International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 6, Issue 6, June 2017 ISSN 2319 - 4847

A Review of Techniques for Power Optimization at Different Levels of Abstraction of Low Power Embedded Systems

A. Nazarudeen[1] , P. Jayaprakash[2]

[1]Research Scholar, Banasthali University, Rajasthan, India

[2]Professor & Head, Mohandas College of Engineering & Technology, Trivandrum, Kerala, India

ABSTRACT

The markets for "Anywhere-Anytime" computing devices and the Internet of Things (IoT) are being established rapidly as advanced versions of these devices have become available to the consumers. The proliferation of these intelligent portable computing devices has created an increasing demand for low power embedded systems, which has resulted in the fabrication of tiny high-speed processors with higher packaging density and powerful computing capabilities. These portable devices are expected not only to be small, lightweight and cool but also to provide incredibly long battery life. This requirement has presented significant challenges for managing the power in low power embedded systems, and it has shifted the designers' focus from traditional constraints to power consumption. This paper reviews the various power optimization techniques that can be applied at different levels of abstraction to provide an in-depth view into the working of power optimization techniques for low power embedded systems.

Keywords: Abstraction levels, Embedded systems, Low power design, power optimization techniques**.**

1.INTRODUCTION

Power consumption has become one of the main design metrics among other metrics used to characterize the quality of the embedded system design.Nowadays, mobile computing devices are packed with thousands of processors and millions of transistors in a single chip to reduce the size and increase the system performance. The frequency of the system clock has also been increasing consistently to enhance the speed of operation. This increase in frequency indirectly aids the power consumption and thereby decreases the battery life. Also, it affects the cooling and the packing cost of the system. Since high-performance computing leads to increased power dissipation in the system, designing high-performance portable computing devices with extended battery life has always been a major challenge for the designers. While addressing the power issues in the embedded design, many significant factors associated with the systems such as delay, system performance, reliability, functional requirements, size, weight, cost, and quality of service are to be taken into consideration. Hence the designers must first understand the major sources and factors affecting the power dissipation to estimate and optimize the power. The knowledge about the different levels of abstractions at which the system can be modeled is also important for optimizing the power consumption in an embedded system.

2.SOURCES OF POWER CONSUMPTION

In an embedded system a base hardware platform (SoC) executes the system and application software, and interacts with the peripherals through communication channels or buses. Therefore, the total power consumption in an embedded system is the sum of power consumed by all parts of the system. The choice of software implementation also has a significant role in the power dissipation [16]. It is well known that the SoC is implemented using CMOS technology, and the dynamic power (P_{DY}) and the static power (P_{ST}) are the two primary components of the total power (P_T) in a CMOS circuit. Hence, the embedded designers normally focus on reducing P_{DY} and P_{ST} when the system is in active and standby modes. P_{DY} refers to the power dissipated when the circuit is normally operating and produces a meaningful output. P_{ST} is the power dissipated whenever the processing elements are switched on, but no active computations are carried out. P_{DY} can further be sub-divided in two components: switching power (P_{SW}) and short circuit power (P_{SC}). So the total power, $P_T = P_{SW} + P_{SC} + P_{ST}$ and it can be modelled by equation 1;

$$P_T = 0.5\, \alpha f C V^2 + \alpha f Q_{SC} V + I_L V \qquad (1)$$

where α is the activity factor, f is the frequency of the system, V is the supply voltage, C is the load capacitance, Q_{SC} is the short circuit charge that flows during a transition, and $\mathbf{I_L}$ is the leakage current which originates from various phenomena. Typically, the P_{SC} increases with increasing V and decreasing with threshold voltage V_{Th}, whereas the P_{ST} increases exponentially with decreasing threshold voltage and increasing temperature [4],[5]. In practice, the share of P_{SW} is approximately 90% of the P_T, and hence the power minimization mainly involves reduction of P_{SW}. From equation 1, it can be seen that reduction of V, f and α are the feasible solutions for reducing the P_{SW}. However, there are limitations for scaling of these parameters as they affect the performance of CMOS circuits [6].

