut down the latency and power consumption with more imprecisions are incorporated in the process, representing a trade-off between power-energy-quality (PEQ).

Table I shows the synthesized power, delay, area and power-delay-product (PDP) comparisons of the different MAC units used in the four convolution circuits. As can be seen, the accurate MAC (row 2) has significantly higher power consumption and delay compared with the approximate MAC implementations (rows 3-5). As the logic compression level is increased from 2-level (2L) to 3-level (3L) and 4-level (4L), the critical path is incrementally cut down in favor of reduced dynamic and leakage power, coupled with latency. As a result, up to 5.5x energy efficiency (expressed in terms of PDP) can be achieved in the case of 4L SDLC MAC. Note that the energy reductions are achieved at the cost of reduced quality.

TABLE I
POWER, DELAY AND AREA COMPARISONS OF DIFFERENT LOGIC COMPONENTS

Circuit	P_{dyn} (uW)	P_{leak} (uW)	Delay (ns)	Area (um^2)	PDP (fJ)
Accurate MAC	58.19	4.23	2.63	1417.47	174.27
2L SDLC MAC	36.24	2.97	2.11	904.56	76.86
3L SDLC MAC	28.90	2.40	1.73	672.31	49.66
4L SDLC MAC	23.41	2.01	1.35	501.37	32.32

Fig. 17 presents the simulation results of an example application using the convolution circuitry in Fig. 15. In a hard real-power scenario, the system requires the signal convolution tasks to be completed in units of 2700 MACs (300-sample packets being convoluted by a signal with 9 samples). No deadline is imposed for the number of packets to be processed over a given time in this example. Fig. 17(a) depicts the input power (P_{in}) scavenged using a simulated source, together with the effective logic power (P_{logic}) for consecutive time intervals of $50ms$ each. The logic selection and management subsystem (Fig. 15) estimates P_{budget} for the convolution circuit discounting the power losses in power delivery (P_{DC}).

The P_{logic} determines which logic mode can be operated to ensure each signal packet is processed with the available power, while also providing with the best possible quality of outcomes. The excess power (P_{loss}) is bypassed through a RC network parallel to the logic block (not shown). As can be seen, when higher power is available initially, the accurate

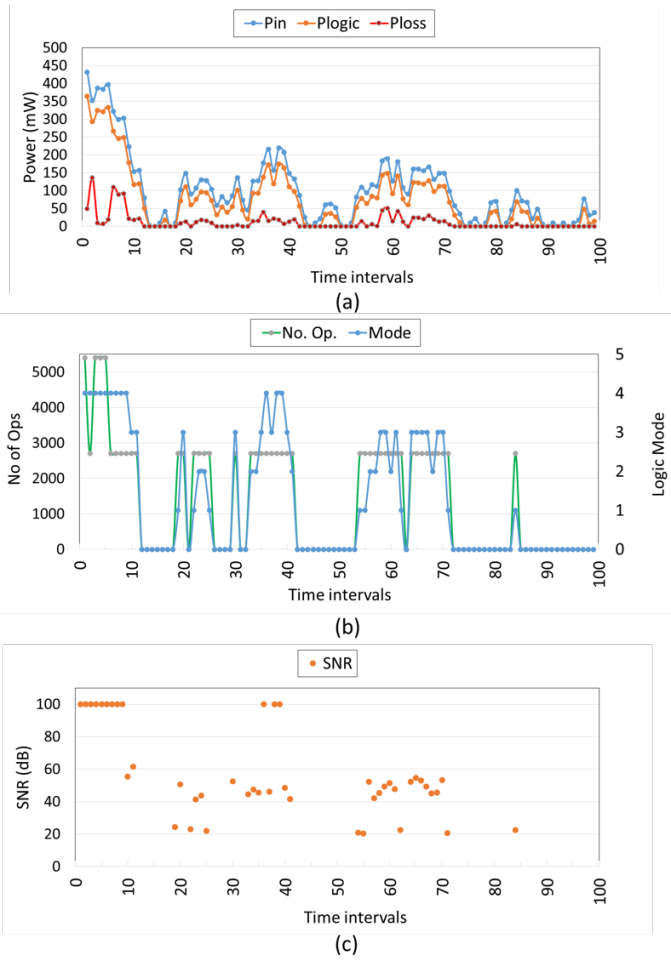


Fig. 17. (a) Input power (P_{in}) and effective logic power (P_{logic}), together with the logic bypass losses (P_{loss}), (b) logic mode selection [0: none selected, 1: 4L SDLC MACs, 2: 3L SDLC MACs, 3: 2L SDLC MACs, and 4: accurate MACs] and the total number of MAC operations performed for the given real-power budget for logic subsystem in (a), and (c) Valid signal-to-noise (SNR) points for the selected convolution tasks.

