

# A Methodology for Designing Low Power Sensor Node Hardware Systems

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik  
der Brandenburgischen Technischen Universität Cottbus-Senftenberg

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften  
(Dr. -Ing.)

genehmigte Dissertation

vorgelegt von

Dipl.-Ing.

Goran Panić

geboren am 31.05.1976 in Sarajewo, Bosnien und Herzegowina.

Gutachter: Prof. Dr.-Ing. Rolf Kraemer

Gutachter: Prof. Dr.-Ing. Heinrich Theodor Vierhaus

Gutachter: Prof. Dr. Mile Stojčev

Tag der mündliche Prüfung: 18.12.2014



*I dedicate this thesis to my beloved wife, Branka, who supported and encouraged me each step of the way. I'm truly grateful for having you in my life. This work is also dedicated to my parents, Zoran and Danica, my brother Milan, sister Ana, and my aunt Milica. Thanks for never stopped believing in me.*



## **Acknowledgments**

I want to express my deepest gratitude to all persons that were involved in any way in the development process of this thesis.

First of all, I want to thank to my mentors, Prof. Dr. Rolf Kraemer and Prof. Dr. Peter Langendörfer. Without their support this work would not be possible. Thank you for all your patience, useful advices, and guidance to the final goal.

I'm especially grateful to Dr. Zoran Stamenković whose help was critical for giving the thesis its final shape and preparing the defence.

I want to thank to my colleagues Dr. Miloš Krstić, Dr. Steffen Ortmann, and Frank Vater for their useful tips and advices regarding thesis writing.

I'm very thankful to Prof. Dr. Heinrich Theodor Vierhaus and Prof. Dr. Mile Stojčev for the time and effort they invested in the review process of this work.

Also, I want to thank all other colleagues from IHP that contributed in different aspects of this work either by providing their technical support or useful hints.

Finally, I want to thank my wife and my family for their unconditional support and love.



# Table of Contents

Abstract .....	1
Zusammenfassung.....	2
<b>1. Introduction.....</b>	<b>5</b>
1.1. Motivation .....	5
1.2. Objectives .....	8
1.3. Contributions.....	9
1.4. Structure.....	10
<b>2. State of the Art .....</b>	<b>11</b>
2.1. Power Consumption .....	11
2.1.1. Power vs Energy .....	12
2.1.2. Dynamic Power Loss.....	13
2.1.3. Static Power Loss.....	15
2.2. Standard Low Power Techniques .....	17
2.2.1. Supply Voltage Scaling.....	17
2.2.2. Multi-Threshold Voltage.....	18
2.2.3. Clock Gating.....	19
2.2.4. Operand Isolation.....	19
2.2.5. Gate-Level Optimization.....	20
2.2.6. Overview.....	20
2.3. Advanced Low Power Techniques.....	21
2.3.1. Process Techniques .....	21
2.3.2. Multi-Voltage Design.....	24
2.3.3. Power Gating .....	28
2.3.4. Body Biasing .....	36
2.3.5. Stacked Transistors.....	38
2.3.6. Subthreshold Logic .....	39
2.3.7. Dynamic Power Management .....	39
2.3.8. Overview.....	39
2.4. Power Aware Design Methodologies .....	40
2.4.1. CPF-Enabled Cadence Low Power Flow .....	41
2.4.2. UPF-Enabled Synopsys Low Power Flow .....	42
2.4.3. High-Level Power Estimation.....	43
2.4.4. Energy Overhead Modelling.....	45

2.5.	Sensor Node .....	47
2.5.1.	Sensor Node Platforms.....	47
2.5.2.	Embedded Sensor Node Processors.....	50
2.5.3.	Overview.....	53
2.6.	Future Trends in Low Power Design.....	54
<b>3.</b>	<b>A Methodology for Choosing Optimal Power Saving Strategy in SoC Design.....</b>	<b>59</b>
3.1.	Methodology Overview.....	59
3.2.	Power Estimation Models .....	60
3.3.	Implementation Margin .....	62
3.3.1.	Technology Limits.....	63
3.3.2.	Break-Even Point .....	64
3.4.	Implementation Overhead .....	65
3.4.1.	Power Gating Overhead .....	65
3.4.2.	Multi-Voltage Overhead.....	70
3.4.3.	DVFS Overhead.....	70
3.5.	Activity Profiling .....	71
3.6.	Activity Profiling in Sensor Nodes .....	73
3.6.1.	Example of Activity Profiling in Schedule-Based WSN .....	75
3.6.2.	Example of Activity Profiling in Wake-Up Controlled WSN.....	77
3.6.3.	Impact of Network Topology.....	79
<b>4.</b>	<b>Power-Driven Design Flow .....</b>	<b>83</b>
4.1.	Overview.....	83
4.2.	Power-Aware System-Level Design.....	84
4.3.	Power-Aware RTL Design .....	85
4.4.	Power-Aware Implementation.....	86
4.5.	Power-Aware Verification .....	87
4.6.	Tools .....	89
<b>5.</b>	<b>Design of Embedded Sensor Node Microcontroller .....</b>	<b>91</b>
5.1.	Introduction.....	91
5.2.	Development of Sensor Node Microcontroller .....	91
5.3.	Requirements .....	93
5.4.	Sensor Node Architecture .....	94
5.4.1.	Processor Core and Peripheral Interface .....	95
5.4.2.	Clock Module and DCO.....	96
5.4.3.	Digital IO .....	97

5.4.4.	Timers .....	97
5.4.5.	Serial Ports.....	97
5.4.6.	Memory Subsystem.....	98
5.4.7.	Crypto Accelerators.....	98
5.4.8.	Communication Accelerator.....	99
5.4.9.	Analogue-to-Digital Converter .....	100
5.5.	Power Consideration .....	100
5.6.	Design of Power-Gating Components .....	101
5.6.1.	Power Switches .....	101
5.6.2.	Isolation Cells.....	104
5.6.3.	Power-Gating Controller .....	105
5.6.4.	AES Test Circuit.....	107
5.7.	Application of Power Saving Methodology to Sensor Node Design .....	110
5.7.1.	Selection of Power-Gating Candidates.....	110
5.7.2.	Determination of Break-Even Point .....	112
5.7.3.	Definition of Target Application .....	114
5.7.4.	Extraction of Activity Profiles .....	116
5.7.5.	Target Network Topology and Energy Estimation.....	118
5.7.6.	Selection of Power Saving Strategy.....	122
<b>6.</b>	<b>Implementation and Evaluation .....</b>	<b>131</b>
6.1.	System Implementation .....	131
6.1.1.	Synthesis.....	131
6.1.2.	Layout .....	132
6.2.	Verification and Measurements.....	133
6.2.1.	System Level Tests.....	133
6.2.2.	Turn-On Time and Rush Currents.....	134
6.2.3.	Production Tests.....	135
6.2.4.	Power Consumption .....	135
6.2.5.	Performance .....	137
6.2.6.	Evaluation Platform.....	137
<b>7.</b>	<b>Conclusion .....</b>	<b>139</b>
	References.....	141
	List of Figures.....	155
	List of Tables.....	159
	List of Abbreviations.....	161



## Abstract

The design of embedded sensor node hardware systems is a challenging task driven by the increasing demands for low power, high efficiency, low cost and small size. These unique requirements make the usage of off-the-shelf general purpose microcontrollers fairly inefficient. For many wireless sensor network applications, the design of a dedicated low power sensor node microcontroller is the only way to answer specific application requirements. According to the trends in device, process and design technology, the development of sensor node devices is relying on a cheap planar bulk-CMOS technology, where power consumption is dominated by static power loss caused by high leakage currents. To keep the power at acceptable level, designers are compelled to apply the methodologies based on advanced low power techniques that target both static and dynamic power in the chip. The decisions made early in design phase are likely to determine the energy efficiency of the final design. Therefore, the choice of power saving strategy is the key challenge in designing energy-efficient sensor node hardware.

This work presents a methodology that assists designers meeting the critical design decisions regarding power, early in the design process. The presented methodology extracts the activity profiles of single system components and applies them in the developed models for energy estimation of particular low power implementation. The energy estimation models account for the energy overhead introduced by specific low power techniques, enabling comprehensive exploration of system's energy efficiency in a given application scenario. Special attention is paid to the methodology utilization in typical wireless sensor network applications. Accordingly, the examples of activity profiling in wireless sensor node systems are presented. The proposed methodology is integrated within a power-driven design flow and applied to the design of an embedded sensor node microcontroller. This methodology is used to perform the cross comparison of alternative low power implementations for the target system architecture. The implementation relying on concurrent clock and power gating is selected as the most energy efficient and consequently realised. Power switching cells and power control logic have been designed and characterized. Also, the final system architecture, basic system components and applied design process are described. Finally, the developed power-gated sensor node microcontroller is implemented, fabricated and successfully tested. The chip measurements results are presented and analyzed.

The analysis of different low power approaches applied to the target system architecture has shown large impact of clock gating on the system energy. In a given application scenario, the clock gating implementation has reduced 72 times the dynamic energy and 12 times the total energy of the system. The implementation of power gating technique has gained 2.8 times reduction of the leakage energy and 2 times reduction of the total system energy compared to the clock gating only implementation. The analysis of two alternative power gating approaches has emphasized the significance of partitioning in power-gated design. A heuristic partitioning that combines two specific blocks having successive activity phases into a single power domain, thereby reducing design complexity and chip area, has been shown to have positive impact on the energy efficiency of the target design.

## Zusammenfassung

Das Design von eingebetteten Hardware-Systemen für Low-Power-Sensorknoten ist eine anspruchsvolle Aufgabe, die durch stetig steigende Anforderungen an geringe Leistungsaufnahme, niedrige Kosten, hohe Performance und hohe Energieeffizienz getrieben wird. Aus diesem Grund ist die Verwendung von Off-the-Shelf Mikrocontrollern sehr ineffizient. Für viele drahtlose Sensornetzwerkanwendungen ist die Integration dedizierter Low-Power-Mikrocontroller der einzige Weg die spezifischen Anwendungsanforderungen entsprechend zu erfüllen. Die technologischen Trends in der Fertigung von Sensorknoten auf Basis von Standard-CMOS-Technologien führen zu einem zunehmend dominierenden Anteil an Leckströmen bei der gesamten Leistungsaufnahme. Um die Leistungsaufnahme auf einem akzeptablen Niveau zu halten sind Hardwareentwickler gezwungen, fortschrittliche Methoden und Low-Power Techniken einzusetzen, die sowohl die statische als auch die dynamische Leistungsaufnahme der Hardware reduzieren. Dabei bestimmen viele Design-Entscheidungen in der frühen Phase des Hardwareentwurfes maßgeblich die Energieeffizienz des endgültigen Produktes. Daher stellt die Wahl der richtigen Energiesparstrategie die zentrale Herausforderung zum Entwurf energie-effizienter Sensorknoten-Hardware dar.

Diese Arbeit stellt eine Entwurfsmethodik für energieeffiziente Low-Power Hardware vor, die den Designer bereits frühzeitig im Designprozess bei kritischen Entscheidungen unterstützt. Die hier vorgestellte Methode extrahiert Aktivitätsprofile einzelner Systemkomponenten und verwendet diese als Grundlage zur Modellierung und Abschätzung der Leistungsaufnahme ausgewählter Low-Power Implementierungen. Die entwickelten Modelle ermöglichen eine Abschätzung des Energieaufwands der verschiedenen Implementierungen im Hinblick auf die einzelne Komponente und auf die zu erwartende Energieeffizienz des Gesamtsystems für das jeweilige Anwendungsszenario. In der vorliegenden Arbeit wird besonderes Augenmerk auf die Anwendung der Entwurfsmethodik für typische drahtlose Sensornetzwerkanwendungen gelegt. Daher werden konkrete Anwendungsbeispiele für die Erstellung von Aktivitätsprofilen von drahtlosen Sensorknoten eingeführt. Die vorgeschlagene Methodik ist in einem Low-Power Entwurfsprozess integriert und wird zur Implementierung eines eingebetteten Sensorknoten angewendet. Dabei erlaubt die Methodik zwischen alternativen Low-Power-Implementierungen für die Architektur des Zielsystems zu wählen. Als Anwendungsbeispiel wird die Umsetzung einer kombinierter Clock- und Power-Gating Architektur mit der besten Energieeffizienz im gegebenen Anwendungsszenario aufgezeigt und auf das Zielsystem angewendet. Die Konstruktion und Charakterisierung der dafür benötigten Power-Gating-Zellen und der zugehörigen Kontrolllogik, die zur Umsetzung erforderlich sind, werden ausführlich dargestellt. Die finale Systemarchitektur, die Grundkomponenten des Systems und der angewandte Entwurfsprozess werden ebenso beschrieben. Der mit der vorgestellten Methodik entworfene Mikrocontroller wurde in Standard-CMOS Technologie gefertigt und getestet. Die damit erzielten Ergebnisse der Funktionstests und der Messungen zur Leistungsaufnahme der gefertigten Chips werden präsentiert und ausgewertet.

Die Analyse der verschiedenen Low-Power Ansätze, die auf die Zielsystem-Architektur angewendet wurden, zeigte insbesondere beim Clock-Gating einen erheblichen Einfluss auf die benötigte Energie. Im spezifischen Anwendungsszenario führte das Clock-Gating zu einer Verbesserung der dynamischen Energie um den Faktor 72 und eine Verbesserung der Gesamtenergie des Systems um den Faktor 12. Die Anwendung der Power-Gating Technik resultierte in einer 2.8-fachen Verbesserung der Leckstromenergie und eine Halbierung der Energie des Gesamtsystems im

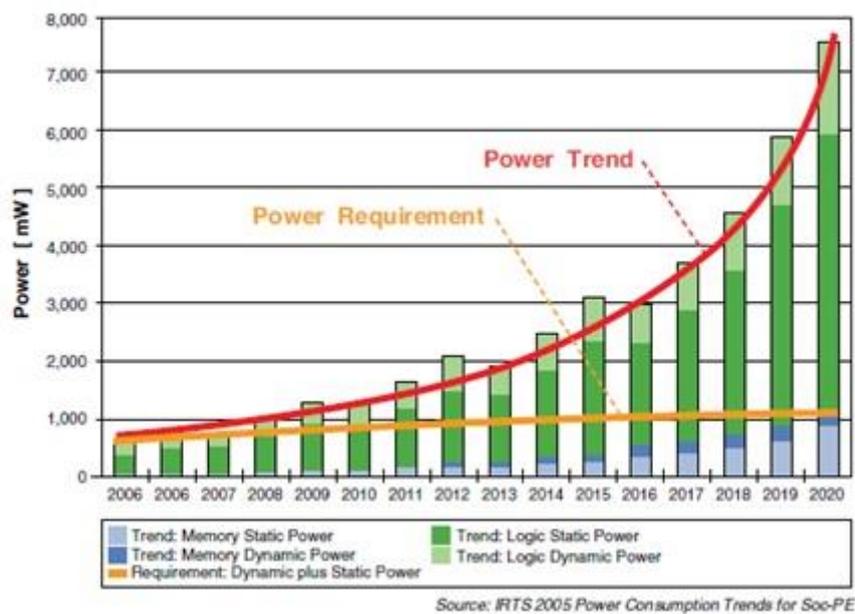
Vergleich mit einer Implementierung in der nur Clock-Gating angewendet wurde. Die Analyse von zwei alternativen Anwendungen der Power-Gating-Technik in der untersuchten Anwendung unterstrich die Bedeutung der Partitionierung bei Verwendung der Power-Gating Technik. Eine Kombination von zwei spezifischen Blöcken mit aufeinander folgenden Aktivitätsphasen, die zuvor in einer getrennten Power Domains waren, in eine gemeinsame Power-Insel, reduzierte die Designkomplexität und die benötigte Chipfläche. Darüber hinaus führte dies zu positiven Auswirkungen auf die Energieeffizienz des Designs.