Sometimes, designers concentrate too much on processing and memory units, ignoring the power consumption in peripheral devices. However, peripheral devices such as displays, audio/video devices, keyboards, sensors, network devices, and other I/O devices and their interconnections consume a significant fraction of the total power. Hence, the designers must consider the power consumption in peripherals also into account while addressing the power issues [3].

The methods of software implementation also affect the power dissipation of base hardware and peripheral devices. For example, the choice of operating system calls, memory access patterns, the number of instructions in the program, and the power efficiency of compiled codes play significant roles in the overall power dissipation of the systems because all these factors affect the switching activity of the various units in the system. Hence, the designers must use optimized software for minimizing the power in hardware, which necessitates a tradeoff between the hardware and software [12].

3.DIFFERENT LEVELS OF ABSTRACTIONS AND POWER ESTIMATION

The knowledge about the various levels of abstractions at which an embedded system can be modeled is beneficial for optimizing the power consumption. In 1983, Gajski and Kuhn derived a chart, called Y-chart [7] which illustrates five different abstraction levels at which an embedded system can be modeled and analyzed. The most important properties of a digital system can be well defined with three different domains represented by lines and five abstraction levels represented by concentric circles as shown in figure 3.1. The behavior domain describes the functional behavior of the system, the structural domain defines various sub-systems and their interconnections, and the physical domain describe the geometric properties of the system.

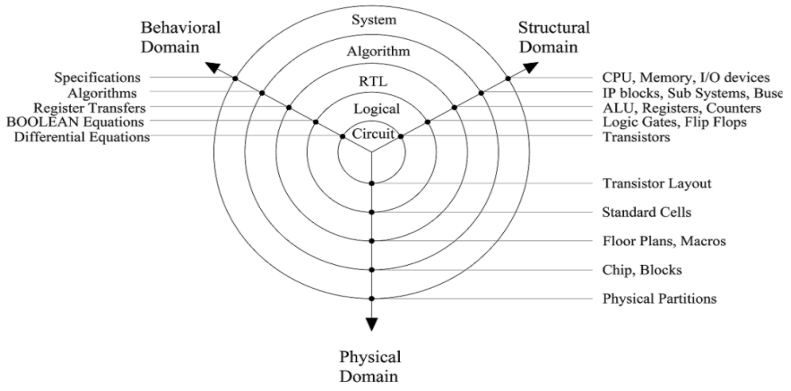

Figure 3.1: Y-Chart - Illustration of different levels of abstraction

The innermost circle which represents the physical level or Circuit/device level corresponds to the lowest abstraction level, while the outermost one, the system-level corresponds to the highest level. Since the amount of complexity was very limited in the past, it was possible to describe a system at the circuit level manually. However, as the complexity increased, manual description became impossible, and the starting point of the design flow was thus raised to logic or gate level, which translates a gate-level description into a layout design. As the complexity kept increasing, the starting point of the design flow was raised to Register Transfer Level (RTL) where hardware description languages are used for RTL description.The starting point of the design flow was further shifted to the algorithmic level and then up to the system level as the complexity kept growing. In general, it is possible to make the implementation faster by abstracting away from the lower-levels. Since the degree of possible optimum performance is directly proportional to the level of abstractions and indirectly proportional to the point in the design flow where decisions are taken, the decisions made at the higher level (system-level) have an unyielding impact on the quality of the end product. The main features of different levels of abstraction and approaches for power estimation at these levels are briefly described in the following sections.

System Level

At this level, the designers have to find the best architecture on which the given set of applications can be mapped, and hard choices such as the impact of power and performance on varying the number of abstraction levels, the interconnect to be used (bus or NoC), memory hierarchy, advantages and disadvantages of implementing the whole set of applications with hardware or hardware/software co-design are taken at this level. It is critical as well as very hard to take decisions at this level due to the limited information regarding the requirement of the design space and any error at this stage may lead to design reiterations which will result in loss of time, money, and probably cause failure of the entire system itself. Considerations at this level are the decisions such as the number of processors, the topology of the memory subsystem, memory hierarchy (cache levels and sizes), and communication models [1],[7].