MACs are selected for the convolution task, allowing up to 2 packets being processed (Fig. 17(b)). However, as scavenged power becomes scarce, lower logic modes (such as 2L, 3L or 4L SDLC MACs) are selected. When the available power is low for processing a packet, the computation is skipped to the next interval. The selection of logic modes has a direct impact on the quality of the outcomes as shown in Fig. 17(c). Processing convolution using 4L SDLC MACs causes the SNR to degrade to as low as 19dB. Note that the effective SNR for a given packet can still vary despite having the similar logic mode selection. This is because the error introduced by the approximate logic block is dependent on the signal values. Signals with higher numeric values (i.e. higher '1's in the significant parts of the logic) can incur marginally higher errors than those with less numeric values due to progressive bit significance-driven logic compression (Fig. 16). Overall the SLDC-based switching of the circuit reacting to instantaneous power levels provides ≈60X better convolution functionality (from 2.7k MACs to 150k MACs) when compared with the accurate convolution circuit alone (observed over 10 signal processing experiments of 30-min each).

C. Case Study 3: Delay Elements with Survival Instincts

Supply voltage variations not only pose functionality challenges, but also introduce challenges in retaining parametric values of the circuit, such as buffer delays [51]. Configurable delay matching approaches, such as [52], [53], have been proposed by researchers to address delay variation challenges. The basic premise is to provide programmable delays that can match the propagation delay of the original circuits [54].

Designing delay elements with deeply embedded survival instincts is key to ensuring continued computational functionality in real-power systems. The instinct must allow for learning of the circuit delay properties adjusted to different voltage levels, and remember these properties when there is no current or voltage. In this case study, we briefly describe the design of a pulse controlled tunable delay element that has the above desirable attributes, based on our ongoing research in [55]. Fundamental to this tunable delay are two key modifications to existing buffers. These are: a) placing a memristor element between two CMOS inverters, and b) introducing a tunable interface for different supply voltages.

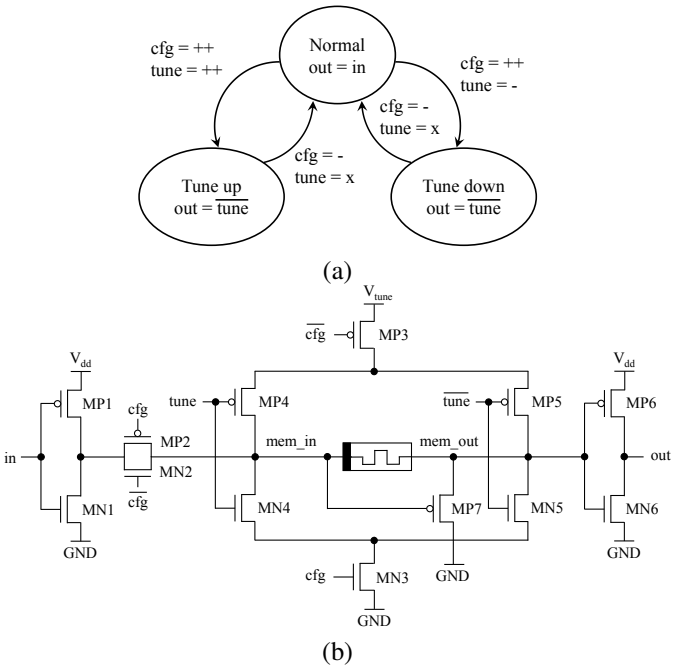
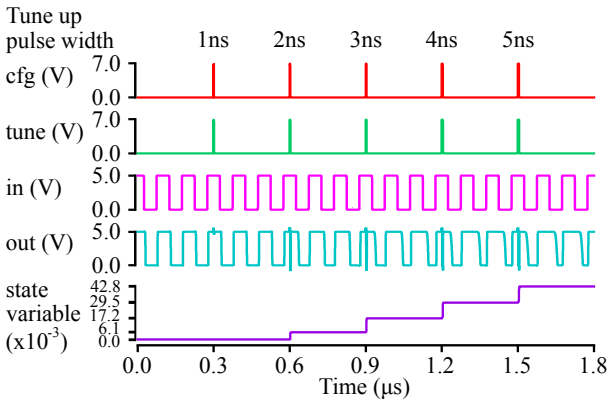


Fig. 18. (a) State diagram for switching the operating mode of the memristor-based delay element, and (b) circuit schematic for the pulse controlled memristor-based delay element.

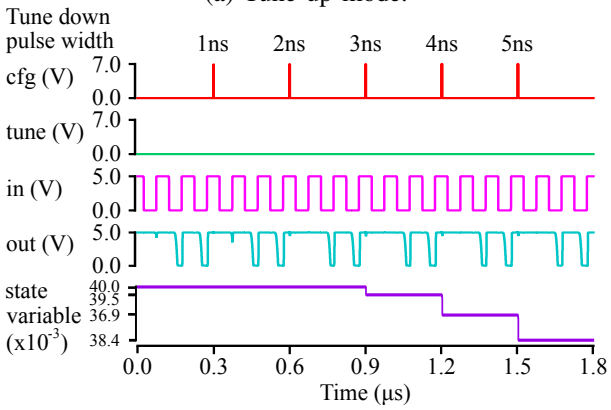
Fig. 18 shows (a) the state diagram of the tunable delay element, together with (b) its schematic. As can be seen, in the normal mode the cfg is "$-$" and turns on both MP2 and MN2, which form a pass gate, to pass the normal signal from the input buffer to the memristor and then to the output buffer. The cfg also turns off both MP3 and MN3 to cut the tuning network from V_{tune} and ground whether the $tune$ is "$++$" or "$-$". In the tune up mode, the cfg switches to "$++$" and turns off the pass gate while both MP3 and MN3 are turned on. At the same time, the $tune$ is "$++$" and turns on both MP5 and MN4. This connects mem_out and mem_in to V_{tune} and ground respectively and causes the state variable to go higher. On the other hand, in tune down mode, the cfg is "$++$" which turns on both MP3 and MN3 while the $turn$ changes to "$-$"

and turns on MP4 and MN5. This also connects the memristor to V_{tune} and ground but in the opposite direction and causes the state variable to go lower.

