Ultra-Low-Power Digital Architectures for the Internet of Things

3

Davide Rossi, Igor Loi, Antonio Pullini, and Luca Benini

This chapter introduces the architectures implementing the digital processing platforms and control for Internet of things applications. It will provide a review of the state of the art Ultra-Low-Power (ULP) micro-controllers architecture, highlighting the main challenges and perspectives, and introducing the potential of exploiting parallelism in this field currently dominated by single issue processors.

3.1 Definitions and Motivations

The last years have seen an explosive growth of small, battery powered devices that sense the environment and communicate wirelessly the sensed data after some data processing, recognition, or classification. Collectively referred to as the Internet of Things (IoT), all these devices share the need for extreme energy efficiency and power envelopes of few milliwatts. From a system level perspective, in architectures targeting such applications, the data acquisition part is implemented by a sensing subsystem, realized with low-power sensors such as visual imagers, microphone arrays, Micro Electro-Mechanical Systems (MEMS), or bioelectrical sensors. Sensed data is then digitally processed with low-power microcontrollers (MCUs), and transmitted through a wireless communication subsystem consisting of low-energy TX/RX radio transceivers implementing low-energy stacks. While these three subsystem are sometimes split over more than one chip, the market is trending toward fully-integrated single-chip solution. The entire IoT node is powered by harvesters or small form factor batteries (power supply/conversion subsystem). Despite of an almost 10x reduction of the transceiver power in just a few years, its share in the overall power budget of most wireless sensor nodes and wearables remains dominant (De Groot 2015). In this scenario, a high-potential approach to reduce the system energy is to increase the complexity of near-sensor data analysis and filtering by providing more computational power to the processing sub-system. This approach can dramatically reduce the amount of wireless data transmitted, that could be reduced to a class, a signature, or even just a simple event. For this reason, the availability of powerful, flexible and energy-efficient digital processing hardware in close proximity to sensors plays a key role in the internet of things revolution. The aim of this chapter is to review the state of the art of

D. Rossi (✉) • I. Loi
University of Bologna, Bologna, Italy
e-mail: davide.rossi@unibo.it

A. Pullini
ETH, Zurich, Switzerland

L. Benini
University of Bologna, Bologna, Italy

ETH, Zurich, Switzerland

M. Alioto (ed.), *Enabling the Internet of Things*, DOI 10.1007/978-3-319-51482-6_3

ultra-low power computing platforms for near-sensor processing, and highlight the main challenges and perspectives related to these architectures.

Most of low-power commercial microcontrollers cannot provide the required performance levels for several applications within the power budgets offered by small form factor coin batteries and energy harvesters. A promising approach to achieve up to one order of magnitude of improvement in energy efficiency of integrated circuits with respect to "business as usual" CMOS in strong inversion is ultra-low voltage, near-threshold computing. The key idea is to lower the supply voltage of chips to a value only slightly higher than the threshold voltage. Aggressive voltage scaling has been extensively analyzed in the literature, including its limitations and disadvantages (Dreslinski et al. 2010). One of the main issues with low-voltage operation is performance degradation, which can limit the degree of use of voltage-scaling for a given processing requirement.

A commonly adopted approach to overcome the performance loss in ultra-low voltage devices on hardwired functions implemented in *Application Specific Integrated Circuits* (ASICs). By exploiting dedicated circuits, digital processing systems are able to match the performance requirements of applications, even at a very low operating voltage, with frequencies of tens to hundreds of KHz and power consumptions in the range of few μW to hundreds μW. In some cases, these dedicated systems, implemented as *System-On-Chip* (SoC) or *System-In-Package* (SiP), integrate digital signal processing circuits, analog front end, analog signal processing circuits, and power supply circuits (batteries, harvester or both) leading to extremely compact form factors.

The dedicated ASIC approach has been extensively used in traditional fields of applications of ultra-low power devices, such as wearable or implantable sensors for health monitoring (Zhang et al. 2012; Yoo et al. 2012; Yakovlev et al. 2012). Although these devices minimize power consumption, they are not flexible as their function is limited to the specific purpose for which they are designed. Also, their performance is not scalable as they are designed with a specific use case in mind. While adoption of dedicated circuits is attractive to tackle the stringent constraints of implantable applications for health monitoring (tens of microwatts), these two important aspects limit the exploitation of dedicated circuits for the majority of IoT applications since the Non Recurrent Engineering (NRE) costs required for their development cannot be amortized over large volume products.

When the targeted algorithms are more generic, so that can they be re-utilized for more than a single application, the above described restrictions can be relaxed by providing some run-time configurability to the integrated circuits, and by increasing the operating range via voltage and frequency scaling. As this approach has mainly been applied to the signal processing field, this class of computing devices is usually referred as *Application (or domain) Specific Signal Processors* (ASSPs). Several examples of this class of devices apply to visual sensors, where several basic functions implemented with dedicated accelerators or specialized processors can be shared among different applications (Park et al. 2013; Hsu et al. 2012; Jeon et al. 2013).

The ultimate step toward flexibility is given by the exploitation of the software programmability of instruction processors. In last few years some instruction processors working in the near-threshold or sub-threshold operating region have been presented from both industry (Ambiq 2015) and research (Bol et al. 2013). Some of the proposed devices are also able to work on a wide range of operating points, as the low voltage operation might not be always suitable to match the requirements of the application targets (Gammie et al. 2011). Again, when operating at low voltage, sequential instruction execution coupled with very slow operating frequency may lead to insufficient performance for the application requirements. Explored solutions to improve performance of ultra-low power processors while maintain a high degree of flexibility also rely on the exploitation of

software parallelism. The exploitation of multi core platforms can provide benefits with respect to single processor cores for high application workloads as it has been demonstrated that parallel computing at low voltage can be more energy efficient than sequential computing at a higher voltage under certain assumptions (Dogan et al. 2011). This concept has been extensively exploited for high-end embedded applications, where multi-core architecture has become the de-facto standard. On the other hand, when the application workload is low, or when the workloads are not easily parallelizable, the introduced multi-core platforms suffer from energy efficiency losses with respect to single core platforms, mainly caused by static (primarily leakage-induced) power consumption, due to the larger area and architectural overheads. Hence, while multi-core platforms are starting to be seen with interest in the world of ultra-low-power applications, a great effort still needs to be done to target the applications driving the IoT domain.

The rest of the chapter is organized as follow: Section 3.1 describes the architecture of off-the-shelf microcontrollers typically employed in IoT applications. Section 3.2 describes the main challenges of IoT computing platforms, mainly related to the exploitation of performance scalability exploiting parallel processing. Section 3.3 provides an example of research Parallel Ultra-Low-Power computing platform (PULP). Finally, Sect. 3.4 provides some concluding remarks, and highlights challenges and perspectives.

3.2 Ultra-Low-Power Microcontroller Architectures

The applications targeted by current off-the-shelf microcontroller (MCUs) architectures require to periodically fetch environmental information from a wide variety of sensors, which is then transmitted via wireless through low-power antennas after a limited amount of processing. In this scenario, to maximize the power efficiency of the system, the data processing hardware should be active only for the small amount of time required to read, process and transmit the information, while it is *idle* for the rest of the time. It is usual to refer to this mechanism as *duty cycling*. Figure 3.1 shows the typical power consumption pattern of a microcontroller employed for a generic IoT application. For most of the time, the MCU is in a *deep sleep* state where it is waiting for an event to wake up, triggered by an internal timer or by an external event generated by one of the sensors. Once the event is captured the device restores the power supply and restarts clocks (wake-up), restores the state of the CPU (stack, data etc.) and then can start fetching new data from I/Os, process and transmit it. Once done with the processing, it can save the state and go back to sleep.

Today microcontrollers (MCUs) feature several power modes to deal with different application scenarios where the various states may have a very different duration, duty cycles and performance requirements. For example if an application has to deal with frequent wake-ups, paying at each wake-up the price of saving and restoring

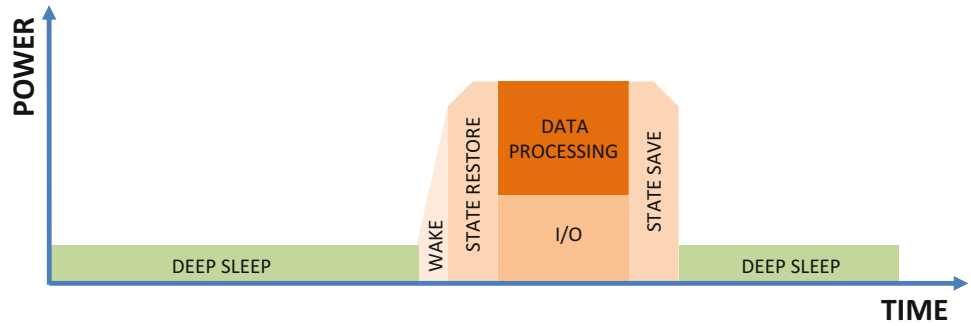

Fig. 3.1 Typical power profile of IoT applications

its full state is highly inefficient and a power mode with higher off current but with state retention in this case is much more valuable.

3.2.1 Power Management

Having an efficient off and idle states is a key requirements for micro controllers. In many IoT applications sleep-mode current is the biggest contributor to the overall power consumption. When considering sleep-mode current we should consider also the current required to reactivate the circuit. Wakeup currents have a big impact on the choice of the sleep mode. There are many different techniques to reduce the power consumption when idle (Sect. 10).

The simplest approach is *clock gating*, which only implies stopping the clock. In this mode dynamic power is cut and only leakage current is flowing. This mode is state retentive and requires no time to go back to active if the clock generator is still running. Clock gating usually has a very fine grained granularity, mostly all blocks in the system can be put in idle mode independently since it does not have any overhead at circuit level. Wakeup time after an idle state is dependent only on the status of the clock generator. If the clock generator is kept on, the wakeup time is negligible; but the clock generator has also been stopped for more aggressive power savings. The wake-up time is strongly impacted by the type of clock generator. In today's microcontrollers there are different types of clock generation unit to deal with the different operating corners that always guarantee the maximum efficiency. A good example of state of the art clocking offer in microcontrollers is the latest STM32L4 family. In the micro of the L4 family there are five available clock sources (see Fig. 3.2). There are two low speed generators that generate a 32 kHz frequency: one oscillator using an external quartz (LSE) and another one using an internal RC oscillator (LSI), which is more power-efficient but less precise. The same is done for the faster 48 MHz clock, where there is an oscillator using an external clock (HSE) and an internal configurable RC oscillator (MSI). The fifth clock source is a fixed 16 MHz internal RC oscillator. The L4 has also three PLLs capable of multiplying the frequency and reaching up to 180 MHz (Microelectronics