# 1. Introduction

## 1.1. Motivation

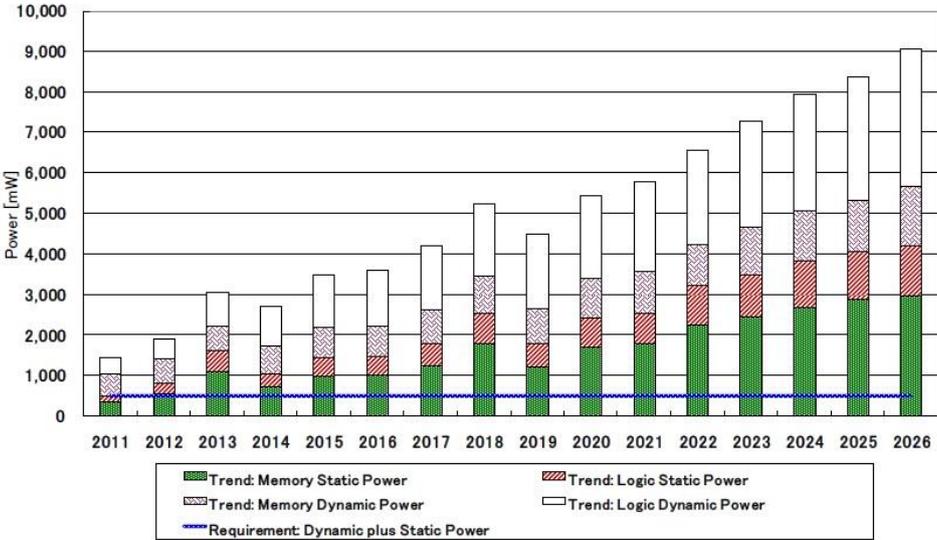
The last decade saw a tremendous growth in the market of wireless and handheld devices. The requirements to operate a device for a long time with a single battery charge made the need for low power being one of the most important challenges designers are facing today. Traditionally, system-on-chip (SoC) design was primarily focused on improving system performance, where the power was considered to be the second order design constraint. When applied, traditional power saving techniques targeted only dynamic power, i.e., the power consumed during the active mode of operation, making no interest in reducing static power. However, as the technology features continued to shrink, the contribution of static power increased exponentially [1]. The early ITRS (International Technology Roadmap for Semiconductors) projections of power trends in planar bulk-CMOS predicted static power to reach and even exceed the level of dynamic power consumption in chip [2]. This is illustrated in Figure 1.



**Figure 1.** Projected power trends vs power requirements in planar bulk-CMOS [2].

The widening gap between power trends and power requirements became the most critical challenge that designers of wireless and portable electronic devices face today. The necessity to design for low power led to the introduction of advanced low power techniques that can aggressively manage both dynamic and static power loss. The application of those advanced techniques can greatly impact design and implementation choices. However, for all designs of 90 nm and below, aggressive management of leakage is a must [73].

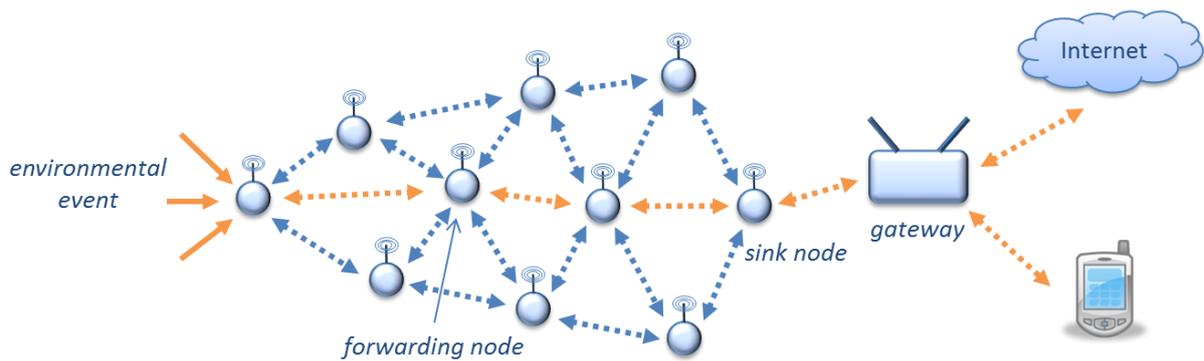
The latest innovations in material, device and process technologies foresee the ultra-thin-body silicon-on-insulator (SOI) and multi-gate CMOS (MG-CMOS) to be the dominant technologies of future, starting from 32 nm. According to the latest ITRS projections, these new technologies are expected to reduce the leakage contribution in total power of the chip. Nevertheless, the static power loss will still remain a significant source of power consumption (Figure 2). The ITRS also predicts that the most important advanced low power techniques developed for planar bulk-CMOS will continue to apply in future designs as well [172].



**Figure 2.** Projected power trends in future CMOS. The projection accounts for the impact of future technologies such as SOI and MG-CMOS [172].

At the moment, the production costs and diverse technology problems restrict the application of SOI and MG-CMOS only to high-end products. The low cost electronic devices and wireless products from niche markets, such as sensor nodes used in wireless sensor networks, are expected to future rely on mature planar bulk-CMOS processing, where the power consumption is dominated by leakage currents. Therefore, an efficient implementation of low power techniques that target both static and dynamic power remains the most critical design challenge towards power-efficient design.

Wireless sensor networks (WSN) consist of a number of distributed wireless sensor nodes (Figure 3). The nodes monitor events and collect data of interest. The monitored data is transmitted through the wireless network towards the sink node. Wireless sensor networks find their application in different areas, from industry automation and environmental monitoring to military surveillance and medicine [4]. Typical sensor node devices combine computing, sensing and communication features within a tiny, battery-powered hardware system. Once deployed, sensor nodes are organized in a network where the data is seamlessly routed among all the nodes. The size of the network depends on specific application, and in some cases it might scale to hundreds or even thousands of nodes. Since a periodic recharge of battery is usually not maintainable, the sensor nodes have to provide long-time operation with a very limited power budget. That makes power being the most important constraint when designing sensor nodes.



**Figure 3.** Wireless sensor network.

Sensor nodes play the crucial role in power efficient implementation of wireless sensor networks. A simple classification of sensor nodes [5] divides them into the three categories: a) adapted general-purpose computers such as PDAs and embedded PCs used for high-performance applications; b) embedded sensor modules built entirely from the commodity-of-the-shelf (COTS) components; and c) systems-on-chip that integrate micro electromechanical systems (MEMS) sensors, processor and wireless transceiver into a single application-specific integrated circuit (ASIC). The general-purpose computers are usually used in high performance applications or as gateway nodes, which perform data aggregation and provide link to the end user. The generic sensor node platforms based on COTS components are widely used by research community to develop applications and communication protocols for wireless sensor networks. They offer good flexibility and software support, but in most cases they cannot fully answer the specific requirements for low power and high performance. Also, the size and the cost of generic nodes are not acceptable in many application fields. To overcome these limitations, many researchers focused their work on novel architectures for embedded sensor node processor design. The efforts were based on non-standard system and circuit techniques such as asynchronous design and subthreshold logic as well as on the application of advanced power saving techniques and hardware accelerators to standard processor architectures. Although, the use of asynchronous logic in sensor processor design might simplify the clocking related issues, it also introduces significant design overhead. The implementation of asynchronous logic is not straightforward and therefore, it is not supported by any industrial design flow. Additionally, the implementation of advanced leakage saving techniques might be difficult with an asynchronous design. The processors running near or below threshold voltage promise ultra-low power operation, but their performance is greatly sacrificed for power. Additionally, the subthreshold logic strongly depends on the process and temperature variations and it requires the implementation of specialized logic cells and memory blocks to provide correct operation in subthreshold regime. Also, interfacing subthreshold logic to common I/O and radio devices might be difficult. The design of an embedded sensor node microcontroller that utilizes advanced low power techniques and hardware acceleration for specific sensor network tasks promises good compromise between performance, power and applicability. However, the design of an embedded sensor node microcontroller is not straightforward. It requires a holistic approach that considers all aspects of design and envisioned application scenario in order to define energy-efficient system architecture based on the optimal

selection of power saving strategy. A methodology to assist designer choose an efficient power saving strategy at early design phase is addressed by this work.

## **1.2. Objectives**

When designing a system, designer has to make a number of design decisions that define the quality of the final design. The design decisions made at the early stage of design process can greatly impact the power efficiency of design. In application-specific systems, power efficiency depends on target system architecture, specific application requirements and applied power saving strategy. For different applications, the power efficiency of the implemented power saving strategy might not always be optimal. An early exploration of the impact of particular low power implementation in given application scenario is required before making critical design decisions. The development of methodology to assist designer select appropriate power saving strategy for design at early design stage is in the focus of this work. The primary goal is to provide methodology that can be efficiently applied to the design of low power sensor node hardware systems.

In battery-powered systems, the battery life is determined by the effective power consumed over the time, i.e., the system energy. Therefore, proposed methodology must consider the energy efficiency of particular low power implementation. This requires development of analytical models for power estimation that address power in time domain, e.g., the models must consider the energy efficiency of specific low power implementation. The energy estimation models must include both static and dynamic energy contributions of single system components. Furthermore, the energy estimation has to address the energy overhead introduced by the implementation of advanced low power techniques.

The time-related data necessary for energy estimation are determined by the system activity defined by target application requirements. The development of methods for the extraction of system activity profiles is another goal of this work. Since many low power techniques apply at the block level, the profiling methodology has to provide the means for mapping of system activity to specific block activity. In addition, the proposed methodology must be suitable for use in typical WSN application scenarios.

The power saving strategy for design relies on the implementation of advanced low power techniques in the system. Therefore, proposed methodology has to be integrated in design flow that supports power-aware design implementation. The goal of this work is to enable the methodology integration to the design flow driven by industry standard tools and to define basic actions of the flow.

Another objective of the work is to provide low power cells for implementation of the advanced low power techniques. The power gating technique is seen to have the highest power saving potential in low-duty cycle sensor node applications. Therefore, it is required to design and characterize power switching cells and integrate them in the existing design kit. The power control logic is to be designed as well. The characterized power cells are to be used for power gating implementation and in the estimation of energy overhead required by the methodology application.

The final objective of the work is to validate the methodology by applying it in the design of a sensor node microcontroller system-on-chip. The chip is to be designed following established design flow, produced and tested.

### 1.3. Contributions

To answer the thesis objectives, a methodology is proposed that enables early exploration of the energy efficiency of alternative power saving strategies applied to the target system architecture in given application scenario. The methodology is applied to design an embedded sensor node microcontroller. The designed chip implements the advanced low power techniques and architectural solutions that enable high performance and low power operation.

The main contributions of this work are briefly described.

**Development of low power methodology and analysis of design energy efficiency.** A methodology is developed to estimate the energy efficiency of alternative power saving strategies applied to the system-on-chip design. The methodology employs the models for energy estimation based on the estimated dynamic and static power of partitioned system blocks and the extracted block activity in given application scenario. The activity profiling maps the extracted system activity to the activity of single blocks enabling both system and block-level energy exploration. The energy estimation models account for the impact of the energy overhead introduced by the implementation of low power techniques to the overall system energy. The detailed models for the estimation of the energy overhead introduced by power gating implementation have been developed. Finally, the application of activity profiling methodology in typical sensor node applications is demonstrated.

**Set up of a power-driven design flow.** A power-driven design flow that integrates the developed methodology has been setup. Implementation of the advanced low power techniques is supported by the flow. The established design flow employs the industry standard design tools and relies on the application of power intent file specification required by the tools. The basic actions in the flow are described in the light of the system power consideration.

**Design of power gating components.** To support the implementation of power-gated design, special power switching cells have been developed and characterized. The cells are designed for IHP 0.25  $\mu\text{m}$  CMOS technology and used as header-type power switches in a column-style based power-gating implementation. Additionally, the power control logic has been developed as well. The implementation of the designed power gating components is made a part of the developed implementation flow. The developed power gating cells have been used in the implementation of low power sensor node microcontroller.

**Design of low power sensor node microcontroller.** The application of the developed methodology is demonstrated through the design of an embedded sensor node microcontroller. The methodology is used to select the target power saving strategy for the microcontroller chip design. The design and implementation process for the chip have been presented in the work. The chip is designed for sensor node applications with strong security demands. It implements five switchable power domains, which can be individually controlled. The designed low power system architecture additionally includes asynchronous on-chip communication, power-driven hardware acceleration,

frequency islands and global clock gating. The sensor node microcontroller chip containing on-chip Flash memory and analogue-to-digital converter has been fabricated and successfully tested. The measurement results are presented and evaluated.

#### **1.4. Structure**

The thesis consists of seven chapters. Chapter 1 gives an introduction to the motivation and the objectives of the work and presents a short description of the main contributions. Chapter 2 presents current state-of-the-art in the existing techniques for low power design and recent advances in the design of sensor node hardware systems. Additionally, the existing power-aware design methodologies are presented as well as the work related to the estimation of the overhead introduced by implementation of the advanced low power techniques. The last section of Chapter 2 discusses the future trends in low power design making the guidelines to the future of sensor node hardware design. Chapter 3 presents the first contribution of this work, a low power methodology used for early estimation of the energy efficiency of specific low power system implementations. It also presents the development of energy estimation models including the modelling of the energy overhead introduced by the implementation of advanced low power techniques. Furthermore, the methodology for activity profiling is presented and its application to typical sensor network application is discussed. Chapter 4 describes the integration of presented methodology in the established power-driven design flow and describes the basic steps of the flow. The low power sensor node microcontroller and power-gating cells are described in Chapter 5. This chapter also illustrates use of the proposed methodology for selection of the target power saving strategy in design of a sensor node. Chapter 6 gives the implementation and measurements results for the designed chip. The results of chip measurements are discussed and analysed. Finally, Chapter 7 gives the summary of the conducted work and draws the conclusions.

## 2. State of the Art

### 2.1. Power Consumption

The demand for portable electronic devices that offer greater functionality and performance at lower costs and smaller sizes has increased rapidly. This market trend is driving the need for efficient System-on-Chip (SoC) designs where the power arises as the one of the biggest problems [6]. The microprocessor design has traditionally focused on dynamic power (active power) consumption as the limiting factor in system integration. As feature sizes shrink below 0.1 micron, static power (leakage power) is posing new low-power design challenges [7].

Historically, CMOS technology has dissipated much less power than earlier technologies such as transistor-transistor and emitter-coupled logic. In fact, when not switching, CMOS transistors consume negligible power. However, the power in CMOS is increased dramatically with increases of the device speed and chip area. For some designs and libraries, the leakage current exceeds the switching currents, thus becoming the primary source of power dissipation in CMOS, as shown in Figure 4.

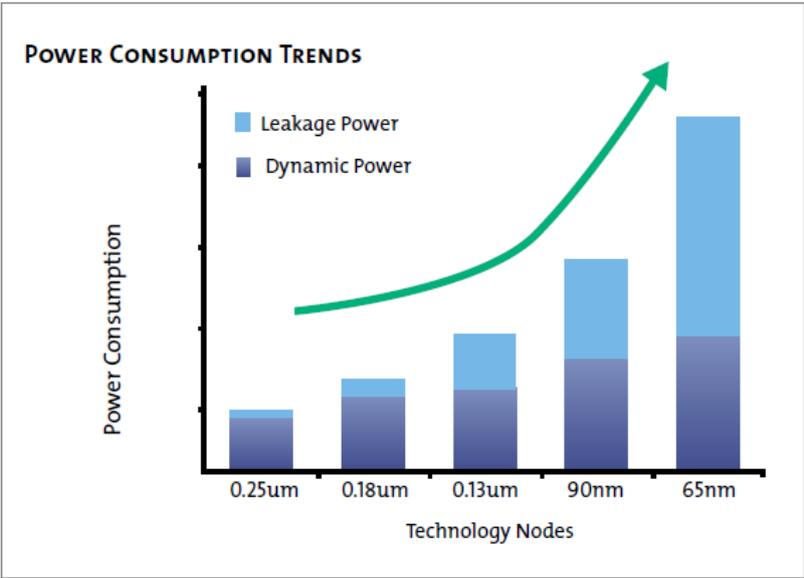


Figure 4. Power consumption trends for different technology nodes [3].