Algorithmic Level

The algorithmic level is described by the definition of concurrent algorithms that describes a procedure for implementing a function as a pre-defined sequence of arithmetic operations, conditions, and loops [2]. The functional descriptions of the system and how the different subsystems interact with it are defined at this level through a set of instructions that are executed in sequence. Since only the casual relationship between the events or operations is known, the designers can only do some estimation about the temporal behavior of both the hardware and the software functions at this level. The various considerations at this level are the sequences of arithmetic operations, conditions, loops, language used for behavior modeling [7],[9].

Register Transfer Level (RTL)

The register-transfer level is a more detailed abstraction level which defines the micro-architecture of the system using hardware description language. The main objective of this level is to describe the data and control path, data transfer operations and their timing between various functional units with registers, register files, and arithmetic-logic units. Although the accuracy of estimation is lost at this level due to limited physical details of the design, estimation at this level provides valuable information. Considerations on RTL include controls such as single-cycle, multi-cycle, pipeline, internal data storage structures, clocking scheme, frequency, and phase signals [2], [9].

Logic Level or Gate Level

This level defines the behavior of RTL with Boolean equations. In the structural domain, the logical level is described with logic gates and flip-flops, and in the geometric domain, it is described by standard cells. The power estimation at this level is simplified because at this level standard cells with pre-characterized physical properties are used. Only the impact of cell-to-cell connecting wires need to be estimated separately at this level, and it can be done using wire load models. At this level, the power consumption of somewhat bigger sized designs can be estimated faster than the circuit-level approaches but at the expense of accuracy. Considerations on gate-level include handling logical hazards, two-level or multilevel logic realization, and gate-level architecture optimization of functional units [9].

Circuit and Device Level

The circuit and device level describe the behavior of a system by a mathematical model using differential equations. The function of a logic gate can be implemented by a circuit level model of interconnected transistors or a logical network of transistors called “standard cell.” This level is the actual hardware level where electric characteristics of the transistors are used to describe the system. Information on this level is printed on a silicon layout to fabricate the chip [2],[8]. At this level, power can be estimated by simulating the circuit at the transistor or switch level and monitoring the current from the supply. Typically, at this level designers seek very precise and accurate estimates. Considerations on this level include; threshold voltage, switching frequency, power consumption, size, and scaling which includes short-channel and straight-channel effects, hot-electron effect, gate-depletion, latch-up, preventing or exploiting quantum effects [9].

Estimating power at different levels of abstraction earlier in the design flow is desirable to take important design decisions and to identify problems in the design. The stages higher in the flow are at a higher abstraction level and do not involve implementation details. As we get lower in the flow, more physical particulars of the design will have been determined. Performing estimation of power at the lower levels in the flow will give more accurate estimates, but it will take more computational resources. The information required for estimating the power at different levels of abstraction are; activity estimates to compute the dynamic power dissipation, description of what the design looks like, and models of various components and connections. In general, the increase of the level of abstraction is directly proportional to the speed of estimation and inversely proportional to the accuracy of estimation. With growth in size and complexity of designs, it is desirable for designers to be able to estimate the power at higher levels of abstraction so that the information can guide early architectural decisions. [9],[27].

3.POWER REDUCTION TECHNIQUES

A set of hardware and software techniques used to optimize the power consumption of a system while maintaining its performance requirement is referred to as power management. The various techniques for optimizing the power consumption in a low power embedded system at different levels of abstraction during the design, fabrication, and run-time are briefly described in the following section. This includes both hardware and software techniques for reducing power in the hardware platform, communication links, memory, peripherals, and power delivery networks (PDN).

Transistor Sizing

This technique is based on the fact that reducing transistor width can cause a reduction in dynamic power. However, reducing the transistor size increases the circuit delay. So, normally the transistors that are on the non-critical paths are the best candidates for this technique. Considering the delay constraint, finding the minimum size of a transistor to minimize the power consumption is computationally a difficult task. Algorithms for this technique usually associate a tolerable delay, which varies depending on how close the transistor is to the critical path. The practical approach is to compute the slack at each logic gate to find how much the gate can be slowed down without a significant effect on the critical delay of the circuit [11],[21].