The transistor MP7, whose the gate and source are connected to mem_in and mem_out respectively, is used to deal with the backward tuning issue commonly seen in existing solutions [56]. In addition, the pass gate is necessary to block the leakage current that flows from V_{tune} to V_{dd} via the body of MP1 which occurs in both tuning operations. Note that the memristor can be placed in both directions depending on the thresholds. The side that has the threshold above the normal signal amplitude must be attached at mem_in to avoid the memristance change. However, the memristor can be placed in any directions if both of its thresholds are greater than the mentioned amplitude. All transistors except MP7 are sized to balance the rise and fall times. For high voltage analog and mixed signal (AMS) CMOS 0.35μm technology, the proper W_p/W_n ratio is 1. Therefore, the widths of 40μm are selected for MP1-MP5 and MN1-MN5 while both MP6 and MN6 are sized as 20μm.



(a) Tune up mode.



(b) Tune down mode.

Fig. 19. Identification of the minimum tuning pulse width in (a) tune up, and in (b) tune down modes.

Using the tunable delay circuit (Fig. 18(b)) extensive experiments were conducted using high voltage CMOS 0.35μm technology and memristor model with Biolek window function [57]. The ferroelectric fitting parameter for the memristor was chosen because of its wide memristance range and ON threshold that fits with the operating voltage. The V_{tune} was set above the highest threshold of ≈7V while V_{dd} was set as ≈5V to let the transistors operate correctly. Ten identical devices were connected in parallel resulting in high 15KΩ and 5MΩ as the actual minimum and maximum memristance respectively. The normal signal frequency in all experiments was set to 10MHz.

Experiments were conducted initially to identify the maximum effective memristance and the maximum delay to determine their upper limits. The maximum memristance was observed by applying tuning pulses on $3ns$ width. The average delay grows exponentially and saturates on the 6th pulse, indicating the maximum delay of 13.54ns and the state variable value of 40×10^{-3}. This value can be converted to the maximum effective memristance of 214KΩ. The minimum and average delay are obtained as 5.48ns and 1.34ns per step.

To identify minimum and average timing pulse widths for the delay element, next simulations were run by sweeping the tuning pulse width in tune up mode from 1ns to 5ns with 1ns increment per step. Fig. 19 shows the waveforms, which indicate that the state variable starts to increase at the 2ns pulse width. In tune down mode the state variable was initialized to the maximum effective memristance, determined earlier. The results also reveal the minimum pulse width settles to 3ns. The variation comes from the difference in memristances: the lower one allows the signal to swing faster. Hence, 3ns pulse width was assigned as the minimum tuning pulse width for all the experiments.

Besides retention of delay properties, the introduction of memristor in the tunable delay circuit also ensured low static power of 14pW and dynamic power of 203μW.

VI. Related Works

Significant research works are being carried within the realm of real-power computing. These works have found different terminologies in literature, namely transient computing [58]–[60], power-/energy-neutral computing [61], power-/energy-proportional computing [18], [62], energy-modulated computing [42], ultra low-power computing [63] and normally-off computing [64], [65]. These terminologies continue to highlight specific issues surfacing around the overall need to design new breed of computing systems that can go beyond the state-of-the-art in terms of energy efficiency and survivability [66]. In the following we give a brief account of the relevant works to date.

The term *transient computing*, proposed by Gomez *et al.* [60], refers to opportunistic computing for energy harvesting systems. The aim is to ensure computation and communication tasks can be carried out based on the available energy in the battery or the supercapacitor. Faster wake up and reaction times are critical for transient computing as these allow for better predictions of harvested energy availability [59]. To enable computation at challenging energy levels, the ability to operate at ultra low-power is also of profound importance [63]. An ultra low power micro-controller architecture, named PULP (URL: http://www.pulp-platform.org), is proposed by Conti *et al.* in [67]. The PULP processor has built-in parallelization features, and can interact with a power controller to react to energy critical situations. Additionally, it supports checkpointing, coupled with run-time routines, to retain the stable state of the computation [68], [69].

Supercapacitors or batteries pose pollution, sustainability and design geometry issues. Hence, operating at instantaneous power levels is often desirable, particularly for systems that need to operate autonomously without supercapacitors or batteries. Underpinning this motivation, Balsamo *et al.* proposed a *power-neutral computing* in [61]. The computation tasks in this system are instantaneously adapted based on available power using dynamic frequency scaling (DFS). For an uninterrupted operation, the system also needs to have state retention feature enabled by on-demand check-pointing [70]. A variant of power-neutral computing is energy-neutral computing, which assumes the presence of supercapacitors or small batteries with limited capacity. Similar to transient computing, energy-neutral computing needs power controllers that closely interact with these supercapacitors or batteries to ensure uninterrupted computation under varying energy situations.

