Low Power Processor Architectures and Contemporary Techniques for Power Optimization – A Review

Muhammad Yasir Qadri, Hemal S Gujarathi and Klaus D. McDonald-Maier
School of Computer Science and Electronic Engineering, University of Essex, CO4 3SQ, UK
Email: yasirqadri@acm.org, hemalgujarathi@gmail.com, kdm@essex.ac.uk

***Abstract*—The technological evolution has increased the number of transistors for a given die area significantly and increased the switching speed from few MHz to GHz range. Such inversely proportional decline in size and boost in performance consequently demands shrinking of supply voltage and effective power dissipation in chips with millions of transistors. This has triggered substantial amount of research in power reduction techniques into almost every aspect of the chip and particularly the processor cores contained in the chip. This paper presents an overview of techniques for achieving the power efficiency mainly at the processor core level but also visits related domains such as buses and memories. There are various processor parameters and features such as supply voltage, clock frequency, cache and pipelining which can be optimized to reduce the power consumption of the processor. This paper discusses various ways in which these parameters can be optimized. Also, emerging power efficient processor architectures are overviewed and research activities are discussed which should help reader identify how these factors in a processor contribute to power consumption. Some of these concepts have been already established whereas others are still active research areas.**

***Index Terms*—Low power, processor architecture, power optimization techniques**

I. Introduction

The ever increasing range of battery operated devices with often complex functionality is a major catalyst for driving the research in the field of low power system design. Battery technology advances at a slower pace and hence it is logical to find the ways to optimize power usage in the hardware architectures. Particularly stand alone embedded systems are being optimized for the energy efficiency during their design. The rapidly evolving field of pervasive computing is major patron for energy efficient devices or technologies. Many advanced systems such as mobile phones and personal digital assistants (PDAs) have multiple processors for achieving the required computational speed or performance. In such cases achieving energy optimization becomes a mandatory requirement rather than a feature. Current research for low power spans the areas of semiconductor technologies, VLSI, processor architecture, complier design and operating systems.

The paper is divided mainly into five chapters: the introductory chapter is followed by a discussion of various techniques for power optimization of processor components and a brief introduction to alternative methods. Chapter 4 overviews various novel low power architectures, and in the final chapter a conclusion is drawn based on these techniques and architectures. In line with the review focus of this paper, the core attention is on the principle concepts and provides literature pointers that allow the reader to probe further.

A. Relation between energy and power

Performance drives the market for the processors [2] and the system advancement relies on increasing performance while retaining energy at an acceptable level. Since $Energy = Power\ x\ Time$, simply reducing the power consumption in a processor may not decrease the energy demand if the task now takes longer to execute. Richard et al. [3] noted a correlation between energy and performance. However, reducing performance does not always reduce the overall energy. With a potential objective of minimum power at a given performance level, or increased performance at a given power level, both quantities need to be considered concurrently. This can be achieved by taking the product of energy and delay [4].

The power consumption in a processor can be classified as 1) static and 2) dynamic. Firstly, the static power consumption of a device is defined as the amount of power that it consumes while there is no switching activity. If the operating voltage is assumed fixed then the static power consumption mainly comprises of leakage current, and is fabrication process dependent; its general optimization techniques are beyond the scope of this paper. Secondly, the dynamic power of a CMOS device comprises of short circuit power and switching power. The short circuit current is observed during input signal transitions, when both the NMOS and PMOS blocks in static CMOS circuits conduct simultaneously for a short period of time. This causes a direct current flow from power rails resulting in short-circuit power [5]. The switching power consumption is given as

$$P_{Switching} = C_L . V_{dd}^2 . a . f, \tag{1}$$

where
C_L = Load Capacitance,
V_{dd} = Drain to source voltage,
a = activity factor,
f = clock frequency.

Based on (1) it is possible to infer that dynamic power consumption can be decreased, by decreasing any one of the factors. Once again V_{dd} and C_L are fabrication technology dependent; however there are techniques such as assigning a lower V_{dd} to noncritical paths that could significantly reduce overall dynamic power consumption. However, lowering V_{dd} increases circuit delay and consequently reduces the overall throughput of device [6]. The circuit delay is given as [7],

$$Delay = k . \frac{V_{dd}}{(V_{dd} - V_{th})^2}, \tag{2}$$

where k is a constant that depends on capacitance, mobility, and transistor dimensions [8] and V_{th} is the threshold voltage. Principle processor core level power optimization techniques are discussed in the following.

B. *Dynamic Voltage-Frequency Scaling (DVFS)*

Dynamic Voltage and Frequency Scaling (DVFS) is an effective technique to attain low power consumption while meeting the performance requirements. Energy dissipation is reduced by dynamically scaling the supply voltage of the CPU, so that it operates at a minimum speed required by the specific task executed (i.e. maintaining real-timeliness) [9].

The DVFS technique principally involves scheduling in order to determine when each request of the task is to be executed by the processor and allows to slow down the processor, so that it consumes less power and takes greater time to execute. The tasks can be assigned priorities statically (when the priorities are fixed) or dynamically (when priorities are changed from one request to another). An example for well known dynamic priority assignment is Earliest Deadline First (EDF) and an example for static assignment is Rate-Monotonic Scheduling (RMS) [10] for periodic tasks.

DVFS can be classified based on timing constraints, scaling granularity, and scheduling policy basis (Fig. 1). The timing constraint based scheduling can be either non-real time or real time. Pillai et al. [11] present a class of novel algorithms for real-time DVS that modify the operating systems' real-time scheduler and task management service to provide significant energy savings while meeting real-time deadlines. Their results show a reduction of energy consumption by 20% to 40%. Based on scaling granularity, DVFS can be classified as pre-emptive and non-pre-emptive. In pre-emptive scheduling,

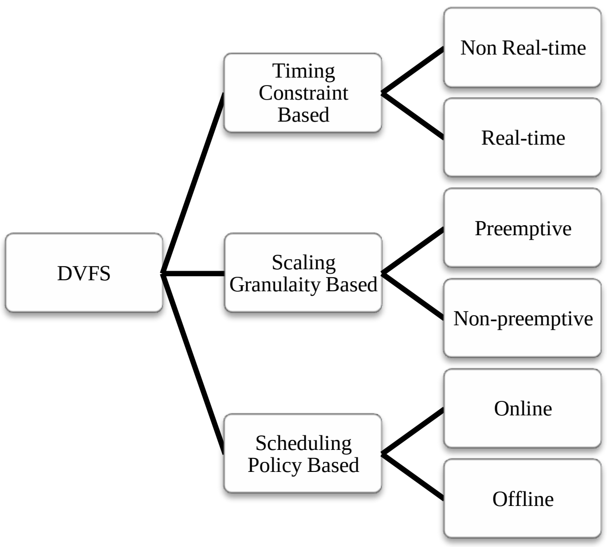


Figure 1. DVFS Classification. Adapted from [1]

the time is typically divided into fixed intervals, as proposed in [12, 13]. DVFS is then applied by a task level scheduler based on processor utilization statistics over preceding interval. Smoothing (or spreading processing cycles) is used to utilize idle cycles. While in non-pre-emptive scheduling tasks run to completion and cannot be interrupted during execution [14]. This in turn compels DVFS decisions to be applied after the completion or before the start of individual tasks. A non pre-emptive scheduling technique that treats voltage as a variable is e.g. discussed in [7]. Jeffay et al. [15] analyzed scheduling of a set of periodic and sporadic tasks on a uniprocessor without pre-emption and without inserted idle time. Their results showed the universal viability of earliest deadline first algorithm for given sets of sporadic and periodic tasks.

DVFS scheduling can also be classified into offline and online techniques [9]. Offline scheduling [12] has a lower runtime processing overhead, by assuming the worst case scenario or pessimistic scheduling approach, thus resulting in lower power savings. Online scheduling techniques as proposed in [16] could result in higher power savings than their static counterparts, however they are associated with a higher processing overhead. Jung et al. [17] present a supervised learning based dynamic power management (DPM) framework for multicore processors. The framework predicts on available input features such as the state of service queue occupancy and the task arrival rate. Then the predicted state is used to look up the optimal power management action from a pre-computed policy lookup table. This technique - in spite of being online - could result in smaller run time energy overhead. Sasaki et al. [18] propose a hybrid technique which takes advantage of lower processing overhead along with greater power efficiency. The technique uses both static and dynamic information and based on appropriate frequency/voltage set points inserted as a run-time code by the compiler. As per claim, the proposed technique can significantly reduce the energy consumption while satisfying soft timing constraints. Interestingly, DVFS can also be applied to secure the device from power attacks, which infer program behaviour from observing power supply current into a processor core: In [19] a power attack resistant cryptosystem is presented, which uses dynamic voltage and frequency switching to make the power traces show random properties and prevent the power attackers from carrying out time correlation between different power traces, while at the same time facilitating the energy saving task.

