Ultra Low Energy Microcontroller Architectures

Aditya Tandon, *at3g10@soton.ac.uk, University of Southampton*

***Abstract*—Wireless sensor network (WSN) applications and mobile processors require low energy architectures for them to be commercially viable. In wireless networks, a device can run for years without change in battery and batteries on mobile phones need to have an extended lifetime so that the user does not have to constantly charge the phone. To cater to these needs, various low power design techniques have cropped up to reduce power on processors. This paper highlights some of these techniques that can be useful for WSN applications and mobile processors. Concepts such as asynchronous design, subthreshold design, low power memory design and power domains have been explained to show the different ways in which energy can be saved. Every scheme has shown a reduction in energy with trade-offs for performance, cost and area. This is a minor trade-off as the life of a product is increased due to these low power techniques.**

I. Introduction

MANY applications in todays world demand minimal energy consumption. In wireless sensor networks, a system might be placed in a remote location where the battery cannot be changed periodically. It could also derive energy from the environment and hence has to function on the least amount of energy as possible. On the other end, mobile processors these days have become very sophisticated. They have to perform a variety of tasks such as calling, streaming videos, playing music etc. and at the same time ensure that the battery lasts for a long time. This report looks into the different approaches that can be employed to design low energy microcontrollers and microprocessors. There are many established techniques that are being used in industry and there are other topics that are still in the research stage. This report aims to provide a mix of both approaches to highlight the most relevant methods that can be adopted to produce an efficient architecture.

Section II of the report deals with asynchronous processors and how they can be applied in wireless sensor applications. Section III describes different low power memory techniques to reduce power and Section IV talks about how design in the subthreshold region can lead to energy savings. Section V highlights some emerging power management schemes such as the use of power domains and also explains how established techniques such as dynamic voltage and frequency scaling can be improved. Section VI highlights other alternative methods for low power design such as the use of FPGAs and Section VII concludes this paper.

II. Asynchronous design

In the field of sensor networks asynchronous processors and microcontrollers have been gaining popularity as they lead to energy efficient designs. The basic principle is that these designs function without a global clock and hence reduce the number of unwanted switching activities in the circuit [1], [2], [3]. To compensate for no clock, the designs usually employ extra hardware for a handshaking protocol [1], [3]. To further conserve energy these designs are event driven [1], [2], [3], [4]. In such a system, the controller is mostly in a state of sleep until it is asked to perform a computation by an event. After performing the task the controller goes back to a sleep state thereby minimizing its active energy [1], [2]. Research has shown that there is no necessary software overhead for these systems due to their event driven nature [1], [2]. This is because these microcontrollers are used for a set of pre-defined tasks and can be simplified. For example, interrupts can be processed as events and there is no overhead to handle concurrent tasks [1], [2]. Designs can be further simplified by employing an in-order design which reduces the amount of hardware and therefore lowers energy [2], [5].

Event driven architectures should have a minimal transition time from a sleep state to an active state. The SNAP/LE architecture addresses this concern by employing an event queue [1]. This resembles a FIFO handler and tasks are executed if there is an event token present in the queue. An event token signifies an event. If there is a token, the appropriate event handler associated with the token is looked up and the task is executed. After executing the task the processor goes into a 'sleep' state if there is no token present [1]. The time taken by the processor to transition between an active and sleep state is the time taken for a token to go through the event queue [1]. The length of the queue can be optimized so that this process is in the order of tens of nanoseconds and therefore this procedure saves energy [1].

Typically these designs can be modularised and stress can be taken off the microcontroller by employing hardware accelerators [1], [2]. Hempstead et al. [2] used the microcontroller only for computational intensive tasks and a separate event processor was employed which was effectively a hard-coded state machine to handle events which required light computation. This reduced the active and leakage power of the main controller. Another type of accelerator used was the 'Message Coprocessor' which was responsible for forming and forwarding incoming messages from the radio unit. Timing operations commonly found in wireless applications can be handled by a Timer Coprocessor and therefore can lead to a simplistic, energy efficient implementation for the microcontroller at the expensive of some additional hardware [2]. An architectural implementation of such a system can be seen from Figure 1.