Energy- or power-proportional computing stems from similar principles of energy-neutral or power-neutral computing. However, the main impetus of such computing is the transparency of energy usage profile of every system component, particularly the memory and input/output (IO) subsystems [18], [42]. The aim is to achieve tighter control over the energy consumption of hardware/software systems when subjected to different workloads. Liqat *et al.* and Kerrison *et al.* proposed software energy modeling and verification approach using execution statistics together with instruction set architecture (ISA) in [32], [71], showing minor deviation with from hardware energy measurements. The model elaborated the impact of different instructions on the hardware components, including processors, arithmetic/logic units, memories and pipelines for multi-threaded XMOS-based embedded systems. Flinn and Satyanarayanan proposed another modeling tool in [72], called Powerscape, which can combine execution statistics and hardware instrumentation to generate a detailed energy usage footprint. Tools like these can reason for better hardware/software energy efficiency using a number of different approaches. For example, a compiler based approach for optimized register cache sizing for modern superscalar processors is proposed in [73]. Based on the energy/performance profiles obtained from ISA, expressed as a energy-delay product (EDP), and using the detailed characterization of cache associativity, the authors demonstrate how EDP can be improved with energy-proportional considerations.

Energy-modulated computing, recently proposed in [7], [42], extends the above concepts further by adding elasticity in computing. The general argument is that computation must continue to provide intended functionality of its equivalent even when energy is scarce, particularly in energy harvesting systems. The elasticity in computation is achieved through two aspects. Firstly, the system must have a layered design with heterogeneous computing units to enable control over the quality of intended computing functionality. When more energy is available, the layer with high-complexity and high-accuracy hardware/software resources will be active. Conversely, when energy is scarce, the layer of low-complexity, energy-efficient and less-accurate resources will be active, powering off the other layers [74] (see Fig. 8). A key aspect of achieving full control in energy-modulated computing is approximate computing system design that can operate with variable precision based on the data or logic significance [50].

Recently *normally-off computing* has been proposed by Nakamura *et al.* [64], [65]. The main principle is to design a new generation of computing system with faster non-volatile memories. Integration of these memories enable aggressive shutting down the computing components (as opposed to power gating using of leaky switches) when energy/power is low. The power management features are incorporated at micro-architectural level, providing the system with survivability features (as highlighted in Section IV). Table II shows a brief summary of the existing works relevant to real-power computing, with classifications based on the taxonomies shown earlier in Section III.

TABLE II
CLASSIFICATION OF EXISTING APPROACHES WITHIN THE REMITS OF REAL-POWER COMPUTING

Approach	Soft RPC	Hard RPC	Instincts & features
[18], [42]	✓		power-adaptive approximation
[23]	✓		energy-adaptive, no retention
[60]	✓		Supercapacitors without retention
[61]		✓	no retention, power-adaptive
[64], [65]		✓	non-volatile computing
[70]	✓		retention through check-pointing
[68], [69]	✓		ultra low-power check-pointing

We are already seeing the penetration of survivability-based systems design in industrial products, increasingly following the real-power approach. Two relevant examples are: a) modern smartphones, in which services are reduced progressively when the battery is low or unreliable [75], and b) autonomous drones or vehicles, in which reduced flying/driving capabilities would still be required to ensure safe landing/parking when the power supply is about to fail [76]. Although these products do not have autonomous survival instincts reacting to instantaneous power supply variations, real-power approach is expected to make this possible.

VII. CHALLENGES AND OPPORTUNITIES

Despite progress in different aspects of real-power computing (Section VI), the full-scale design and implementation of real-power computing systems will need holistic and concerted efforts across the entire system stack: from hardware to software. We envisage the following key research challenges and opportunities covering different aspects:

A. Design Automation Tools

Power-centric controls and management introduced in real-power computing is clearly a departure from the existing power agnostic controls. As such, existing electronic design automation (EDA) tools will not be able to meet the needs of the new paradigm, largely due to the reversal of the controls. Hence, new EDA tools and techniques will need to be developed to facilitate power-compute co-design, validation and verification. These tools will use power-compute co-design policies to generate power controllers and layers of hardware/software resources.

B. Instrumentation and Run-time Optimization

To orchestrate run-time adaptability and control features, careful instrumentation of knobs and monitors is a key requirement. Existing design principles of power/performance knobs/ monitors can be used. However, new knobs and monitors need to be developed to build systems with survival instincts. These can be used in conjunction with light-weight run-time optimization routines for feedback based reactive control [66], [77], or run-time model based proactive controls using machine learning principles [78], [79].

C. Architectures for Survivability

The concept of power-proportional computing introduced in Section IV-A requires development of new system architectures for real-power computing. These architectures are expected to feature heterogeneous computing resources, including traditional (CPUs, DSPs, FPGAs/ASICs) and emerging circuits and systems, such as programmable approximate computing units [50]. Based on the available energy, the computation and communication resource(s) would be suitably chosen to provide required functionality or its gracefully degraded equivalent. To allow for run-time survivability, the architectures will also need to integrate hardware/software support (such as non-volatile registers or tightly-coupled memories) for dynamic retention capabilities.