C. *Clock Gating*

A straightforward technique to reduce dynamic power consumption is to reduce gate toggling either by reducing the number of gates in a device or minimizing the number of times each gate toggles i.e. the clock frequency. This technique achieves a power reduction by reducing the switching capacitance at the cost of computational speed. Typically, the clock accounts for 20% to 40% of the total power consumption [20]. Clock gating is used to bypass the unused components of the system as shown in Fig. 2. It shows a combinational logic where ENABLE controls when the clock signal is passed to the further stages. Clock-gating algorithms can be grouped into three categories [21]: 1) system-level, 2) sequential and 3) combinational. System-level clock-gating stops the clock for an entire block and effectively disables its' entire functionality. On the contrary, combinational and sequential clock-gating selectively suspend clocking while the block continues to produce output.

In [22] a logic synthesis approach for domino/skewed logic styles based on Shannon expansion is proposed that dynamically identifies idle parts of logic and applies clock gating to them to reduce power in the active mode of operation, which results improvements of 15% to 64% in total power with minimal overhead in terms of delay and area compared to conventionally synthesized domino/skewed logic.

D. *Power Gating*

Power gating is one of the most effective techniques to reduce both sub-threshold leakage and gate leakage as it cuts off the path to the supply [23].

Fig. 3 shows a simple schematic of a logic block that has been power gated by a header switch or a footer switch. While the logic block is not active, assertion of the SLEEP signal results in turning off the either of the switches, thus disconnecting the logic block from supply, and reducing the leakage by orders of magnitude [24]. This technique is widely applied for implementing various sleep modes in CPUs. The examples of power gating architectures can be found in [25, 26].

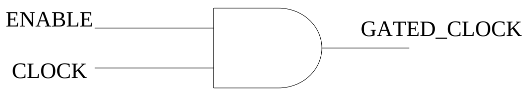


Figure 2. Gating circuit

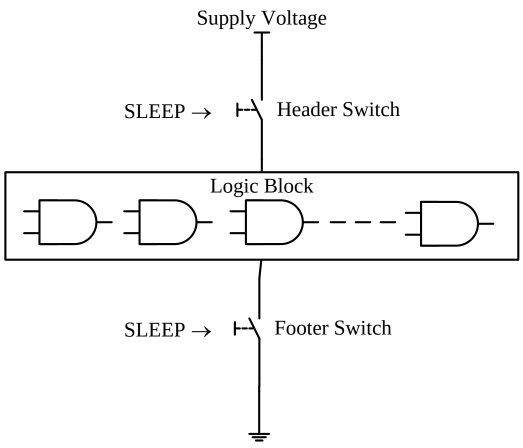


Figure 3. Power Gating using Header/Footer Switches (Adapted from [23])

In the following chapter power optimization techniques for different components are discussed.

II. Component level power reduction techniques

The major power consuming components such as the cache, pipeline, and buses are explored below and schemes to optimize the power in these components are discussed.

A. *Cache*

Cache power dissipation is contributed to by the tag and data arrays [27] since these are employed as static RAMS so that the cache access rate matches the pipeline clock rate. A further contribution is the frequency of access. Lastly the cache area is more densely packed than other areas on the die, which means more transistors and hence more power dissipation. A significant amount of power is dissipated by on–chip caches, as exemplified in the following [28]:

- The on–chip caches in the DEC 21164 microprocessor account for 25% of the total power dissipation
- In the bipolar, multi–chip implementation of a 300–MHz. CPU, 50% of the total dissipated power is due to the primary caches.
- A low–power microprocessor the DEC SA–110, dissipates 27% and 16% of the total power, respectively, in the on–chip instruction cache and data cache respectively [29].

Caches clearly present one of the most attractive targets for power reduction, this power reduction in caches can be achieved through several means: semiconductor process improvements, memory cell redesign, voltage reduction, and optimized cache structures [30].

Important additional techniques are:

- Adding buffers/filter cache/block buffering/Horizontal partitioning
- Turn off cache lines
- Sub-banking/Vertical partitioning

- Efficient tagging scheme (i.e. reducing tag comparisons)

(1) Adding buffers

A large number of low power cache ideas have been centered on the principle that adding an extra cache or buffer, usually small in size, and designing the system to fetch data directly from this buffer, allows thereby preventing an access to the original cache altogether. Since the buffer is relatively small, it is possible to achieve significant power savings if one can ensure a high hit rate to the buffer [9]. Another term that is used in the same context is *filter cache*. It has been observed that the instruction fetch and decode can consume more than 40% of processor power [31]. Experimental results across a wide range of embedded applications show that the filter cache results in improved memory system energy efficiency. For example, a direct mapped 256-byte filter cache achieves a 58% power reduction while reducing performance by 21%, corresponding to a 51% reduction in the energy-delay product over a conventional design [32]. An instruction filter cache or so called level zero cache [30] can be placed between the CPU core and the instruction cache to service the instruction stream. Power savings in instruction fetch result from accesses to a small cache [31]. Fig. 4 shows the effect of additional filter cache on the memory organization [30], where parts of the memory hierarchy can be on-chip or off-chip. The filter cache can be more efficiently optimized according to Kin et al. [32], who propose a scheme to predict the subsequent fetch addresses at run-time to identify whether the next fetch address is in the filter cache. In case a miss is predicted, it reduces the miss penalty by accessing the instruction cache directly.

(2) Turn off cache lines and Cache decay

During a fixed period of time the activity in a cache is often only centered on a small subset of the lines. This behavior can be exploited to cut the leakage power of large caches by putting the so called cold cache lines (with no recent activity) into a state preserving, low-power drowsy mode [33]. Various policies for turning lines off are based on generational aspects of cache line usage [34]. Since some architectures take advantage of dual caches (level one - L1 & level 2 - L2) different policies apply to these. Abella et al. [35] propose a technique to reduce leakage for L2 cache which occupies large area and hence more power consumption. A new approach, IATAC (inter access time per access count) introduced to reduce the leakage of L2 caches by turning off those cache lines that are not expected to be reused. For each cache line, it keeps track of the inter-access time and the number of accesses, providing decay times for each individual line.

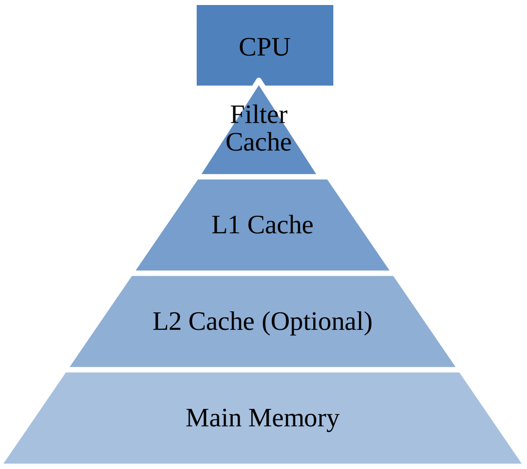


Figure 4. Filter cache & memory hierarchy (Adapted from [32])

In [36] the authors explore options for reducing leakage power by proactively discarding items from the cache, marking the lines invalid, and then putting the cache lines "to sleep" in a way that dramatically reduces their leakage current. Another policy described in [36], is *cache decay* that turns a cache line off if a preset number of cycles have elapsed since its last access.

(3) Sub-banking and Partitioned cache architecture

As feature sizes shrink, leakage power constitutes an increasing fraction of total processor power [37]. This affects cache structures disproportionately, since they have large area requirements. Subdividing the memory into smaller memories is one of the mechanisms used to reduce the effective length of the bit lines driven during memory operations, thereby reducing power dissipation [9]. In [38] the authors show a way to split the cache into several smaller units, each of which is a cache by itself. These so-called *Subcache* architectures not only reduce the per-access energy costs, but can potentially improve the locality behavior as well.

Various paradigms for memory partitioning are discussed by Golubeva et al. [39]. One such paradigm is to define portions of a cache of suitable granularity (e.g., a cache line, a way, a bank); second, they put unused portions into a low-leakage state, based on their idleness. Another scheme proposed is to build a multi-bank *cache architecture*, rather than a single cache internally consisting of multiple banks. A basic structure for sub-banking [40] is presented in Fig. 5.

Su et al. [40] explain various partitioning schemes for cache. Vertical partitioning is identical to buffering which is discussed in "Adding Buffers" section. Horizontal partitioning saves power by eliminating unnecessary access to caches (sub-banking techniques). Gray code addressing takes the advantage of the spatial locality of sequential memory address.

Since address buses poses a challenge for power optimization gray code addressing tries to reduce the bit switching activities for the same number of cache hits.