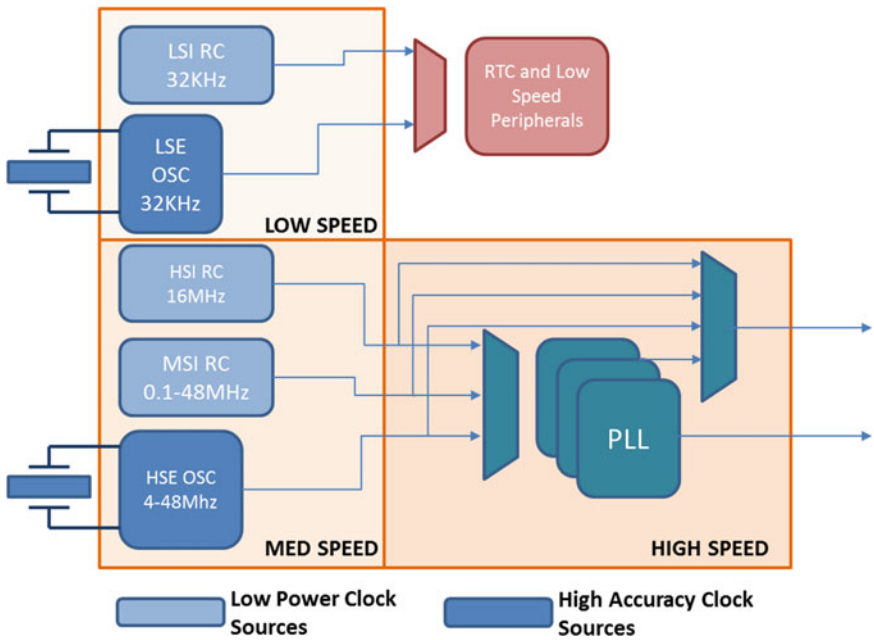

Fig. 3.2 Example of state of the art clock distribution

2016). Sleep mode turns the clocks to the core off, but the user has the option to leave on the peripherals' clocks. The power in this mode is not just leakage but dynamic current of the peripherals that are left on. In this mode data can be still received, and the core retains its state and continues operation when required. All clocks to the digital logic are turned off and the analog sub-systems can be controlled to have flexible wake up times depending on the application requirements. The lowest power mode is when all the analog clocking elements are turned off. Wake up time is determined by the selection of the wake up clock source. The fastest time is from the low-power oscillator and the slowest time is from the crystal oscillator and the PLL.

To reduce leakage in idle mode, clock gating is used in conjunction with voltage scaling. When the clock is stopped we can afford to lower the voltage as low as the retention voltage for the memory elements. The cost of this at circuit level is the required flexibility in the power supply. The effect on the application is that we require more time to exit the idle mode due to the settling time of the power supply. Further reduction can be obtained by not only lowering the voltage but also applying a reverse body biasing to increase the threshold voltage of the devices (Rossi et al. 2016a).

The lowest leakage is obtainable only with power gating, where the power supply is turned off. All microcontrollers available today feature a deep-sleep mode where the device power supply is disconnected and only a small always-on part of the chip controlling the wake up is kept alive. Always-on domain usually have an RTC to have the possibility to wake up the system after a certain amount of time and have a set of memory elements (flops or SRAMs) to keep track of the previous state. The amount of features active on the always-on domain is usually selectable by the user to give maximum flexibility depending on the application requirements. The startup behavior of the analog modules can have a major impact on the amount of time spent in active mode; voltage regulators or references utilizing external decoupling caps can take milliseconds to settle. Therefore, it is important for a systems designer to analyze the overall wake-up and settling time for both the digital and analog circuitry to factor in the true cost of this wasted energy. State of the art devices as the Ambiq Apollo have deep sleep currents as low as 100 nA with RTC on (Ambiq 2015).

3.2.2 IO Architecture

Peripheral subsystems in microcontrollers include an extensive set of peripherals needed to connect to the wide variety of sensors available on the market. Although MCUs traditionally feature low bandwidth interfaces like UART, I2C, I2S or standard SPI, they lately include also higher bandwidth peripherals like USB or camera and display interfaces. Low bandwidth peripherals are usually attached to a shared bus and high bandwidth peripherals are usually connected to the system bus. Traditionally, the I/O was constantly supervised by the CPU and the CPU was responsible of handling peripherals events and regulate data transfer to/from peripheral and memory. Recent MCUs have increased the complexity of the I/O subsystem to support more power modes to be able to selectively turn on peripherals only when needed (Figure 3.3).

Further reduction in power can be obtained by increasing the intelligence of the peripherals and have they run without CPU supervision while the CPU is in sleep mode. The "smart peripheral" approach is used more and more in the most advanced MCUs. It has many variants with different names, but the general idea is the same. For example ATMEL "SleepWalking" features in the latest SAM-L2 family enable events to wake up peripherals and put them back to idle when data transfer is done without any CPU intervention for instance using a DMA (Atmel 2015). Another example is the Renesas' RL78 which has a "snooze mode" where the analog-to-digital converter (ADC) operates while the processor is asleep. An I2C slave or CAN controller can watch for an address before it captures incoming data and then wakes up the CPU (Renesas 2014). Cypress Semiconductor's PSoC flexible family was one of the pioneers

with their configurable digital and analog peripherals. Part of the configuration is the ability to link peripherals together without the CPU being involved. The company's latest PSoC 4 BLE (Bluetooth Low Energy) can operate the BLE radio while the CPU is idle (Cypress et al. 2016). Microchip's Configurable Logic Cell (CLC) highlights the more conventional approach to configurable peripherals. A microcontroller may have one or more CLC blocks. Each block has a selectable set of inputs and outputs with limited logic capability so the output of one peripheral can be fed to another. As with the PSoC, linked peripherals can often handle simple algorithms faster than the CPU. ST Microelectronics' STM32 has "autonomous peripherals" that use a "peripheral interconnect matrix" to link peripherals together. Timers, DMAs, ADCs, and DACs can be linked together. Or with their latest "Analog Chain" where comparators, references, DACs and ADCs can be interconnected to generate complex triggers (Microelectronics 2016). The trend towards more functionality and complexity is highlighted by Microchip's PIC16F18877 family. It has a 10-ADC tied to a computational unit that can do accumulation and averaging. It can even do low-pass filter calculations in hardware. The CPU can sleep until a filtered result exceeds programmed limits (Microchip 2016).

3.2.3 Data Processing

Reduction of energy consumption for the data processing part should be improved under different aspects, by improving the micro-architecture of the CPU increasing the amount of data that can be processed per cycle, or by optimizing the circuit design to reduce the power consumed per cycle.

The Architecture of CPUs used in MCUs has rapidly evolved in the last years, moving quickly from the 8 bit architectures to the now widespread 32 bit ARM Cortex-M architectures. The architectural evolutions are driven by the constantly increasing demand for performance for the always more complex applications. The vast majority of CPUs used in microcontrollers are single issue machine in which instructions are executed in order. The only exception today is the Cortex-M7 recently released by ARM which has a dual-issue pipeline. The choice of simple single issue micro-architecture is mainly dictated by energy efficiency for the target performance. We are not yet in the performance range to justify multiple-issue superscalar architectures. Engineers focused more on optimization to the micro architecture to improve as much as possible the IPC (instruction per cycle) and the data level parallelism.

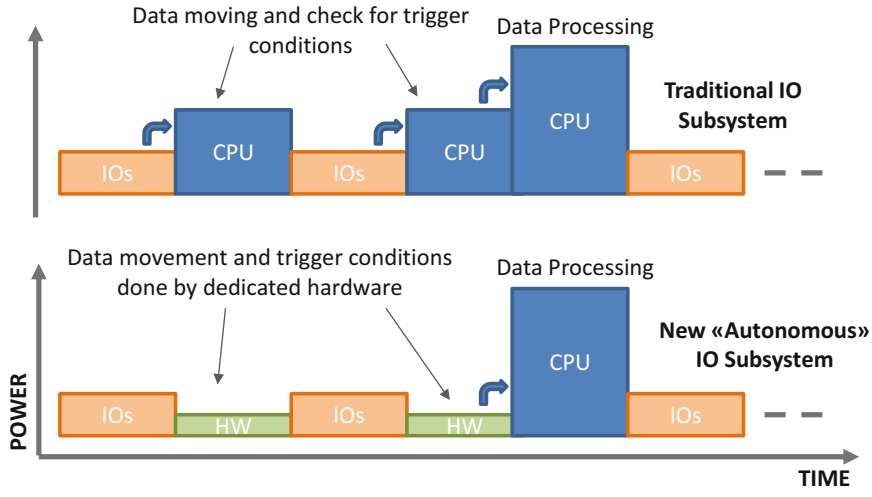

Fig. 3.3 Power profile of new "Autonomous" I/O subsystem

The most common improvements include for example hardware loops to speed up loops on tiny kernels typical of most DSP applications. It is usually done with one or more instructions that setup a dedicated logic which controls the number of interactions of a loop and the fetch stage, with the benefit of reducing the amount of cycles needed to compare and jump back at the end of a loop. The benefit is huge when small kernels are considered (Gautschi et al. 2015).

Other non DSP-specific improvements are loads and stores with pre- and post-increment whose main usage is to reduce the number of instructions needed to access consecutive memory locations.

More on the DSP optimizations we have for example the use of SIMD (Single instruction Multiple Data) where a single arithmetic instruction can operate on multiple data. On high-end CPUs this is often coupled with wider access to memory to handle multiple data at the data path width. On microcontrollers it is much more common to use their less power hungry version based on data size lower than the data path width. It is common to have the possibility to process 2 half-words or 4 chars with a single 32bit instruction. Good examples are the DSP extensions introduced in the Cortex-M4 (ARM 2010).

Architectural optimization to support single cycle multiplications and dedicated instructions for MACs, as well as hardware support for saturation arithmetic to better handle fixed point arithmetic, are other examples of how CPUs are extended to increase energy efficiency during the execution of DSP algorithms.

Those were the optimizations for the data and control logic but with the introduction of the ARM cores we also saw improvements on the instruction side. The ARM7TDMI ARM introduced the THUMB instructions, which allows some instructions to be coded using 16-bit instead of 32-bit, reducing the pressure on the instruction memory, the overall code size and instruction memory requirements. Table 3.1 summarizes the main features of few MCUs widely used for ultra-low-power applications, while Figure 3.4 shows their typical architecture.