The ITRS predicts the dynamic power per device to decrease over the time. However, if it is assumed that the number of on-chip devices doubles every two years, total dynamic power will increase on per-chip basis. The packaging and cooling costs as well as the limited power capacity of batteries become unsustainable.

These problems are all expected to get worse as we move to the new technology nodes. The ITRS makes the following predictions (Table 2-1):

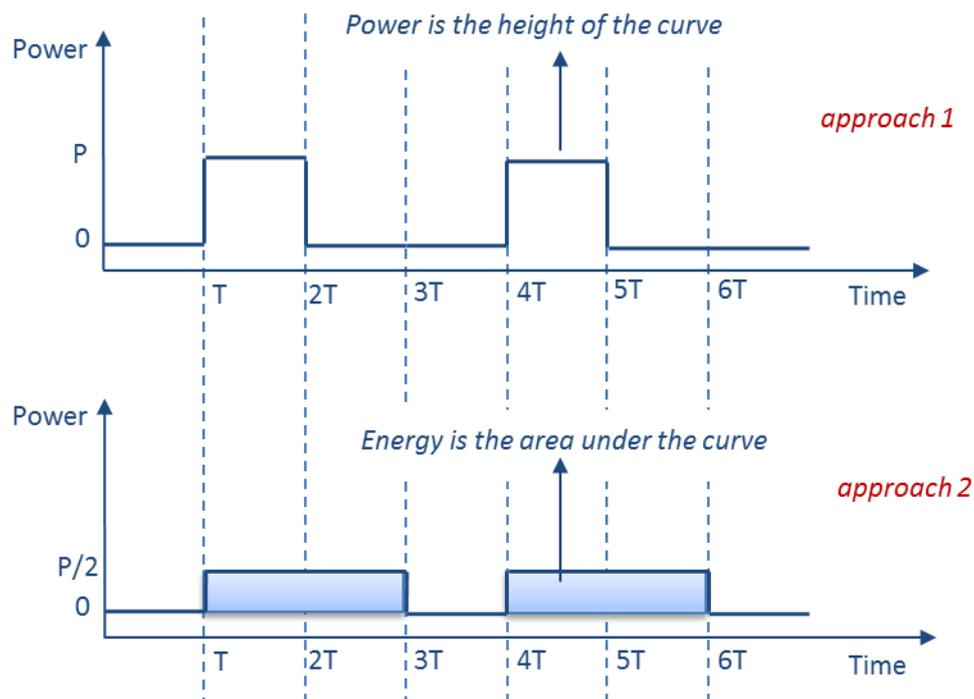
**Table 2-1.** Power increase projection for different technology process nodes [73].

Node	90nm	65nm	45nm
Dynamic Power per cm <sup>2</sup>	1X	1.4X	2X
Static Power per cm <sup>2</sup>	1X	2.5X	6.5X
Total Power per cm <sup>2</sup>	1X	2X	4X

It is a task of designers to try to reduce the growth in power below the predicted numbers. Therefore, advanced power saving techniques have been developed that target both static and dynamic power in system.

### 2.1.1. Power vs Energy

For the battery operated devices, the distinction between power and energy is critical. Power is the instantaneous power in the device, and energy is the integral of power over the time. The example in Figure 5 presents two approaches that have same impact on the battery life.



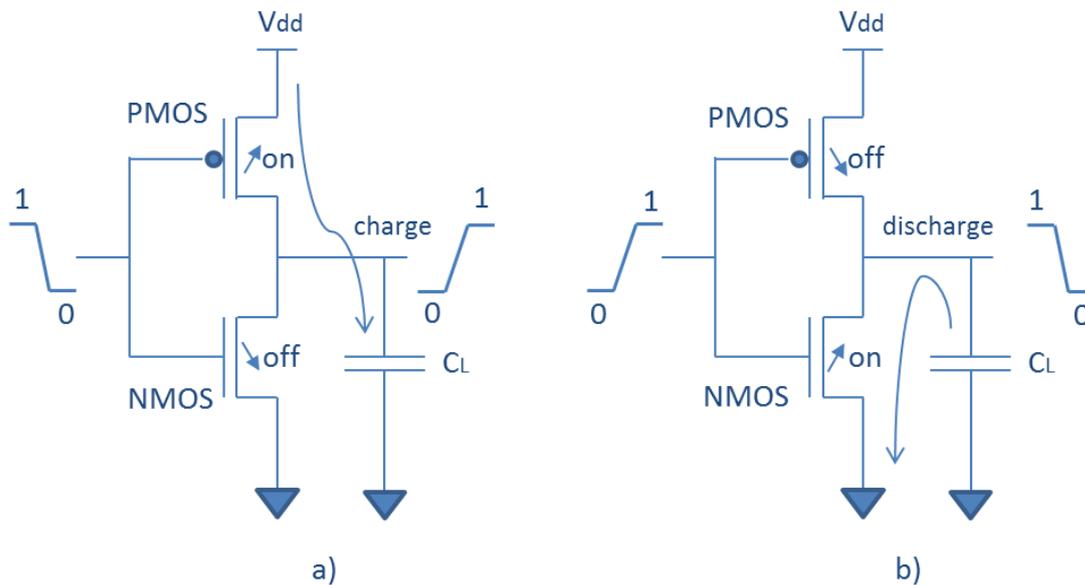
**Figure 5.** Power vs. Energy. Two approaches have different power but same energy.

In the first approach, a task takes time  $T$ , with average power consumption  $P$ , to complete. The consumed energy for the task completion is  $E_1 = P \times T$ . In the second approach, the average power for the task completion is  $P/2$ , but the task needs  $2T$  time to complete. The consumed energy for the task completion is in this case  $E_2 = (P/2) \times 2T = P \times T = E_1$ . It is clear that it's the energy that determines the battery life (the area under the curve in Figure 5).

### 2.1.2. Dynamic Power Loss

The power dissipation in CMOS consists of two components: dynamic and static power dissipation. Dynamic power ( $P_{dyn}$ ) is power dissipated when the logic states of gates are switching. It is associated with the active mode of operation and consists of two components, switching power and internal power. Switching power results from the charging and discharging of the external capacitive load on the output of a cell. Internal power is caused by short-circuit (crowbar) current flowing through the PMOS-NMOS stack during a transition.

The cause of switching power is illustrated in Figure 6. A transition from 0 to 1 on the output of the inverter charges the capacitive load of the output net through the PMOS transistor (Figure 6a). A transition from 1 to 0 discharges the same capacitive load through the NMOS transistor (Figure 6b).



**Figure 6.** Switching power in CMOS: a) 0-to-1 transition, b) 1-to-0 transition.

The energy per transition in a CMOS device is equal to the charge transferred from  $V_{dd}$  to ground during a transition. It is described as:

$$Energy/Transition = C_L \cdot V_{dd}^2 \quad (2.1)$$

Where  $C_L$  is the load capacitance and  $V_{dd}$  is the supply voltage. The switching power is defined by the frequency of transitions and it is described as:

$$P_{sw} = Energy/Transition \cdot f = C_L \cdot V_{dd}^2 \cdot \alpha \cdot f_{clock} \quad (2.2)$$

Where  $f$  is the frequency of transitions,  $\alpha$  is the probability of an output transition, and  $f_{clock}$  is the clock frequency. If effective capacitance  $C_{eff}$  is defined as:

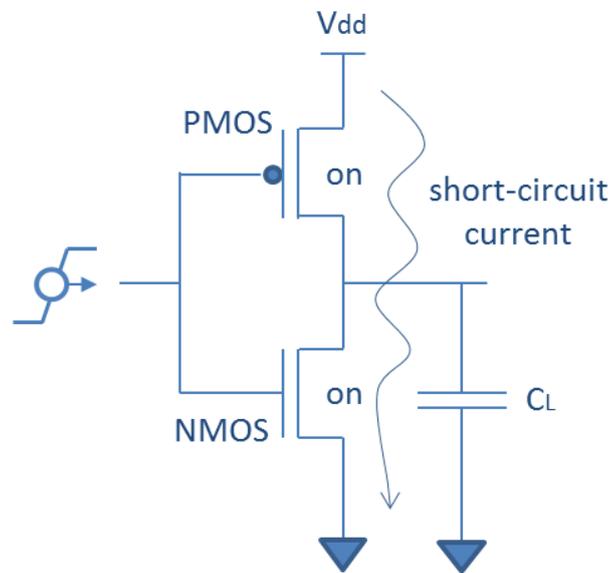
$$C_{eff} = C_L \cdot \alpha \quad (2.3)$$

Then the switching power can be described with more familiar expression:

$$P_{sw} = C_{eff} \cdot V_{dd}^2 \cdot f_{clock} \quad (2.4)$$

It has to be noticed that the switching power is not a function of the transistor size, but it rather depends on the capacitive load at the transistor output.

In addition to switching power, internal power also contributes to dynamic power. Internal power is consumed during the short period of time when the input signal is at intermediate voltage level, at which both the PMOS and NMOS transistors are conducting. This condition results in a nearly short-circuit conductive path from  $V_{dd}$  to ground, as illustrated in Figure 7.



**Figure 7.** Internal power.

The internal power can be described as:

$$P_{sc} = t_{sc} \cdot V_{dd} \cdot I_{peak} \cdot f_{clock} \quad (2.5)$$

Where  $t_{sc}$  is the time duration of the short circuit current and  $I_{peak}$  is the total internal switching current (short circuit current plus the current required to charge the internal capacitance).

Thus, dynamic power becomes:

$$P_{dyn} = P_{sw} + P_{sc} = C_{eff} \cdot V_{dd}^2 \cdot f_{clock} + t_{sc} \cdot V_{dd} \cdot I_{peak} \cdot f_{clock} \quad (2.6)$$

As long as the ramp time of an input signal is kept short, the short circuit current occurs for only a short time during each transition. In that case, the total dynamic power is dominated by the switching power. For this reason, the formula for dynamic power can be simplified as:

$$P_{dyn} = C_{eff} \cdot V_{dd}^2 \cdot f_{clock} \quad (2.7)$$

There is a number of low power techniques used to reduce dynamic power consumption in a circuit. Most of them are trying to reduce dynamic power by focusing on voltage and frequency components of the equation. Some other techniques are trying to reduce the data-dependant switching activity. The techniques for power reduction are discussed later.

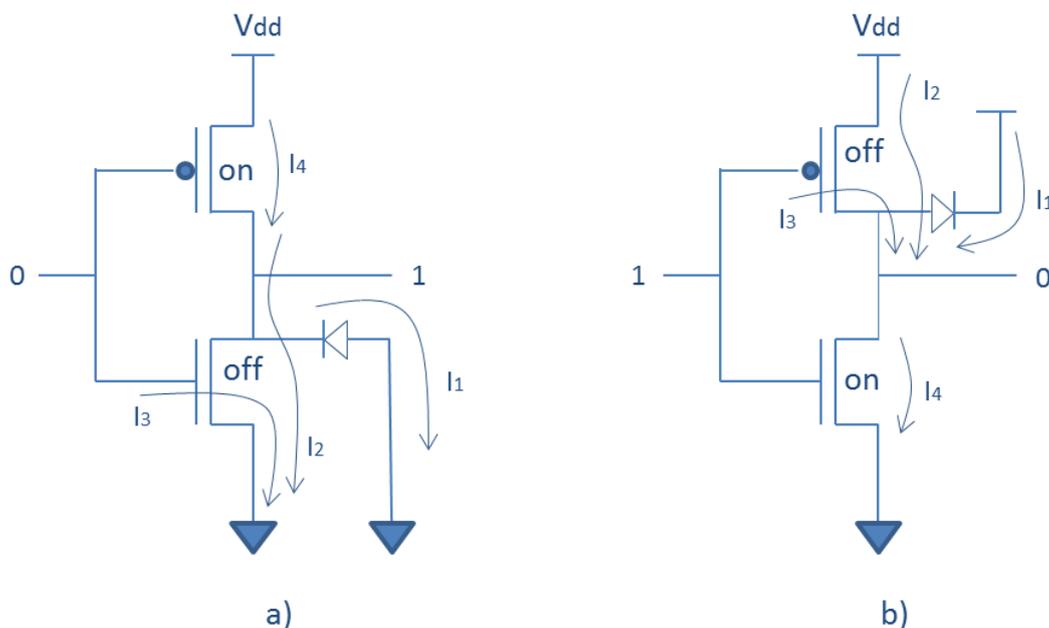
### 2.1.3. Static Power Loss

Static power (also referred to as leakage power) results from current that leaks through transistors even when they are turned off. Leakage current was negligible in earlier CMOS technologies. However, with shrinking device geometries and reduced threshold voltages, leakage power is becoming increasingly significant, sometimes approaching the levels of dynamic power dissipation.

The main causes of leakage power in CMOS are

1. reverse-bias p-n junction diode leakage ( $I_1$ ),
2. subthreshold leakage ( $I_2$ ),
3. gate leakage through the oxide ( $I_3$ ), and
4. gate induced drain leakage ( $I_4$ ).

The schematic representation of leakage paths in CMOS inverter is illustrated in Figure 8.



**Figure 8.** Static leakage currents: a) gate at logic level 0, b) gate at logic level 1.

*Leakage at reverse-biased p-n junctions* (diode leakage) has always existed in CMOS circuits. This is the leakage from the n-type drain of NMOS transistor to the grounded p-type substrate, and from the n-well (held at  $V_{dd}$ ) to the p-type drain of PMOS transistor. This leakage is relatively small.

*Subthreshold leakage* is the small source-to-drain current that flows even when the transistor is held in the “off” state. As technology scales, the power supply voltage is getting lower and closer to the threshold voltage. This makes subthreshold leakage current increasing exponentially. The subthreshold leakage current can be approximated by [135]:

$$I_{sub} = \mu C_{ox} V_{\theta}^2 \frac{W}{L} \cdot e^{\frac{V_{GS} - V_{th}}{nV_{\theta}}} \quad (2.8)$$

Where  $V_{\theta}$  is the thermal voltage  $kT/q$  (25.9 mV at room temperature), and  $W$  and  $L$  are the dimensions of the transistor. Parameter  $n$  is the function of the fabrication process. The last equation can be approximated as the function of supply voltage:

$$I_{sub} = K_1 W e^{\frac{-V_{th}}{nV_{\theta}}} \left(1 - e^{\frac{-V_{dd}}{V_{\theta}}}\right) \quad (2.9)$$

Where  $K_1$  is experimentally derived. The last equation suggests that the increase of threshold voltage  $V_{th}$  or the reduction of  $V_{dd}$  will reduce the leakage.

*Gate leakage* is the tunnelling current through the oxide. It can be approximated with the following Equation [135]:

$$I_{ox} = K_2 W \left(\frac{V_{dd}}{T_{ox}}\right)^2 e^{\frac{\alpha T_{ox}}{V_{dd}}} \quad (2.10)$$

Where  $K_2$  and  $\alpha$  are experimentally derived, and  $T_{ox}$  is the oxide thickness. The oxide leakage strongly depends on  $T_{ox}$  and  $V_{dd}$ . With the introduction of high-k dielectrics in the mainstream production, this source of leakage is significantly reduced.

The *gate induced drain leakage* (GIDL) is caused by high field effect in the drain junction of MOS transistors. For an NMOS transistor with grounded gate and drain potential at  $V_{dd}$ , significant band bending in the drain allows electron-hole pair generation through avalanche multiplication and band-to-band tunnelling. A deep depletion condition is created since the holes are rapidly swept out to the substrate. At the same time, electrons are collected by the drain, resulting in GIDL current.

Leakage currents occur whenever power is applied to the transistor, irrespective of the clock speed or switching activity. Leakage cannot be reduced by slowing or stopping the clock. However, it can be reduced or eliminated by lowering the supply voltage or by switching off the power to the transistors entirely. Increasing the transistor threshold voltage or using stacked transistor configuration can reduce leakage as well. These and other methods to reduce leakage are discussed in the following sections.