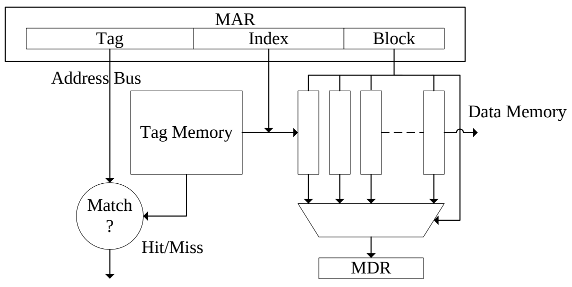


Figure 5. Cache sub-banking (Adapted from [9])

(4) Efficient tagging scheme and efficient set associative cache

Reading and comparing the tags in the tag arrays consumes also significant power. For highly associative caches, the power consumption is a particular concern since large number of tag comparisons are carried out in parallel [41]. Tag comparison elimination (TCE) techniques have been proposed to remove unnecessary tag comparisons to achieve power reduction. A tag comparison is considered unnecessary if a match or a mismatch can be determined without a real comparison. The decision is made based on the runtime information that is maintained in low cost hardware.

A fundamental cache design trade-off is between a direct-mapped cache and set-associative cache. A conventional direct mapped cache accesses only one tag array and one data array per cache access, whereas a conventional four-way set-associative cache accesses four tags arrays and four data arrays per cache access [42]. Inoue et al. [43] use way prediction for achieving high performance and low energy consumption of set-associative caches. It improves the energy delay product by 60-70% compared to a conventional set associative cache. The scheme speculatively predicts the cache way using the MRU (Most Recently Used) algorithm. If it is a cache hit there is a significant energy saving but if it is a prediction miss, the cache-access time of the way-predicting cache increases.

Panwar et al. [44] propose three modifications to reduce the tag comparison frequency. The first modification involves comparing cache tags for only those instructions that result in fetches from a new cache block. The second modification involves the tagging of those branches that cause instructions to be fetched from a new cache block. The third modification involves augmenting the I-cache with a small on-chip memory called the S-cache. The most frequently executed basic blocks of code are statically allocated to the S-cache before program execution. A non uniform cache architecture [45] is wherein an algorithm determines the optimum number of cache-ways for each cache-set and generates object code suitable for the non-uniform cache memory. A novel cache tagging scheme was introduced by Zhou et al. [46], where both virtual and physical tags co-exist in the cache tag arrays. Physical tags and special handling for the super-set cache index bits are used for references to shared data regions in order to avoid cache consistency problems. By eliminating the need for address translation on cache access for the majority of references, a significant power reduction is achieved.

B. Pipelining

The primary method of reducing the effective switched capacitance, even in a single-issue processor, is to control speculation, i.e. any operation which is performed before it is known to be required. In this sense the most 'profound' type of speculation is pipelining; letting instructions into the processor pipeline before knowing for certain that the instruction flow is not about to branch is clearly speculative. In most instances this is justified, however discarded instructions that are fetched and partially processed waste energy [47]. Here power performance issues are discussed that relate to pipelining with focus on branch prediction, pipeline gating and pipeline depth.

Pipelining a design involves [48] the addition of registers to create stages with the aim of improving throughput when the pipeline is fully utilized. Crucially the location of the pipeline stage will dictate the number of registers required to implement the stage. Since registers are power hungry elements, reduction in the total number of registers will yield reduction in power consumption. Shimada et al. deploy another technique called as PSU [49] or *pipeline stage unification*. PSU dynamically scales the clock frequency to reduce energy consumption as with DVFS, but unlike DVFS, it unifies multiple pipeline stages by bypassing pipeline registers, instead of scaling down the supply voltage. PSU saves energy consumption in two ways: 1) by reducing the total load capacitance of the clock driver; this is accomplished by stopping the clock signal to bypassed pipeline registers and 2) PSU reduces the clock cycle count of program execution by reducing the number of pipeline stages.

Ruan et al. [50] take advantage of bi-partitioning and encoding techniques toward optimizing power consumption of pipelined circuits. A pipeline architecture can be seen as combinational logic blocks separated by edge-triggered registers that are driven by a single clock signal. They propose a bipartition dual-encoding architecture to decrease power consumption of pipelined CMOS designs. The approach is based on the observation that the pipeline registers take a large fraction of total power dissipation for most of the circuits. In order to address this issue, a given circuit is bi-partitioned by using Shannon expansion to minimize the number of different outputs of both sub-circuits [51]. Subsequently, both partitions are encoded to reduce the switching activities of the pipeline registers and logic blocks.

To take advantage of the parallelism at the instruction level it is necessary to have some kind of prediction or speculation techniques to identify which instructions can be executed in future. This speculation can cost substantial energy if instructions are executed as a result of wrong prediction. A hardware mechanism called *pipeline gating* [2] to reduce overgrowing speculations in the pipeline can help to avoid this energy wastage. In this technique a confidence estimator is used to assess the quality of each branch prediction. A "high confidence" estimate means the branch predictor is likely to be correct. A "low confidence" estimate means the branch predictor has incorrectly predicted the branch. These confidence estimates decide when the processor is likely to be executing instructions that will not commit; once that decision has been reached, the pipeline is "gated", stalling specific pipeline stages.

Similarly Bahar et al. [52] introduce a *pipeline balancing* (PLB) technique to reduce component and full chip power without negatively impacting performance. *PLB* takes advantage of the inherent IPC (Instruction Per

Cycle) variations within a program to adjust the pipeline issue rate to the dynamic needs of each program.

Hartstein and Puzak [53] show that the optimum pipeline depth is found to be much shorter when power is taken into account. They define an appropriate power/performance metric, and optimize the design to that metric. Decreasing the pipeline depth slows down the processor but also saves energy because fewer 'speculative' instructions are fetched and decoded. In addition, hardware that supports pipelining, such as branch prediction, can be turned off as it is less effective in shallow pipelines [47]. Efthymiou et al. questions the energy saving accomplished using the branch predictors and hence they exploit the slack times in the applications with soft real time deadlines. Two techniques are presented to control the pipeline depth, using an asynchronous design style [47] because of its capability to adapt to changing energy/performance requirements at run time. One technique uses token passing as a method to control the pipeline occupancy, and thus indirectly, the pipeline depth. The other enables the selective and dynamic merging of adjacent pipeline stages by making the pipeline latches between them 'permanently' transparent.

C. Low power Buses

As the technology paradigm shifts towards globally asynchronous locally synchronous (GALS) and network-on-chip (NoC), interconnect mechanism increasingly contribute to power efficiency. Such interconnect mechanism (e.g. buses) can become a major bottleneck if not designed to be energy efficient. Generally, a significant portion of power in a chip is consumed by its bus system. For example, the bus wires dissipate about 15% to 30% [54] of the total power in DEC Alpha 21064 and Intel 80386 [55]. The delay and power dissipation of global buses [56] is increasing with technology scaling. This is bound to increase as the resistance of the wire increases with reduced size of the wire. With increasing bandwidth and communication efficiency, network fabrics [57] are becoming more attractive as the principle interconnect. Such fabrics are incorporated with high speed router and data links with bandwidth of several Gb/s [57]. Interconnect power optimization can be achieved at all the levels of design as shown [58] in Fig. 6.

The architecture level and circuit level techniques will be illustrated. These two categories can be sub divided into [54]

- Designing bus drivers/receivers to reduce the bus swing [59, 60]
- Encoding techniques to reduce the bus switching activity [61, 62], and
- Bus structure redesign to take advantage of local communications [63]

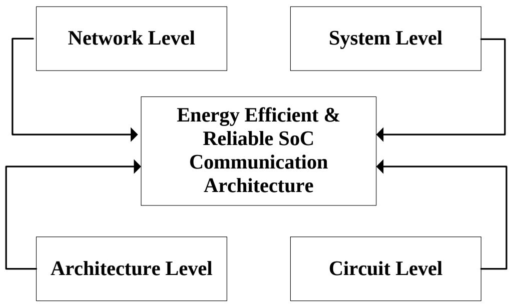


Figure 6. Interconnect optimizations at different levels (Adapted from [58])

(1) Reducing bus swing

The power consumption related to interconnect is given by [6]:

$$P_w = \frac{1}{2} \propto f_c C_w \Delta V^2, \qquad (3)$$

Where $\propto$ is the signal activity (the probability that the signal will change per clock cycle), f_c is the clock frequency, C_w is the interconnect wire capacitance and ΔV the signal voltage swing. It is apparent from the equation that the interconnect power can be reduced by decreasing the voltage swing on the wires. In [64] Kursun et al. propose a low power bi-directional CMOS voltage interface circuit which acts as the level converter between the driver and receiver, reducing the voltage level at the transmitter and amplifying it back at the receiver. A simple schematic representing this scheme [65] is shown in Fig. 7.

The driver converts a full-swing input into a reduced-swing interconnect signal, which is converted back to a full-swing output by the receiver. A detailed analysis of the low swing architectures has been undertaken by Zhang et al. [65], where methodologies have been introduced which involve reduced driver supply voltage, dynamically enabled drivers, level converters with low threshold voltage etc.