Table 3.1 Summary of commercial low power MCUs

MCU	16 F1503	MSP430FR6x	SAML21x	Kinetis KL17	EMF32 Pearl Gecko	Apollo	STM32F745xx
Instruction set architecture	PIC	MSP430	ARM Cortex M0 +	ARM Cortex M0 +	ARM Cortex M4	ARM Cortex M4	ARM Cortex M7
Datapath	8-bit	16-bit	32-bit	32-bit	32-bit + FPU	32-bit + FPU	32-bit + FPU (dual issue)
Memory	I: 2 bB SRAM	128 kB FRAM	256 kB FLASH	256 kB FLASH	256 kB FLASH	512 kB FLASH	1 MB FLASH
	D: 128 B SRAM		64 kB SRAM	32 kB SRAM	32 kB SRAM	64 kB SRAM	340 kB SRAM
Max Freq. (MHz)	20	16	48	48	40	24	216
Current (µA/MHz)	30	100	35	54	60	34	700
Deep sleep current (µA)	0.02	0.02	0.2	0.28	0.02	0.12	2
State-retentive current (µA)	NA	0.02	1.3	1.96	1.4	0.193	2.75
Retentive deep sleep	No	Yes	No	No	No	No	No

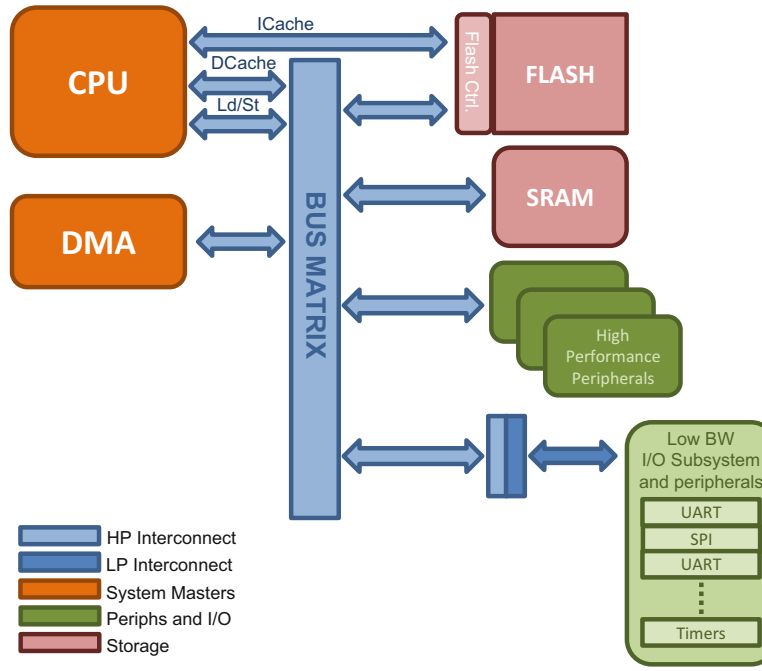

Fig. 3.4 Typical architecture of off-the-shelf MCUs for IoT

3.2.4 Non-volatile Memories

As briefly mentioned, before energy spent to save and restore the state before and after a deep sleep state has a deep impact on how often and for how long a deep sleep state could be used. Such cost depends almost entirely on the type of memory used for the storage. Embedded flash memories are the most common type of non-volatile memory used but, due to their high write energy and low endurance, they are not suitable to be used for state retention during frequent power cycles (Chap. 6). In today microcontrollers, state retention is implemented by keeping active part of the circuit, with the consequent reduction of the effectiveness of the deep sleep state.

Recent developments in non-volatile memories show many possible solution to this problem. Texas Instruments has recently included in their MSP430 family some microcontrollers based on FeRAM (Texas 2011). FeRAM have a structure very similar to DRAM where a bit cell is made of one transistor and one capacitor and they share with DRAM the high speed. The capacitor is implemented by using a ferroelectric material (usually PZT lead-zirconate-titanate) which is polarized in two possible states that are kept without requiring any refresh. Other types of memory currently under study and close to commercial applications are MRAM, STT-MRAM, and PCRAM. Magnetic RAM (MRAM) are based on memory cells constituted of two magnetic storage elements, one with fixed polarity and one with switchable polarity. Depending on the state of the switchable element the resistance of the overall cell changes and that is what is sensed by the read circuitry. STT-MRAM is a particular implementation of MRAMs more suitable for scaled technology. It uses spin-polarized currents enabling smaller and less power demanding bit cells. Currently, STT-RAM is being developed in various companies including Everspin, Grandis, Hynix, IBM, Samsung, TDK, and Toshiba. Due to its easy integration in the CMOS process as well as its performance, power and scalability properties,

it is one of the most appealing solutions. Chapter 7 will present a detailed explanation. Another example of NVM currently investigated by the industry is the Phase Change RAM (PCRAM) in which the bit cell is made of materials that can exist in two phases (e.g., crystalline and amorphous) and result in different resistance. A more detailed description of NVM memories in the context of IoT architectures is presented in Chaps. 6 and 7.

3.2.5 A Step Forward: Near-Threshold MCU Architectures

Today's low-power MCU architectures operate in the super-threshold domain during active phases of computation, relying on duty cycling and heavily optimized deep sleep modes to improve energy consumption. For example, some commercial devices, such as Ambiq Apollo (Ambiq 2015) provides a sub-threshold RTC to minimize deep-sleep power (~300 nW), but it operates at 0.9 V, which is only 100 mV below the nominal operating voltage for the 90 nm process technology utilized for its implementation, providing active power efficiency similar to other commercial MCUs. Although this approach provides a very low power for applications requiring low computational workloads (e.g., a temperature sensor wakes up the microcontroller once per minute to transmit sensed data) the situation drastically changes when the applications require nearly always-active operation. In this scenario, the energy consumption of a microcontroller can increase by up to three orders of magnitude (i.e., from tens of μW to tens of mW, on average). Near threshold computing is emerging as a promising approach to achieve major energy efficiency improvements of ultra-low-power digital architectures (Dreslinski et al. 2010). However, this comes at the cost of performance degradation and increased sensitivity with respect to process, voltage and temperature (PVT) variations. While the performance degradation can be managed by adjusting the operating voltage of the device according to the required performance target, compensation of PVT variations has to be achieved in a transparent way with respect to the end user, which cannot be aware of the operating conditions of the device.

3.2.6 Compensation of Process and Environmental Variations in Near-Threshold

The variability of ultra-low-power devices operating at low voltage has been extensively analyzed in the last few years from the research community (Alioto 2012). Some of the approaches leverage design-time techniques to improve resiliency of the circuits with respect to process and temperature variations, including standard-cell design, clock tree optimization, and automatic synthesis. All these techniques are extensively analyzed in Chap. 4. On top of design level techniques to mitigate the impact of variations, their compensation is usually addressed at the architectural level, integrating mechanisms able to probe the PVT conditions of the circuit and aging, and provide a feedback to knobs exposed at system level that allow to compensate the variations by adjusting the supply voltage of the circuit.

Widely explored approaches to implement the probing circuits to adapt the supply voltage are process monitoring blocks (PMBs), canary circuits, or razor flip-flops (Ernst et al. 2005). PMBs are generic structures implementing ring oscillators with different characteristics (e.g., PMOS only, NMOS only, inverters with long wires). Depending on the specific oscillator probed from a PMB it is possible to acquire specific information about the process condition of PMOS and NMOS, temperature, or voltage. Although this approach is generic, as it does not require design of specific hardware for each chip, some additional logic (e.g., a lookup table) or software processing is required to merge the data and provide a feedback to the actuator. A canary circuit is a replica of the critical path plus some delay element, often programmable, which makes the monitor super-critical (Calhoun and

Chandrakasan 2004). As canary circuits are designed to clone a specific critical path, they are not generic components, but they provide a more direct information on the criticality of the operating point. The global process variations as well the environmental conditions can be monitored with both PMBs and canary circuits, and the supply voltage can be adapted according to these conditions. However, there are still considerable margins to be assumed to ensure a reliable operation of the circuit. For example, local process variations and IR drop cannot be covered, as the monitor circuit is a copy of the critical path placed in a different location. A yet more direct approach is to use the speed monitor within the respective circuit block. The razor concept (Ernst et al. 2003; Blaauw et al. 2008) provides energy reduction guaranteeing reliable operation by lowering the supply voltage to the point of first failure. Timing failures are detected by shadow latches placed on the critical path of the design controlled using a delayed clock, eliminating all margins due to global and local PVT variations. By comparing the values latched by the flip-flop and the shadow latch, a delay error in the main flip-flop is detected. The value in the shadow latch, which is guaranteed to be correct, is then used to correct the delay failure. The main drawback of the razor approach is that is very design specific, requiring major manual adjustments to the microarchitecture of the SoC, as no automatic razor flops mechanism is available in commercial synthesis tools. This makes the adoption of this approach very challenging for a wide set of designs, and impossible when the IP cores are provided as hard macros or encrypted netlist by processors or IPs providers.

Although the most traditional way of compensation for PVT variations leads to adjust the supply voltage of the circuits, other approaches have been explored at the architectural level: clock gating the specific pipeline stage where a fault is detected (Ernst et al. 2003), using counter flow pipelining (Charles et al. 1994), through architectural replay (Blaauw et al. 2008), or through insertion of reconfigurable pipeline stages (Bortolotti et al. 2013). The main benefit of the architectural compensation or error recovery mechanism is that they can react in a single clock cycle to the detection of a fault or a critical situation. On the other hand, once again, they are very invasive from the micro-architectural viewpoint. Alternative approaches to compensate for PVT variations rely on the adoption of adaptive body biasing (Tschanz et al. 2002, 2007). This approach has several advantages with respect to modulation through supply voltage. First of all, dynamically adapting threshold voltage of transistors only increases leakage power of the circuit, while adaptation of supply voltage has an impact on both leakage and dynamic power. With body biasing it is possible to apply independent polarization to PMOS and NMOS. Hence, it is possible to optimize the leakage power consumption of a circuit by applying asymmetric body biasing. Moreover, body biasing is very effective when the device operates in near-threshold, as in this operating region small changes of the threshold voltage of transistor provide significant increase (or reduction) of the operating frequency (Rossi et al. 2016a). Finally, since the polarization of the PWELL and NWELL only requires transient currents, modulation of body biasing can be implemented with simple and energy efficient circuits (e.g., charge pumps), as opposed to supply voltage control, which requires the adoption of DC/DC or voltage regulators. As a drawback, the adoption of body biasing to address large PVT variations is somehow challenging due to the limited capabilities of bulk technology to provide extended body bias ranges. For this reason, joining the degrees of freedom provided by voltage scaling and adaptive body biasing appear as a suitable solution to compensate for PVT variations while tracking the best energy operation.

3.3 From Single Core to Multi Core

The trend to develop multicore-systems is a general tendency for several high-end embedded, desktop and server platforms. Energy efficiency requirements have forced processor developers to add multi-core capabilities instead of increasing the system clock frequency in single-core

systems (Parkhurst et al. 2006). Cache coherency and memory bandwidth determine the architecture of such multi-core systems, and when private caches are involved, support for cache coherency across the entire multi-core system is desired, as it enables the software running on embedded processing system to balance load and allocate tasks seamlessly between cores.

Cache coherence protocols must react immediately to writes, and invalidate all cached read copies in the private cache banks, and this is source of significant complexity. Complexity translates into cost, both from silicon and energy perspective (Martin et al. 2012). Data synchronization in multi-core systems prevents data from being invalidated by parallel access whereas event synchronization coordinates concurrent execution. One common mechanism to achieve data synchronization is a lock. Event synchronization forces processes to join at a certain point of execution. Barriers can be used to separate distinct phases of computation and they are normally implemented without special hardware using locks and shared memory (Culler and Singh 1999). Efficient data transfer between cores in a multi-core system is critical for balanced system performance. A multi-core environment introduces high demands on the interconnect infrastructure and the interconnect needs to be able to handle multiple streams simultaneously. Complex interconnect are widely used to sustain bandwidth and QoS requirements (e.g., AXI ACE), and again complexity is translated into cost.