Transistor Reordering

The power dissipation in a CMOS circuit depends on the switching activities of the transistors; hence the arrangement of transistors in the circuit affects the power consumption. The basic idea of this technique is to place the transistors closer to the output of the circuit if they frequently switch to prevent the domino effect where the switching activity from one transistor propagates into many other transistors in the circuit causing dynamic power dissipation. This approach requires profiling techniques to determine how frequently the different transistors are likely to switch [21],[23]

Low Power Clocks (Half Frequency - Half Swing Clocks)

Low power clocks, also called half frequency and half swing clocks is another technique for reducing the power consumption at the circuit level. Normally, register file write operations occur during the rising edge of a clock signal. However, half frequency and half swing clocks synchronize the events using both edges of the clock, and also they tick at half the speed of the normal clocks which reduces the switching power to half. The reduced-swing clocks use a low voltage signal which further reduces the dynamic power quadratically [22].

Transistor Stacking

Transistor stacking is a technique used in an active mode to reduce the sub-threshold leakage current flowing through a stack of transistors by disconnecting one or more transistors connected in series. Since an increase in the source voltage of CMOS transistor reduces the sub-threshold leakage current exponentially, the dependence of the subthreshold current can be exploited by the transistor stacking effect. Also, it is possible to save more leakage power by increasing the number of transistors connected in series in a stack structure [20].

Logic Resizing or Gate Sizing

The switching power dissipated by a CMOS logic gate is directly proportional to the capacitive load C. Since the input/output capacitance of a gate is proportional to the size of the gate, reducing the gate size can reduce C as well as the switching powerconsumed by the gate. However, reduction in size increases the operation delay. So to preserve the timing behavior of the gate, only the size of the gates that do not belong to the critical path can be reduced [10],[17]. With logic resizing, switching power can be reduced without much impact on the design flow.

Logic Restructuring

Logic restructuring is a gate-level dynamic power optimization technique which helps mainly to reduce the glitching power of a logic circuit. The restructuring through logic equivalence transformation (for example three-stage logic to two-stage logic) can reduce the switching transitions in the logic circuit by moving high switching operations up in the logic cone and the low switching operations down in the logic cone [17]. This reduction in switching activities at the gate level saves the dynamic powerin the system without much impact on any other aspects of the design flow.

Pin Swapping

In this method, the gate pins are swapped so that the most frequent switching activity occurs at the pins with a lower capacitive load. Typically, an automatic pin swapping algorithm is used for pin swapping. By this technique, switching powercan be reduced without any significant impact on other aspects of the design flow [17].

Operand Isolation

In certain control-dominated designs, the arithmetic data path components are used only for a few clock states and stay in the idle state for a significant amount of time. Whenever a circuit performs an operation whose output is not utilized in the downstream module, this operation is termed as a redundant operation. This redundant operation unnecessarily dissipates power and causes significant overhead in system power consumption. This technique reduces power dissipation in the data path components by isolating the redundant operations with little or no impact on the other aspects [17].

Clock Gating

Normally, all circuit blocks in a system such as ALU, flip-flops, counters, registers, and memory in a system are not accessed in every clock cycle, but the clock signal continues to toggle these units at every clock cycle. Since the system clocks switch at the maximum rate and typically have large capacitive loads, clock trees consume a significant amount of dynamic power. If data is not updated into certain units frequently, a substantial amount of power can be saved by shutting off the units by disabling the clocks to ensure that the power is not wasted during the idle time [10].