(2) Encoding techniques

Signal transitions in terms of logic level bit values, is another source of power consumption in CMOS circuits [66]. Short length interconnect wires present a parasitic capacitance similar to gate or drain capacitance which is negligible. However, for wires that are connecting more distant blocks the interconnect capacitance is of substantial measure [6]. A remedy for this is to reduce the switching activity on the wires effectively reducing the capacitance.

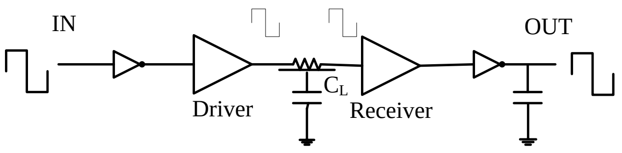


Figure 7. Programmable interconnect architecture (Adapted from [65])

Bus encoding schemes have been investigated as a means to downsize the total number of transitions on interconnects. Several encoding approaches have been proposed [67] according to the type of bus to which the encoding schemes are applied (address/data/control bus), the class of architectures considered (general-purpose, special-purpose), the complexity of the encoder, which determines whether it is applicable on-chip and/or off-chip, etc. Stan et al. [61] propose a *bus invert* method to reduce the power consumption by half. This method uses the hamming distance between the present data value on the bus and the next data value.

In case of off-chip communication such as external memory access Gray code encoding achieves the minimum switching activity as it takes the advantage of the spatial locality of the addresses generated by the microprocessor [68]. However Benini et al. have devised a better address encoding scheme *T0 code*[69] which outperforms the Gray code in the case of streams of consecutive addresses of limited and unlimited lengths.

In [70] a review of some of the encoding techniques has been undertaken. The encoding function can be optimized for specific access patterns such as sequential access (Gray [71, 72], T0 [68], Pyramid [73]) or random data (Bus Invert [61]), for special data types such as floating point numbers in DSP applications [74]. The encoding may be fully customized to the target data (Working Zone [75], Beach [76]). The key idea behind all of these techniques is to reduce the Hamming distance between consecutive address and data transfers.

(3) Bus structures

As previously discussed, the primary bottleneck is the capacitance during the switching activity. There has been research being done on redesigning the complete bus structure to reduce this capacitance. Shin et al. [77] developed a design methodology where it is assumed that capacitance components of buses are available from the layout of the processor core and the address streams from typical runs of the application code. Based on this information the optimal bus order is derived and this order is used to modify the processor layout. In [78] the split shared bus architecture is described. As shown in Fig. 8 the long bus is divided into two smaller buses and the dual port bus driver passes the data between the two buses. The power consumption is reduced in a sense that the effective capacitance for complete bus is divided into smaller portions and only active portions contribute to the power. As the effective length is shortened the parasitic components are also reduced. Similar idea is elaborated in [54] under the name of bus segmentation.

Network on Chip is considered to be one of the potential forerunners in coming years due to the performance demands with multiple processing blocks operating within same chip. This opens up flurry of opportunities for designing efficient network and routing architectures. Apart from the above mentioned methods there are different Network topologies and routing architectures/algorithms [79] which are exploited for energy/performance. Network topologies determine the number of hops and wire length [80] required for data

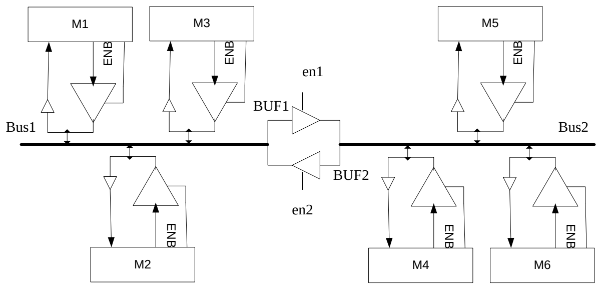


Figure 8. Split shared bus architecture (Adapted from [78])

transfer. In [80] Hangsheng et al. discuss the energy performance of the four main topologies: 2-D meshes, high-dimensional meshes, hierarchical meshes and express cubes. In [81] new micro architectures are introduced for routers in Network on Chip such as the segmented crossbar, cut-through crossbar and writethrough buffer.

III. Additional approaches

Although the focus of the paper is processor and chip level technologies for achieving low power but it is necessary to briefly introduce well known methodologies which can be applied into other stages of design. With new design ideas optimization part into compiler is being equally weighed as having power efficient hardware. Chakrapani et al. [82] identify two classes A and B of the compiler optimizations for energy reduction. Class A [82] consists of energy saving due to performance improvement such as reductions in the number of loads and stores, procedure cloning, loop unrolling, procedure inlining, and loop transformations. Class B [82] uses techniques like instruction scheduling; register pipelining, innovations in code selection to replace high power dissipating instructions with other instructions.

One further method centers on the application of compression algorithms to instruction code, cache blocks. Different code compression schemes for embedded systems are discussed in *[83]*. Arithmetic circuits are generously used in Arithmetic Logic Unit (ALU), Floating Point Unit (FPU) designs as well as DSP. Piguet [6] discusses various power efficient CMOS circuit design issues for basic arithmetic operators such as adder and their use in multiplier, square etc.

In the following some novel low power architectures are introduced.

IV. Low Power Architectures

In this chapter an overview of asynchronous, Reconfigurable Instruction Set Processors (RISP), Application Specific Instruction Set Processors (ASIP), extensible and No Instruction Set Computer (NISC) architectures is provided.

A. *Asynchronous Processors*

Asynchronous processor is a major development in low power architectures. The absence of a clock addresses the fact that the clock is a major power consumer in a synchronous processor. A clockless design does not only save clock power consumption but also protects from clock skew problem (which is more evident as the size of IC grows), reduces noise, and ease the component modularity and reuse. However the absence of clock also implicates careful design of handshake protocols; their failure could result in hazards [84] and race conditions [85]. Additionally, asynchronous circuits require more area for implementation as compared to their synchronous counterparts; thus resulting in higher production costs.

An early example of an asynchronous processor is the AMULET, which is a pioneer Asynchronous ARM architecture debuted in 1993 [86]. Internally the processor is constructed from several function units which operate independently and concurrently, exchanging information through a handshake protocol. This basic theme remains common with other asynchronous counterparts like SNAP [87], Bit SNAP [88], ARM996HS [89], HT80C51, and SUN Sproull Pipeline Processor architecture [90] etc.

Asynchronous circuits may have several advantages but the absence of suitable commercial grade synthesis tools have limited their wide spread usage to date. The TiDE - Timeless Design Environment [89] and BALSA [91] are addressing some of these issues. Also prototype development platforms for asynchronous design are still in the research phase e.g. [92, 93].

GALS (Globally Asynchronous and Locally Synchronous) is an approach aimed to combine the advantages of synchronous and asynchronous design methodologies while avoiding their disadvantages. GALS can be classified into three main implementation techniques, namely pausible clock generators, FIFO buffers, and boundary synchronization [94].

- Pausible-clock generators: The pausible-clock design style relies on locally generated, data driven clocks that can be stretched or paused either to prevent metastability or to let a transmitter or receiver stall because of a full or empty channel [95, 96].
- FIFO buffers: using asynchronous FIFO buffers between locally synchronous blocks to hide the synchronization problem. This technique can be seen in various NoC designs as [97, 98].
- Boundary synchronization: performing boundary synchronization on the signals crossing the borders of the locally synchronous island without stopping the complete locally synchronous block during data transfer [94].

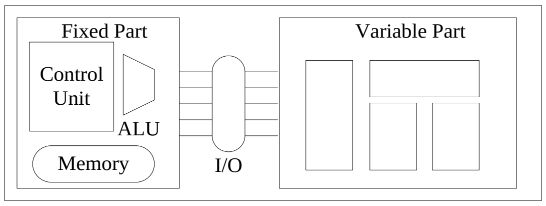
GALS has been actively researched for more than 20 years. But despite several successful implementations, GALS has had little impact on commercial products [94]. In [94] the authors have analyzed the actual challenges and problems for wider adoption of the currently proposed GALS methods. Their analysis shows that significant improvement can be achieved in terms of system integration and EMI reduction; but with marginal improvements in power savings. Additionally, introduction of a GALS approach leads to relatively small area increases, and in some cases even causes certain performance losses. GSLA or Globally Synchronous Locally Asynchronous is a relatively new scheme with the same targets in mind; however this also suffers with large area overheads; examples of GSLA can be found in [99-101].