## 2.2. Standard Low Power Techniques

Standard low power techniques are established for mature technologies where dynamic power is considered to be the main source of power consumption in device. The early techniques are focused on the reduction of supply voltage, switching activity and load capacitance. They can be classified as supply voltage scaling, multi-threshold voltage, clock gating, operand isolation and gate level optimization.

### 2.2.1. Supply Voltage Scaling

Supply voltage scaling (also called multi- $V_{dd}$ ) benefits from the reduction of supply voltage and is considered to be very effective because of the quadratic dependence of  $P_{dyn}$  from  $V_{dd}$ . However, in order to maintain performance, early voltage scaling techniques required massive parallelization and pipelining resulting in significant area overhead [8]. A variation of this technique, named clustered voltage scaling, proposed the usage of dual supply voltages (standard and reduced) to be applied to specific circuit clusters [9]. The standard supply voltage is applied to the circuit cluster containing logic on critical paths and the reduced supply voltage is applied to the cluster with logic on non-critical paths. The special latch cells are implemented to shift the voltage levels between two clusters. The reported power savings are 10-20%. The multi-voltage techniques are generally difficult to implement since they have large impact on design complexity.

In a multi- $V_{dd}$  implementation different blocks operate at different supply voltages. Running a block at lower voltage reduces its power consumption, but at the expense of speed. For example, in a CPU-based system, processor and cache memories require high speed operation, but peripherals can operate at lower speed under the technology limit. Therefore, it is possible to apply different voltage supplies to those blocks (Figure 9).

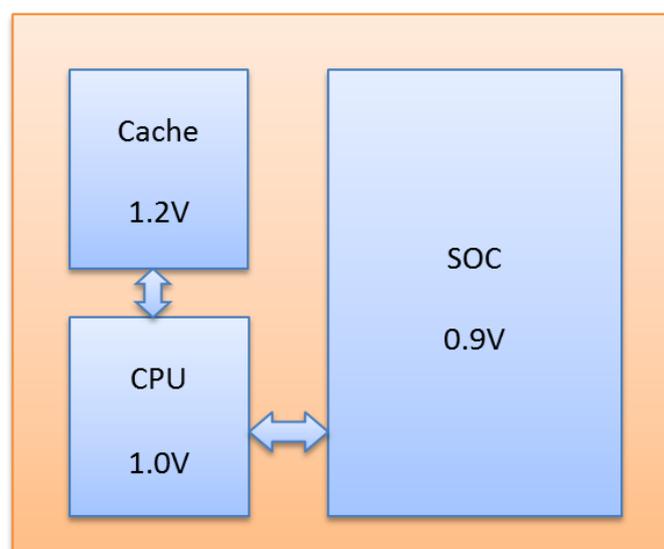


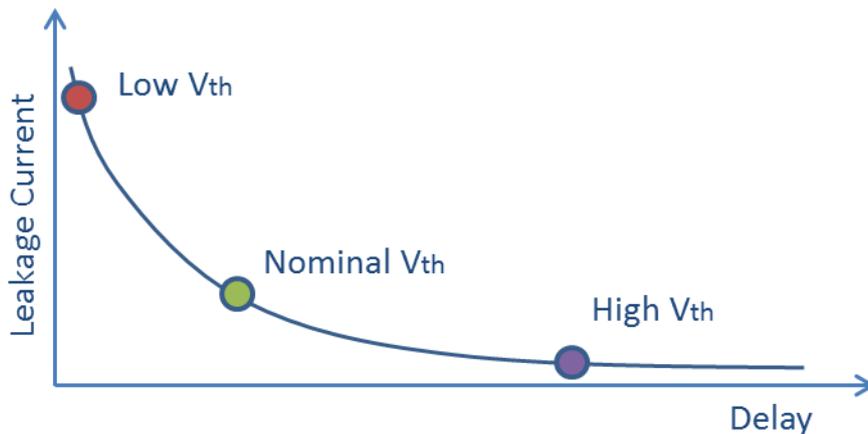
Figure 9. Multi-voltage architecture.

The implementation of supply voltage scaling makes significant impact on system architecture and design complexity. The implementation of the multi- $V_{dd}$  technique requires voltage shifters at signals crossing between different voltage levels. Additionally, the IO pads that can provide different supply voltages are required as well. There are also some issues related to power grid design, timing closure and distribution of clock.

### 2.2.2. Multi-Threshold Voltage

The multi-threshold voltage technique (also called multi- $V_{th}$ ) proposed the usage of both high and low threshold voltage transistors in a single chip [10]. The power reduction is achieved by replacing power hungry low- $V_{th}$  transistors with high- $V_{th}$  transistors, wherever it is possible. This technique is primarily proposed to reduce static power dissipation by power shut-down, but even without switching the power off, a significant reduction in total power can be achieved. The results presented in [11] show the static power savings of up to 80% and the dynamic power savings between 20% and 50%, depending on the switching activity of benchmark circuits. However, dynamic power savings are in average much lower and usually not more than 5%.

Many vendors provide libraries with high, low and standard- $V_{th}$  cells. The implementation tools can take advantage of those libraries to optimize for timing and power simultaneously. The reduction in threshold voltage improves the leakage, but it impacts the delay in device (Figure 10).

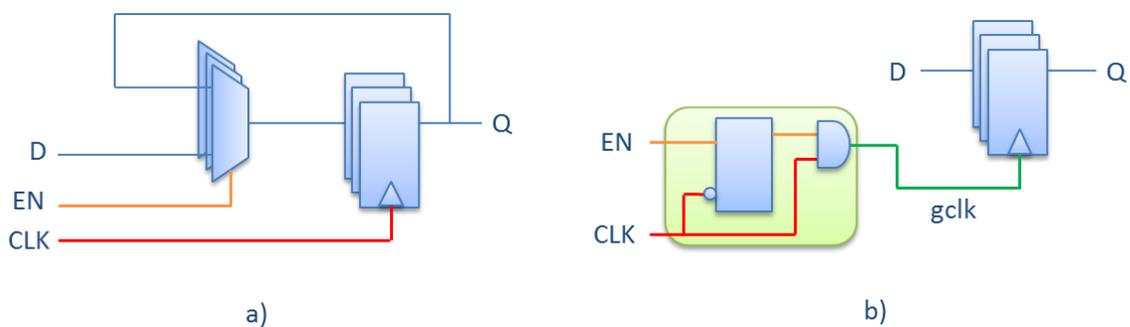


**Figure 10.**  $V_{th}$ -dependency of transistor leakage current and transistor delay.

A standard multi- $V_{th}$  implementation flow usually starts with the logic synthesis in a single library. It is either a high- $V_{th}$  library when designing for low power systems or a low- $V_{th}$  library when designing for high performance systems. Then, the second synthesis run is applied to replace high- $V_{th}$  components with low- $V_{th}$  in the first case, or to replace low- $V_{th}$  components with high- $V_{th}$  in the second case.

### 2.2.3. Clock Gating

Clock gating is the most popular technique to reduce dynamic power dissipation in synchronous circuits [12], [13]. It reduces power by disabling the switching of clock tree net in the parts of circuit that are at particular time inactive. Normally, the inputs of inactive blocks do not change. However, power is still dissipated in the clock tree and flip-flops due to the switching of clock net (Figure 11a). Since the switching activity of those elements is 100%, their power consumption may reach up to 30% of the overall dynamic power dissipation [14]. Consequently, applying clock gating may reduce dynamic power dissipation significantly (Figure 11b). In the example from [15], the clock gating implementation reduced the power of a 200k-gate design for 72%. Clock gating is illustrated in Figure 11.



**Figure 11.** Clock gating: a) no clock gating implemented, b) clock gating implemented.

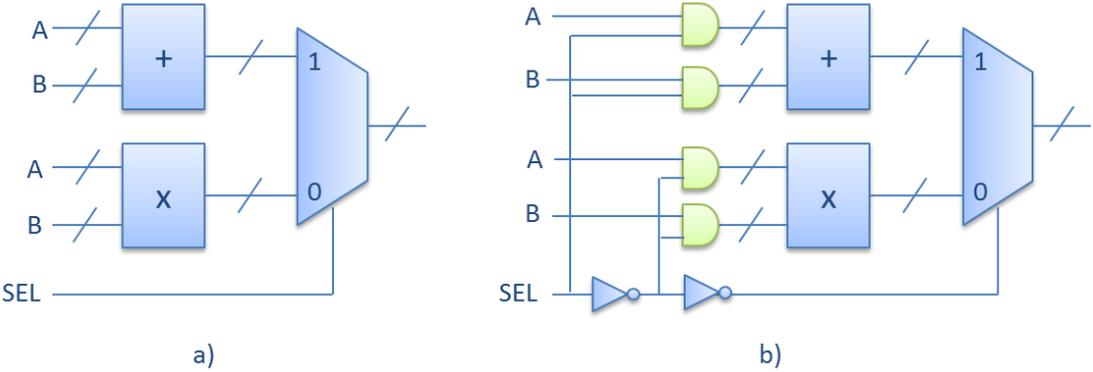
A clock gating element could be built as a simple AND gate or as a latch-based structure (Figure 11b). The simple AND gate is prone to glitch generation. Therefore, the latch-based structure, which prevents the glitching, is preferred solution. The modern electronic design automation (EDA) tools support automatic insertion of clock gating that goes deep into design hierarchy. However, the insertion of global clock gating early in the clock tree is more effective, since it disconnects larger portion of clock tree with the minimal area overhead.

### 2.2.4. Operand Isolation

Similar to the clock gating technique, it is also possible to reduce the switching activity of inactive datapath blocks controlled by an enable signal. This technique is known as operand isolation. The experimental results using the operand isolation technique presented in [16] and [17] show power savings of up to 30% at relatively small cost of area. However, large power savings due to operand isolation are possible only in datapath architectures. For predominantly clocked systems, the savings are not larger than 5%.

The idea behind operand isolation is to set the inputs of datapath block to constant value when enable signal is inactive. This prevents from switching in datapath and saves power. Figure 12 shows a simplified ALU block consisting of a multiplier and an adder. Signal SEL selects the operation to be performed on the input signals A and B. The result of the operation is propagated to the ALU output.

When no operand isolation is applied (Figure 12a), both multiplier and adder will respond to the change in input signals. This will cause undesired switching in non-selected block and increase the total power consumption. With the operand isolation applied (Figure 12b), only the selected block will propagate the changes in input signals, reducing the total switching power. Modern EDA tools apply this technique automatically.



**Figure 12.** Example of operand isolation applied to simple ALU block: a) no operand isolation applied, b) operand isolation applied.

**2.2.5. Gate-Level Optimization**

There are a number of techniques to optimize power at gate level. Those techniques are usually applied by modern synthesis tools to improve the power of synthesized design [18]. Standard gate-level methods to reduce power are logic restructuring and pin swapping. Logic restructuring is based on the principle of composition, where sets of cells are composed into a single complex cell making the high activity net internal to that cell and thus reducing power [19]. The pin swapping technique exploits the functional symmetries in the gates in order to perform rewiring of gate pins without affecting the functionality. In that way, the high activity nets can be rewired to the pins with lower load capacitance resulting in reduced power dissipation [20]. Other examples of gate level power optimization include cell sizing and buffer insertion. The both methods try to increase the transition time of a net that in certain situations leads to considerable power savings. Gate-level power-saving techniques may improve dynamic power for 10-15% [21].

**2.2.6. Overview**

Standard low power techniques are mainly used to reduce dynamic power in CMOS. The largest savings of dynamic power are accomplished by clock gating and multi- $V_{dd}$ . Clock gating is easy to implement and therefore has the largest popularity among all standard low power techniques. Multi- $V_{th}$ , operand isolation and gate-level techniques make low implementation effort as well. At the other hand, multi- $V_{dd}$  makes significant impact on design and architecture. This is mainly due to the increased complexity of power supply grid and need for signal level shifting on block boundaries. The reduction of supply voltage in multi- $V_{dd}$  introduces some timing penalty in design, but it also reduces

leakage power. Leakage power is reduced by multi- $V_{th}$  as well. The leakage reduction results from the increased threshold voltage in high- $V_{th}$  cells. Apart from multi- $V_{dd}$ , all standard low power techniques have minor impact on timing, area and verification, and their implementation is automatized in most modern EDA tools. Table 2-2 summarizes the costs and benefits of standard low power techniques.

**Table 2-2.** Overview of standard low power techniques.

Technique	Dynamic Power Savings	Leakage Power Savings	Timing Penalty	Area Penalty	Impact: Architecture	Impact: Design	Impact: Verification	Impact: Place &Route
Multi- $V_{th}$	low (<5%)	2-3x	little	little	low	low	none	low
Clock Gating	medium (<30%)	none	little	little	low	low	none	low
Multi- $V_{dd}$	large (<50%)	2x	some	little*	high	medium	low	medium
Operand Isolation	low (<5%)	none	little	little	low	low	none	low
Gate-Level Techniques	low (<15%)	none	little	little	none	none	none	none

\* supply voltages are driven externally

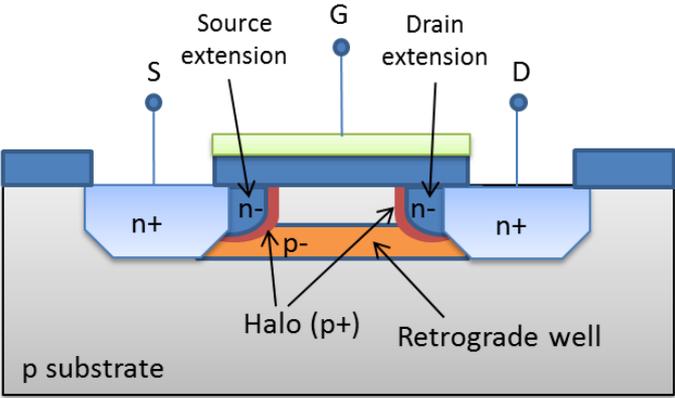
### 2.3. Advanced Low Power Techniques

Advanced low power techniques are developed to deal with the increasing contribution of leakage currents to the total power consumption of deep-submicron CMOS. Over the years, the supply voltage has been scaling down with the technology keeping the power consumption under control. In order to maintain high driving currents and to improve performance, the threshold voltage has been scaling down as well. The reduction in threshold voltage contributed to the increase in subthreshold leakage current [22]. The subthreshold current is the weak inversion conduction current and it occurs when the gate voltage is below  $V_{th}$ . The increase in  $V_{th}$  or the lowering of the supply voltage results in the reduction of subthreshold leakage. The techniques to reduce leakage can be applied at both process and circuit level. The process-level techniques apply the changes in process dimensions (oxide thickness, length, etc.) and in the doping profiles of transistors. The circuit-level techniques try to control threshold voltage and leakage currents by controlling the voltages of different device terminals (drain, source, gate and body (substrate)). The most important circuit-level advanced low power techniques are multi-voltage design, power gating, body-biasing and transistor-stacking.