D. Programming Model

For high-level scheduling between power supply and compute units, new programming models will need to be developed. These models will consist of a set of annotations and run-time routines. Annotations will dictate the power budgets of modular tasks in heterogeneous computing resources, either statically (for hard real-power systems) or dynamically (for soft real-power systems), while run-time routines will manage the system (and its survivability) based around the given power budgets. Interacting with knobs and monitors will be exposed to the run-time through application programming interfaces.

VIII. Conclusions

We are entering an era of massively ubiquitous computing (e.g. swarms of devices such as sensors, monitors, actuators, markers, smart tags) at the smallest possible granularity level. The quantity of devices that form such swarms will be in the order of trillions with inherent requirement of autonomous survival capabilities. Cumulatively, they are expected to consume enormous amounts of power, which cannot be expected with current and predicted battery technology scaling. Existing low power design methods largely use performance-constrained power/energy minimization without considering the survivability under energy/power variabilities. Indeed, for these promising devices we need to design and build systems that can operate uninterruptedly under a wide range of power constraints. Underpinning these motivations, we defined and proposed a new computing paradigm, named *Real-Power Computing*. Our definitions have been complemented with different case studies and exemplars, coupled with reflections and experiences from existing research efforts in the form of power- or energy-aware design. Although this new paradigm has direct relevance to current and future generations of ubiquitous systems, we believe that there is a strong impetus for this paradigm to be useful in other computing applications for cost and energy-efficiency considerations.

We have considered an electronic design approach inspired by the survivability instinct seen in many natural phenomena, e.g. microbes, to ensure continued functionality under such unreliable energy. We have also shown in Case Study 1 that the new paradigm lends itself to the principle of using the available energy to the maximum amount of computation in systems. According to [80], "this principle of least action is an expression not of nature's parsimony, but of nature's prodigality: a system's natural trajectory is the one that will hog the most computational resources".

Our approach develops a holistic view and a research framework for the Real-Power paradigm covering challenges and opportunities of power schedulability, tools and algorithms for power-centric design, design- and run-time optimization and adaptation for ensuring continued execution of battery-less ubiquitous systems. We strongly believe the proposed paradigm will broaden the scopes for extensive future research with definitive pathways.

Acknowledgement

The authors would like to thank EPSRC PRiME project (EP/K034448/1) for supporting this work and appreciate the useful discussions with Geoff Merrett (Uni. Southampton), George Constantinides (Imperial College, London), Kerstin Eder (Uni. Bristol) and members of the Microsystems Research Group (Uni. Newcastle).

References

[1] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE TCAD*, vol. 18, no. 12, pp. 1702–1714, Dec 1999.

[2] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 195–237, Sep. 2005.

[3] M. Bauer, N. Bui, C. Jardak, and A. Nettstrter, "The IoT ARM reference manual," 12 2013.

[4] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, no. 4, pp. 6–15, 2011.

[5] T. Simunic, L. Benini, and G. De Micheli, "Energy-efficient design of battery-powered embedded systems," *IEEE Trans. on Very Large Scale Integration (TVLSI) Systems*, vol. 9, no. 1, pp. 15–28, 2001.

[6] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. Al Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *TVLSI*, vol. 15, no. 3, pp. 262–275, 2007.

[7] A. Yakovlev, "Energy-modulated computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2011, pp. 1–6.

[8] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. of the 41st annual Design Automation Conference (DAC)*. ACM, 2004, pp. 868–873.

[9] F. Xia, A. Rafiev, A. Aalsaud, M. Al-Hayanni, J. Davis, J. Levine, A. Mokhov, A. Romanovsky, R. Shafik, A. Yakovlev, and S. Yang, "Voltage, throughput, power, reliability, and multicore scaling," *Computer*, vol. 50, no. 8, pp. 34–45, 2017.

[10] K. Lyytinen and Y. Yoo, "Ubiquitous computing," *Communications of the ACM*, vol. 45, no. 12, pp. 63–96, 2002.

[11] R. Iyer and E. Ozer, "Visual iot: Architectural challenges and opportunities; toward a self-learning and energy-neutral iot," *IEEE Micro*, vol. 36, no. 6, pp. 45–49, 2016.

[12] S. P. Beeby, R. Torah, M. Tudor, P. Glynne-Jones, T. O'donnell, C. Saha, and S. Roy, "A micro electromagnetic generator for vibration energy harvesting," *J. of Micromechanics and microengineering*, vol. 17, no. 7, p. 1257, 2007.

[13] J. A. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," *IEEE Perv. Comp.*, vol. 4, no. 1, pp. 18–27, 2005.

[14] S. Chalasani and J. M. Conrad, "A survey of energy harvesting sources for embedded systems," in *SouthEastCon*. IEEE, 2008, pp. 442–447.

[15] L. Mateu and F. Moll, "Review of energy harvesting techniques and applications for microelectronics (keynote address)," in *Microtechnologies for the New Millennium 2005*, 2005, pp. 359–373.