B. Reconfigurable Instruction Set Processors (RISP)

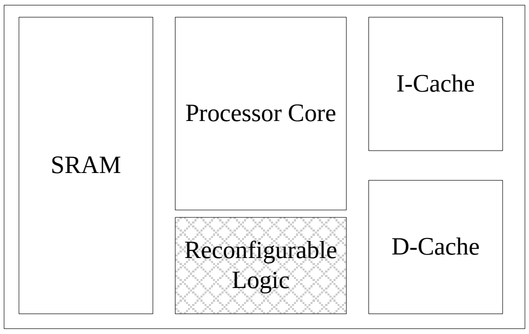
The idea of reconfigurable computing dates back to 1960 when Gerald Estrin at UCLA; presented [102] a model of a fixed machine with some variable (or configurable) structure [103] (see Fig. 9(a)). Work in this area has progressed substantially, but it is only in recent years that reconfigurable architectures are considered as potential runners for the low power race. RISPs are rather different to mainstream processors; they consist of fixed, simple functional units and a reconfigurable logic fabric (see Fig. 9(b)), that enables the software to modify the datapath as per requirement. The underlying idea of an RISP is to avoid higher non-recurring engineering (NRE) costs as in the case of application specific IP and at the same time providing the flexibility of custom datapath design based on application at runtime [104]; thus providing a highly power efficient and fast processor. Depending on the size of their configurable logic blocks (CLBs), RISPs can be classified as fine grained and coarse grained; while fine grained RISP give higher flexibility by handling data at bit (or several bits) level; however this complicates the handling algorithms and may result in higher power consumption, more area requirement and increased delay due to greater interconnects. Coarse grained RISP on the other hand are not very flexible as it consists of larger CLBs and can handle several bits simultaneously but may result in power losses on bit level computations [105]. The CLBs in coarse grained processors are sometimes alternatively referred to as rDPUs (reconfigurable DataPath Units) [106] to contrast from CLBs in a fine grained system.

There are various issues in the design of such processors, i.e. the design of fixed part or the core, the design of reconfigurable logic, its granularity, the highly flexible or dynamic datapath, and last but not least, the interface between the fixed and reconfigurable part [104]. Examples of RISP can be found in [107, 108], while in [109] a software development toolchain, based on opensource gcc complier, is proposed to target RISPs.

A specialized form of RISP is the Warp processor; that uses dynamic software optimization for better speed/energy performance. When a program first runs on the fixed processor, the system monitors the program to detect its most frequently-executed parts. The processor then automatically tries to move those parts to the reconfigurable logic or FPGA, all in runtime, thus eliminating tool flow restrictions and extra designer effort associated with traditional compile-time optimizations [110]. This approach gives an edge on the established approaches like the Binary-translation Optimized Architecture (BOA) [111] and the Dynamic Optimization System (Dynamo) [112]; by performing binary level hardware/software partitioning at runtime [113]. Warp



(a) First Proposed Design by G. Estrin in 1960 (Adapted from [102]



(b) A Modern Reconfigurable Processor (Adapted from [104])

Figure 9. Reconfigurable Processor, from earlier concept to reality

processors are currently capable of achieving speedups averaging 6.0 and energy savings up to 60% [114, 115]. The only difference between a traditional RISP and a Warp is that the reconfigurable part is located off-the-chip i.e. a separate Warp oriented FPGA or W-FPGA [110].

Other examples of RISP architectures are RICA [116], Matrix [117], and Montium [118]. Among these, RICA (Reconfigurable Instruction Cell Array) [116] is a novel cell [119, 120] based architecture; comprising of array of dynamically reconfigurable instruction cells (ICs), programmable with a high level language. All this is attained with an innovative design of the silicon fabric in a similar way to reconfigurable arrays but with a closer equivalence to software, thus achieving the high performance as coarse-grain FPGA architectures, while maintaining the flexibility, low cost, and programmability as of general purpose processors.

C. Application Specific Instruction Set Processors (ASIPs) and Extensible Processors

For more than a decade ASIPs have been researched as a compromise between General Purpose Processors and ASICs; providing reasonable performance, and power savings. The ASIP generally comprise of some basic instructions and the use of a code profiler to facilitate instruction extension, and logic parameterization. The instruction extension gives the flexibility of designing custom instructions based on application, thus resulting in a very efficient code execution. The ASIP Profiler may also provide flexibility of inclusion or exclusion of certain registers or logic blocks, or selection between big-endian or little endian [9].

Designing an optimal extended instruction set (IS) is the most challenging aspect, and has a direct impact on the performance of the ASIP. For larger software, it is very difficult to manually extend the IS, and this is further complicated by various design constraints, such as the format of the extended instructions (e.g., the number of input and output operands), clock period, available chip area, etc [121]. Fig. 10 shows a typical design flow for extensible processor generation. Cong et al. present a set of algorithms, including pattern generation, pattern selection, and application mapping, to efficiently utilize the instruction set extensibility of the target configurable processor [121]. There have been several successful commercial implementations of ASIPs like Tensilica’s Extensa Processors [122], Improv Systems Jazz DSP, Altera’s NIOS and Xilinx Microblaze processors.

D. No Instruction Set Computer (NISC)

A very different type of architecture that has no instructions at all, is presented in [123, 124] where authors suggest to build a custom datapath based on application code without any fixed instruction set. In this approach the datapath of the architecture is fully allocated before scheduling and binding. It compiles a C program directly to the datapath, the compilation results in Control Words (CWs) instead of instructions, and that’s basically a data path specified as a net-list of resource instances and their connections.

Fig. 11 elaborates the difference between typical RISC and NISC architectures. In [124] an algorithm is presented that maps an application on a given datapath by performing scheduling and binding simultaneously. In [125] two algorithms are presented, the first algorithm starts with an architecture that supports maximum parallelism for implementation of the input C code and iteratively refines it until an efficient resource utilization

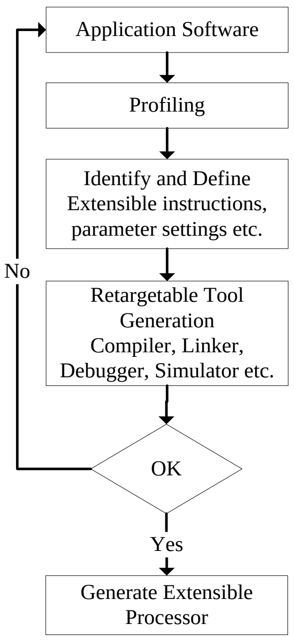

Application Software
Profiling
Identify and Define Extensible instructions, parameter settings etc.
Retargetable Tool Generation Compiler, Linker, Debugger, Simulator etc.
No
OK
Yes
Generate Extensible Processor

Figure 10. A typical extensible processor design flow (Adapted from [9])

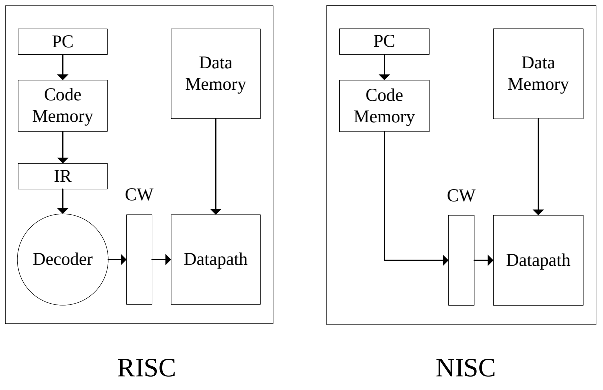


Figure 11. A Comparison between RISC and NISC (Adapted from [123])

is obtained while maintaining the performance constraint. The other algorithm is used to choose the priority of application basic blocks for optimization.

V. SUMMARY & CONCLUSION

As more and more portable devices become part of everyday life, power efficiency poses a challenge for the hardware and software designers. Power optimization for low power devices and processor cores has dominated research and will continue to be the focus for embedded systems in future. Solutions have been proposed in terms of new power saving architectures as well as power optimization techniques in different processor subsystems. This paper has reviewed such techniques as well as the architectures. Although not all the techniques can be applied simultaneously possible scenarios are outlined.

The use of DVFS to dynamically reduce the clock speed as well as supply voltage of CMOS circuits is discussed for systems where task based scheduling is possible. Caches can be helpful in systems where frequent external memory access is anticipated. They prove advantageous in reducing the off chip memory accesses which increases delay as well as bus power. Filter caches or buffers which can be seen as low level cache for level 1 and level 2 caches are powerful in boosting the performance and reducing the power consumption. Cache partitioning and turning off unused cache lines to optimize the power have also been reviewed. Adding pipeline stages is a commonly used method to improve the processor throughput so that processor speed can be utilized to its fullest. Power can be saved in pipelines by reducing the number of registers, branch speculation and run time adaptations of the pipeline stages which is helpful in soft real time systems. The processor's throughput and power are highly dependent on the underlying bus architectures as well. Buses have empowered the communication in NoC and will continue to do so. Various bus architectures, encoding techniques and also network topologies supported by power efficient routers are being developed, however we feel that there is still much more to be explored in the upcoming field of power efficient NoCs.

Novel architecture concepts such as Asynchronous processors, RISP, ASIP and NISC have been reviewed. These architectures present unconventional approaches, viz. clockless operation, application based reconfiguration, or absence of Instruction Set. A possible route forward could be to investigate the combination of these architectures supplemented with above mentioned techniques.

It can be inferred that independent techniques targeting a system component such as memory, supply voltage, instruction scheduling, cache etc remain more popular due to less development time and cost involved in design and test.