#### 2.3.1. Process Techniques

The most common process-level techniques applied to standard bulk-CMOS are retrograde doping and halo doping [23], [24]. The retrograde channel doping improves short-channel effects and increases surface channel mobility by creating a low surface channel concentration followed by a highly doped subsurface region. The existence of subthreshold region reduces the threshold voltage since it acts as a punchthrough barrier. At the other hand, low surface concentration improves mobility maintaining performance. The halo doping introduces highly-doped p-type-regions at the

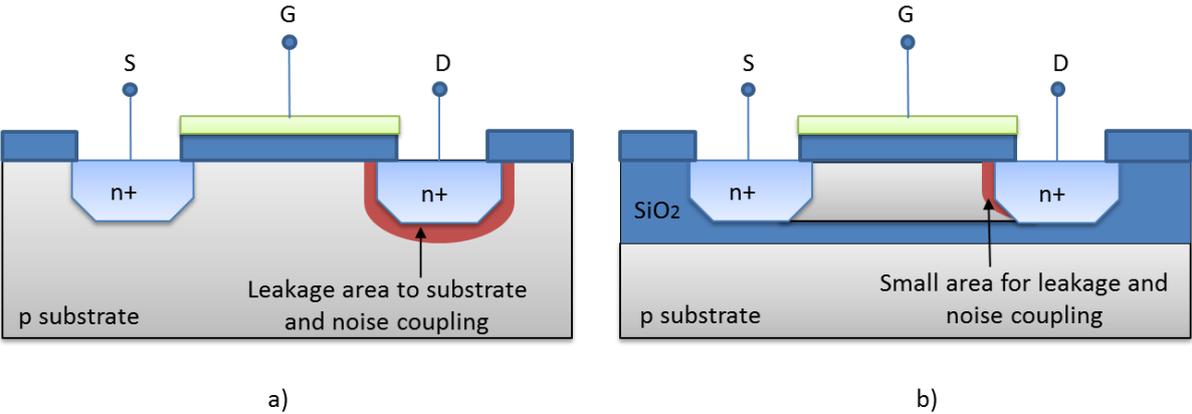
edges of the channel reducing the width of the depletion regions in the drain and source substrates. This reduces the threshold-voltage dependence on channel length improving subthreshold leakage. This is illustrated in Figure 13.



**Figure 13.** Different aspects of well engineering.

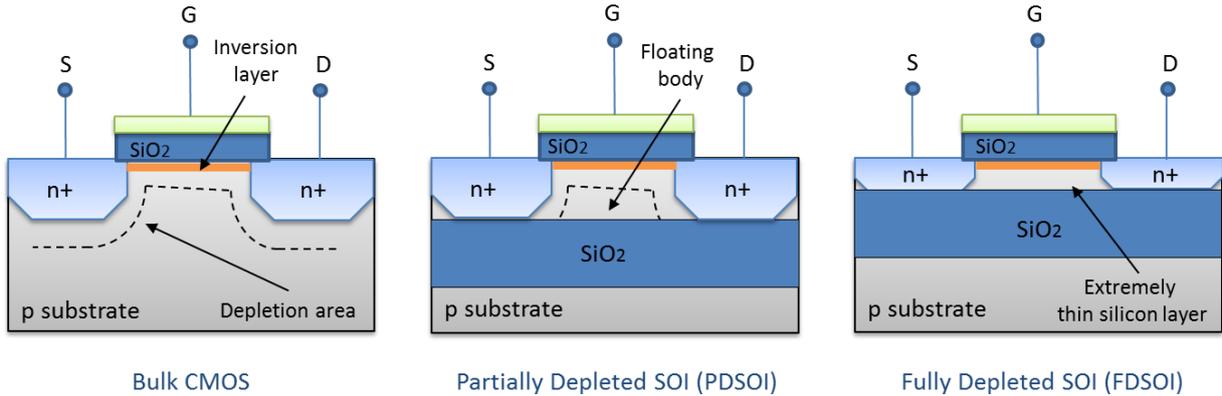
In order to reduce power and improve performance, novel technologies have been proposed as an extension or substitute to standard planar bulk-CMOS. The most promising are silicon-on-insulator and non-planar technologies such as multi-gate SOI and FinFET.

In SOI, the insulating substrate is used instead of silicon to reduce parasitic effects and improve performance and power [25], [26], [27]. Compared to bulk silicon, SOI provides up to 90% lower junction capacitance, near ideal subthreshold swing, reduced device cross-talk, lower junction leakage, no latch-up, increased radiation hardness, and full dielectric isolation of the transistor [27]. The silicon-on-insulator methodology is illustrated in Figure 14.



**Figure 14.** Bulk-CMOS (a) vs SOI (b) technology.

The idea of silicon on insulator is very old. However, due to the restrictions in material and device quality and due to the continuing advances in bulk-CMOS, SOI didn't enter the mainstream production until the late 1990s. The leading research in SOI technology is performed by IBM Research Group. After several demonstrations performed over the years, IBM released their first SOI product in 1998, a 64-bit PowerPC microprocessor. The IBM research on SOI continued with 0.18  $\mu\text{m}$ , 0.13  $\mu\text{m}$  and 0.1  $\mu\text{m}$  processes as well as for SiGe devices [28]. There are two types of SOI devices, partially depleted (PDSOI) and fully depleted (FDSOI). In PDSOI, the body area of transistor is only partially depleted. In FDSOI, the silicon area over the insulator is very thin, so the depleted area fills full width of the body (Figure 15).

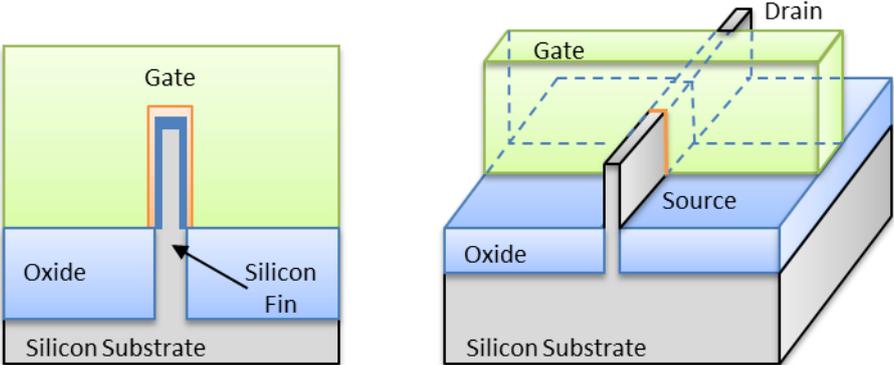


**Figure 15.** Bulk-CMOS vs PDSOI vs FDSOI.

The major drawback of SOI are body floating effects that cause shifts in threshold voltage, subthreshold swing and kink effect [29], [30]. At sub-0.25  $\mu\text{m}$  technology nodes, the body and short-channel effects in PDSOI start to dominate increasing the leakage and making the power consumption high. FDSOI reduces those effects, but it is more difficult to fabricate. PDSOI has been successfully used for high-performance microprocessor design in technologies up to 22 nm. The recent advances in FDSOI made it preferable SOI solution at lower scale technology nodes. Compared to bulk CMOS, SOI clearly has its advantages, but it is more expensive to fabricate. An early experiment comparing DSP designs in 0.6  $\mu\text{m}$  bulk CMOS and 0.5  $\mu\text{m}$  SOI reported improvement in performance of 51% and reduction in power of 35% [31]. With the later SOI technologies, the gain in performance of SOI over bulk-CMOS is estimated to 25-30%. For a given CMOS generation, SOI outperforms bulk-CMOS making extra headroom for low-power implementation. In [32], having reduced supply voltages and using PDSOI process, the reported power gain for a high-frequency mixed signal design was 2-3x.

The multiple-gate MOSFET devices present the evolution from classical planar, single-gate devices into three-dimensional, multiple-gate (two-, three, all-around-, etc.) devices [33]. The multiple-gate devices have better control of channel allowing better suppression of short-channel effects. Also the drive current of multi-gate devices is better enabling the increase in performance. A special case of multi-gate device, called FinFET, is presented in [34]. The heart of FinFET is a thin silicon fin, wrapped around with a conducting channel. The fin-shaped silicon structure serves as the

body of MOSFET device. FinFETs have self-aligned gates that help reducing parasitic effects of typical double-gate devices suffering from the gate alignment problem. FinFETs can be produced either in SOI or in bulk substrate employing a planar-like technology process. SOI FinFETs show better performance due to the dielectric isolation compared to the junction isolation of bulk FinFET [35]. However, the bulk FinFET is cheaper to fabricate. A tri-gate bulk FinFET structure used in new generation Intel microprocessors is illustrated in Figure 16.



**Figure 16.** Tri-gate FinFET.

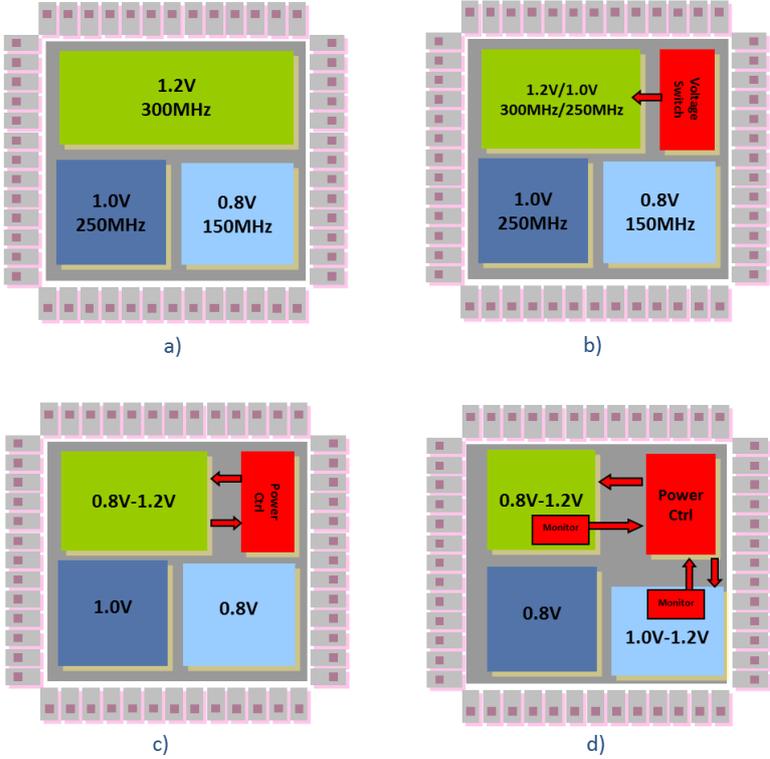
In tri-gate FinFET, the gate electrodes control the silicon fin from three sides providing improved sub-threshold slope, i.e., lower power in subthreshold region. At the other hand, the inversion layer area is increased enabling high drive current, i.e., better performance. According to Intel, the process costs for FinFET are only 2-3% higher than for bulk-CMOS (vs. ~10% for FDSOI).

**2.3.2. Multi-Voltage Design**

Advanced multi-voltage design techniques extend the idea of supply voltage scaling by introducing the concept of voltage islands [36]. Voltage islands, also called power islands or power domains, are blocks of logic on chip that have their separate supply voltage and clock frequency. Having different power regions on chip enables the application of complex power saving strategies that can manipulate power and frequency of each power domain independently.

The simplest case of multi-voltage design is static voltage scaling (SVS). In static voltage scaling, different power islands run at different fixed supply voltages [37]. In some cases, SVS can be applied to multi- $V_{th}$  design [38]. SVS has usually fast logic blocks such as processor or memories supplied by high voltage and slow logic blocks such as peripherals supplied by low voltage. If voltage blocks are running at different frequencies, then the approach is called static voltage and frequency scaling (SVFS). The extension of static voltage scaling, called multi voltage scaling (MVS), applies multiple fixed voltage levels to different power domains. If multiple voltage levels are associated with multiple operating frequencies, then the approach is called multi voltage and frequency scaling (MVFS).

Dynamic voltage scaling (DVS) technique is used to improve power of high performance logic. DVS takes advantage of variable processor workload, where computationally intensive processing tasks are followed by idle periods. This enables supply voltage and operating frequency to be scaled down simultaneously to the point where the task execution can be performed within available time margins. The DVS technique is often referred as dynamic voltage and frequency scaling (DVFS), and it is commonly used technique for power reduction in high performance processors. The usage of DVFS is first proposed in [39]. The potential of voltage scaling is discussed in [8] and some of the first algorithms for DVFS are presented in [40]. In [41], authors investigated practical and theoretical limits of dynamic voltage scaling. The results show that the theoretical minimum of energy is achieved in subthreshold region. However, the impact of process variation in subthreshold regime [42] is not considered in the work. Many implementations of DVFS have been reported over the years achieving reported power savings from 40-80% [43], [44], [45]. As technology scales, supply voltage margin for DVFS implementation reduces. Therewith, the effectiveness of DVFS on power saving diminishes [46]. In adaptive voltage and frequency scaling (AVFS), the control of frequency/voltage levels is provided by performance monitors implemented in hardware. The performance of silicon is measured based on process and temperature variation and the information is fed back to power controller that precisely adjusts the levels of voltage and frequency according to the needs. An example of AVFS applied to self-timed circuitry is presented in [50]. Other examples of AVFS presented in [51] and [52] report 20-40% power improvement over DVS. The multi-voltage design techniques are illustrated in Figure 17.

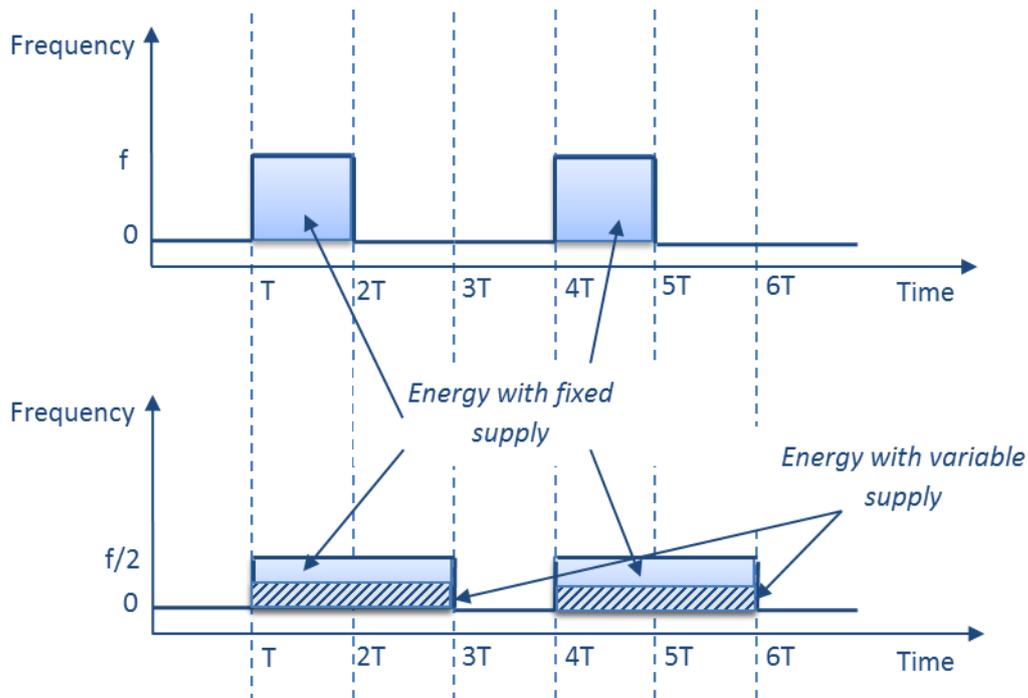


**Figure 17.** Multi-voltage design techniques: a) static voltage scaling, b) multi voltage scaling, c) dynamic frequency and voltage scaling, and d) adaptive frequency and voltage scaling.

The idea behind DVFS technique is to take advantage of variable workload in CPU task processing. Since CPU tasks are often followed by idle periods, it leaves the space to slow down the task execution speed without affecting the system performance. The task execution speed is adjusted by changing the clock frequency. Generally, the reduction of clock frequency reduces dynamic power that is given by:

$$P_{dyn} = C_{eff} \cdot V_{dd}^2 \cdot f_{clock} \quad (2.11)$$

However, slower clock speed implies longer task execution time giving no effective improvement in the energy needed for task completion. For example, if the frequency is reduced by the factor of 2, the time needed to complete a task is increased twice. The consumed energy remains in the best case the same. Therefore, dynamic frequency scaling makes sense only if it allows for simultaneous scaling of supply voltage with frequency. Thus, dynamic power improves quadratically. If the frequency and voltage are scaled down by factor of 2, the total energy required for task completion becomes only a quarter of the equivalent energy when no variable supply is applied (Figure 18).



**Figure 18.** Example of frequency and voltage scaling. The frequency scaling with fixed supply voltage does not save the energy (blue). In the case of variable supply voltage, the energy consumption is reduced (dashed).