Multi-Threshold Voltage (MTV)

This technique utilizes gates with different thresholds voltages to optimize the power, timing, and area. The circuit paths with high speed are designed using low V_{th} devices while devices with high V_{th} are applied at the other paths to reduce the sub-threshold leakage current. It requires additional fabrication steps to support multi-V_{th} cells, which eventually increases the design time, fabrication complexity and may reduce yield [10]. Typically, the low-V_{th} gates switch more quickly, but consume more leakage power and provide high performance, whereas the high-V_{th} gates switch more slowly but consume less leakage power and provide lower performance. So, usually the tradeoff favors power. By using a high-V_{th} cell in place of the low-V_{th} cell, it is possible to achieve a significant reduction in leakage powerwithout any significant impact on any other aspects of the design flow [17],[18]

Substrate Biasing or Adaptive Body Biasing

In this approach, the substrate or the appropriate well is biased to raise the transistor threshold voltage in active mode thereby reduce the leakage current. Since the leakage power is a function of V_{th}, substrate biasing can reduce leakage power. This method incurred area and routing penalties, because body cells are placed throughout the design to provide voltages for transistor bulk. Also, a bias voltage generator is needed, which also dissipates some dynamic power, partially offsetting the leakage power [10],[17].

Memory Splitting

In most of the systems, only a portion of the memory is used at a given time. So, memory capacity is set for the peak and normal usages. If the software and data are persistent in one section of memory but not in another, it is possible to split that section of memory into two or more blocks. Then it is possible to selectively shut off the unused sections of the memory block to save the leakage power dissipation during the idle period. The leakage power saving by memory splitting depends on the software, and there is an additional cost related to the insertion of the power shut-off circuitry [10],[17].

Bus Segmentation

Bus segmentation is an effective technique for power reduction in interconnects. Buses in a system consume a significant amount of dynamic power due to high switching activities and large capacitive load. In shared bus architecture, the entire bus system is charged and discharged during every access even though the devices connected to some segments of the bus system are not active. This technique saves power by allowing most of the buses to remain powered down and only the buses essential for communication are activated independently. Researchers developed many algorithms for segmenting the buses to benefit from this technique[19].

Multi-Supply Voltage (MSV) or Voltage Islands

Multi-Supply Voltage technique uses different supply voltages for various blocks in the chip based on their performance requirements – low voltage supply for low-performance circuit and high voltage for the high-performance circuit. This technique is very effective in saving power because lowering the supply voltage has a squared effect on active power consumption in CMOS circuit. However, reduction in power consumption is only possible at the expense of speed and performance. Typically, in a multi-supply-voltage circuit, the normal or high voltage is applied to the circuit on the critical paths, while the reduced voltage is applied to the circuit on the non-critical path. Here the critical path of a circuit is not changed so; this transformation does not affect the performance and speed. This technique may reduce dynamic power with a significant improvement in leakage power. However, the additional level shifters add area overhead which may impact clock scheduling. Also, the design turn-around time may increase due to the added design complexity [17],[18]

Dynamic/Adaptive Voltage and Frequency Scaling (DVFS/AVFS)

DVFS technique and its associated techniques such as dynamic voltage scaling (DVS) and adaptive voltage and frequency scaling (AVFS) are very effective in reducing dynamic power because lowering the voltage has a squared effect on the active power consumption in a CMOS circuit. These techniques provide different ways to minimize power consumption in a processor core by scaling down the voltage and frequency based on the performance requirements of the targeted application. Since DVFS and AVFS techniques optimize both the voltage and frequency, they are highly effective in minimizing both active and static power. In DVFS, the voltages of the targeted system are scaled in fixed discrete steps, and a frequency-based voltage table determines the voltage levels. The DVFS is an open-loop system with large margins built in so that the power reduction is not optimal. However, AVFS adopts a closed-loop voltage scaling method that constantly monitors the system performance and provides a positive feedback to compensate for the variations in process, temperature, IR drop using dedicated circuitry. The AVFS technique is more complex, and the payoff in terms of power reduction is much higher than that of DVFS technique. Typically, DVFS can save a significant amount of total power with little area impact [17],[18].

Power Shut-Off (PSO) or Power Gating

Power gating – also called power shut-off (PSO) **-** is the single most effective technique for reducing the leakage power of the idle circuit blocks in a chip. Power gating is employed to temporarily shut off power to parts of the circuit blocks that are not in use (or in sleep, deep-sleep, standby modes) and the power is supplied again when the blocks are required for operations. During the shut-off period, the circuit block is not operational hence no leakage power is dissipated in this block. The power gating can be implemented in two ways; On-chip power gating which means power switches are within the chip and off-chip power gating which means the power switches are external to the chip. This technique can save up to 96% of the leakage power without any impact on dynamic power. However, it has an impact on timing and area[17],[18].