ACKNOWLEDGMENT

This research is in part supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under Grants EP/C005686/1, EP/C014790/1 and EP/C54630X/1.

REFERENCES

[1] E. Mocii and M. Pedram, "Best practices in low power design. 1. Power reduction techniques [Tutorial 1]," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* , 2004, pp. xi- xi.

[2] S. Manne, A. Klauser, and D. Grunwald, "Pipeline gating: speculation control for energy reduction," *SIGARCH Comput. Archit. News*, vol. 26, pp. 132-141, 1998.

[3] F. Richard, S. Perissakis, N. Cardwell, C. Kozyrakis, B. McGaughy, D. Patterson, T. Anderson, and K. Yelick, "The energy efficiency of IRAM architectures," in *Proceedings of the 24th annual international symposium on Computer architecture* Denver, Colorado, United States: ACM, 1997.

[4] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid-State Circuits,* vol. 31, pp. 1277-1284, Sep 1996.

[5] E. Acar, R. Arunachalam, and S. R. Nassif, "Predicting Short Circuit Power From Timing Models," *Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific*, vol. 21-24 pp. 277 - 282, 2003.

[6] C. Piguet, *Low-Power CMOS Circuits: Technology, Logic Design and CAD Tools*. Boca Raton, FL: CRC Press, Taylor & Francis Group, 2006.

[7] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power Optimization of Variable-Voltage Core-Based Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 1702-1714, December 1999.

[8] A. P. Chandrakasan, S. Sheng, and R. W. Broderson, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 473–484, 1992.

[9] J. Henkel and S. Parameswaran, *Designing Embedded Processors: A Low Power Perspective*: Springer Netherlands, 2007.

[10] G. C. Buttazzo, "Rate monotonic vs. EDF: judgment day," *Real-Time Syst.*, vol. 29, pp. 5-26, 2005.

[11] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the eighteenth ACM symposium on Operating systems principles* Banff, Alberta, Canada: ACM, 2001.

[12] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*, pp. 13–23, 1994.

[13] J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems," in *Real-Time Systems Symposium, 1992*, 1992, pp. 110-123.

[14] J.-W. Dai, "The Scheduling to Achieve Optimized Performance of Randomly Addressed Polling Protocol," *Wireless Personal Communications*, vol. 15, pp. 161-179, 2000.

[15] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of period and sporadic tasks," in *Proceedings of Twelfth Real-Time Systems Symposium, 1991.* , 1991, pp. 129-139.

[16] K. Choi, W. Lee, R. Soma, and M. Pedram, "Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*: IEEE Computer Society, 2004.

[17] H. Jung and M. Pedram, "Improving the Efficiency of Power Management Techniques by Using Bayesian Classification," in *9th International Symposium on Quality Electronic Design, 2008. ISQED 2008.* San Jose, CA, , 2008.

[18] H. Sasaki, Y. Ikeda, M. Kondo, and H. Nakamura, "An intra-task dvfs technique based on statistical analysis of hardware events," in *Proceedings of the 4th international conference on Computing frontiers* Ischia, Italy: ACM, 2007.

[19] S. Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Y. Xie, "Power Attack Resistant Cryptosystem Design: A Dynamic Voltage and Frequency Switching Approach," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 3*: IEEE Computer Society, 2005.

[20] F. Emnett and M. Biegel, "Power Reduction Through RTL Clock Gating," in *SNUG(Synopsis User Group) Conference* San Jose, 2000.

[21] M. Dale, "The Power of RTL Clock-gating," in *Chip Design Magazine*. vol. 2008 [cited 2008 February]: available from http://www.chipdesignmag.com/display.php?articleId=915.

[22] N. Banerjee, K. Roy, H. Mahmoodi, and S. Bhunia, "Low power synthesis of dynamic logic circuits using fine-grained clock gating," in *Proceedings of the conference on Design, automation and test in Europe: Proceedings* Munich, Germany: European Design and Automation Association, 2006.

[23] S. V. Kosonocky, A. J. Bhavnagarwala, K. Chin, G. D. Gristede, A.-M. Haen, W. Hwang, M. B. Ketchen, S. Kim, D. R. Knebel, K. W. Warren, and V. Zyuban, "Low-power circuits and technology for wireless digital systems," *IBM Journal of Research and Development*, vol. 47, pp. 283-298, 2003.

[24] R. Puri, L. Stok, and S. Bhattacharya, "Keeping hot chips cool," in *Proceedings of the 42nd annual conference on Design automation* Anaheim, California, USA: ACM, 2005.

[25] A. Calimera, A. Pullini, A. V. Sathanur, L. Benini, A. Macii, E. Macii, and M. Poncino, "Design of a family of sleep transistor cells for a clustered power-gating flow in 65nm technology," in *Proceedings of the 17th great lakes symposium on Great lakes symposium on VLSI* Stresa-Lago Maggiore, Italy: ACM, 2007.

[26] H.-O. Kim, Y. Shin, H. Kim, and I. Eo, "Physical design methodology of power gating circuits for standard-cell-based design," in *Proceedings of the 43rd annual conference on Design automation* San Francisco, CA, USA: ACM, 2006.

[27] K. Ghose and M. B. Kamble, "Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation," in *Proceedings of the 1999 international symposium on Low power electronics and design* San Diego, California, United States: ACM, 1999.

[28] K. Ghose and M. B. Kamble, "Energy Efficient Cache Organizations for Superscalar Processors," in *Power-Driven Microarchitecture Workshop In Conjunction With ISCA98 in Barcelona*, 1998.

[29] J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, G. W. Hoeppner, D. Kruckemyer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stephany, and S. C. Thierauf, "A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor," *Digital Tech. J.*, vol. 9, pp. 49-62, 1997.

[30] N. Bellas, I. Hajj, and C. Polychronopoulos, "Using dynamic cache management techniques to reduce energy in a high-performance processor," in *Proceedings of the 1999 international symposium on Low power electronics and design* San Diego, California, United States: ACM, 1999.

[31] W. Tang, R. Gupta, and A. Nicolau, "Power Savings in Embedded Processors through Decode Filer Cache," in *Proceedings of the conference on Design, automation and test in Europe*: IEEE Computer Society, 2002.

[32] J. Kin, M. Gupta, and W. H. Mangione-Smith, "The filter cache: an energy efficient memory structure," in *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture* Research Triangle Park, North Carolina, United States: IEEE Computer Society, 1997.

[33] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Proceedings of the 29th annual international symposium on Computer architecture* Anchorage, Alaska: IEEE Computer Society, 2002.

[34] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: exploiting generational behavior to reduce cache leakage power," in *Proceedings of the 28th annual international symposium on Computer architecture* G\&\#246;teborg, Sweden: ACM, 2001.

[35] J. Abella, A. Gonzalez, X. Vera, and M. F. P. O'Boyle, "IATAC: a smart predictor to turn-off L2 cache lines," *ACM Trans. Archit. Code Optim.*, vol. 2, pp. 55-77, 2005.

[36] Z. Hu, S. Kaxiras, and M. Martonosi, "Let caches decay: reducing leakage energy via exploitation of cache generational behavior," *ACM Trans. Comput. Syst.*, vol. 20, pp. 161-190, 2002.

[37] M. J. Geiger, S. A. McKee, and G. S. Tyson, "Drowsy region-based caches: minimizing both dynamic and static power dissipation," in *Proceedings of the 2nd conference on Computing frontiers* Ischia, Italy: ACM, 2005.

[38] S. Kim, N. Vijaykrishnan, M. Kandemir, A. Sivasubramaniam, and M. J. Irwin, "Partitioned instruction cache architecture for energy efficiency," *Trans. on Embedded Computing Sys.*, vol. 2, pp. 163-185, 2003.

[39] O. Golubeva, M. Loghi, E. Macii, and M. Poncino, "Locality-driven architectural cache sub-banking for leakage energy reduction," in *Proceedings of the 2007 international symposium on Low power electronics and design* Portland, OR, USA: ACM, 2007.

[40] C.-L. Su and A. M. Despain, "Cache design trade-offs for power and performance optimization: a case study," in *Proceedings of the 1995 international symposium on Low power design* Dana Point, California, United States: ACM, 1995.

[41] Y. Zhang and J. Yang, "Low cost instruction cache designs for tag comparison elimination," in *Proceedings of the 2003 international symposium on Low power electronics and design* Seoul, Korea: ACM, 2003.

[42] C. Zhang, F. Vahid, J. Yang, and W. Najjar, "A way-halting cache for low-energy high-performance systems," *ACM Trans. Archit. Code Optim.*, vol. 2, pp. 34-54, 2005.

[43] K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in *Proceedings of the 1999 international symposium on Low power electronics and design* San Diego, California, United States: ACM, 1999.

[44] R. Panwar and D. Rennels, "Reducing the frequency of tag compares for low power I-cache design," in *Proceedings of the 1995 international symposium on Low power design* Dana Point, California, United States: ACM, 1995.