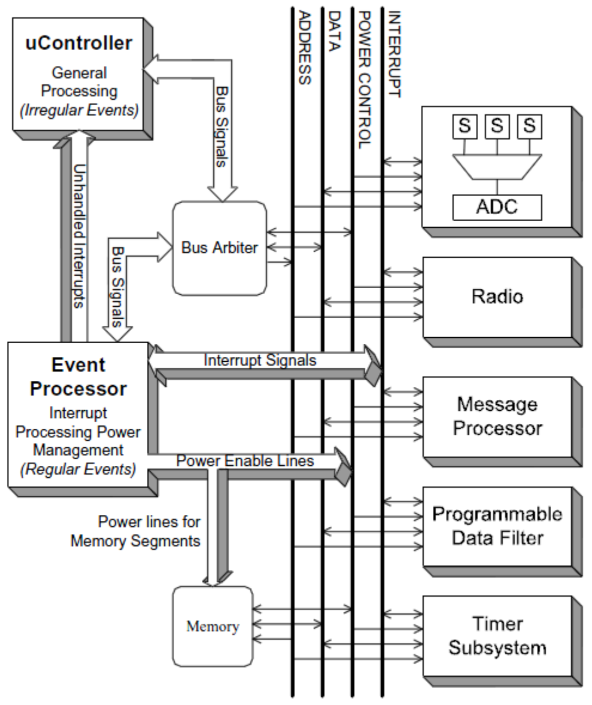


Fig. 1. Event driven design (reproduced from [2]).

The steps taken in desynchronization involve replacing the clock tree with local, asynchronous controllers and converting the flip-flops into latches [4]. This was demonstrated by Neechi et al. [4] where a synchronous AVR microcontroller was used as a template to create an asynchronous version. It was found that the asynchronous controller was about 5 times more energy efficient than the synchronous one [4]. The amount of energy that these prototype processors have taken to execute a particular instruction has been in the range of 10pJ 14pJ assuming an operating voltage of 1.2V and about 2.7pJ/instruction at 0.54V [1], [4], [5].

III. Low power memory

Over the years researchers have come up with different techniques to mitigate energy loss due to memory. A proposal for an adaptive cache for mobile processors could help reduce power [6]. The L2 caches on mobile phones have been found to have access patterns that are not correlated or balanced and therefore there is scope to dynamically adjusting the cache to match the application using it [6]. The compiler does an offline analysis of the application before run-time to determine parameters such as global average miss rates and access rates. Cache access is also monitored during run-time and this information in conjunction with the offline material is used to enlarge or decrease the size of the cache dynamically depending on the need [6]. This proves useful as memory is used much more efficiently. Also, the leakage power is reduced as there are fewer idle cells. This technique was found to give a 13% - 29% reduction in power consumption (using benchmark programs) but there was a small trade-off for speed and area to incorporate this [6].

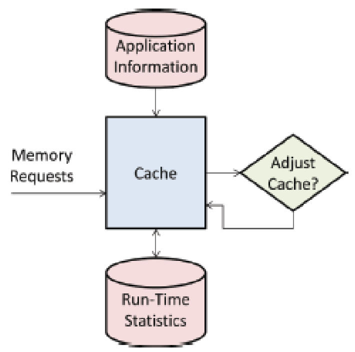


Fig. 2. Adaptive cache design (reproduced from [6]).

Another technique to reduce power is memory compression [7]. Here, a part of the data in volatile memory is compressed in order to reduce the number of logic elements that have to be self-refreshed when a device is turned into a low power state. Rest of the memory can be powered off and therefore the battery life is extended [7]. When there is a request to put a device into a low power state, the memory compression logic takes blocks from a designated memory, compresses it using a compression algorithm and then stores these blocks back to memory. A decompression procedure is followed when the device moves into an active state [7]. The compression logic can be implemented in hardware or software and induces some overhead on the battery while performing compression but the power saved by compressing data outweighs this overhead [7].