In reality, there is an operation region where the frequency increases monotonically over voltage, with a maximum voltage specified for the specific technology and the minimum voltage under which the circuit does not operate reliably. This has to be considered when applying DVFS. If this region is too short, the efficiency of DVFS is reduced. If it's too large, it may be difficult to design voltage shifters and efficient power logic that can supply large blocks with sufficient current. It is shown that for the process geometries under 90 nm, DVFS does not benefit in energy reduction.

However, DVFS is still being used in high performance processors to balance the workload and reduce heat dissipation. The best power efficiency of DVFS can be achieved in the technology nodes that offer significant voltage headroom, like 180 nm and 130 nm. Furthermore, the reduction of supply voltage reduces the leakage to certain extent. To reduce static leakage during idle periods more aggressively, other techniques have to be applied.

A number of design issues have been related to the implementation of DVFS [63]. In physical layer, the signal levels between two voltage domains have to be shifted to match the voltage levels [64], [65]. At architecture level, power controller must be designed to take care of the voltage transition times in order to prevent malfunction. During design phase, in order to perform accurate timing closure on design, the libraries characterized for different delay corners are required. Also, choosing optimal voltage/frequency pairs is a challenging task [66].

The most important challenges associated with the implementation of DVFS are:

1) Voltage regulators - The supply voltage regulation in voltage scaled circuits is traditionally performed off-chip. Off-chip regulators have good efficiency, but they take significant space on board and have slow transition times that impact the DVFS implementation. The advantage of on-chip regulators over conventional off-chip implementation on a 4-core multiprocessor is presented in [53] showing improvement of 21% in energy savings. Two widely used voltage regulator topologies in multi-voltage designs are switched and linear regulators. Linear regulators are easier to integrate, have smaller size, better noise suppression and fast transient response [54]. However, the conversion efficiency of linear regulators is proportional to the difference between output and input voltage making them preferable only for low voltage swing implementations. Low-dropout regulator (LDO) is the most popular on-chip linear regulator topology [55], [56], [57]. It offers good performance and can be completely integrated without need for external passive components. Switched regulators, at the other hand, can manage high efficiency across wide range of output voltages. There are two types of commonly used switched regulator topologies: buck and switched-capacitor (SC) converters. Buck converters can provide wide range of output voltages, but they require a large, high quality inductor, which is difficult to integrate on chip [58]. In contrast, SC converters use flying capacitors to nominally divide high input voltage by predetermined integer ratios [59]. A comprehensive overview of integrated switched regulators is given in [60]. In order to improve conversion efficiency, a hybrid solution combining buck and SC regulator is presented in [61]. A combination of linear and SC regulator is given in [62].

2) Task scheduling - The implementation of DVFS requires an efficient algorithm for task scheduling that is usually implemented in software as part of the operating system [47], [48], [49]. It is a challenging task making workload prediction and performing safe and efficient task scheduling. Usually, task scheduling is performed in software, where the scheduling algorithm is incorporated in a power-aware operating system. Thus, the system software is capable to predict and schedule the tasks in order to balance the workload. It is also possible to perform the workload prediction locally, where the hardware logic is used for task scheduling. Examples are the filters that monitor tasks collected in a FIFO scheduler, where each task is bounded to some weighted coefficient. By performing some averaging function over the task collection, it is possible to make prediction of the expected workload. The equivalent methodology is used in DVFS for task prediction.

3) Transition time - The voltage regulators need some time to complete voltage transitions. This has to be considered when predicting task scheduling. During transition, the power control unit must keep a block isolated (inactive) until its supply voltage becomes stable.

4) Voltage shifters - All signals between two power domains with different voltage supply must be shifted to match signal levels. Therefore, low-to-high and high-to-low voltage shifters have to be designed and implemented on the interfaces between power domains.

5) Timing/Voltage Values - It is a key decision to determine the frequency/voltage in DVFS design. It is possible to decide between discrete values or to change the values continually. This affects the decision on the voltage regulator architecture.

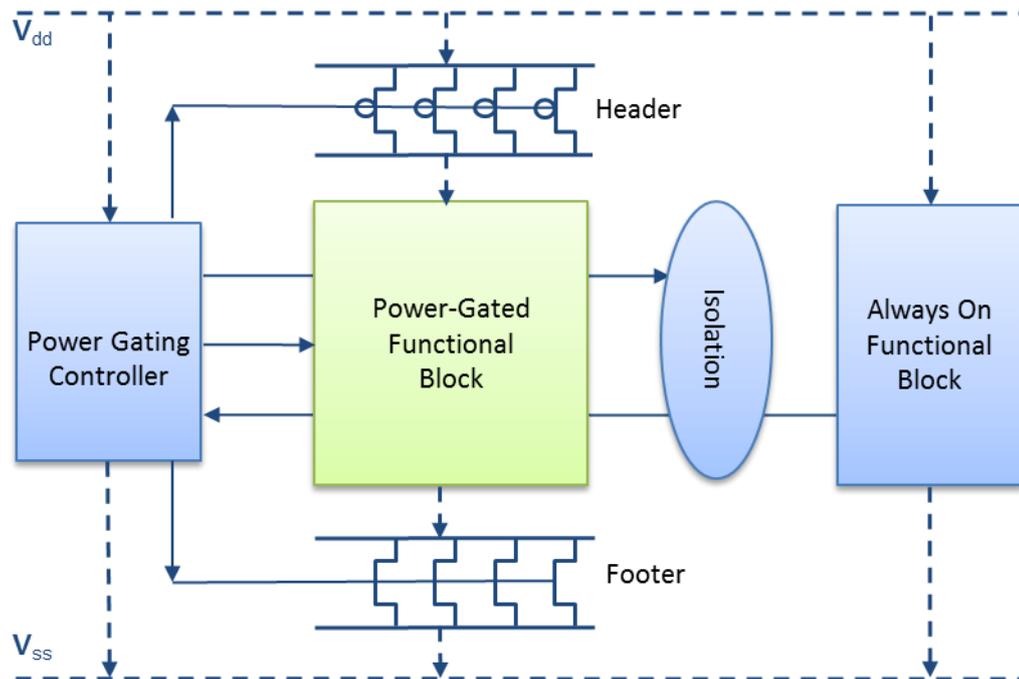
6) Libraries - To determine the voltage levels to be applied with specific frequency, a trial timing analysis has to be performed. Therefore, the timing libraries must be characterized beyond standard supply voltages.

7) Power-up sequencing - During power up, there is some time needed to settle all voltages and frequencies to desired level. The power-on-reset or some other circuitry has to be implemented to ensure safe start-up.

### **2.3.3. Power Gating**

The standard techniques to reduce dynamic power have none or a very limited impact on leakage. The most effective way to reduce leakage is to shut off the power of a block when it is inactive. This technique is called power shut-off (PSO) or power gating and it is illustrated in Figure 19.

In power gating, high- $V_{th}$  transistors are used to shut-down the power of inactive low- $V_{th}$  circuit reducing the leakage. A power gating circuit is first proposed in [66] in form of switched-source impedance circuit (SSI). SSI used a switch in parallel with a resistor to control the leakage during standby. This circuit had some limitations such as need to retain the state when it is in standby mode. Those limitations are thoroughly discussed in [67]. The multi-threshold CMOS (MTCMOS) technology presented in [10] was the first to propose the usage of high- $V_{th}$  transistors for power switching. Sleep transistors (also called power gates or power switches) in MTCMOS are implemented by PMOS device separating circuit from  $V_{dd}$  (header) and NMOS device cutting-off  $V_{ss}$  (footer). Modern power gating implementations use only one type of power switches, footer or header. Alternatively, combined or alternating switch topologies are also possible [68]. Power gating gained on popularity as the contribution of leakage in deep-submicron CMOS increased. The proposed implementations of power gating include fine grain power gating, where each MTCMOS cell integrates power switches [69], [70], [71] or coarse grain power gating, where power switches are applied at block level [72], [73]. An advantage of fine-grain power gating is the ability to use standard design flows, since the virtual power nets are hidden within the cells. However, fine-grain power gating requires designing of new libraries, and it is more sensitive to process, voltage and temperature (PVT) variations affecting voltage drop [74]. At the other hand, coarse-grain power gating makes less impact on voltage drop and chip area, but it introduces additional design complexity. Modern power-gated designs implement coarse-grain power gating.

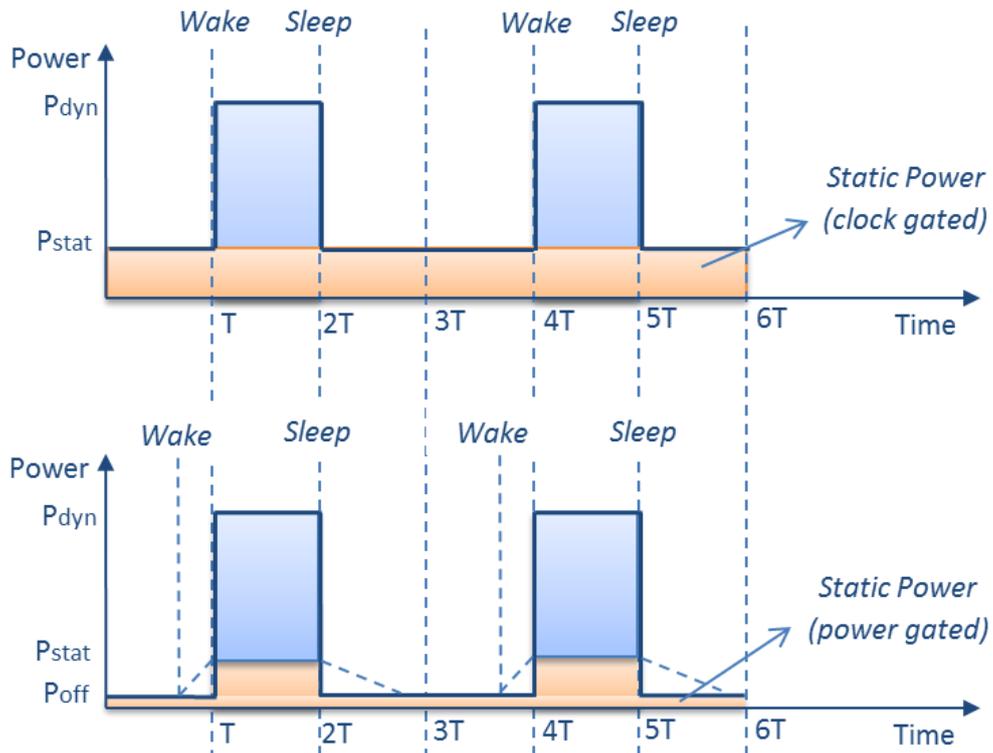


**Figure 19.** Basic concept of power gating. PMOS (header) and NMOS (footer) power switches disconnect a block from the global power supply network. The block outputs to always-on logic are electrically isolated to prevent crowbar currents. The power sequencing is controlled by power gating controller.

Power gating can be applied in memories as well. The implementation of power gating in cache memories is discussed in [84]. A power-gated DRAM is presented in [85]. A common technique to implement power gating in SRAMs, called memory splitting, applies power gating to the idle memory banks. Another approach uses multi-mode techniques in SRAM that reduce supply voltage to the level of data retention voltage [86]. A combination of those two techniques is also possible [87].

There are different styles to distribute power switches in power-gated design. The most popular are ring and column based style. The power can be switched either by NMOS transistors in footer configuration or by PMOS transistors in header configuration. It is also possible to use both headers and footers, but it would have larger impact on voltage drop. The signals from a power-gated block to an always-on block must be electrically isolated to prevent crowbar currents in the active block. Power switching is controlled by power gating controller. The efficiency of power gating depends on the size of the gated logic, power switching frequency and activity profile. The longer are the inactive periods the better is the power saving efficiency. The illustrated examples of activity profiles with and without power gating are given in Figure 20.

The implementation of power-gating technique is related to a number of design challenges including sleep transistor sizing, signal isolation and state retention.



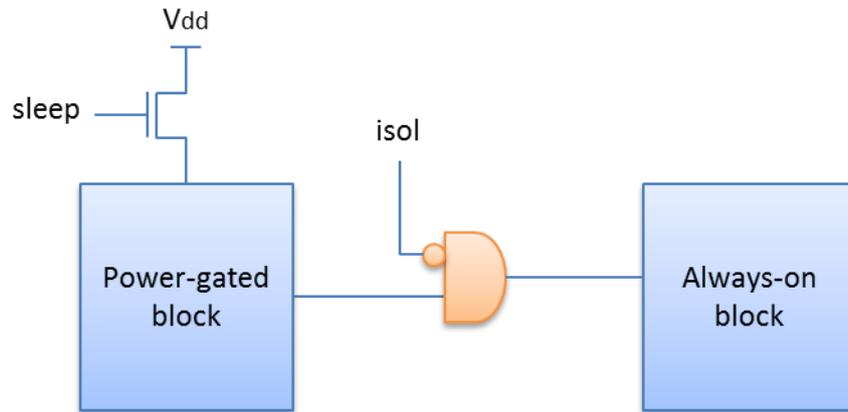
**Figure 20.** Activity profiles: clock gating vs power gating.

### *Sleep Transistor Sizing*

Sleep transistor sizing tries to find an optimal switch transistor size for a given MTCMOS implementation. In [75], authors proposed a tool that analyses the delay in fine-grain MTCMOS and calculates the optimal sleep transistor size. Another algorithm, presented in [76], applies a slack-based approach promising additional power savings compared to conventional fixed-delay implementation. In coarse-grain power gating, sleep transistors are designed and characterized as special cells in digital library. Accordingly, modern implementation tools are able to calculate the optimal number of sleep transistors to be applied to a power-gated block.

### *Signal Isolation*

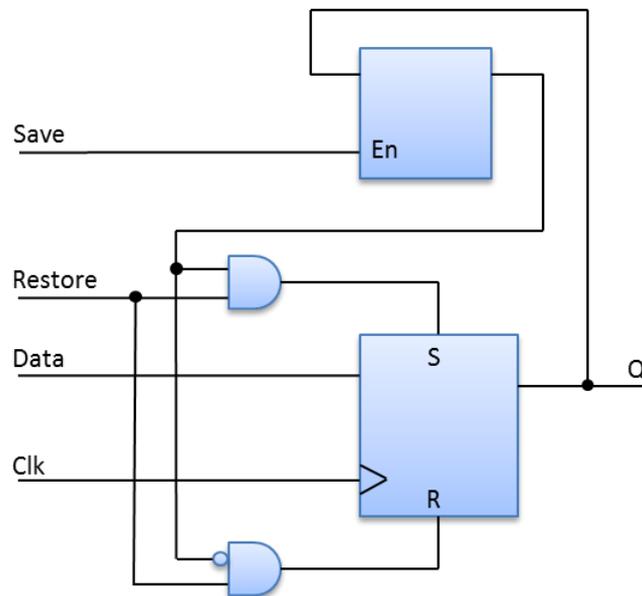
The implementation of power gating requires the output signals from a powered-down block to be electrically isolated in order to prevent crowbar currents in active blocks. This is usually done by implementing isolation clamps on block output signals that set the signal levels to logic 0 or logic 1 before the block enters power-down state (Figure 21). The standard cells (AND, OR) can be used for signal isolation or special cells can be designed [77].



**Figure 21.** Isolation cell.

*Retention*

When a block is powered down, the internal state of its register is lost. If it's required to preserve the state of registers, state retention techniques have to be implemented. Usually, state retention is provided by some sort of shadow logic (latch) that can store the data during sleep periods [78], [79] (Figure 22).



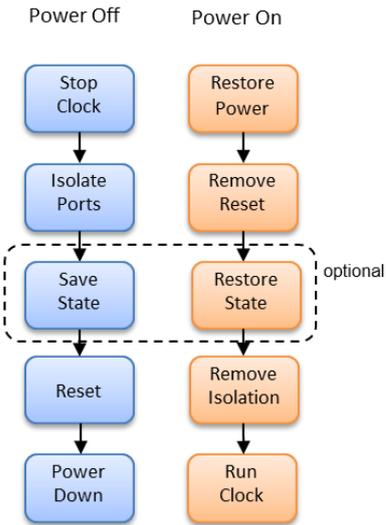
**Figure 22.** Basic architecture of retention flip-flop.