On the other side, micro-controller systems can take benefits of lean a simple architecture with respect to high-end single/multi-core systems. For this reason, they are much more attractive for a wide category of IoT devices. For instance basic micro-controllers typically have no data/instruction caches, while multicores needs complex memory hierarchies to sustain bandwidth and computational power requirements. But the current trend in IoT devices is that, modern use cases are demanding more and more computational power (eg. speech and image recognition, surveillance etc.). To achieve this target with a prefixed energy budget, standard micro-controllers are not sufficient to provide this amount of computational power, and the trend is to migrate from a single-core to a multi-core architecture, while maintaining low the complexity of the memory hierarchy to target high energy efficiency.

For this reason, multi-core architectures are beginning to penetrate the microcontroller business segment: recently, a new class of heterogeneous dual-core MCU products appeared in the market. These devices have cores with different instruction set architectures (i.e., Cortex M0 and Cortex M4) (NXP Semiconductors 2015) and rely on the architectural heterogeneity to achieve energy efficiency with a principle similar to the mid-to-high-end big-little multi-cores (Peter 2011). In these architectures, the little core is mainly meant for control of peripherals and low workload tasks, while the big processor perform heavy data processing tasks. The main advantage of these architectures is that cores are completely decoupled, easing the partial shut-down of the platform when one of the cores is not used. On the other hand, this decoupled architecture, where each core has its own binary and process data on private memories, doesn't allow for easy and fair workload distribution among cores, and requires to keep private copies of data-buffers, which dramatically reduces the computational efficiency of the platform when true data level parallelism has to be exploited.

A more convenient approach to the design of parallel low-power architectures is a tightly coupled clusters sharing, similarly to what happens in GPGPUs, a multi-banked L1 memory. With respect to traditional architectures featuring per-core private data memory, where data buffers are processed on the local, low-latency L1 memory, and shared with the other cores through a higher high-latency L2 memory, the tightly coupled data memory approach significantly reduces the data-sharing overhead, as the memory banks can be accessed by all cores with a fixed latency (usually one cycle). This way allows to efficiently exploit both data and task parallelism, as opposed to more traditional private L1 memory scheme that allows to run efficiently only task parallel applications, enabling better performance scalability when the required workloads allow for true data-level parallelism.

3.3.1 Energy Benefits and Challenges for Parallel ULP Processors

Figure 3.5 shows the tradeoff for multi-core vs. -single-core systems, under different workload conditions. Each workload represents a specific voltage-frequency pair that exploits the required computation power. Under high workload requirements the single-core is less energy efficient when compared to multi-core due to the quadratic dependency of supply voltage with dynamic power (Dogan et al. 2011). Indeed, to achieve the same throughput the single-core needs to operate with a supply voltage much higher than that of the multi-core, where CMOS devices are far from their maximum energy efficiency point. However, when the voltage is close to the threshold, leakage become dominant and circuit slowdown increases, so from an energy perspective, single core is much more efficient, therefore, this region is not interesting for always-on parallel computing over multiple cores (Fig. 3.5).

On the other hand, when we introduce a deep-sleep mode typical of the MCU domain, under low workloads, the multicore solution is still attractive. In Fig. 3.6, in case of a single core execution, energy is consumed in the whole active period, then when computation is done, the system is put in deep-sleep. In the active period, the energy drained by the core is given by the sum of core and system energy. Assuming to run the same application on a multi-core platform, then the application runtime (active region) will decrease (in the ideal case, will be N times smaller, where N is the number of cores). In this region, we have multiple cores and the system draining energy from the power supply, and assuming that the system power is the same in both cases, and that core energy is the same, by reducing the active period, some amount of energy is saved, thus better energy efficiency is achieved.

This scenario highlights the importance of a power management strategy at the system-level to develop efficient shutdown policies of unused cores and workload consolidation policies for minimizing the dynamic and leakage power consumption for the idle cores (Rossi et al. 2016a). An effective strategy to eliminate dynamic power of unused processors during the execution of sequential portions of code is architectural clock gating. Indeed, joining hardware support for synchronization mechanism with architectural clock gating of cores provides significant energy savings during execution of sequential or not perfectly balance code, while minimizing the synchronization overhead, not present in single-core platforms. However, even at low speed levels a processor consumes a significant amount

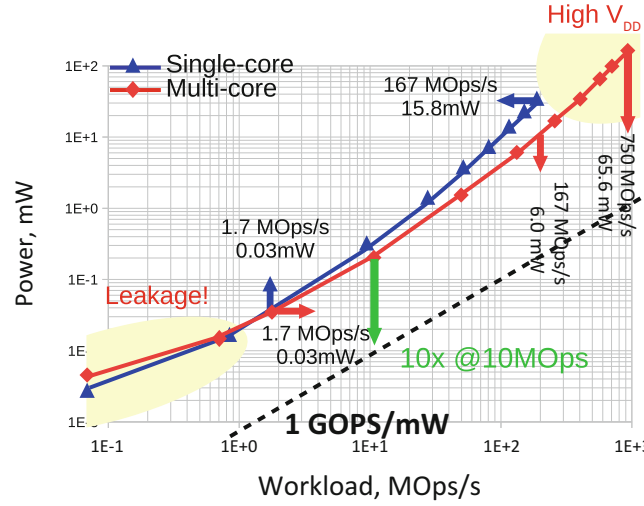

Fig. 3.5 Power efficiency of multi-core vs. single core under different operating points. Each operating point is function of both workload and frequency/voltage that enable such performance

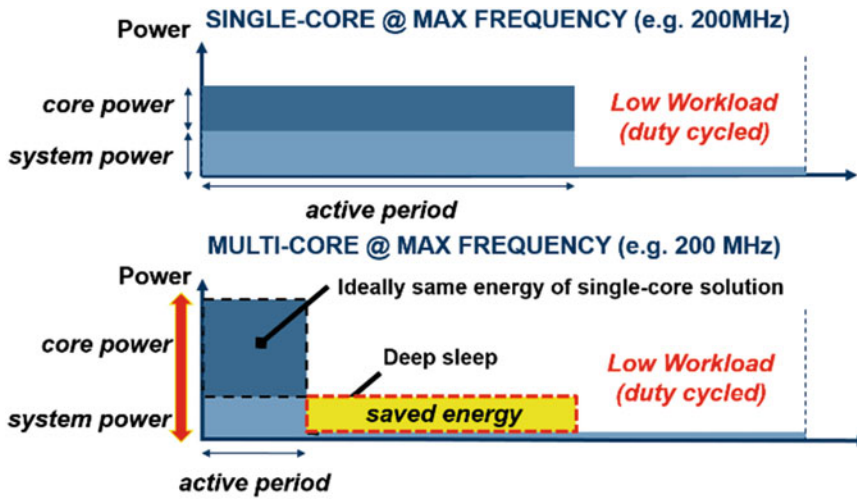

Fig. 3.6 Energy efficiency Multi-core vs Single core on parallel application

of static energy, caused, for example, by leakage current. A possible alternative to reduce both leakage and dynamic power consumption of multi-core platform is per-core DVFS. Exploiting a per-core DVFS scheme, each core is allowed to operate with and independent voltage and frequency, thus theoretically achieving the best possible energy efficiency for each given workload, and can be shut down when idle. However, this comes with level shifters and dual-clock FIFOs at the boundaries of every core introducing significant data sharing overhead, due to the handshaking required to implement the clock domain crossing between the processors and the data memory. Moreover, the small form factor of processors typically adopted in this domain might not justify the costs of a so complex solution (i.e., per-core DC/DC converters or LDOs are required). A simpler but effective approach to reduce leakage power of unused cores lies in per-core power gating. Exploiting a per-core power gating architecture all the cores belong to the same frequency domain, and operate at the same voltage when active, eliminating clock synchronization and level shifting overhead, but they are able to shut down independently when idle. Although this approach is effective as it can reduce leakage power by several orders of magnitude, it requires a ring of PMOS transistors around each core to implement the power gating. This significantly increases the area of the cores, and most important significantly degrades the performance of the circuits when operating at low voltage. Indeed, the presence of one additional PMOS stacked over the pull-up network of standard cells has been shown to lead significant performance degradation in the near threshold operating region (Alioto 2012). Finally, power gating does not allow for state retention, as during the idle phases all the cells within a gated region are not supplied. In this context, implementing a state retention mechanism requires more complex state-retentive memories and flip-flops, with the associated overheads in terms of area, power, and additional routing required to bring to all these components the retention voltage. Another alternative to manage leakage power of idle cores is reverse body biasing (RBB). RBB can reduce leakage power by up to one order of magnitude (Rossi et al. 2016a). Although this is not enough for implementing deep-sleep modes typical of MCUs, it provides an effective and fast (less than 100 ns settling time) way to reduce the leakage power of idle cores in a ULP parallel architecture. Indeed, in contrast to multiple voltage domains and power gating approaches, this architecture has minimal overhead in term of isolation, as it usually requires a small ring of deep N-well around the region to isolate P-wells

and N-wells in a typical triple well process. Moreover, it does not require level shifters and isolation with power gating transistors, since all the regions belong to the same voltage domain. Finally, employing leakage reduction through body biasing allows standard registers to maintain the state during the sleep phases, thus avoiding the usage of more complex state-retentive flip-flops and the related overhead in terms of power routing and area.

3.3.2 Memory Hierarchy for Parallel ULP Processors

IoT is a network of physical objects that can refer to a wide variety of devices. In such scenario it is not possible to derive a single architecture that fits all the possible cases, hence the device architecture is tailored upon the computation requirements of the use case, and energy constraints. The memory hierarchy of such architectures plays a dominant role in the performance/energy metrics: for example, in real time systems caches would never exist for the fact that different layers of memory hierarchy have such different access/speed characteristics, and the global access time, e.g., in case of miss, is not predictable, and may lead to deadline violations and system failures (safety issues). Moreover, it is hard for the software to explicitly control and optimize data locality and transfers (Kalokerinos et al. 2009). Scalability issues are real in multicore system, where the cache coherency complexity puts a limit to the number of cores that can be integrated in the chip.