Non-volatile memories are another area of interest for power reduction and even in performance enhancement [8]. Resistive Random Access Memory (RRAM) is a piece of memory that could be used as a substitute for SRAMs on mobile devices [8], [9]. It was found that an RRAM with a crosspoint structure that uses a diode as a select cell reduces leakage current. This is because the resistance goes up as voltage decreases and the leakage current paths are cut-off [8]. This structure is also area efficient as multi-layered structures can be made [8]. Investigation is still going into this area as peripheral circuit design is harder if RRAMs are used but it could be used as a technique to reduce energy and power [8]. Alternative RRAM technologies are also being researched and a switching energy of 6fJ has been achieved and this is close to the switching energy of Flash Memory [9].

Many other techniques can be used to reduce power. An Intel processor for mobile devices reduces the leakage power in the L2 cache [10]. The data arrays in the cache continue to be in the 'sleep' mode until a 'Hit' signal is generated. Even when there is a hit, only the relevant data array is charged so that it can be activated while the other arrays continue to be in the low power mode [10]. This is a memory partitioning technique and is widely used to mitigate leakage power [10], [11].

IV. Subthreshold design

Devices operating in the subthreshold region show a reduction in energy because the switching activity decreases with supply voltage [12], [13]. However, the propagation delay increases in this region which gives rise to leakage currents. This can cause energy dissipation [12], [13]. There exists an optimal operating point in the subthreshold region at which devices can operate. This point can be found through a minimum energy analysis technique in which the energy is plotted against the supply voltage as the voltage is scaled down [12]. Wang and Chandrakasan [12] implemented this idea in a FFT processor to demonstrate how devices can operate in subthreshold regions. Logic elements have to be specifically modified to be catered for subthreshold operation to avoid leakage current. Parallel leakage is a major contributor to leakage current and it occurs when the idle current is comparable to the drive current in circuits [12]. This effect can be mitigated by reducing or balancing the number of parallel devices in the pull up and pull down path to avoid leakage [12]. Devices should avoid being stacked as this reduces the effective drive current of each transistor in a stack [12].

RAM blocks usually contain a six transistor (6T) scheme to enable reading and writing of data. Wang and Chandrakasan have demonstrated that subthreshold conditions make read and write operations harder as they place a sizing constraints on the transistors used [12]. There are also other considerations such as bitline leakage that comes into effect when operating in this region [12]. Therefore, an alternative structure to the RAM is needed to address this problem. In the FFT processor, the RAM uses tristate inverters to create a latch and this is used for write operations. The read operation uses parallel tristate gates and a hierarchical read bitline to mitigate parallel leakage [12].

Therefore, it can be observed that operating in the subthreshold region can help in energy savings but only if the logic is suitably catered for it. Designers might have to construct subthreshold libraries if they want their devices operating in this region. The gains in the long run are satisfactory as it results in an energy efficient device. The subthreshold FFT chip that was made was found to be “350 times more energy efficient than the low-power microprocessor implementation” which was a microprocessor that did not include subthreshold logic [12].

V. Power management schemes

A. Power domains and other techniques

Chips on mobile phones are now moving towards a multi-core implementation to support the vast functionality that is in demand. The cores are usually heterogeneous so that each of them can run at an independent frequency and hence be utilized in the best possible way and also reduce dynamic power [14]. A single chip implementation makes it hard for power management schemes to achieve power reduction due to leakage currents. A multi-chip implementation paves way to implementing a partial power off scheme where unused chips can be powered off if they are not in use [14]. This gives rise to the concept of a power domain where different parts of a chip and also different chips can be isolated from each other in terms of power management [14]. Many domains can thus be created and power can be saved. Implementing power domains does pose some problems. For example, the shutdown elements between power domains need to be robust and reliable [14]. μI/Os are used to route signals from domains that might be powered off. These are special circuits used to isolate such signals [14]. To minimize such additional circuitry, a hierarchical power domain scheme can be adopted which sets a level of precedence for certain domains [14]. So no μI/Os are needed from a higher hierarchy to a lower one as the lower hierarchy cannot be active while the upper one is switched off. Figure 3 illustrates a hierarchical structure that can be implemented on mobile phones.