[16] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Proc. IPSN*. IEEE Press, 2005, p. 64.

[17] S. Haruta and N. Kanno, "Survivability of microbes in natural environments and their ecological impacts," *Microbes and environments*, vol. 30, no. 2, pp. 123–125, 2015.

[18] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec 2007.

[19] O. Heaviside, *Electrical papers*. Cambridge Uni. Press, 2011, vol. 2.

[20] M. Kleanthous, Y. Sazeides, E. Özer, C. Nicopoulos, P. Nikolaou, and Z. Hadjilambrou, "Toward multi-layer holistic evaluation of system designs," *IEEE Comp. Arch. Letters*, vol. 15, no. 1, pp. 58–61, 2016.

[21] H. Lhermet, C. Condemine, M. Plissonnier, R. Salot, P. Audebert, and M. Rosset, "Efficient power management circuit: From thermal energy harvesting to above-IC microbattery energy storage," *IEEE J. of solid-state circuits*, vol. 43, no. 1, pp. 246–255, 2008.

[22] R. A. Shafik, S. Yang, A. Das, L. A. Maeda-Nunez, G. V. Merrett, and B. M. Al-Hashimi, "Learning transfer-based adaptive energy minimization in embedded systems," *IEEE Trans. on Comp.-Aided Des. of Integ. Circuits and Systems (TCAD)*, vol. 35, no. 6, pp. 877–890, 2016.

[23] A. Baz, D. Shang, F. Xia, X. Gu, and A. Yakovlev, "Energy efficiency of micropipelines under wide dynamic supply voltages," in *2014 IEEE Faible Tension Faible Consommation*, May 2014, pp. 1–4.

[24] A. S. Weddell, M. Magno, G. V. Merrett, D. Brunelli, B. M. Al-Hashimi, and L. Benini, "A survey of multi-source energy harvesting systems," in *DATE*. EDA Consortium, 2013, pp. 905–908.

[25] Y. Li and J. Henkel, "A framework for estimating and minimizing energy dissipation of embedded hw/sw systems," in *DAC*, 1998, pp. 188–193.

[26] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 4, Sep. 2007.

[27] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, "Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems," in *Sensor Mesh and Ad Hoc Communications and Networks (SECON), IEEE Comms. Society Conf. on*. IEEE, 2010, pp. 1–9.

[28] R. J. Vullers, R. Van Schaijk, H. J. Visser, J. Penders, and C. Van Hoof, "Energy harvesting for autonomous wireless sensor networks," *IEEE Solid-State Circuits Magazine*, vol. 2, no. 2, pp. 29–38, 2010.

[29] D. Zhu, S. Roberts, M. J. Tudor, and S. P. Beeby, "Design and experimental characterization of a tunable vibration-based electromagnetic micro-generator," *Sensors and Actuators A: Physical*, vol. 158, no. 2, pp. 284–293, 2010.

[30] P. D. Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, "Energy harvesting from human and machine motion for wireless electronic devices," *Proc. IEEE*, vol. 96, no. 9, pp. 1457–1486, 2008.

[31] D. C. Hoang, Y. K. Tan, H. B. Chng, and S. K. Panda, "Thermal energy harvesting from human warmth for wireless body area network in medical healthcare system," in *IEEE PEDS*, 2009, pp. 1277–1282.

[32] U. Liqat, S. Kerrison, A. Serrano, K. Georgiou, P. Lopez-Garcia, N. Grech, M. V. Hermenegildo, and K. Eder, "Energy consumption analysis of programs based on xmos isa-level models," in *Intl. Sym. on Logic-Based Program Synth. & Transf.*, 2013, pp. 72–90.

[33] B. H. Cheng, K. I. Eder, M. Gogolla, L. Grunske, M. Litoiu, H. A. Müller, P. Pelliccione, A. Perini, N. A. Qureshi, B. Rumpe *et al.*, "Using models at runtime to address assurance for self-adaptive systems," in *Models@ run. time*. Springer, 2014, pp. 101–136.

[34] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded cpus," *TCAD*, vol. 36, no. 1, pp. 106–119, Jan 2017.

[35] S. Tozlu, M. Senel, W. Mao, and A. Keshavarzian, "Wi-Fi enabled sensors for Internet of Things: A practical approach," *IEEE Communications Magazine*, vol. 50, no. 6, 2012.

[36] F. Xia, A. V. Yakovlev, I. G. Clark, and D. Shang, "Data communication in systems with heterogeneous timing," *IEEE Micro*, vol. 22, no. 6, pp. 58–69, 2002.

[37] Y.-K. Chen, "Challenges and opportunities of Internet of Things," in *Design Automation Conference, 2012 17th Asia and South Pacific (ASP-DAC)*. IEEE, 2012, pp. 383–388.

[38] R. A. Shafik, B. M. Al-Hashimi, and K. Chakrabarty, "Soft error-aware design optimization of low power and time-constrained embedded systems," in *DATE*, 2010, pp. 1462–1467.

[39] A. Yakovlev, *Enabling Survival Instincts in Electronic Sys...*, 2015, ch. 13, pp. 237–263.