Using Hardware Accelerators

Power consumption in the hardware can be minimized by putting the functionality in software. However, it may significantly affect the system performance. Mapping functionality into hardware accelerators is an efficient technique for optimizing the power, but some tradeoffs must be allowed between power and performance.The rule of thumb in an embedded system is that adding circuits or hardware means more power consumption. However, by analyzing the algorithms and realizing appropriate hardware in programmable logic (called hardware accelerators), the designers can increase the performance of an embedded computing system while reducing the power consumption. Also, the rule of thumb is that the power consumption in a CMOS circuit is roughly proportional to the area of the chip for the given process technology. Hence the design implemented in programmable logic tends to be larger in size and consumes more power than the one implemented in hard-wired logic. Practically, the frequency at which a CMOS circuit operates has more impact on power consumption than the chip size, because it draws most of the current when transistors are switching. Since the switching speed depends on frequency, the higher the frequency, the greater the power consumption. This opens the possibility that power consumption in CMOS circuit can be reduced by adding hardware with a significant reduction in clock speed. If the hardware can be applied to accelerate software algorithms and decrease the clock frequency, power can be minimized while meeting system performance. This technique provides either a better performance for a given clock speed or lower the clock speed for a given performance [13].

Power Savings Modes

Many embedded control applications do not require the system to be processing data at all the time. Traditionally, for small applications, the processor simply runs an idle loop when there is no useful work to be done. However, for large systems, the leakage power is a major component of the total power consumption, making it essential to power the processor down when it is idle. To address this issue, most embedded processors have three instruction-based power saving modes such as sleep, idle, and deep sleep. Sleep mode is the low power mode of the system in which, the processor, system clock, and peripherals that operate on the clock are disabled. In idle mode, the processor is disabled, but the system clock continues to operate. The peripherals also continue to operate, but can optionally be disabled. The deep sleep mode is also the lowest power mode in which the processor, system clock, and all the peripherals except real-time clock and calendar and deep sleep watchdog timer are disabled. Depending on the application these power modes can be activated by interrupts, watchdog timer output or device reset request managed by software. The wake-up time is the main drawback of this method. Normally, the processor responds quickly to interrupts, but the start-up time of the peripherals might slow down the processor operations [12].

Resource (Peripheral) Hibernation

The embedded application requires a number of peripheral devices such as displays, cameras, sensors, multimedia devices, and network interfaces, but only certain devices are needed for a certain time. Although switching activity is the main cause of power consumption, the peripheral devices consume power even during their idle period. Therefore, if the processor can switch off the devices that are not in use, the static power consumed by the devices can be reduced to zero. This is the basic idea behind the resource hibernation technique [11],[12]. Selective peripheral clocking is one of the most commonly used techniques used for saving the static power consumption in peripherals.

Selective Peripheral Clocking

The sleep, idle and deep sleep modes reduce the processor power consumption significantly by stopping or slowing the processor clock. However, the peripheral devices remain clocked and thus consume some amount of power. Also, there may be cases where certain peripheral devices are not needed for a period, but remain powered and clocked leading to a significant amount of power dissipation. The selective peripheral clocking techniques address this issue by allowing peripheral devices to selectively power down to reduce or eliminate the power consumption. Typically, most microcontrollers have individual control over the clocking of each peripheral, and most peripheral devices have software controlled enable/disable pin to selectively enable/disable each device on demand during all operating modes of the processor. The drawback of this method is the start-up time of the peripherals, which might slow down the performance of the processor [12],[26]

Configuring Power Delivery Network

The power conversion efficiency of the power delivery network (PDN) is a critical factor that is often overlooked while traditional power optimization techniques have been heavily investigated. The PDN is an essential component in an embedded system, which delivers power to all devices in the system from the power source. Modern applications like real-time processing of multimedia streams use multicore processors (chip multicore processors –CMPs) and a variety of configurable hardware accelerator blocks to achieve high computing density at the rate of giga-operations per second. Applying DVFS to each core (per core DVFS) and hardware accelerator is the most efficient techniques for reducing the power consumption in CMPs. It necessitates a PDN with the same number of DC-DC converters as the number of cores and hardware accelerators, which in turn dissipates a significant amount of power in the PDN. Thus, reducing power in the PDN can considerably reduce the total power consumption in a system. Reconfigurable PDN is the best solution to minimize the power consumption in the power delivery network [14],[15].