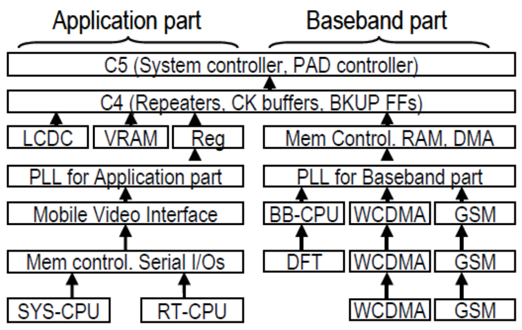


Fig. 3. Hierarchical power domains (reproduced from [14]).

Another problem is the rush current generated while switching on power domains [14]. Hattori et al. [14] described an efficient power switch design to minimize this rush current so that it can be made negligible. They observed very low leakage currents using this design and so the use of hierarchical power domains could be an effective way to save energy [14].

Power gating is another strategy that has been around for quite a while. Here, the processor is switched off due to switches inserted into the power rail [15]. This causes a reduction in power but other components, such as state retention registers, power management unit and Low Drop Regulator (LDO) still remain active and contribute to leakage current [15]. To mitigate this, Lueders et al. [15] proposed a scheme based around the LDO. The idea was that a digitally adaptive LDO would drive the micro-controller unit and based on the requirements, it would adapt its drive current and hence reduce power management overhead in low frequency operations [15]. Also, as the LDO is integrated onto the chip, it would be designed with a low output capacitance and therefore take up very little power during sleep and also have a quick transition wake up time [15]. During sleep mode, the LDO can be disabled and it was shown that a power saving of a factor of 4.3 was achieved as compared to power gating [15]. This was easier to implement than power gating as system partitioning was simple and no power switches had to be used [15].

Other power management schemes, include the implementation of different power states in a system [10]. For example, an Intel processor for mobile phones has 6 power states (C0 C6) [10]. The C0 state is the high frequency state and in C6 state the core power is shutdown [10]. The intermediate states involve the power gating of different components such as the core clock, phase locked loops and flushing of L1 caches to reduce dynamic power [10].

B. Dynamic voltage and frequency scaling

Dynamic voltage and frequency scaling (DVFS) is used ubiquitously to improve energy performance in processors [16], [17], [18]. Power is proportional to frequency and the square of supply voltage so scaling down these parameters saves energy [17]. Although delay increases with a reduced voltage, applications that are not time critical can be performed at a lower voltage to avoid performance hits [16]. Multicores on mobile phones can be made much more energy efficient if they ran at an optimum operating point. This operating point is a combination of choosing the correct operating frequency and the number of cores used for an application [16]. Quite often, cores are under-utilized in order to save energy but this actually dissipates more energy as fewer cores are running intensive programs at a higher frequency. Instead of this, more cores can be made available and can operate at a lower frequency [16], [19]. This increases the number of computation resources and also reduces the energy as the operating frequency is low. Also, the work gets executed faster and power dissipation outside the cores can be minimized as these components can be turned off once the computation is finished [16], [19]. Carol and Heiser [16] have come up with a linux governor that implements this concept. Frequency is increased if a core is being over-utilized and vice versa. It also disables and enables cores based on their need and this could be a useful tool to incorporate onto devices as an energy saving of upto 25% was observed in this case [16].