A shadow latch can be implemented as external cell or as a part of complex flip-flop cell. In [80], authors discussed the problem of leakage sneak paths in different retention flip-flop topologies and proposed a topology based on leakage feedback gate that ensures low leakage of a cell in standby

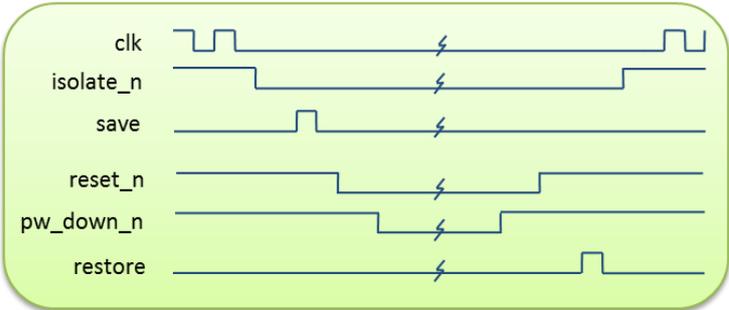
mode. However, this topology introduced significant overhead to the size of flip-flop. The optimized retention cell topology having minimal impact on delay, area and power, is presented in [81]. A special case of retention cell that unifies retention latch and a scan flip-flop is presented in [82]. Another commonly used technique to retain the state during power-down mode is scan-based approach [83], [73]. Using the scan chain infrastructure, it is possible to efficiently store the state of registers in an always-on memory.

*Power Gating Control*

In a power gated system, it is critical to perform safe power-up and power-down procedures. In systems with fine grain power gating, it may take significant time until the power-up is completed. Therefore, power controller must ensure that power switching is performed safely. Basic steps of simple power-on and power-off sequences are shown in Figure 23a. The power-off sequence usually starts by stopping the clock and applying the isolation to the output signals of power-gated block.



a)



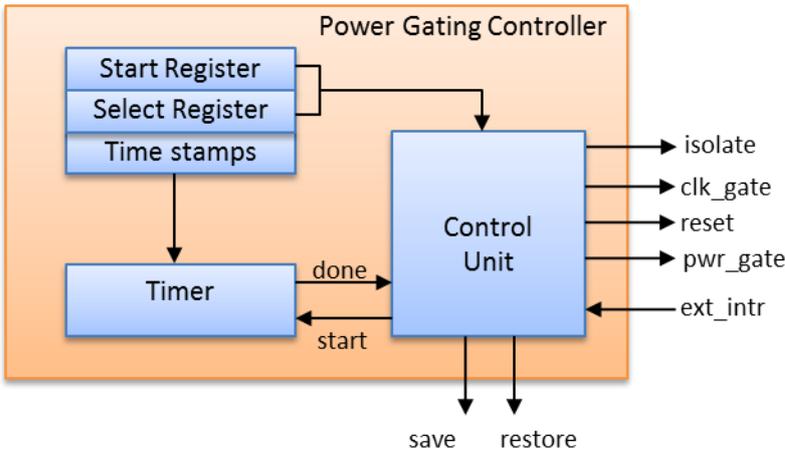
b)

**Figure 23.** Example of power switching sequencing in a power-gated design: a) basic steps of power-on and power-off procedures, b) control signals from power gating controller.

If the retention is implemented, the state of the retention registers is saved and the reset signal is applied. Then, the power is turned-off. The steps to turn the power on are opposite to those of the power-off sequence. The changes in control signals of power gating controller are shown in Figure 23b.

Power gating controller has to be carefully designed in order to provide sufficient time for each step of a power control sequence to execute safely. The controller actions are usually controlled by software (e.g., processor). It is also possible to implement hardware monitors that initiate actions in power controller. In software-controlled systems, the duration of power-down states is often controlled by timer functions. The timer generates an interrupt that initiates processor to program consecutive actions in the controller. In hardware-controlled system, the hardware interrupts may trigger the controller directly, without disturbing processor. The architecture of a complex power-gating controller is shown in Figure 24.

A typical power gating controller consists of power control unit (a finite state machine), register file storing control data and a timer that controls duration of power-down states. Usually, it is not required to have a dedicated timer, but the system timer can be used instead. In that case, the controller logic might be simplified to contain only the state machine and target select register. It is also common to omit timer functions by controlling power-down states directly in application software or by external interrupts.



**Figure 24.** Architecture of a complex power gating controller.

*Power Switches and Switching Network*

One of the critical design decisions when implementing power gating is to determine size, type and configuration of the power switches and switching network.

a) Header vs Footer - The first architectural issue is whether to switch  $V_{dd}$  (with a P-type “header” switch), or to switch  $V_{ss}$  (with an N-type “footer” switch), or to use both. Some researches recommend simultaneous use of both types of switches to improve leakage, but this comes at the cost of increased voltage drop. In header configuration, switching off the supply rail will collapse the

internal nodes towards the ground rail. The internal nodes will never be discharged completely but the equilibrium will be established when the leakage currents from the internal nodes reach the leakage current of the switches. In footer configuration, the ground rail of the internal nodes is pushed toward the supply voltage level. The designers have to decide which type of power switches to implement.

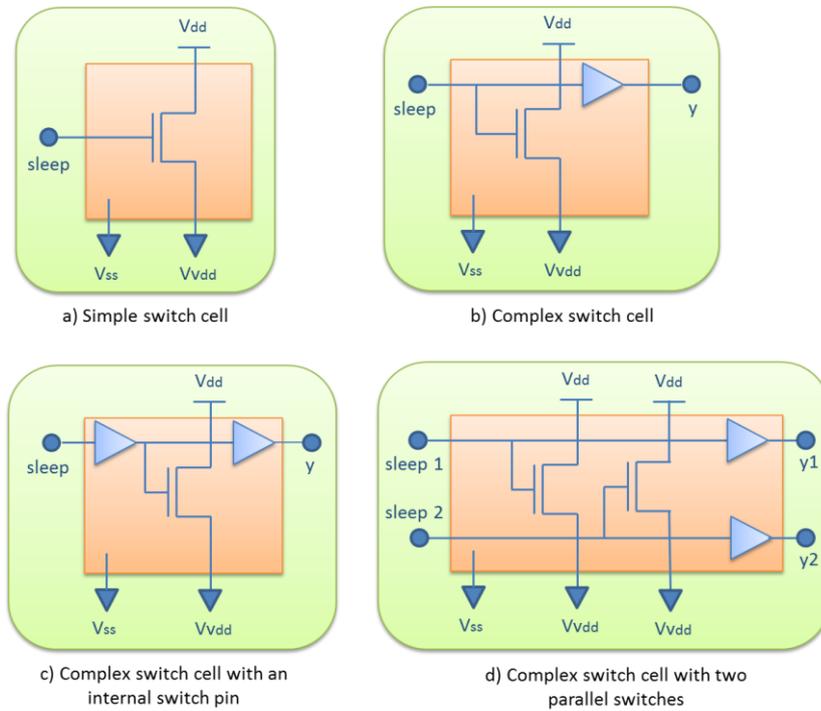
There are some architectural and technological issues that must be considered when choosing between footer or header configuration. The architectural issues are related to the target architecture. For example, if DFVS or MVS techniques are to be used along with power-gating, then the header type of switch transistors may be appropriate choice, since the voltage shifters cells are usually designed to have two voltage supplies and common ground. The technological issues are related to specific features of target technology. For example, the p-substrate technology where N-type transistors have their sources connected to the substrate is inappropriate for the footer cell insertion.

Another parameter to be considered is the efficiency of switch transistors. The efficiency is defined as the ratio between drain currents in ON and OFF state ( $I_{on}/I_{off}$ ). The goal is to maximize the switch efficiency in active mode and to minimize the leakage in sleep mode. The NMOS sleep transistor usually has better efficiency and smaller size compared to its PMOS counterpart. However, from the system perspective, the header type of sleep transistor seems to be preferred solution, since the system designers refer to  $V_{ss}$  as logic '0', and they are used to think of  $V_{dd}$  as power supply that has to be switched on and off.

b) Fine vs Coarse Grain - In fine grain power gating, switch transistor is placed inside each logic cell along with an isolation clamp. This simplifies implementation, since the standard design flow can be used for implementation. Fine grain power gating has less impact on voltage drop, but it introduces significant impact on the chip area. Also, the switching time in fine grain power gating is longer compared to coarse grain. However, it is not clear if the design area penalty of fine grain approach is worth the power savings.

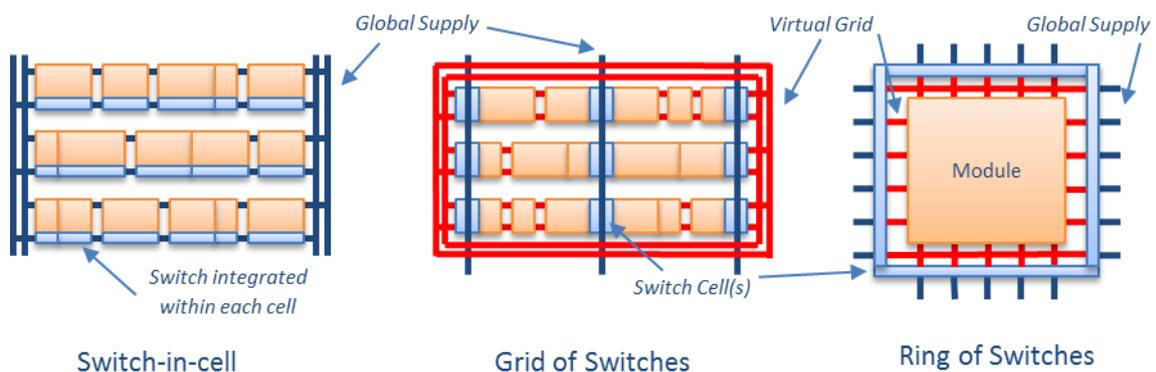
In coarse grain power gating, a collection of power switches controls the power of a block. The area impact is much less than in fine grain power gating, but the sizing of switching network is a challenge. The estimation of the size of switching network is somewhat automatized in modern EDA implementation tools. Today, almost all designs implement coarse grain power gating.

One of the most critical challenges in coarse grain power gating is to manage the in-rush current during power-on transition. The excessive current flow can make significant impact on voltage drop affecting the function and reliability of active blocks. Therefore, coarse grain power switches are usually designed to include buffering on control sleep signals, enabling sequential switching of power-gating transistors (Figure 25b-d). This helps reducing the peak in-rush current.



**Figure 25.** Examples of coarse-grain power switches.

c) *Ring vs Grid Style* - The power switches can be implemented in ring- or column-based style power network. The ring-based implementation places the switches outside the power-gated block, effectively forming a ring of switches around the block. In the grid-based implementation, switches are placed inside the power-gated block forming an array (usually a column or daisy-chain formation). The ring-style approach has some advantages when it comes to the implementation since the power networks are separated and there is no placement and routing impact in the power-gated block. At the other hand, the grid-style approach makes less area impact since the switches are placed inside the power-gated blocks. Also, the switches do not have to drive long metal interconnects making the impact on voltage drop lower when compared to the ring-style implementation. Figure 26 shows different configurations of power switching networks.



**Figure 26.** Examples of power switching networks.

### *Design Challenges*

Implementation of power gating is associated with different design challenges. Some of those challenges have been already discussed, and they are related to the insertion of power switches, isolation logic, state retention and power control logic. Additional challenges include the design flow related tasks such as RTL design, synthesis, floorplanning, testability and system level issues.

The industrial design flows for low power design usually require the description of design power intent in corresponding file format. The power intent file contains a set of *tcl* commands, which describe power domains and supply voltages in the design. It also contains the information on low-power specific cells and libraries to be used. The details on the power intent specification and related low power flows are discussed later.

Power gating creates some additional challenges for design for test. These challenges include the testing of power switching network, the testing of current and power limitations, the testing of power controller functions, the testing of power switching sequences, and the testing of isolation and retention behaviour.

In the case of scan test insertion, the observability of power-gating related signals has to be provided, and the controllability of reset and clock signals has to be ensured. It is important to create scan structures in that way that they do not interfere with power control logic and thus trigger undesired power switching in the circuit. Also, the isolation signals and retention flip-flops have to be protected from corruption during scan tests. A good practice is to have separate scan chains for each power domain.

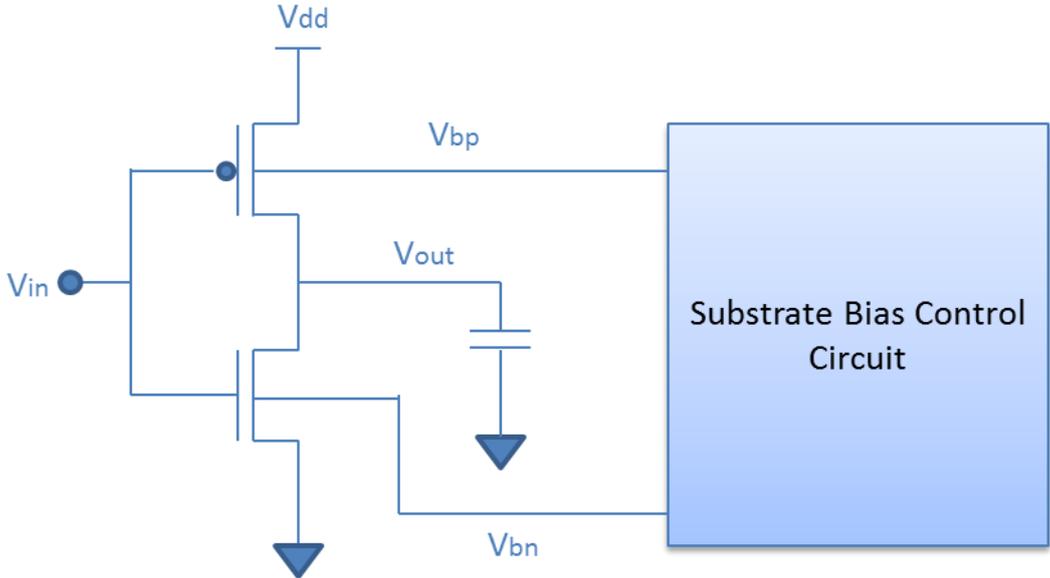
#### **2.3.4. Body Biasing**

The body biasing technique is used to control transistor threshold voltage by connecting the body of transistor to a bias network rather than power or ground. The change in the threshold voltage results from the voltage difference between source and body of transistor. This is also known as body effect. The body bias can be supplied from an external off-chip source or from an internal on-chip source, usually a charge pump. Reverse body biasing (RBB) applies negative body-to-source voltage to n-type of transistor, raises its threshold voltage, and makes it both less-leaky and slower. Forward body biasing (FBB), at the other hand, lowers the threshold voltage of n-type transistor by applying positive body-to-source voltage making it both faster and leakier. The polarities are opposite for p-type of transistor. Usually, body biasing techniques are used to compensate for variations in process parameters [88].

The simplest body biasing methodology is to apply the fixed body bias voltage to all devices in production lot. For example, additional power savings can be achieved in a power-gated design if fixed forward bias is applied to sleep transistors during the on-state and fixed reverse bias is applied during the off-state [89].

Adaptive body biasing (ABB) is more advanced methodology allowing calibration of the body bias of each chip during production tests. Adaptive body bias is often used to adjust chip performance in post-production phase, compensating for PVT variations [90].

Dynamic bias techniques can change the body bias voltages during operation time. Dynamic biasing can be used to reduce temperature and aging-effects and make power management modes more efficient. An example of dynamic ABB and DVFS is presented in [91]. The presented algorithm claims 48% improvement in the average power consumption over standard DVFS.