[40] Y. Cai, "Survivability," in *Instinctive Computing*. Springer, 2016, pp. 353–371.

[41] M. Magno, D. Boyle, D. Brunelli, E. Popovici, and L. Benini, "Ensuring survivability of resource-intensive sensor networks through ultra-low power overlays," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 2, pp. 946–956, 2014.

[42] R. Ramezani, D. Sokolov, F. Xia, and A. Yakovlev, "Energy-modulated quality of service: New scheduling approach," in *Faible Tension Faible Consommation (FTFC), 2012 IEEE*. IEEE, 2012, pp. 1–4.

[43] Q. Liu, T. Mak, J. Luo, W. Luk, and A. Yakovlev, "Power adaptive computing system design in energy harvesting environment," in *Embedded Computer Systems (SAMOS), 2011 International Conference on*. IEEE, 2011, pp. 33–40.

[44] E. Beigne, P. Vivet, Y. Thonnart, J.-F. Christmann, and F. Clermidy, "Asynchronous circuit designs for the Internet of Everything: A methodology for ultralow-power circuits with gals architecture," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 4, pp. 39–47, 2016.

[45] D. Sokolov and A. Yakovlev, "Quality of service in power proportional computing," Citeseer, Tech. Rep., 2011.

[46] D. Chabi, W. Zhao, E. Deng, Y. Zhang, N. B. Romdhane, J.-O. Klein, and C. Chappert, "Ultra low power magnetic flip-flop based on checkpointing/power gating and self-enable mechanisms," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1755–1765, 2014.

[47] R. Ramezani, A. Yakovlev, F. Xia, J. P. Murphy, and D. Shang, "Voltage sensing using an asynchronous charge-to-digital converter for energy-autonomous environments," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 3, pp. 35–44, 2013.

[48] T. E. Williams, "Self-timed rings and their application to division," Ph.D. dissertation, Stanford, CA, USA, 1991, uMI Order No. GAX92-05744.

[49] X. Zhang, D. Shang, F. Xia, and A. Yakovlev, "A novel power delivery method for asynchronous loads in energy harvesting systems," *J. Emerg. Technol. Comput. Syst.*, vol. 7, no. 4, pp. 16:1–16:22, Dec. 2011.

[50] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *DATE*, March 2017, pp. 7–12.

[51] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Proc. of MICRO-36.*, Dec 2003, pp. 7–18.

[52] M. Maymandi-Nejad and M. Sachdev, "A monotonic digitally controlled delay element," *IEEE J. of Solid-State Circuits*, vol. 40, no. 11, pp. 2212–2219, Nov 2005.

[53] J. Christiansen, "An integrated high resolution cmos timing generator based on an array of delay locked loops," *IEEE J. of Solid-State Circuits*, vol. 31, no. 7, pp. 952–957, Jul 1996.

[54] J.-S. Chiang and K.-Y. Chen, "The design of an all-digital phase-locked loop with small dco hardware and fast phase lock," *IEEE Trans. on Circuits and Systems (TCAS) II: Analog and Digital Signal Processing*, vol. 46, no. 7, pp. 945–950, Jul 1999.

[55] T. Bunnam, A. Soltan, D. Sokolov, and A. Yakovlev, "Pulse controlled memristor-based delay element," in *PATMOS*, Sept 2017, p. (in press).

[56] X. Zhang, Z. Ma, J. Yu, and L. Xie, "Memristor-based programmable delay element," in *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Oct 2014, pp. 1–3.

[57] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "Vteam: A general model for voltage-controlled memristors," *IEEE TCAS II: Express Briefs*, vol. 62, no. 8, pp. 786–790, 2015.

[58] A. Rodriguez, D. Balsamo, A. Das, A. S. Weddell, D. Brunelli, B. Al-Hashimi, and G. V. Merrett, "Approaches to transient computing for energy harvesting systems..." in *ENSsys 2015*, 2015.

[59] D. Spenza, M. Magno, S. Basagni, L. Benini, M. Paoli, and C. Petrioli, "Beyond duty cycling: Wake-up radio with selective awakenings for long-lived wireless sensing systems," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 522–530.

[60] A. Gomez, L. Sigrist, M. Magno, L. Benini, and L. Thiele, "Dynamic energy burst scaling for transiently powered systems," in *DATE*, 2016, pp. 349–354.

[61] D. Balsamo, A. Das, A. S. Weddell, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini, "Graceful performance modulation for power-neutral transient computing systems," *IEEE TCAD*, vol. 35, no. 5, pp. 738–749, 2016.