Dynamic Thermal Management (DTM) is another important aspect to consider when discussing DVFS [17]. If the temperature of a chip exceeds a certain thermal threshold the frequency has to be scaled down again to prevent the chip from over-heating. Once the temperature goes down the frequency is increased again and this would result in the thermal threshold being exceeded [17]. It can be observed that in some scenarios the operating frequency can oscillate between two frequencies due to DTM and this causes a degradation in power as the chip operates at an unstable frequency [17]. To mitigate this, Kim et al. [17] proposed a DVFS scheme based on an average frequency operating point so that the frequency is stable. Frequencies are sampled on a periodic basis when they exceed the thermal threshold and an average is formed once enough samples are collected. The samples are stored in a 'frequency window' that gets updated until its full [17]. The average of the frequency window is set as the operating point or the 'target frequency' [17]. This operating point can be re-sampled for different criteria if, for example, the frequency needs to be lowered or raised. This leads to a reduction in energy and an energy saving of 12.7% was observed in this scheme [17].

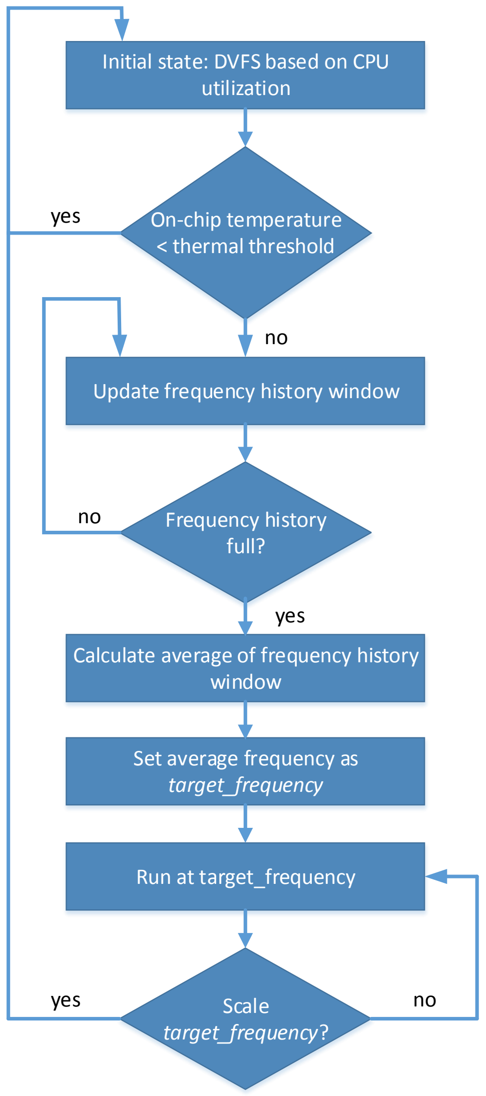

Initial state: DVFS based on CPU utilization
On-chip temperature < thermal threshold
yes
no
Update frequency history window
Frequency history full?
no
yes
Calculate average of frequency history window
Set average frequency as *target_frequency*
Run at target_frequency
Scale *target_frequency*?
yes
no

Fig. 4. Temperature aware DVFS scheme (adapted from [17]).