Data Scratch-Pad Memories (SPMs on Fig. 3.7a) are a valid alternative to caches for on-chip data memories. SPM is small on-chip memory bank (e.g., SRAM) mapped into the processor's address space, and tightly coupled with the processor pipeline. SPMs are faster and consume less energy with respect to larger or off-chip memories, and their inherent predictability have made them popular in real-time systems for instance. SPM also offer better scalability by allowing explicit control and optimization of data placement and transfers. These systems, then employ specialized data memories hierarchies for better efficiency for targeted data. However these memory structures need a mechanism to move data from the different levels of the memory hierarchy to the SPM, and transferring data to/from this address space (explicit communication) may lead to inefficiencies that can vanish the benefits of specialization. Remote direct memory accesses (RDMA) is a wide used technique to move data from/to the SPM from the different levels of the memory hierarchy. This technique is efficient if the programming overhead to trigger the DMA transfer is minimized, and it is affordable in the cases when the producer knows who the consumers will be, or when the consumer knows its input data set ahead of time. Moreover,

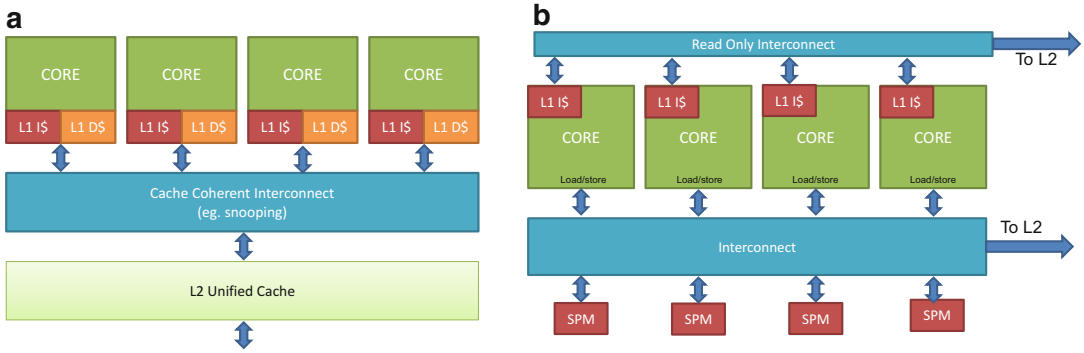

Fig. 3.7 Overview of cached (**a**) vs. cacheless (**b**) multicore system. Private caches pose a limit to scalability, increase complexity (e.g., snooping), and introduce non uniform access time. Cacheless approach removes any coherency issues, and improve energy efficiency (simple architecture), and predictability

SPMs can replace data caches only if they are supported by an effective compiler. The mapping of memory elements of the application benchmarks to the SPM is done mostly for data, since for instructions there are no coherency issues, therefore having a cache is highly desirable on this side. However, the major benefit to adopt SPMs vs. on-chip caches is the better energy efficiency, better access timing, and better silicon area usage. Caches using static RAM consume power in the range of 25–45% of the total chip power, and energy can be reduced by 40% if replaced by SPM (Banakar et al. 2002). The Area-Time product metric can be reduced to 46% in favor of the SPM.

3.3.3 SPMs in the Near-Threshold Region

On the other side, with the introduction of near-threshold or even sub-threshold operation to digital circuits, memory design has gained renewed attention as being more susceptible to the side effects of low voltage operation.

A classic SRAM cell is a ratioed circuit relying on the relative drive strength of the transistors involved, and parametric variations of the individual devices can lead to functional failures of the cell. Low-voltage SRAMs use different supply voltages for the digital domain and memories. This approach entails additional complexity on system level (level shifters and power distribution). Other approaches for designing low-voltage SRAMs include design of 7 T, 8 T, or 10 T bit-cells, decoupled read/write operations, read and write assist techniques, and adaptive and resilient SRAMs design (Chap. 5).

One important factor is given by the leakage power, which is proportional to the chip area, which generally is dominated by memories. Scaling the voltage in the NTC area can leverage to 10X better static power consumption, which in IoT devices can lead to huge power improvement since those devices are most of the time in standby. If, from one side, scaling the voltage can lead to some power benefit, on the other side, SRAMs become slower and slower and more susceptible to process variation, leading to failures when the voltage is below 0.6 V in a 28 nm process (Teman et al. 2015).

Figure 3.8 shows the power breakdown for a multicore system, a cluster of four RISC processors, equipped with private instruction

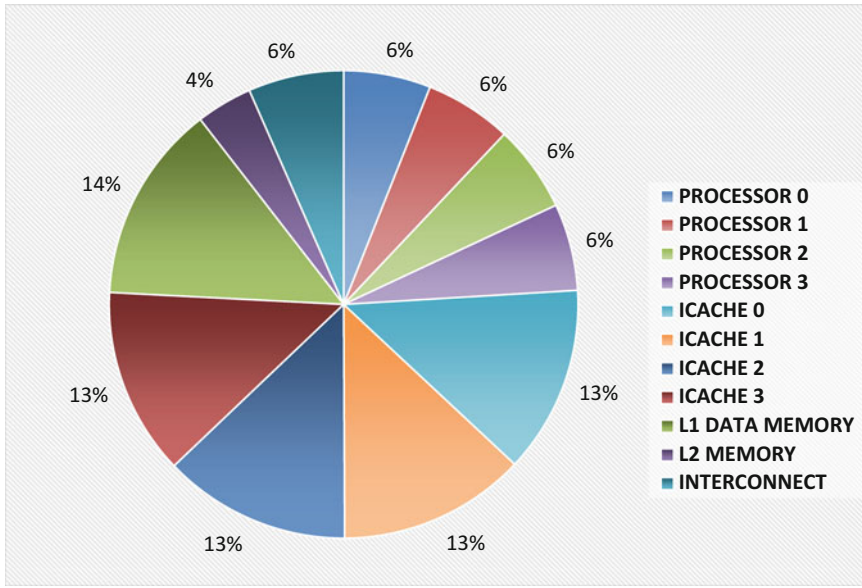

Fig. 3.8 Power Breakdown for a QuadCore with private instruction caches, based on monolithic RAM memories

caches, based on SRAM memories (TAG and DATA). As shown in the figure, private I$ are responsible for more than 50% of the power budget. One of the alternatives that has been proposed in recent years to overcome some of these shortcomings is to synthesize memories from standard cells. Standard Cell Memories (SCMs) are soft-digital blocks, described in a hardware description language (HDL) at register-transfer level (RTL), and mapped to Standard Cell (SC) libraries. By using SCMs instead of SRAMs, designers can define each memory block according to the specific needs of each component and achieve the specifications required by their design as part of the standard digital design implementation flow. One of the primary advantages that SCMs have over traditional SRAM is their robustness, especially at low-voltages. Since SCMs are constructed exclusively from SCs, they scale along with the core digital logic, and continue to operate well below the limit of standard 6 T SRAM arrays (∼600–700 mV). This advantage is accompanied by a loss of memory density, since the basic SCM storage cell is much larger than a standard 6 T SRAM bitcell. However, this trade-off is common to all low-voltage SRAM solutions (Teman et al. 2015). SCM offers better energy efficiency with respect to SRAM based SPMs In the application example provided in (Meinerzhagen et al. 2010) it was shown that the use of the considered SCM architecture reduces the power consumption 37% compared to the use of SRAM.

3.3.4 Architecture of Memory Subsystem for Parallel ULP Processors

The conventional memory hierarchy for low-power multi-core architectures is generally composed by a private L1 subsystem and a shared L2 level (as depicted in Fig. 3.7a). In the private L1 subsystem, local data keeps a copy of the accessed data, potentially replicating the same data in the different private local memory subsystem. If the L1 is based on data caches, and cores are working on the same dataset, then all those access patterns are cached locally, leading to data replication, which results in reduced effective memory capacity, and poor energy efficiency. Secondly, L1 private cache need to be coherent, and as highlighted in the previous section, cache coherency pose serious limit to multi-core scalability, and it brings a not negligible intrinsic cost (power).

Moreover, for each processor, the available memory capacity is limited by its local memory size, and each inter-core communication is expensive because is done though message-passing mechanism (mailbox etc.). The major benefit of the private caching scheme is a lower cache hit latency.

On the other hand, the shared caching scheme always maps data to a fixed location. Because there is no replication of data, this scheme achieves a lower on-chip miss rate than private caching (because of large aggregate cache capacity), and simple and efficient mechanism for inter-core communication, and finally no coherency issues (at L1 level). However, the average cache hit latency is larger, because cache blocks are simply distributed to all available cache slices. To mitigate this penalty, several architectural solutions have been proposed (e.g., victim cache).

However, both in case of shared or private scenarios, the data cache is not attractive for an energetic point of view. Secondly a wide range of applications from image and video processing domains have significant data storage requirements in addition to their computational requirements and recent and studies (Benini et al. 2000) have shown that regular data access patterns found in array-dominated applications can be better captured if SPM is employed, instead of a more traditional data cache.

Based on the assumption discussed before, in ultra-low-power multi-core systems, the data-cache is replaced by several SPM blocks (Fig. 3.7b) that are linked together in a multi-banked shared data memory, and referred as Tightly Coupled Data Memory (TCDM). TCDM is tightly integrated on the processors load/store unit interface, and shared through a lean, fast and thin distributed crossbar, that provides shared access to several SPM banks.

The TCDM therefore implements a multi-ported multi-banked shared data memory, where each processor load/store interface is plugged directly on one master port of the TCDM, while the SPM memory banks are connected on the slave side. The internal arbiter handles routing and flow control of request coming from different processors and directed to the same SPM bank. Memory banks are mapped on the global address space, and are accessible from each master port with a simple flow control protocol (req/grant). To reduce the collision probability, those banks are mapped with a word level interleaving scheme, meaning that adjacent addresses are mapped on adjacent banks, with a typical granularity of 32 bits (same as the processor architecture). To further decrease the pressure on memory side, the number of SPM banks are doubled with respect the number of processors. Since TCDM is a shared architecture, it loses the determinist latency feature of private SPM, but on the other side, the embedded crossbar ensures a maximum worst case latency (with round robin arbitration, the maximum latency for the worst case is equal to the number of cores).

To make this approach affordable, shared and private SPM schemes must behave in the same way in the case of best case (no collision on shared accesses). The processor load/store interface requires to be carefully optimized, since SPM are now shared, and there is the crossbar logic in between processor pipeline and SPM. Second, in case of conflicts (e.g., two or more processors making a request on the same shared memory bank) the processor pipeline must be frozen until the request is granted, therefore and additional stall must is added in the processor pipeline architecture.

Private instruction caches are usually able to achieve higher speed, due to their simpler design (deep integration with processor pipeline), but similarly to private data cache, the reduced capacity (vs. the aggregate capacity of shared instruction cache) lead to an increase in the miss ratio. Although large private instruction caches can significantly improve performance, they have the potential to increase power consumption, therefore are not affordable for ultra-low-power multicore systems.

The optimal solution from a point of view of the energy saving is the shared instruction cache

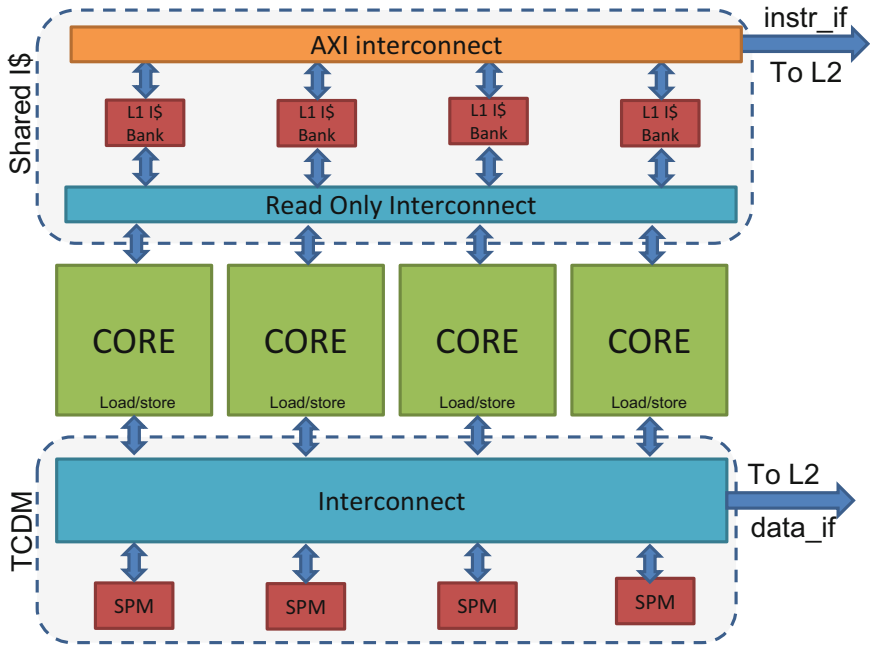

Fig. 3.9 Cluster of four processing elements, with tightly coupled data memory (SPM) and shared instruction cache

(Fig. 3.9). This approach can is an attractive solution to improve performance and energy efficiency while reducing the footprint area of the chip, since the total capacity seen by each processor is given by the aggregate capacity of the whole shared instruction cache (Loi et al. 2015). The shared instruction cache has no coherency issues (cache is read only), and it is implemented as multi-banked, multi ported read-only cache. Each cache bank is shared through a thin, fast read-only interconnect (which arbitrates the fetch request among the available cores), and mapped at cache line level address interleaving to better distribute accesses in all available cache banks (to reduce the collision rate). Similarly to the TCDM, the latency is not deterministic because both hot/miss and local collision may happen. Local contention may lower the instruction per cycle metric (IPC) in each core, so some additional features are required to reduce performance degradations.