Software Power Optimization

Software constitutes a major component and it plays a significant role in reducing the power consumption of low power embedded systems, because developing power-efficient embedded applications heavily depends on software that uses the hardware resources most appropriately. Even if the hardware facilitates power reduction techniques such as power gating, DVFS, clock gating, sleep modes, and external peripheral clocking, if the software is not shutting down the clock or not powering down the supply voltage or not leveraging the voltage or frequency scaling, the system is doing nothing in terms of power minimization. Also, the optimal solution can be achieved only if the software requirements are accounted for in the hardware during the design. The more flexible the hardware is in terms of processor, clock systems, memory, buses, peripherals and power supply, the higher the potential energy savings the designer can achieve. The way in which various functional blocks of an embedded processor are arranged has a great impact on overall power consumption. The main areas, where the software has an influence on power consumption are; size of the program, code compression, memory, processor utilization and modes of operation, system buses, peripherals, power supply, and system clocks[24],[25].

4.CONCLUSION

In this paper, we reviewed several techniques for optimizing the power consumption at different levels of abstraction of low power embedded systems. Some of these techniques are easy to integrate during the design flow, but others are hard to adopt. The criteria for the selection of these techniques depends on their suitability at different levels of abstraction and the type of applications. Some techniques are not specifically utilized at a particular abstraction level but can be realized by the use of other techniques. We hope that the review presented in this paper would help the readers in gaining a greater insight into the implementation of power reduction techniques at different levels of abstraction of the low power embedded systems.

REFERENCES

[1] Richard Zurawski, “Embedded Systems Handbook”, Embedded Systems Design & Verification (IIT, 2nd Ed, CRC Press, ISBN 9781439807552

[2] F. Vahid and T. Givargis “Embedded System Design: A Unified Hardware/ Software Introduction”, John Wiley & Sons,New York, NY, USA: Inc., 2001.

[3] Sparsh Mittal: “A survey of techniques for improving energy efficiency in embedded computing systems”, International Journal of Computer Aided Engineering and Technology, 6 (4), pp.440-459, 2014.

[4] R. Mendoza; A. Ferre; L. Balado; J. Figueras: “CMOS leakage power at cell level”, IEEE International Conference on Design and Test of Integrated Systems inNano-scale Technology, DTIS 2006, Year: 2006 Pages: 194 – 199, 2006.

[5] S. Borkar, “Low power design challenges for the decade”,Proceedings of the ASP-DAC, IEEE Asia and South Pacific Design Automation Conference, Pages: 293 - 296, 2001

[6] Tadahiro Kuroda, “ Low Power CMOS Design Challenges”,Invited paper, Special issue on Silicon Nano-devices, IEICE Trans. Electron, Vol.E84-C, 2001

[7] Santhosh Kumar Rethinagiri, “System-Level Power Estimation Methodology for MPSoC based Platforms” https://tel.archives-ouvertes.fr/tel-00943272, 2014. Universit_e de Valencienneset du Hainaut-Cambresis

[8] C. Talarico; J. W. Rozenblit; V. Malhotra; A. Stritter; “A new framework for power estimation of embedded systems”,Computer , Volume: 38, Issue: 2 Pages: 71 - 78, IEEE Journals & Magazines, Year: 2005

[9] Nathalie Chan King Choy, “Activity-based Power Estimation and Characterization of DSP and Multiplier Blocks in FPGAs”,Thesis 2006, The University of British Columbia.