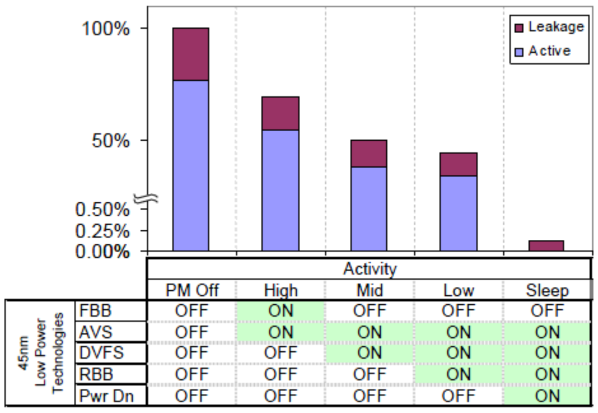
VI. Other techniques

FPGAs have been used to provide quick, cost effective solutions as they have re-programmable capabilities. Yet this re-configurable overhead is also the reason why they consume more power than ASIC designs as power management is more complex [20]. Tuan et al. [20] have investigated low power FPGA applications for battery powered devices and have used a variety of techniques to reduce power. They designed a low power FPGA called *Pika* [20]. Voltage scaling was used to scale the core operating voltage to drastically reduce energy. A 1V operating voltage was found to give the best reductions without severely affecting performance [20]. It was found that SRAM cells were a major contributor towards leakage current. Subthreshold leakage was reduced by using a higher voltage threshold (Vt) and gate leakage was reduced by using thicker gate oxides [20]. Though this increases cost and area, the overall energy savings outweigh the cons. Finally, power gating was also used to reduce leakage current [20]. Unused blocks were turned off to save power. NMOS transistors were used as power gates as they are faster. Both NMOS and PMOS were not used in conjunction to save area [20]. Power gating in FPGAs is complex due to the amount of logic that can be gated. Therefore establishing what the smallest block that can be power gated is important [20]. In *Pika*, a tile was the smallest unit that was power gated [20]. A tile here is used to define a configurable logic block (CLB) along with its relevant programmable switch matrix that connects it to other CLBs [20]. The SRAM cells in the switch matrix are not power gated to enable state retention when the rest of the core is powered down [20]. Power gating individual tiles helps in implementing a partial standby mode wherein some logic elements can be powered off and the rest can still remain active. This feature is implemented by having a programmable bit per tile [20]. The overall power savings for all these schemes is illustrated in Table I and it can be seen that a 46% in active power reduction and 99% standby power reduction was observed when compared to a normal FPGA with no power management [20]. There was however a trade-off with performance and area to enable these schemes. A 27% reduction in performance was observed along with a 40% increase in area. [20]

Technique	Active Power	Standby Power
Voltage scaling	35%	23%
Low-leakage SRAM config	13%	43%
Power gating	7%	N/A
Standby mode	N/A	51%
Total power reduction	46%	99%

TABLE I
IMPACT OF POWER REDUCTION TECHNIQUES (REPRODUCED FROM [20])

Adaptive body biasing is another technique that was found to be beneficial. It is based on the simple idea that forward body biasing (FBB) decreases the threshold voltage and therefore increases performance and power while reverse body biasing (RBB) increases threshold voltage and reduces leakage current [21]. This adaptive biasing is only applied to certain areas of the chip to suitably alter performance or power and this can give better power savings. Gammie et al. [21] have described a tool, *SmartPriMer*, which inserts power management modules into a piece of RTL code. These modules include power domains, adaptive body biasing and other such techniques to reduce power consumption [21]. The tool also generates UPF information for the design that can be used on the later stages of the design flow [21]. These techniques were tested on mobile applications and a 37% reduction in active power was observed [21]. Figure 5 shows the power reduction achieved when the different techniques are used in a 45nm system on chip (SoC) designed for a mobile phone [21]. It can be observed that a mix of Adapative voltage scaling (AVS), RBB and DVFS are used to cut down power when the activity level of a processor is low and least power is consumed when the core is powered down [21].



45nm Low Power Technologies	Activity				
	PM Off	High	Mid	Low	Sleep
FBB	OFF	ON	OFF	OFF	OFF
AVS	OFF	ON	ON	ON	ON
DVFS	OFF	OFF	ON	ON	ON
RBB	OFF	OFF	OFF	ON	ON
Pwr Dn	OFF	OFF	OFF	OFF	ON

Fig. 5. Power reduction techniques during different processor activities (reproduced from [21]).

AMD have designed an x86 processor, *Bobcat*, for the low power, mobile phone market [22]. This processor employs different techniques for it to consume low power and yet be energy efficient. Instructions are executed as Complex Micro-operations (COPs). Here, the decode, execute, memory read and write-back can be implied by the same instruction and hardware that takes up fewer resources, can be specifically designed for such instructions [22]. Sophisticated branch prediction is another way power is reduced as the power spent on flushing incorrectly fetched instructions is minimized [22]. The register file of this processor uses a pointer based scheme. No data is actually moved, only pointer references are updated during register renaming and this saves power [22]. Another novel approach is the clock gating of the data-size itself. For example, if only 32 bits of 64 bits of the data are used then the upper 32 bits can be gated to save power [22]. The L2 cache saves power by running at half the core frequency as it does not affect performance to a significant extent [22]. Overall, the *Bobcat* processor was found to be efficient in terms of area and power [22].