[62] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic knobs for responsive power-aware computing," in *ACM SIGPLAN Notices*, vol. 46, no. 3. ACM, 2011, pp. 199–212.
[63] A. Abnous and J. Rabaey, "Ultra-low-power domain-specific multimedia processors," in *VLSI Signal Processing, IX, 1996.,[Workshop on]*. IEEE, 1996, pp. 461–470.
[64] T. Nakada and H. Nakamura, "Normally-off computing," in *Normally-Off Computing*. Springer, 2017, pp. 57–63.
[65] T. Nakada, T. Shimizu, and H. Nakamura, "Normally-off computing for iot systems," in *SoC Design Conference (ISOCC), 2015 International*. IEEE, 2015, pp. 147–148.
[66] J.-F. Christmann, E. Beigné, C. Condemine, J. Willemin, and C. Piguet, "Energy harvesting and power management for autonomous sensor nodes," in *DAC*. ACM, 2012, pp. 1049–1054.
[67] F. Conti, D. Rossi, A. Pullini, I. Loi, and L. Benini, "PULP: A ultra-low power parallel accelerator for energy-efficient and flexible embedded vision," *J. of Signal Proc. Sys.*, vol. 84, no. 3, pp. 339–354, 2016.
[68] G. Tagliavini, A. Marongiu, D. Rossi, and L. Benini, "Always-on motion detection with application-level error control on a near-threshold approximate computing platform," in *ICECS*. IEEE, 2016, pp. 552–555.
[69] M. Rusci, D. Rossi, M. Lecca, M. Gottardi, E. Farella, and L. Benini, "An event-driven ultra-low-power smart visual sensor," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5344–5353, 2016.
[70] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2015.
[71] S. Kerrison and K. Eder, "Energy modeling of software for a hardware multithreaded embedded microprocessor," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 14, no. 3, p. 56, 2015.
[72] J. Flinn and M. Satyanarayanan, "Powerscope: A tool for profiling the energy usage of mobile applications," in *Mobile Computing Systems and Applications, 1999. Proc.. WMCSA'99. Second IEEE Workshop on*. IEEE, 1999, pp. 2–10.
[73] T. M. Jones, M. F. O'Boyle, J. Abella, A. González, and O. Ergin, "Energy-efficient register caching with compiler assistance," *ACM TACO*, vol. 6, no. 4, p. 13, 2009.
[74] K. G. Larsen, S. Laursen, and M. Zimmermann, "Limit your consumption! finding bounds in average-energy games," in *Proc. of QAPL, The Netherlands, April 2-3, 2016.*, 2016, pp. 1–14.
[75] A. Elnashar and M. A. El-Saidny, "Extending the battery life of smartphones and tablets: A practical approach to optimizing the lte network," *IEEE Vehicular Technology Magazine*, vol. 9, no. 2, pp. 38–49, 2014.
[76] I. Schiller and J. S. Draper, "Mission adaptable autonomous vehicles," in *Neural Networks for Ocean Engineering, 1991., IEEE Conference on*. IEEE, 1991, pp. 143–150.
[77] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, "Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems," in *DAC*. ACM, 2014, pp. 1–6.
[78] S. Yang, R. A. Shafik, G. V. Merrett, E. Stott, J. M. Levine, J. Davis, and B. M. Al-Hashimi, "Adaptive energy minimization of embedded heterogeneous systems using regression-based learning," in *PATMOS*. IEEE, 2015, pp. 103–110.
[79] A. Das, A. Kumar, B. Veeravalli, R. Shafik, G. Merrett, and B. Al-Hashimi, "Workload uncertainty characterization and adaptive frequency scaling for energy minimization of embedded systems," in *DATE*. IEEE, 2015, pp. 43–48.
[80] T. Toffoli, *Action, or the fungibility of computation in* Feynman and Computation. Ed. Anthony Hey, Perseus Books, Cambridge, MA, 1999, ch. 21, pp. 349–392.

Rishad Shafik (MIET, MIEEE) is a Lecturer in Electronic Systems within the School of Engineering, Newcastle University, UK. Dr. Rishad received his Ph.D., and M.Sc. (with distinction) degrees from Southampton in 2010, and 2005; and B.Sc. (with distinction) from the IUT, Bangladesh in 2001. He is one of the editors of the book "Energy-efficient Fault-tolerant Systems," published by Springer USA. He is also author/co-author of 85+ IEEE/ACM journal and conference articles, with three best paper nominations. He has recently co-chaired 30th DFT2017 (www.dfts.org) at Cambridge, UK. His research interests include energy-efficiency and adaptability aspects of embedded computing systems.

Alex Yakovlev (FIET, FIEEE) is a Professor of Computer Engineering, who founded and leads the MicroSystems Research Group, and co-founded the Asynchronous Systems Laboratory at Newcastle University. He was awarded an EPSRC Dream Fellowship in 2011–13. He has published 8 edited and co-authored monographs and more than 300 papers in IEEE/ACM journals and conferences, in the area of concurrent and asynchronous systems, with several best paper awards and nominations. He has chaired organizational committees of major international conferences. He has been principal investigator on more than 30 research grants and supervised 40 PhD students. Most recently, he has been elected to the fellowship of Royal Academy of Engineering in the UK.

Shidhartha Das (MIET, MIEEE) is currently the Principal R&D Engineer at ARM, and the recipient of the ARM Inventor of the Year award in 2016. He received the B.Tech degree from the IIT, Bombay in 2002 and the M.S and Ph.D degrees from the University of Michigan, Ann Arbor in 2005 and 2009. His research interests include emerging non-volatile memory technologies, micro-architectural circuit and systems design. He is the recipient of multiple best paper awards; his research also featured in popular IEEE magazines. Dr. Das serves on the technical program committee of several leading international conferences.