Adding a pre-fetcher in between the shared instruction cache and processor instruction-fetch interface, with the capability to fetch one cache line in a single cycle (wide interface) further reduce the local collision rate at cache bank level, leading to reduce the IPC, very close to 1. With these optimizations the shared cache can be considered a valid alternative to private cache subsystem, with a negligible overhead due additional resources, but with the big advantage of an increased memory capacity which finally leads to improve hit ratio and less refills though the L2 level hierarchy (with a sensible improvement of energy/power consumption).

Both shared TCDM and Shared instruction cache, can be implemented using SCM, making these architecture suitable even in the Near Threshold region, where ultra-low-power requirements are mandatory, and where SRAM based memory are not operative.

3.4 Design Example: The PULP Platform

PULP is a multi-core platform achieving leading-edge energy efficiency and featuring tunable performance across a wide range of workloads (Rossi et al. 2016a; Rossi et al. 2016b). The aim of PULP is to satisfy the computational demands of IoT applications requiring flexible processing of data streams generated by multiple sensors, such as accelerometers, low-resolution cameras, microphone arrays, vital signs monitors. As opposed to single-core MCUs, a parallel ultra-low-power programmable accelerator allows to meet the computational requirements of these applications, without exceeding the power envelope of a few mW typical of miniaturized, battery-powered systems.

3.4.1 SoC Architecture

The compute engine is a cluster with a parametric number (2–16) of cores (Fig. 3.10). Cores are based on a power-optimized micro-architecture implementing the OpenRISC ISA. Pipeline depth is optimized at four balanced stages, enabling full forwarding with single stalls only on load-use and mis-predicted branches. Further pipelining would not improve near-threshold energy-efficiency because L1 memory access time dominates cycle time, while register and clocking overhead increase power (Gautschi et al. 2014). The original OpenRISC ISA and the core micro-architecture is extended for energy efficient digital signal processing, supporting zero-overhead hardware loops with L0 I-buffer, load and store operations embedding pointer arithmetic, SIMD vector instructions and power management instructions (Gautschi et al. 2015). The cluster can be configured with either private or shared instruction cache with instruction broadcasting support. By coupling the shared I-cache with L0 buffers, it is possible to greatly reduce cache pressure, resulting in a much higher energy efficiency (Loi et al. 2015). The cores do not have private data caches, avoiding memory coherency overhead and greatly increasing leakage and area efficiency for data memory. A L1 multi-banked Tightly Coupled Data Memory (TCDM) acts as a software-managed shared data scratchpad. The TCDM features a parametric number of word-level interleaved

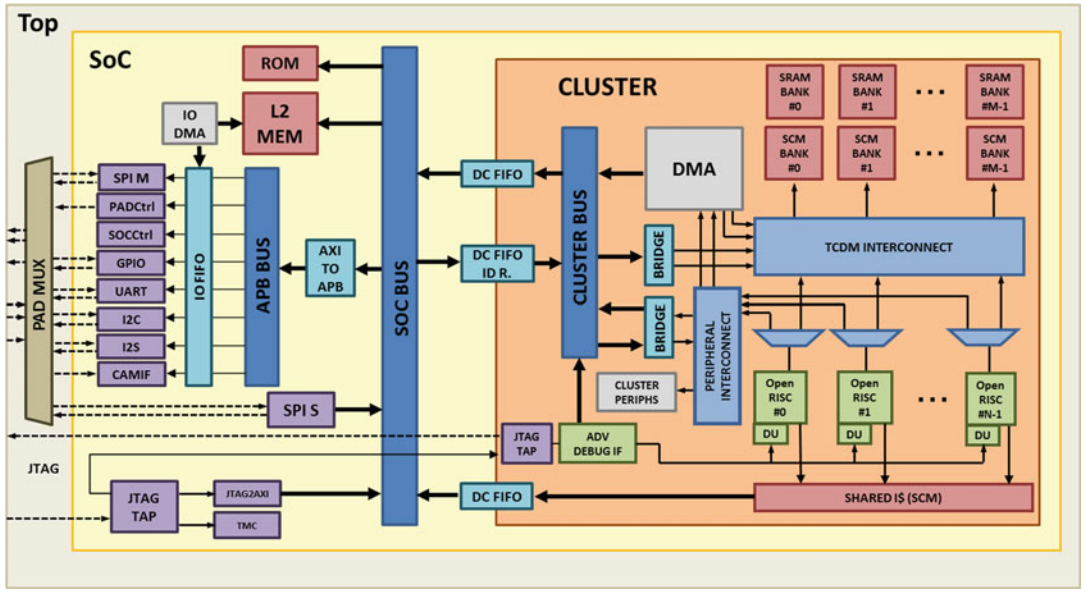

Fig. 3.10 PULP platform architecture

banks connected to the processors through a non-blocking interconnect to minimize banking conflicts (Rahimi et al. 2011). Each logical bank is implemented as a heterogeneous memory, composed of a mix of SRAMs and latch-based Standard Cell Memory (SCM) banks. Instruction caches can also be implemented using SCMs. While SRAMs achieve a higher density than SCMs (3x-4x), SCMs are able to work over the same voltage range as logic, extending the operating range of the whole cluster to the very limit of the technology; moreover, their energy/access is lower than that of SRAMs for the small cuts needed in L1 (Teman et al. 2015). Depending on the availability of low-voltage memories in the targeted implementation technology, different ratios of SCM and SRAM memory can be instantiated at design time. Reconfigurable pipeline stages, controlled by the processors through a memory mapped interface, can be optionally added to the TCDM interconnect to deal with the performance degradation and variability of SRAMs at low voltage.

The cluster has AXI4-compliant interfaces. Off-cluster (L2) memory and peripheral access latency is managed by a tightly coupled DMA optimized for low power with just ten cycles programming latency, up to 16 outstanding transactions and a private physical channel for DMA control for each core. Various peripherals are available for PULP SoCs, including SPI interfaces with streaming support, I2C, I2S, a camera interface, GPIOs, a bootup ROM and a JTAG interface for debug and test purposes. SPI interfaces can be configured in master/slave single or quad mode. PULP SoCs can operate standalone or as a slave accelerator of a standard host processor (e.g., an ARM Cortex-M microcontroller). To reduce the overall number of pads, and make low-cost wire bonding packaging of the SoC suitable for IoT applications, the IO interfaces can be multiplexed.

3.4.2 Power Management Architecture

To provide high energy efficiency across a wide range of workloads, the PULP cluster and the rest of the SoC are in different power and clock domains. Fine-grained tuning of the SoC and cluster frequencies is achieved through two FLLs (Frequency-Locked Loops) (Miro-Panades

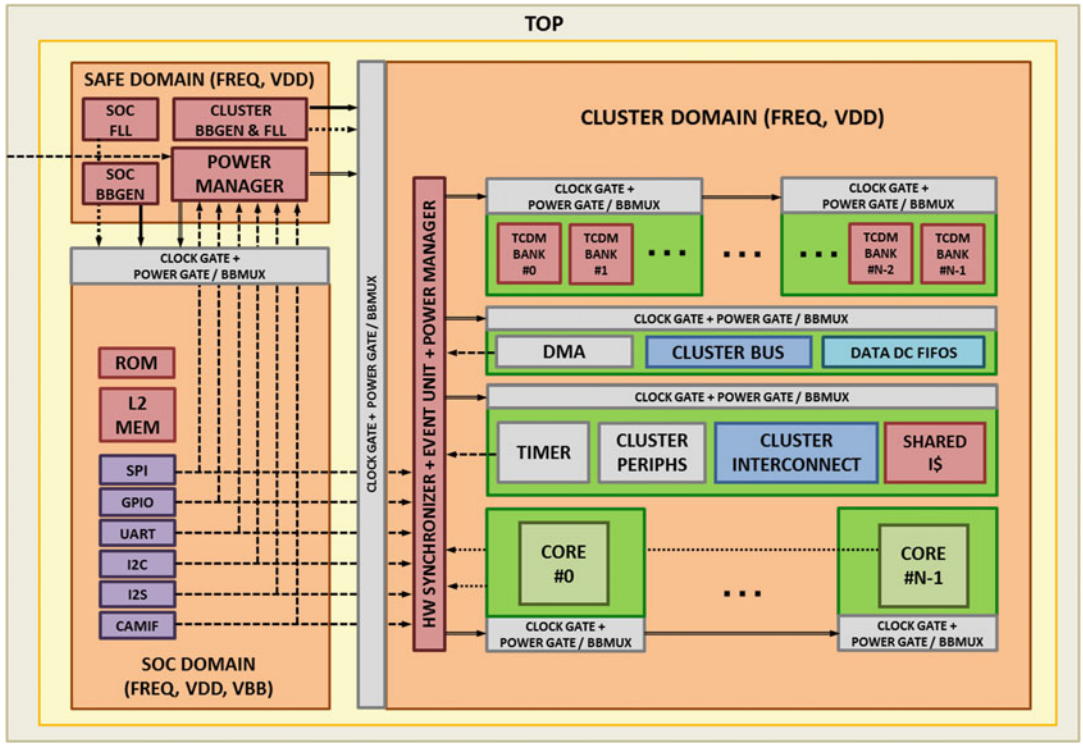

Fig. 3.11 Power management architecture

et al. 2014). Each processor within the cluster, as well as other data transfer, and memory resources can be separately disabled, clock-gated and reverse body biased when idle. The concept of fine-grain power management is shown in Fig. 3.11. In the PULP platform, the body-bias multiplexers allow to dynamically select the back-bias voltage of each processor, enabling ultra-fast transitions between the active mode and the sleep mode. Although the same concept applies also to leakage management with power gating, in the context of a near-threshold processor body biasing offers two key advantages for partial shut-down of small blocks. First it is intrinsically state-retentive, and does not require to increase the complexity of flip flops and memories with additional supply rails. Second, power gating requires a ring of transistors around the digital area to me managed, causing a relevant overhead for small blocks, and add a PMOS to the pull-up network of the digital logic, which degrades the performance in near-threshold. Depending on the required workload, the cluster is able to keep active an arbitrary number of processing elements, while the others consume zero dynamic power and up to 10x less leakage power. An event unit automatically manages transitions of the cores between the active and idle states. Processors can be put in idle state with a write operation on a memory mapped control register. After going in sleep mode, a core remains idle until a configurable event is triggered. Events can be issued by all IO peripherals, the DMA, and timers. Emergency and general-purpose events are also supported. The same approach is replicated hierarchically at cluster and SoC level. A power manager based in a safe domain controls the state of the SoC and cluster domains, managing dynamic and leakage power during sleep states and generating wake-up sequence when a termination event is received, for example due to the completion of a transfer from an IO peripheral to the L2 memory. This mechanism allows to minimize the intrinsic overheads of a relatively complex parallel computing platform, guaranteeing that only

the strictly necessary blocks are active during the phases of parallel or sequential computation, and data transfer. On the other hand, power gating is more effective to implement system-level deep-sleep modes of heavily duty-cycled applications, where the whole system needs to be managed for longer periods (tens to hundreds micro seconds) and leakage power needs to be reduced by several orders of magnitudes (100 nW–10 μW).