Finally, a healthy interaction of hardware and software policies is a good way to reduce power [23]. Most mobile phones have multi core processors so to make optimum use of these resources parallel computing and task scheduling can be used [23]. Parallel computing involves executing instructions concurrently and an efficient task scheduler can map out a sequence of instructions that can be executed with minimal delay [23]. The use of heterogeneous cores aids schedulers as cores can be catered to different applications and they make good use of the available resources and help in reducing energy as tasks are executed faster [23], [24], [25]. A H-tree clocking implementation helps reducing dynamic power as it reduces wiring capacitance and could be another strategy that can be used [24].

VII. Conclusion

The need for low power microcontroller architectures is increasing in the field of wireless sensor networks and mobile applications. Battery life is become a major concern and so different power saving techniques need to be implemented to increase battery life. Various energy efficient techniques were presented in this paper. Asynchronous design highlighted an event based approach for processor design to simplify the architecture and hence reduce power. The memory on any chip is another major contributor towards power consumption and so techniques like adaptive caches, memory compression and memory partitioning were described to reduce power. Subthreshold design was investigated and it was found that for realistic power savings, logic had to be redesigned to cope with low voltages in this region. The use of power domains was another efficient way to reduce power on a chip along with dynamic voltage and frequency scaling (DVFS) although in the latter scheme the on chip temperature had to be considered to make substantial power savings. FPGAs were looked into as an alternative to microcontrollers and it was found that vast power reductions could be made if power management schemes were appropriately added to the FPGA design. Finally, adaptive body biasing and software approaches like parallel computing were proposed as suitable methods to reduce energy consumption in a processor.

Many of these techniques can be used together to provide a robust system which consumes less power. There is always a trade-off with performance, cost and area when implementing these techniques but in the long run the energy savings outweigh the cons. Therefore, a more efficient chip can be made which lasts longer and delivers suitable performance.

Acknowledgment

The author would like to thank Dr. Tom J Kazmierski for his input into this report.

References

[1] V. Ekanayake, C. Kelly, IV, and R. Manohar, "An ultra low-power processor for sensor networks," *SIGARCH Comput. Archit. News*, vol. 32, no. 5, pp. 27–36, Oct. 2004. [Online]. Available: http://doi.acm.org/10.1145/1037947.1024397

[2] 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, no. 2, pp. 208–219, May 2005. [Online]. Available: http://doi.acm.org/10.1145/1080695.1069988

[3] Y. Liu, G. Xie, P. Chen, J. Chen, and Z. Li, "Designing an asynchronous fpga processor for low-power sensor networks," in *Signals, Circuits and Systems, 2009. ISSCS 2009. International Symposium on*, July 2009, pp. 1–6.

[4] L. Necchi, L. Lavagno, D. Pandini, and L. Vanzago, "An ultra-low energy asynchronous processor for wireless sensor networks," in *Asynchronous Circuits and Systems, 2006. 12th IEEE International Symposium on*, March 2006, pp. 8 pp.–85.

[5] B. Warneke and K. Pister, "An ultra-low energy microcontroller for smart dust wireless sensor networks," in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, Feb 2004, pp. 316–317 Vol.1.