[10] http://semiengineering.com/kc/knowledge_center; contents originallyprovided by cadence Design Systems

[11] Vasanth Venkatachalam, Michael Franz, “Power reduction techniques for microprocessor systems”,ACM Computing Surveys, Vol. 37, No. 3, Pages 195–237, 2005

[12] LubomirValerievBogdanov, RachoMarinov Ivanov, “Approaches for reducing the power consumption in embedded systems”,Annual Jrnl. of Electronics, ISSN 1314-0078, 2012.

[13] Altera Corporation, “Adding Hardware Accelerators to Reduce Power in Embedded Systems” White paper, ver. 1.0 1, 2009.

[14] W. Lee, Y. Wang and M. Pedram, "Optimizing a Reconfigurable Power Distribution Network in a Multicore Platform," in IEEE Trans. on Comp.-Aided Design of Integr. Circuits and Syst, vol. 34, no. 7, pp.1110-623,2015.

[15] Woojoo Lee, “Tutorial: Design and Optimization of Power Delivery Networks”,IEIE Transactions on Smart Processing and Computing”, vol. 5, no. 5, 2016. http://dx.doi.org/ 10.5163/IEIESPC.2016.5.5.1

[16] MassoudPedram, “Power optimization and management in embedded systems”,IEEE Design Automation Conference, 2001.Proceedings of the ASP-DAC 2001.Asia and South Pacific.IEEE Xplore: 2002

[17] Chi-Ping Hsu, “A Practical Guide to Low-Power Design-User Experience with CPF”, Silicon Integration Initiative (Si2) https://projects.si2.org/?page=1061,

[18] Sunil Deep maheshwari , Naveen Srivastava, and Rohit Ranjan, “Reducing IC power consumption: Low-power design techniques”http://www.edn.com/design/integrated-circuit-design/4440415/Reducing-IC-power-consumption--Low-power-design-techniques,**EDN network**, September 24, 2015

[19] W.-B. Jone,J. S. Wang, Chia-Yi, Hsueh-I, I. P. Hsu, and J.-Y. Chen, “Design theory and implementation for low-power segmented bus systems”, ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 8 Issue 1, Pages 38-54, 2003

[20] Siva Narendran, Shekhar. Borkar, Vivek De, Dimitri Antoniadisn, and AnanthaChandrakasann, "Scaling of Stack Effect and its Application for Leakage Reduction,"inProc. ISLPED, August 2001, pp. 195-200.

[21] MahrooZandrahimi, Zaid Al-Ars," A Survey on Low-Power Techniques for Single and Multicore Systems" Proceedings of the 3rd International Conference on Context-Aware Systems and Applications, October 2014 ICCASA '14,nstitute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2014

[22] E. Kursun, S. Ghiasi, and M. Sarrafzadeh, "Transistor Level Budgeting for Power Optimization" in ISQED, pp. 116-121,2004 [68] A. Sultania, D. Sylvester, and S. Sapatnekar, "Transistor and Pin Reordering for Gate Oxide Leakage Reduction in Dual Tox circuits" in ICCD, pp. 228-233, 2004.

[23] Bernard Cole, "The challenges ahead in embedded systems design",http://www.embedded.com/ electronics-blogs/cole-bin/4430318/The-challenges-ahead-in-embedded-systems-design, May 19, 2014

[24] HarnsLekatsas, Wayne Wolf, and Joerg Henkel, "Arithmetic coding for low power embedded system design", IEEE Proceedings of the Conference on Data Compression, page 430, Washington, DC, USA, IEEE Computer Society, 2000.

[25] A. Parikh, Soontae Kim, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. "Instruction scheduling for low power" J. VLSI Signal Process. Syst., 37(1):129-149, 2004.

[26] Wang; S. Malik; R. A. Bergamaschi, Shaojie "Modeling and integration of peripheral devices in embedded systems 2003 Design", IEEE Automation and Test in Europe Conference and Exhibition, Pages: 136 – 141, 2003

[27] P. E. Landman and J. M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," IEEE Transactions on VLSI Systems, vol. 3, no. 2, pp. 173–187, 1995.