[45] T. Ishihara and F. Fallah, "A non-uniform cache architecture for low power system design," in *Proceedings of the 2005 international symposium on Low power electronics and design* San Diego, CA, USA: ACM, 2005.

[46] X. Zhou and P. Petrov, "Low-power cache organization through selective tag translation for embedded processors with virtual memory support," in *Proceedings of the 16th ACM Great Lakes symposium on VLSI* Philadelphia, PA, USA: ACM, 2006.

[47] A. Efthymiou and J. D. Garside, "Adaptive Pipeline Depth Control for Processor Power-Management," in *IEEE International Conference on Computer Design (ICCD'02)*, 2002, p. 454.

[48] H. Ali and B. M. Al-Hashimi, "Architecture Level Power-Performance Tradeoffs for Pipelined Designs," in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.* New Orleans, LA, 2007, pp. 1791-1794.

[49] H. Shimada, H. Ando, and T. Shimada, "Pipeline stage unification: a low-energy consumption technique for future mobile processors," in *Proceedings of the 2003 international symposium on Low power electronics and design* Seoul, Korea: ACM, 2003.

[50] S.-J. Ruan, K.-L. Tsai, E. Naroska, and F. Lai, "Bipartitioning and encoding in low-power pipelined circuits," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, pp. 24-32, 2005.

[51] G. D. Micheli, *Synthesis and optimization of digital circuits*. New York, NY.: McGraw-Hill, 1994.

[52] R. I. Bahar and S. Manne, "Power and energy reduction via pipeline balancing," *SIGARCH Comput. Archit. News*, vol. 29, pp. 218-229, 2001.

[53] A. Hartstein and T. R. Puzak, "The optimum pipeline depth considering both power and performance," *ACM Trans. Archit. Code Optim.*, vol. 1, pp. 369-388, 2004.

[54] W. B. Jone, J. S. Wang, H. I. Lu, I. P. Hsu, and J. Y. Chen, "Design theory and implementation for low-power segmented bus systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 8, pp. 38-54, 2003.

[55] C. Dake Liu Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 663-670, Jun 1994.

[56] S. R. Sridhara and N. R. Shanbhag, "A low-power bus design using joint repeater insertion and coding," in *Proceedings of the 2005 international symposium on Low power electronics and design* San Diego, CA, USA: ACM, 2005.

[57] M. Ni and S. O. Memik, "Self-heating-aware optimal wire sizing under Elmore delay model," in

Proceedings of the conference on Design, automation and test in Europe Nice, France: EDA Consortium, 2007.

[58] V. Raghunathan, M. B. Srivastava, and R. K. Gupta, "A survey of techniques for energy efficient on-chip communication," in *Proceedings of the 40th conference on Design automation* Anaheim, CA, USA: ACM, 2003.

[59] G. C. Cardarilli, M. Salmeri, A. Salsano, and O. Simonelli, "Bus architecture for low-power VLSI digital circuits," in *IEEE International Symposium on Circuits and Systems*. vol. 4, 1996.

[60] R. Golshan and B.Haroun, "A novel reduced swing CMOS bus interface circuit for high speed low power VLSI systems," in *IEEE International Symposium on Circuits and Systems. ISCAS '94.*, . vol. 4 London, UK, 1994 pp. 351-354.

[61] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 3, pp. 49-58, 1995.

[62] M. R. Stan and W. P. Burleson, "Coding a terminated bus for low power," in *Proceedings of the Fifth Great Lakes Symposium on VLSI (GLSVLSI'95)*: IEEE Computer Society, 1995.

[63] R. Mehra, L. M. Guerra, and J. M. Rabaey, "A partitioning scheme for optimizing interconnect power," *IEEE Journal of Solid-State Circuits.*, vol. 32, pp. 433-443, Mar 1997.

[64] V. Kursun, R. M. Secareanu, and E. G. Friedman, "Low Power CMOS Bi-Directional Voltage Converter," in *Conference Proceedings of the IEEE EDS/CAS Activities in Western New York* 2001, pp. 6-7.

[65] H. Zhang and J. Rabaey, "Low-swing interconnect interface circuits," in *Proceedings of the 1998 international symposium on Low power electronics and design* Monterey, California, United States: ACM, 1998.

[66] P. R. Panda and N. D. Dutt, "Reducing Address Bus Transitions for Low Power Memory Mapping," in *Proceedings of the 1996 European conference on Design and Test*: IEEE Computer Society, 1996.

[67] G. Ascia, V. Catania, M. Palesi, and A. Parlato, "An evolutionary approach for reducing the energy in address buses," in *Proceedings of the 1st international symposium on Information and communication technologies* Dublin, Ireland: Trinity College Dublin, 2003.

[68] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, "Address bus encoding techniques for system-level power optimization," in *Proceedings of the conference on Design, automation and test in Europe* Le Palais des Congrés de Paris, France: IEEE Computer Society, 1998.

[69] L. Benini, G. d. Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems," in *Proceedings of the 7th Great Lakes Symposium on VLSI*: IEEE Computer Society, 1997.

[70] M. Pedram, "Power optimization and management in embedded systems," in *Proceedings of the 2001 conference on Asia South Pacific design automation* Yokohama, Japan: ACM, 2001.

[71] C.-L. Su, C.-Y. Tsui, and A. M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Des. Test*, vol. 11, pp. 24-30, 1994.

[72] H. Mehta, R. M. Owens, and M. J. Irwin, "Some Issues in Gray Code Addressing," in *Proceedings of the 6th Great Lakes Symposium on VLSI*: IEEE Computer Society, 1996.

[73] W.-C. Cheng and M. Pedram, "Power-optimal encoding for DRAM address bus (poster session)," in *Proceedings of the 2000 international symposium on Low power electronics and design* Rapallo, Italy: ACM, 2000.

[74] P. A. Kyeounsoo Kim Beerel, "A low-power matrix transposer using MSB-controlled inversion coding," in *The First IEEE Asia Pacific Conference on ASICs, 1999. AP-ASIC '99.* Seoul, South Korea, 1999, pp. 194-197.

[75] E. Musoll, T. Lang, and J. Cortadella, "Exploiting the locality of memory references to reduce the address bus energy," in *Proceedings of the 1997 international symposium on Low power electronics and design* Monterey, California, United States: ACM, 1997.

[76] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer, "System-level power optimization of special purpose applications: the beach solution," in *Proceedings of the 1997 international symposium on Low power electronics and design* Monterey, California, United States: ACM, 1997.

[77] Y. Shin and T. Sakurai, "Coupling-driven bus design for low-power application-specific systems," in *Proceedings of the 38th conference on Design automation* Las Vegas, Nevada, United States: ACM, 2001.

[78] C.-T. Hsieh and M. Pedram, "Architectural power optimization by bus splitting," in *Proceedings of the conference on Design, automation and test in Europe* Paris, France: ACM, 2000.

[79] J. Hu and R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*: IEEE Computer Society, 2003.

[80] W. Hangsheng, P. Li-Shiuan, and M. Sharad, "A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 2*: IEEE Computer Society, 2005.

[81] H. Wang, L.-S. Peh, and S. Malik, "Power-driven Design of Router Microarchitectures in On-chip Networks," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*: IEEE Computer Society, 2003.

[82] L. N. Chakrapani, P. Korkmaz, V. J. M. III, K. V. Palem, K. Puttaswamy, and W. F. Wong, "The

emerging power crisis in embedded processors: what can a poor compiler do?," in *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems* Atlanta, Georgia, USA: ACM, 2001.

[83] H. Lekatsas and W. Wolf, "Code compression for embedded systems," in *Proceedings of the 35th annual conference on Design automation* San Francisco, California, United States: ACM, 1998.

[84] S. Furber and J. SparsØ, *Principles of Asynchronous Circuit Design – A Systems Perspective*. Boston: Kluwer Academic Publishers, 2001.

[85] K. Y. Yun and D. L. Dill, "Automatic synthesis of 3D asynchronous state machines," in *Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design* Santa Clara, California, United States: IEEE Computer Society Press, 1992.

[86] S. B. Furber, P. Day, J. D. Garside, N. C. Paver, and J. V. Woods, "AMULET1: a micropipelined ARM," *Compcon Spring '94, Digest of Papers,* pp. 476 - 485, 28 Feb.-4 March 1994

[87] M. Hempstead, N. Tripathi, P. Mauro, G.-Y. Wei, and D. Brooks, "An Ultra Low Power System Architecture for Sensor Network Applications," *SIGARCH Comput. Archit. News,* vol. 33, pp. 208-219, 2005.

[88] V. N. Ekanayake, I. V. Clinton Kelly, and R. Manohar, "BitSNAP: Dynamic Significance Compression for a Low-Energy Sensor Network Asynchronous Processor," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*: IEEE Computer Society, 2005.