3.4.3 Programming PULP

OpenMP, OpenCL and OpenVX programming models are available for PULP, as they are well-known standards for shared memory programming, and widely adopted in embedded MPSoCs (Stotzer et al. 2013). GCC- and LLVM-based tool-chains and light-weight implementations of all these environments have been tailored to PULP's explicitly managed, scratchpad-based memory hierarchy. To achieve energy efficiency, however, it is necessary that the implementation of the different execution models efficiently exploits the ULP features of the hardware. The alternance of sequential and parallel code parts is inherent to all the programming paradigms based on the fork-join model. While minimizing the impact of Amdahl's law by extracting the maximum degree of concurrency in applications is paramount for every parallel system, for ULP parallel system the way idleness is implemented is key to achieve energy efficiency. Idle power of unused cores might be significant, causing huge energy efficiency drop. To this end, special hardware for accelerating key software patterns has been developed. The PULP software runtime integrates a clock-gating based thread docking scheme to eliminate dynamic power, coupled with Reverse Body Bias (RBB) to reduce leakage, when worker threads are idling (e.g., in sequential regions of the program). In addition, the key operations required in fork/join thread management are HW-accelerated, dramatically increasing the performance with respect to polling or event-based synchronization.

3.4.4 Extending PULP

When the application requirements are so strict that they cannot be matched by parallel execution on power-optimized processors, customization of the cluster may be required. It is possible to include hardware accelerators as an extension of the baseline PULP cluster sharing L1 memory (Dehyadegari et al. 2015). Integration of HW accelerators in the PULP cluster is fully modular; the IPs used to couple accelerators with the cluster are parametric and support manually or HLS-designed accelerators using either a streaming dataflow model or a memory-mapped one. In the context of high-performance computing, deep learning (Memisevic 2015) and more specifically Convolutional Neural Network (CNN) algorithms have rapidly grown since 2012 thanks to their outstanding capabilities in the recognition (Hannun 2014), and classification (Russakovsky 2014) fields. Being these algorithms approximation tolerant, a new key challenge is to exploit them as hardware accelerators for deeply embedded applications by replacing their native single- or double- precision implementations with integer or approximated floating point arithmetic to improve energy efficiency. Another advantage of CNNs is their affinity to be rapidly adapted to several application domains, such as embedded audio and video processing (Wu et al. 2015). In addition, the computational patterns typically implemented by CNNs are very similar to those of more traditional filters used in several deeply embedded classification applications such as FIR filtering or FFT. Hence, CNN hardware accelerators join the typical advantages of application specific computing (i.e., significant boost in performance and energy efficiency with respect to equivalent software implementations) with generality, matching two of the key requirements of IoT computing devices. Vision-PULP is an extension of the basic platform oriented to embedded vision that features a 2D convolutional accelerator able to perform two 16-bit 5×5 convolutions per cycle (Conti et al. 2015). This improves energy efficiency by a

factor of 40x or more in convolutional workloads, reaching up to 80 GOPS of performance and 3000 GOPS/W.

3.5 Trends and Perspectives

Table 3.2 provides a summary of recent single-core and multi-core ultra-low-power and energy efficient digital computing architectures targeting the IoT application domain. Until last few years, most IC design has mainly been driven by performance and area to reduce the cost production, while power consumption has always been a secondary concern for optimization. With the coming of for near-sensor processing and IoT we are experiencing a shift from a computation-driven to a data-driven environment, requiring a radical change in the design of the hardware platform architectures. In this scenario data are generated by several sensors, often asynchronously (event-based computing), that activate the computational platform at regular intervals, with long idle periods in between. Hence, the computational platforms have to be always-on, powered by harvester or small coin batteries, and their life-time can be as long as several years, as expected by their target applications. As a consequence, both sleep modes and active modes need to be carefully optimized to reach the goal of zero-energy sense, classify and transmit IoT systems.

In the next generation of IoT nodes energy saving will have to be considered as a system-wide concern. All the components within an IoT node need to be optimized for power, including sensor interfaces (Chaps. 12 and 13), digital platforms (Chap. 9) and RX/TX transceivers (Chap. 14), power supply and conversion subsystem (Chaps. 10, 11, 15). Optimized software also plays a crucial role system-wide to manage power state of all the components forming the IoT node. While DVFS joint with parallelism has been demonstrated to be extremely effective in reducing the energy of digital blocks, this technique can only be applied in a very limited way

Table 3.2 Summary of recent ultra-low-power and energy efficient processors and DSPs

	SLEEP WALKER (Bol et al. 2013)	REISC (Ickes et al. 2011)	DSP (Gammie et al. 2011)	FRISBEE (Wilson et al. 2014)	CENTR P3DE (Fick et al. 2012)	PULP (Rossi et al. 2016a)	PULP2 (Rossi et al. 2016b)
CPU	MSP430	ReiSC	TMS320C64x	FRISBEE	ARM CortexM3	OpenRISC	OpenRISC
Data format	16-bit	32-bit	32-bit VLIW	32-bit VLIW	32-bit	32-bit	32-bit
Number of cores	1	1	1	1	64	4	4
I$/D$/L2 (bytes)	16 k/2 k/n.a.	8 k/8 k/n.a.	32 k/32 k/128 k	4 k/4 k/n.a.	64 k/512 k/n.a.	16 k/4 k/16 k	16 k/4 k/16 k
Technology	CMOS	CMOS	CMOS	FD-SOI	CMOS	FD-SOI	FD-SOI
Node	65 nm GP	65 nm LP	28 nm LP	28 nm HP	130 nm	28 nm LP	28 nm HP
VDD range (mem) [V]	0.4 (1.0)	0.54–1.2 (0.4–1.2)	0.6–1.0	0.4–1.3	0.65–1.15 (0.8–1.65)	0.44–1.2 (0.54–1.2)	0.32–1.2 (0.45–1.2)
Max freq. (MHz)	25	82.5	331	2600	80	475	825
Power dens. (uW/MHz)	7.7	10.2	409	62	317	72	20.7
Best Perf. (MOPS)	25	57.5	662	2600	1600	1800	3300
Energy eff. (MOPS/mW)	64.5	68.6	4.5	16	3.9	60	193

and with some restrictions to analog blocks. For example, memories are very critical IPs, and often form a bottleneck for energy efficiency of ultra-low-power platforms, since the bit-cells, as well as some of the analog IPs in the periphery requires higher voltages than the logic to be operational in a reliable way. Although several solutions has been proposed in research (Chap. 5), silicon vendors still strive to provide memory generators optimized for low-voltage operation in production design kits.

An emerging trend for ultra-low-power systems, supplied only with energy harvesters or small batteries is that of energy-driven computation. This computational paradigm, referred as *transient computing*, can be applied to applications with very low requirements in terms of real-time constraints. For example a sensor that collects and transmit environmental conditions (e.g., temperature, pressure). When an *energy burst* is captured and accumulated in some energy storage (e.g., super-caps) by the harvester the acquisition and computation can start. When the energy on the accumulator—monitored by the hardware platform—is going to finish, the hardware platform goes into a deep-sleep or even shut-down mode, after saving its internal state and temporary data on a non-volatile support. When a further energy burst occurs, the hardware platform restores the state from its non-volatile support and continues acquisition, processing and transmission. This is yet an example showing the new requirements in terms of tighter system-level integration between the MCU and the peripheral parts of the node (e.g., batteries, sensors, transceivers) whose power consumption need to be constantly monitored and predicted to adopt power-management and shut-down strategies, while guaranteeing to always be able to recover processing from a consistent state. In this scenario, a key role will be played by non-volatile memories, which has to be re-designed and optimized for their new scope (Chaps. 6 and 7).

Most of today's IoT systems are fairly low volume, and intended for a cost conscious market. Hence, computing devices for IoT are implemented with very cheap and not scaled technology nodes, ranging from 180 to 65 nm. Advanced nodes such sub 20 nm, or FD-SOI, would have advantages at the transistor level but their mask set and wafer cost will only be justified only with the expected increase of IoT product volumes, that would reduce manufacturing costs as well. A big concern of scaled technology nodes is the availability of embedded Flash as today 40 nm is the most advanced node featuring embedded Flash. Another issue that has to be addressed deals with process and temperature variations. IoT systems are intrinsically subject to variations during their long lifetime due to degradation of batteries and power supply network lowering the supply voltage, or aging. Usage of scaled technology magnifies variations caused by process and temperature. More specifically, scaled transistors operating at low voltage are subject to thermal inversion phenomena, which causes a strong dependency between the ambient temperature and maximum operating frequency. This issues will have to be tackled in next generation of IoT commercial platforms with advanced techniques to tolerate or compensate these variations, such as in situ error detection and correction, adaptive voltage scaling, and adaptive body biasing.