[6] G. Bournoutian and A. Orailoglu, "Application-aware adaptive cache architecture for power-sensitive mobile processors," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3, pp. 41:1–41:26, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2539036.2539037

[7] S. Balasundaram, "Increasing the battery life of a mobile computing system in a reduced power state through memory compression," Dec. 20 2007, uS Patent App. 11/450,214. [Online]. Available: http://www.google.com/patents/US20070291571

[8] P. Chiu, P. Lu, and X. Z, "Energy efficiency enhancement in mobile processor memory design using emerging nonvolatile memory," University of Berkley, Tech. Rep., 2013. [Online]. Available: http://www.eecs.berkeley.edu/ pfchiu/EE241_midtermReport.pdf

[9] C. Cheng, C. Y. Tsai, A. Chin, and F. Yeh, "High performance ultra-low energy rram with good retention and endurance," in *Electron Devices Meeting (IEDM), 2010 IEEE International*, Dec 2010, pp. 19.4.1–19.4.4.

[10] G. Gerosa, S. Curtis, M. D'Addeo, B. Jiang, B. Kuttanna, F. Merchant, B. Patel, M. Taufique, and H. Samarchi, "A sub-2 w low power ia processor for mobile internet devices in 45 nm high-k metal gate cmos," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 73–82, Jan 2009.

[11] N. Sakran, M. Yuffe, M. Mehalel, J. Doweck, E. Knoll, and A. Kovacs, "The implementation of the 65nm dual-core 64b merom processor," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb 2007, pp. 106–590.

[12] A. Wang and A. Chandrakasan, "A 180-mv subthreshold fft processor using a minimum energy design methodology," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 310–319, Jan 2005.

[13] J. Chen, L. Clark, and C. Tai-Hua, "An ultra-low-power memory with a subthreshold power supply voltage," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 10, pp. 2344–2353, Oct 2006.

[14] T. Hattori and et al., "Hierarchical power distribution and power management scheme for a single chip mobile processor," in *Proceedings of the 43rd Annual Design Automation Conference*, ser. DAC '06. New York, NY, USA: ACM, 2006, pp. 292–295. [Online]. Available: http://doi.acm.org/10.1145/1146909.1146986

[15] M. Lueders, B. Eversmann, J. Gerber, K. Huber, R. Kuhn, M. Zwerg, D. Schmitt-Landsiedel, and R. Brederlow, "Architectural and circuit design techniques for power management of ultra-low-power mcu systems," pp. 1–1, 2013.

[16] A. Carroll and G. Heiser, "Mobile multicores: Use them or waste them," in *Proceedings of the Workshop on Power-Aware Computing and Systems*, ser. HotPower '13. New York, NY, USA: ACM, 2013, pp. 12:1–12:5. [Online]. Available: http://doi.acm.org/10.1145/2525526.2525850

[17] J. Kim, Y. Kim, and S. Chung, "Stabilizing cpu frequency and voltage for temperature-aware dvfs in mobile devices," pp. 1–1, 2013.

[18] J. Howard and et al., "A 48-core ia-32 message-passing processor with dvfs in 45nm cmos," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 108–109.

[19] C. H. K. van Berkel, "Multi-core for mobile phones," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '09. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2009, pp. 1260–1265. [Online]. Available: http://dl.acm.org/citation.cfm?id=1874620.1874924

[20] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90nm low-power fpga for battery-powered applications," in *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, ser. FPGA '06. New York, NY, USA: ACM, 2006, pp. 3–11. [Online]. Available: http://doi.acm.org/10.1145/1117201.1117203

[21] G. Gammie and et al., "A 45nm 3.5g baseband-and-multimedia application processor using adaptive body-bias and ultra-low-power techniques," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, Feb 2008, pp. 258–611.

[22] B. Burgess, B. Cohen, M. Denman, J. Dundas, D. Kaplan, and J. Rupley, "Bobcat: Amd's low-power x86 processor," *Micro, IEEE*, vol. 31, no. 2, pp. 16–25, March 2011.

[23] R. Ramirez, E. Rubio, and A. Viveros, "Energy consumption in mobile computing," in *Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on*, March 2013, pp. 132–137.

[24] Igarashi and et al., "10.2 a 28nm hpm heterogeneous multi-core mobile application processor with 2ghz cores and low-power 1ghz cores," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb 2014, pp. 178–179.

[25] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, no. 5, pp. 67–77, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1941487.1941507