[89] A. Bink and R. York, "ARM996HS: The First Licensable, Clockless 32-Bit Processor Core," *IEEE Micro,* vol. 27, pp. 58-68, 2007.

[90] C. E. Molnar, R. F. Sproull, and I. E. Sutherland, "Counterflow Pipeline Processor Architecture," Sun Microsystems, Inc. 1994.

[91] A. Bardsley and D. Edwards, "Compiling the language Balsa to delay insensitive hardware," in *Proceedings of the IFIP TC10 WG10.5 international conference on Hardware description languages and their applications* Toledo, Spain: Chapman; Hall, Ltd., 1997.

[92] D. Fang, J. Teifel, and R. Manohar, "A High-Performance Asynchronous FPGA: Test Results," in *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05) - Volume 00*: IEEE Computer Society, 2005.

[93] S. Hauck, S. Burns, G. Borriello, and C. Ebeling, "An FPGA for Implementing Asynchronous Circuits," *IEEE Des. Test,* vol. 11, pp. 60-69, 1994.

[94] M. Krsti, E. Grass, F. K. Gürkaynak, and P. Vivet, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," *IEEE Des. Test,* vol. 24, pp. 430-441, 2007.

[95] F. K. Gurkaynak, S. Oetiker, H. Kaeslin, N. Felber, and W. Fichtner, "GALS at ETH Zurich: Success or Failure," in *Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems*: IEEE Computer Society, 2006.

[96] P. Teehan, M. Greenstreet, and G. Lemieux, "A Survey and Taxonomy of GALS Design Styles," *Design & Test of Computers, IEEE,* vol. 24, pp. 418-428, 2007.

[97] E. Beigne and P. Vivet, "Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture," in *Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems*: IEEE Computer Society, 2006.

[98] I. M. Panades and A. Greiner, "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures," in *First International Symposium on Networks-on-Chip (NOCS'07),* 2007, pp. 83-94.

[99] A. E. Sjogren and C. J. Myers, "Interfacing synchronous and asynchronous modules within a high-speed pipeline," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 8, pp. 573-583, 2000.

[100] H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, and J. M. Rabaey, "A 1V Heterogeneous Reconfigurable DSP IC for Wireless Baseband Signal Processing," *IEEE Journal of Solid State Circuits,* vol. 35, pp. 1697-2003, November 2000.

[101] Y. Li, Z.-y. Wang, and K. Dai, "A Low-Power Application Specific Instruction Set Processor Using Asynchronous Function Units," in *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*: IEEE Computer Society, 2007.

[102] G. Estrin, "Organization of Computer Systems: The Fixed-plus Variable Structure Computer," in *Proceedings of the Western Joint Computer Conference* New York, 1960, pp. 33-40.

[103] G. Estrin, "Reconfigurable computer origins: the UCLA fixed-plus-variable (F+V) structure computer," *Annals of the History of Computing, IEEE,* vol. 24, pp. 3-9, 2002.

[104] F. Barat, R. Lauwereins, and G. Deconinck, "Reconfigurable instruction set processors from a hardware/software perspective," *Software Engineering, IEEE Transactions on,* vol. 28, pp. 847-862, 2002.

[105] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk, and P. Y. K. Cheung, "Reconfigurable computing: architectures and design methods," *IEE Proceedings - Computers and Digital Techniques,* vol. 152, pp. 193-207, 2005.

[106] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective," in *Proceedings of Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001.* Munich, Germany, 2001, pp. 642-649.

[107] F. Barat, M. Jayapala, T. Vander, R. Lauwereins, G. Deconinck, and H. Corporaal, "Low Power

Coarse-Grained Reconfigurable Instruction Set Processor," *Field-Programmable Logic and Applications; Lecture Notes in Computer Science*, vol. 2778/2003, pp. 230-239, 2003.

[108] M. J. Wirthlin, "A dynamic instruction set computer," in Proceedings of the IEEE Symposium on FPGA's for Custom Computing Machines: IEEE Computer Society, 1995.

[109] A. L. Rosa, L. Lavagno, and C. Passerone, "A software development tool chain for a reconfigurable processor," in *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems* Atlanta, Georgia, USA: ACM, 2001.

[110] R. Lysecky, G. Stitt, and F. Vahid, "Warp Processors," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, pp. 659-681, 2006.

[111] M. Gschwind, E. R. Altman, S. Sathaye, P. Ledak, and D. Appenzeller, "Dynamic and transparent binary translation," *Computer*, vol. 33, pp. 54-59, 2000.

[112] V. Bala, E. Duesterwald, and S. Banerjia, "Dynamo: a transparent dynamic optimization system," SIGPLAN Not., vol. 35, pp. 1-12, 2000.

[113] G. Stitt, R. Lysecky, and F. Vahid, "Dynamic hardware/software partitioning: a first approach," in Proceedings of the 40th conference on Design automation Anaheim, CA, USA: ACM, 2003.

[114] R. Lysecky and F. Vahid, "A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning," in Proceedings of the conference on Design, Automation and Test in Europe - Volume 1: IEEE Computer Society, 2005.

[115] R. Lysecky and F. Vahid, "A Configurable Logic Architecture for Dynamic Hardware/Software Partitioning," in Proceedings of the conference on Design, automation and test in Europe - Volume 1: IEEE Computer Society, 2004.

[116] S. Khawam, I. Nousias, M. Milward, Y. Yi, M. Muir, and T. Arslan, "The Reconfigurable Instruction Cell Array," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 16, pp. 75-85, 2008.

[117] E. Mirsky and A. e. DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources," in IEEE Symposium on FPGAs for Custom Computing Machines, 1996. Proceedings. Napa Valley, CA, USA, 1996, pp. 157-166.

[118] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "Montium - Balancing between Energy-Efficiency, Flexibility and Performance," in Proceedings of ERSA'03, Las Vegas, USA, 2003, pp. 235-241.

[119] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, "Introduction to the cell multiprocessor," IBM J. Res. Dev., vol. 49, pp. 589-604, 2005.

[120] H. P. Hofstee, "Power Efficient Processor Architecture and The Cell Processor," in Proceedings of the 11th International Symposium on High-Performance Computer Architecture: IEEE Computer Society, 2005.

[121] J. Cong, Y. Fan, G. Han, and Z. Zhang, "Application-specific instruction generation for configurable processor architectures," in Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays Monterey, California, USA: ACM, 2004.

[122] G. Martin, "Recent Developments in Configurable and Extensible Processors," in IEEE 17th International Conference on Application-specific Systems, Architectures and Processors (ASAP'06), 2006, pp. 39-44.

[123] D. Gajski, "NISC: The Ultimate Reconfigurable Component," Center for Embedded Computer Systems October 2003.

[124] M. Reshadi and D. Gajski, "A cycle-accurate compilation algorithm for custom pipelined datapaths," in Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis Jersey City, NJ, USA: ACM, 2005.

[125] J. Trajkovic and D. Gajski, "Automatic Data Path Generation from C code for Custom Processors," in International Federation for Information Processing Publications-IFIP, May 2007, pp. 107-120.

Muhammad Yasir Qadri is currently a PhD candidate at School of Computer Science and Electronic Engineering, University of Essex, UK. He received his Bachelor of Engineering in Electronics from Mehran University of Engineering and Technology, Jamshoro, Pakistan in 2002. He has more than 6 years of experience in embedded system design. His research interests include low power processor architectures, cache optimization for power and reconfigurable MPSoC. He is a student member of ACM and ACM SIGARCH.

Hemal Gujarathi got his MSc degree in Embedded Systems at School of Computer Science and Electronic Engineering, University of Essex, UK in 2008. As part of his MSc thesis he developed power measurement strategy for embedded systems to understand how energy is consumed at software level. He received his Bachelor of Engineering in Electronics &

Telecommunications from University of Pune, India in 2001. Prior to his MSc study he worked in the fields of embedded systems design and development for more than 4 years. He also worked at Motorola, USA developing embedded software for their Linux based mobile phones. His areas of interest are low power embedded systems and application of these low power embedded systems in the field of pervasive computing.

Prof. Klaus D. McDonald-Maier received the Dipl. Ing. and MS degrees in electrical engineering from the University of Ulm, Germany, and the École Supérieur de Chimie Physique Électronique de Lyon, France, in 1995, respectively. In 1999, he received the doctorate in computer science from the Friedrich Schiller University, Jena, Germany. Prof. McDonald-Maier worked as a systems architect on reusable microcontroller cores and modules at Infineon Technologies AG's Cores and Modules Division in Munich, Germany and as a lecturer in electronic engineering at the University of Kent, Canterbury, United Kingdom. In 2005, he joined the University of Essex, Colchester, United Kingdom, where he is a Professor in the School of Computer Science and Electronic Engineering. His current research interests include embedded systems and system-on-chip (SoC) design, development support and technology, parallel and distributed architectures, the application of soft computing techniques for real world problems, and robot control. He is a senior member of the IEEE and a Fellow of the IET.