References

M. Alioto, Ultra-low power VLSI circuit design demystified and explained: a tutorial. IEEE Trans. Circuits Syst.—Part I (Invited) **59**(1), 3–29 (2012)

Ambiq Micro, Ultra-low power MCU family. Apollo Datasheet rev 0.45 (Sep. 2015)

ARM, Cortex-M4, Technical Reference Manual r0p0 Issue A (Mar. 2010), pp. 3.8–3.10

Atmel, SMART ARM-based Microcontrollers. SAM L22x Rev. A datasheet (Aug. 2015)

R. Banakar, S. Steinke, B. Lee, M. Balakrishnan, P. Marwedel, Scratchpad memory: a design alternative for cache on-chip memory in embedded systems. in *Tenth International Symposium on Hardware/Software Codesign (CODES)* (Estes Park, 2002)

L. Benini, A. Macii, E. Macii, M. Poncino, Increasing energy efficiency of embedded systems by application-specific memory hierarchy generation. IEEE Design Test Comput. **17**(2), 74–85 (2000)

D. Blaauw, et al., Razor II: in situ error detection and correction for PVT and SER tolerance, in *IEEE*

International Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers (IEEE, 2008)
D. Bol, J. De Vos, C. Hocquet, F. Botman, F. Durvaux, S. Boyd, D. Flandre, J. Legat, SleepWalker: a 25-MHz 0.4-V Sub-mm 2 7-μW/MHz microcontroller in 65-nm LP/GP CMOS for low-carbon wireless sensor nodes. IEEE J. Solid-State Circuits **48**(1), 20–32 (2013)
D. Bortolotti, D. Rossi, A. Bartolini, L. Benini, A variation tolerant architecture for ultra low power multiprocessor cluster, in *2013 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)* (Karlsruhe, 2013), pp. 32–38
B.H. Calhoun, A.P. Chandrakasan, Standby power reduction using dynamic voltage scaling and canary flip-flop structures. IEEE J. Solid-State Circuits **39**(9), 1504–1511 (2004)
F. Conti, L. Benini, A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters, in *Design, Automation & Test in Europe Conference & Exhibition* (9–13 Mar 2015), pp. 683–688
D.E. Culler, J.P. Singh, *Parallel Computer Architecture, a hw/sw Approach* (Morgan Kaufmann, San Francisco, 1999)
Cypress, Designing for low power and estimating battery life for BLE applications, AN92584 Rev. C Application Note (Mar. 2016)
C.E. Molnar, R.F. Sproull, I.E. Sutherland, The counterflow pipeline processor architecture. IEEE Des. Test Comput. **11**, 48–59 (1994)
H. De Groot, IoT and the cloud: a hacked personality and an empty battery head-ache or an intuitive environment to make our lives easier? S3S (2015)
M. Dehyadegari, A. Marongiu, M.R. Kakoee, S. Mohammadi, N. Yazdani, L. Benini, Architecture support for tightly-coupled multi-core clusters with shared-memory HW accelerators. IEEE Trans. Comput. **64**(8), 2132–2144 (2015)
A.Y. Dogan, D. Atienza, A. Burg, I. Loi, L. Benini, Power/performance exploration of single-core and multi-core processor approaches for biomedical signal processing, in *Proceedings of 21st International Workshop*, PATMOS 2011 (Madrid, 26–29 Sept. 2011), pp. 102–111
R.G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, T. Mudge, Near-threshold computing: reclaiming Moore's law through energy efficient integrated circuits. Proc. IEEE **98**(2), 253–266 (2010)
D. Ernst et al., Razor: a low-power pipeline based on circuit-level timing speculation, in *Proceedings of 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36* (2003), pp. 7–18
D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N.S. Kim, Razor: circuit-level correction of timing errors for low-power operation. IEEE Micro **34**(6), 10–20 (2005)
D. Fick et al., Centip3De: a 3930DMIPS/W configurable near-threshold 3D stacked system with 64 ARM Cortex-M3 cores. in *2012 I.E. International Solid-State Circuits Conference* (San Francisco, 2012), pp. 190–192
G. Gammie, N. Ickes, M.E. Sinangil, R. Rithe, J. Gu, A. Wang, H. Mair, S. Datla, B. Rong, S. Honnavara-Prasad, L. Ho, G. Baldwin, D. Buss, A.P. Chandrakasan, Uming Ko, A 28 nm 0.6 V low-power DSP for mobile applications, in *2011 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (20–24 Feb. 2011), pp. 132–134
M. Gautschi, D. Rossi, L. Benini, Customizing an open source processor to fit in an ultra-low power cluster with a shared L1 memory, in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI-GLSVLSI'14* (2014), pp. 87–88
M. Gautschi, A. Traber, A. Pullini, L. Benini, M. Scandale, A. Di Federico, M. Beretta, G. Agosta, Tailoring instruction-set extensions for an ultra-low power tightly-coupled cluster of OpenRISC cores, in *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)* (October 2015)
A. Hannun, Deep speech: scaling up end-to-end speech recognition, arXiv (2014)
S. Hsu, A. Agarwal, M. Anders, S. Mathew, H. Kaul, F. Sheikh, R. Krishnamurthy, A 280 mV-to-1.1 V 256b reconfigurable SIMD vector permutation engine with 2-dimensional shuffle in 22 nm CMOS, in *2012 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (19–23 Feb. 2012), pp. 178–180
N. Ickes, Y. Sinangil, F. Pappalardo, E. Guidetti, A. P. Chandrakasan, "A 10 pJ/cycle ultra-low-voltage 32-bit microprocessor system-on-chip, in *2011 Proceedings of the ESSCIRC (ESSCIRC)* (Helsinki, 2011), pp. 159–162
D. Jeon, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, D. Sylvester, A 470 mV 2.7 mW feature extraction-accelerator for micro-autonomous vehicle navigation in 28 nm CMOS, in *2013 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (17–21 Feb. 2013), pp. 166–167
G. Kalokerinos, V. Papaefstathiou, G. Nikiforos, S. Kavadias, Ma. Katevenis, D. Pnevmatikatos, X. Yang, FPGA implementation of a configurable cache/scratchpad memory with virtualized user-level RDMA capability, in *International Symposium on Systems, Architectures, Modeling, and Simulation, 2009*. SAMOS '09 (Samos, 2009), pp. 149–156
I. Loi, D. Rossi, G. Haugou, Exploring multi-banked shared-L1 program cache on ultra-low power, tightly coupled processor clusters, in *Proceedings of the 11th ACM Conference on Computing Frontiers-CF '15* (July 2015), pp. 1–10
M.M.K. Martin, M.D. Hill, D.J. Sorin, Why on-chip cache coherence is here to stay. Commun. ACM **55**(7), 78–89 (2012)
P. Meinerzhagen, C. Roth, A. Burg, Towards generic lowpower area-efficient standard cell based memory

architectures, in *Proceedings of IEEE International Midwest Symposium on Circuits and Systems* (Aug. 2010), pp. 129–132

R. Memisevic, Deep learning architectures, algorithms and applications, in *Hot Chips: A Symposium on High Performance Chips* (Cupertino, 23–25 Aug. 2015)

Microchip, Analog-to-digital converter with computation technical brief. TB3146 Application Note (Aug. 2016)

ST Microelectronics, STM32L4x5 advanced ARM®-based 32-bit MCUs. RM0395 Reference Manual (Feb. 2016)

I. Miro-Panades, E. Beignè, Y. Thonnart, L. Alacoque, P. Vivet, S. Lesecq, D. Puschini, A. Molnos, F. Thabet, B. Tain, K.B. Chehida, S. Engels, R. Wilson, D. Fuin, A fine-grain variation-aware dynamic Vdd-hopping AVFS architecture on a 32 nm GALS MPSoC. IEEE J. Solid-State Circuits **49**(7), 1475–1486 (2014)

NXP Semiconductors. 2015. LPC5410X Product Data Sheet. NXP. Rev 2.2

J. Park, I. Hong, G. Kim, Y. Kim, K. Lee, S. Park, K. Bong, H.-J. Yoo, A 646GOPS/W multi-classifier many-core processor with cortex-like architecture for super-resolution recognition, in *2013 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (17–21 Feb. 2013), pp. 168–169

J. Parkhurst, J. Darringer, B. Grundmann, From single core to multi-core: preparing for a new exponential, in *2006 IEEE/ACM International Conference on Computer Aided Design* (San Jose, CA, 2006), pp. 67–72

Peter Greenhalgh, big.LITTLE processing with ARM Cortex-A15 & Cortex-A7. Technical Report (ARM Ltd., 2011)

A. Rahimi, I. Loi, M.R. Kakoee, L. Benini, A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters, in *Design, Automation & Test in Europe Conference & Exhibition* (Mar. 2011), pp. 1–6

Renesas, RL78/G13 Rev. 3.20 User's Manual Hardware (July 2014)

D. Rossi, I. Loi, G. Haugou, L. Benini, Ultra-low-latency lightweight DMA for tightly coupled multi-core clusters, in *Proceedings of the 11th ACM Conference on Computing Frontiers—CF '14* (July 2014), pp. 1–10

D. Rossi, A. Pullini, I. Loi, M. Gautschi, F.K. Gurkaynak, A. Bartolini, P. Flatresse, L. Benini, A 60 GOPS/W,−1.8 V to 0.9 V body bias ULP cluster in 28 nm UTBB FD-SOI technology. Solid-State Electron. **117**, 170–184 (2016a)

D. Rossi, A. Pullini, I. Loi, M. Gautschi, F.K. Gurkaynak, A. Teman, J. Constantin, A. Burg, I.M. Panades, E. Beignè, F. Clermidy, F. Abouzeid, P. Flatresse, L. Benini, 193 MOPS/mW @ 162 MOPS, 0.32V to 1.15V voltage range multi-core accelerator for energy-efficient parallel and sequential digital processing, Cool Chips (2016b)

O. Russakovsky, ImageNet large scale visual recognition challenge. Int. J. Comput. Vis (2014)

E. Stotzer, A. Jayaraj, M. Ali, A. Friedmann, G. Mitra, A. P. Rendell, I. Lintault, OpenMP on the low-power TI keystone II ARM/DSP system-on-chip, in *OpenMP in the Era of Low Power Devices and Accelerators* (2013)

A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, A. Burg, Controlled placement of standard cell memory arrays for high density and low power in 28 nm FD-SOI, *in 20th Asia and South Pacific Design Automation Conference (ASP-DAC), 19–22 January, 2015* (2015), pp. 81–86

Texas Instruments, Low-power FRAM microcontrollers and their applications, SLAA502 White Paper (June 2011)

J.W. Tschanz et al., Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. IEEE J. Solid-State Circuits **37**(11), 1396–1402 (2002)

J. Tschanz et al., Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging, in *IEEE International Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers* (11–15 Feb. 2007), pp. 292–604

R. Wilson et al., A 460MHz at 397mV, 2.6GHz at 1.3V, 32b VLIW DSP, embedding F_{MAX} tracking, in *2014 I. E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (San Francisco, 2014), pp. 452–453

M. Wu, R. Iyer, Y. Hoskote, S. Zhang, B. Deadman, M. Bhartiya, Y. Satish, Design of an ultra-low Power SoC testchip for wearables & IOT, in *Hot Chips: A Symposium on High Performance Chips* (Cupertino, 23–25 Aug. 2015)

A. Yakovlev, D. Pivonka, T. Meng, A. Poon, A mm-sized wirelessly powered and remotely controlled locomotive implantable device, in *2012 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (19–23 Feb. 2012), pp. 302–304

J. Yoo, Y. Long, D. El-Damak, M. Bin Altaf, A. Shoeb, Y. Hoi-Jun, A. Chandrakasan, An 8-channel scalable EEG acquisition SoC with fully integrated patient-specific seizure classification and recording processor, in *2012 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (19–23 Feb. 2012), pp. 292–294

F. Zhang, Y. Zhang, J. Silver, Y. Shakhsheer, M. Nagaraju, A. Klinefelter, J. Pandey, J. Boley, E. Carlson, A. Shrivastava, B. Otis, B. Calhoun, A batteryless 19 μW MICS/ISM-band energy harvesting body area sensor node SoC, in *2012 I.E. International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. (Feb. 2012), pp. 298–300