**Figure 27.** Body Biasing.

The effectiveness of reverse body-biasing in  $0.18\ \mu\text{m}$  single- $V_{th}$  and  $0.13\ \mu\text{m}$  dual- $V_{th}$  process is discussed in [92]. The work discusses RBB dependency on the channel length, target  $V_{th}$ , temperature, and technology generation. The analysis shows that RBB effectiveness diminishes with technology scaling, primarily because of the worsening of short channel effects in transistors.

The idea of reverse body biasing is to dynamically change the threshold voltage  $V_{th}$  by changing the substrate or well polarisation of selected transistors. The increase of threshold voltage will reduce the leakage power (Figure 27). This methodology is also known as variable threshold CMOS (VTCMOS).

In PMOS, the body of transistor is biased to a voltage higher than  $V_{dd}$ . In NMOS, the body of transistor is biased to a voltage lower than  $V_{ss}$ . Body-biasing is usually applied during sleep mode to reduce leakage. Moreover, the increase in  $V_{th}$  affects delay of the transistor, so the body biasing technique can also be applied dynamically during the active mode of operation.

The implementation of body biasing is related to some challenges.

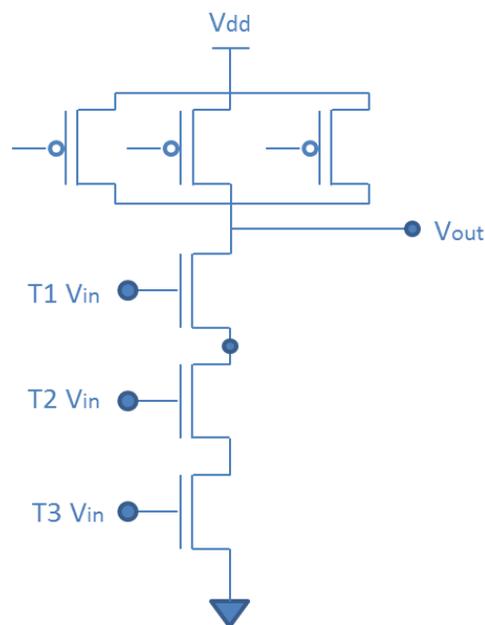
- Body biasing requires twin-tub (twin well) technology where the substrates of individual devices can be independently adjusted.
- To control body biasing, a control circuit is required, typically a charge pump. This circuit introduces additional overhead in power and area.

- There is an impact on routability since the substrate contacts have to be routed.
- The libraries need to be characterized for different threshold voltages.

The efficiency of body biasing decreases with each new technology process. Lately, the body biasing technique is overshadowed by power gating. Alternatively, it is possible to bias the diffusion area instead of substrate.

### 2.3.5. Stacked Transistors

The technique to control standby leakage using a stack of transistors is also known as self-reverse bias. This technique exploits the stacking effect, in which the subthreshold current flowing through a series of transistors reduces when more than one transistor in the stack is turned off [93]. Some gates such as NAND, NOR or other complex gates, have naturally stacked transistor structures (Figure 28). The choice of appropriate block input vectors during the standby mode can maximize the number of stacked transistor structures and additionally save the power. A technique presented in [94], based on an algorithm for input vector selection, results in 44% improvement in power savings over conventional stacking technique. The forced stacking of transistors in a non-stacked gate reduces the driving current of the gate resulting in delay penalty [95]. Therefore, the forced stacked transistor structures have to be inserted only on non-critical paths. The transistor stacking technique is expected to become less effective with further technology scaling [96].



**Figure 28.** Stacked 3-input NAND Gate.

If transistors are connected in a series and one transistor is turned off, then the subthreshold leakage of the transistor stack will be reduced compared to the leakage of a single transistor. For example, the leakage of a two-transistor stack is one order of magnitude less than the leakage of a

single transistor. The leakage reduction takes place because the voltage level of the intermediate node (between the two transistors) is positive. This leads to negative  $V_{gs}$  (gate-to-source voltage) and negative  $V_{bs}$  (body-to-source potential), and it also leads to reduction of  $V_{ds}$  (drain-to-source voltage). Consequently, this yields in lower substrate current  $I_{sub}$ .

The transistor stacking can significantly reduce the subthreshold leakage. However, there is a penalty in cell area and dynamic power associated with this technique.

### **2.3.6. Subthreshold Logic**

Subthreshold logic circuits operate in transistor subthreshold region exploiting the subthreshold leakage current as the operating drive current [97]. The subthreshold current has an exponential dependency on the gate voltage. Operating a circuit in subthreshold mode gives an exponential reduction of power consumption, but it also gives an exponential increase in delay. However, the gain in power is proven to be higher than the decrease in performance. The standard CMOS circuits operating in subthreshold mode are very sensitive to process variations and changes in temperature and supply voltage. This is the main limitation of subthreshold logic. It is shown that pseudo-NMOS [98] and dynamic-threshold MOS (DTMOS) [99] structures have some advantages over standard CMOS when operating in subthreshold region. The pseudo-NMOS logic has better robustness than standard CMOS. However, always-on pull-up PMOS in pseudo-NMOS is highly sensitive to process variations [100]. The DTMOS logic, having gates of transistors connected to the substrate, has simpler architecture and better process variation stability, but its fabrication process is difficult.

### **2.3.7. Dynamic Power Management**

System level dynamic power-saving techniques use runtime behaviour to reduce power when system is idle or having low workload. These techniques, known as dynamic power management (DPM), perform dynamic reconfiguration of system components to provide the requested services and performance levels with a minimum number of active components or a minimum load on such components [101]. Dynamic power management techniques are based on workload prediction using different rules, i.e., policies, to determine whether and when to shut down a device. The survey of the most important DPM policies and their strengths and limitations is given in [102]. DPM has been used in past to optimize power of large multi-chip systems. An example of an embedded on-chip DPM is to be found in the DVFS-based systems [103].

### **2.3.8. Overview**

The advanced low power techniques emerged as an answer to the increasing impact of short-channel effects to the leakage power of deep submicron CMOS. The process level techniques such as halo-doping and retrograde well may improve performance of standard bulk-CMOS to some extent, but they become less effective and more difficult to implement as the technology features continue to scale. The novel process and transistor techniques such as SOI and FinFET promise significant improvement in performance and power and tend to become standard choice of future designs. The

advanced circuit-level low power techniques, today mainly used in planar bulk-CMOS, will continue to apply in future technologies as well. The multi-voltage techniques based on voltage and frequency scaling promise very good dynamic power savings and moderate leakage power savings, but they have significant impact on design complexity. MVS and DVFS add level shifters and voltage source circuitry to design making considerable area penalty of up to 10%. These techniques suffer from clock scheduling issues due to dynamic latency changes in circuit. In DVFS, power management unit has to maintain control of complex power-up sequences defined by task scheduling algorithms making an impact on design and architecture. The body-biasing and power gating can significantly reduce the leakage but at the cost of increased design complexity and additional area and timing penalties. Power gating is the most effective technique for the leakage power reduction. It adds power switches, isolation cells, retention registers, always-on buffers and power management unit to design. In order to prevent excessive voltage drop caused by in-rush currents, the power grid of a power-gated design has to be designed wider making an impact on area. The estimated area penalty introduced by power gating is 5-15%. The implementation of body-biasing requires a twin-well technology. The body-biasing technique adds power control circuit to design and has high impact on routing, due to the large number of substrate contacts that have to be connected to the control circuit. Body biasing becomes less effective with each new technology generation. Table 2-3 gives an overview of the most important advanced low power techniques and their impact on design and power.

**Table 2-3.** Summary of advanced low power techniques.

Technique	Dynamic Power Savings	Leakage Power Savings	Timing Penalty	Area Penalty	Impact: Architecture	Impact: Design	Impact: Verification	Impact: Place&Route
SVS/MVS	high (40-50%)	2X	low	medium	medium	medium	medium	low
DVFS	high (40-70%)	2-3X	low	medium	high	high	high	medium
Power Gating	~0%	10-50X	medium	medium-high	high	medium-high	high	high
Body-Biasing	~0%	10X	high	medium	high	high	medium-high	high

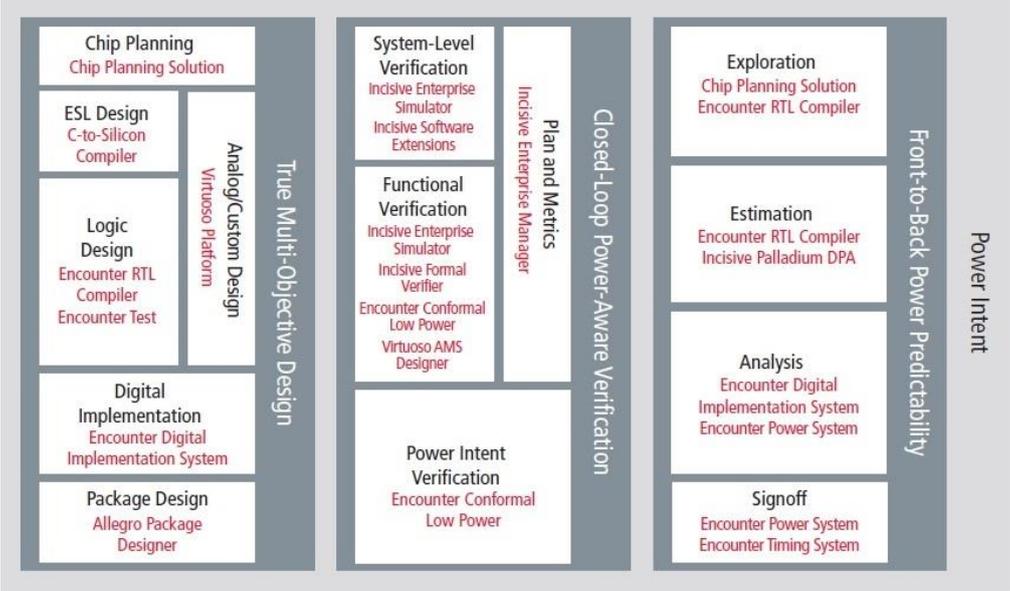
**2.4. Power Aware Design Methodologies**

The need for low power initiated the creation of advanced low power techniques that use multiple, often switchable, power supplies. These advanced techniques drove the need to express specifications and rules, requiring a set of design semantics not envisioned or supported by hardware description languages, which previously abstracted away power and ground connections during logic design phases as unnecessary facets of physical implementation [144]. This power-related semantics is usually referred as the design’s power intent. The power intent describes power domains, voltages, power cells (switches, isolation, and level shifters) and power rail connections in design as well as the power-related implementation rules. The power intent information enables design tools to perform power-aware analysis and optimization through the implementation process. Two widely adopted formats for power intent exist: Si2’s Common Power Format (CPF) [145] and IEEE’s Unified Power Format (UPF) [146]. The two formats are similar but have some major differences in how the power intent is defined. CPF enables higher level of abstraction at RTL level, where UPF requires the exact

physical power network to be described at RTL level. The both formats are used in major industrial design flows. The CPF format is supported by Cadence design tools, and the UPF format is used in Synopsys design tools.

**2.4.1. CPF-Enabled Cadence Low Power Flow**

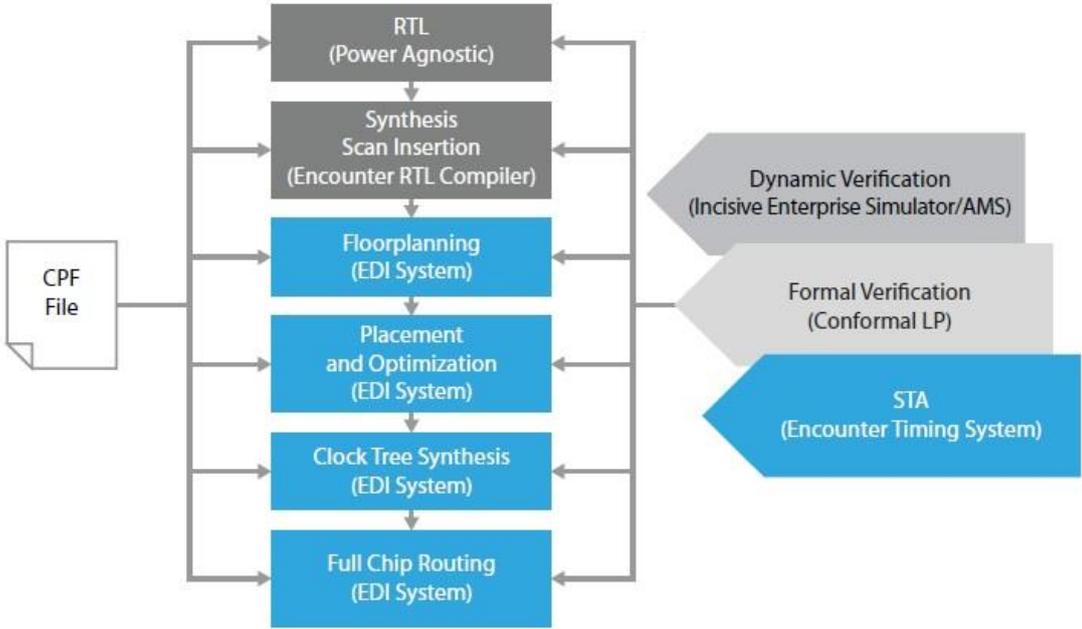
Cadence provides a comprehensive technology that enables the CPF-based low-power solution (Figure 29). The classical technologies included in the solution are Incisive® Enterprise Simulator (for low-power functional verification), Virtuoso® AMS Designer (for power-aware mixed-signal simulation), Encounter® RTL Compiler (for power-aware logic synthesis and design-for-test synthesis), Encounter Conformal Low Power (for power-aware formal verification), Encounter Test (for power-aware automatic test pattern generation), Encounter Digital Implementation System (for power-aware physical implementation), Encounter Power System (for power integrity and signoff analysis), and Encounter Timing System (for timing signoff) [147].



**Figure 29.** Overview of Cadence technologies enabled by CPF [147].

A typical design flow that employs Cadence design tools is shown in Figure 30 [148]. The power intent of the design is specified in CPF format. The CPF enables a “power agnostic” RTL design that preserves the reusability of RTL design for different architectures. A power-aware simulation that verifies shutoff states, retention and isolation is performed with the Cadence Incisive Enterprise Simulator. The RTL compiler tool is used for power-aware synthesis. It infers power aware logic to the design, e.g., retention registers, power management unit, level shifters and isolation cells. The backend design is performed with the Encounter Digital Implementation System (EDI). The formal verification including rigorous structural and functional tests on design is performed with the Conformal Low-Power tool. The static timing analysis is done with Encounter Timing System (ETS).

Both Conformal and ETS are integrated within the EDI system environment. The EDI system also integrates the tools for power and rail analysis, Encounter Power System and VoltageStorm.



**Figure 30.** SoC low power flow with Cadence tools [148].

The Cadence tools for physical design implementation (blue coloured tasks in Figure 30) are also used in the CPF-enabled implementation flow that is proposed in this work (Chapter 4). The Cadence verification tools support the flow as well. Some of the features of the EDI System are also used in developed methodology to enable fast prototyping of required power switches. The methodology is presented in Chapter 3.

**2.4.2. UPF-Enabled Synopsys Low Power Flow**

The low-power solution from Synopsys provides a set of tools integrated within the Galaxy implementation platform that support the UPF-based low power design. The standard tools used in the solution are Design Compiler (for power-aware logic synthesis), IC Compiler (for power-aware place and route), VCS and MVSIM (for power-aware functional verification), MVRC (low power rule checker), Formality (for logic equivalence checking), PrimeTime and PrimeTime PX (for static timing analysis and power analysis) and PrimeRail (for power rail analysis). Figure 31 shows a schematic view of the UPF-enabled low power design flow from Synopsys.

Design Compiler (DC) is the first and the best-known product from Synopsys. The DC solution is used for logic synthesis delivering high quality netlists optimized for timing, area, power and test. Superior quality of results, high flexibility and fast runtime defined DC as the standard logic synthesis solution used in this work. DC is used in the methodology for early power prototyping of system components and in the frontend design flow.

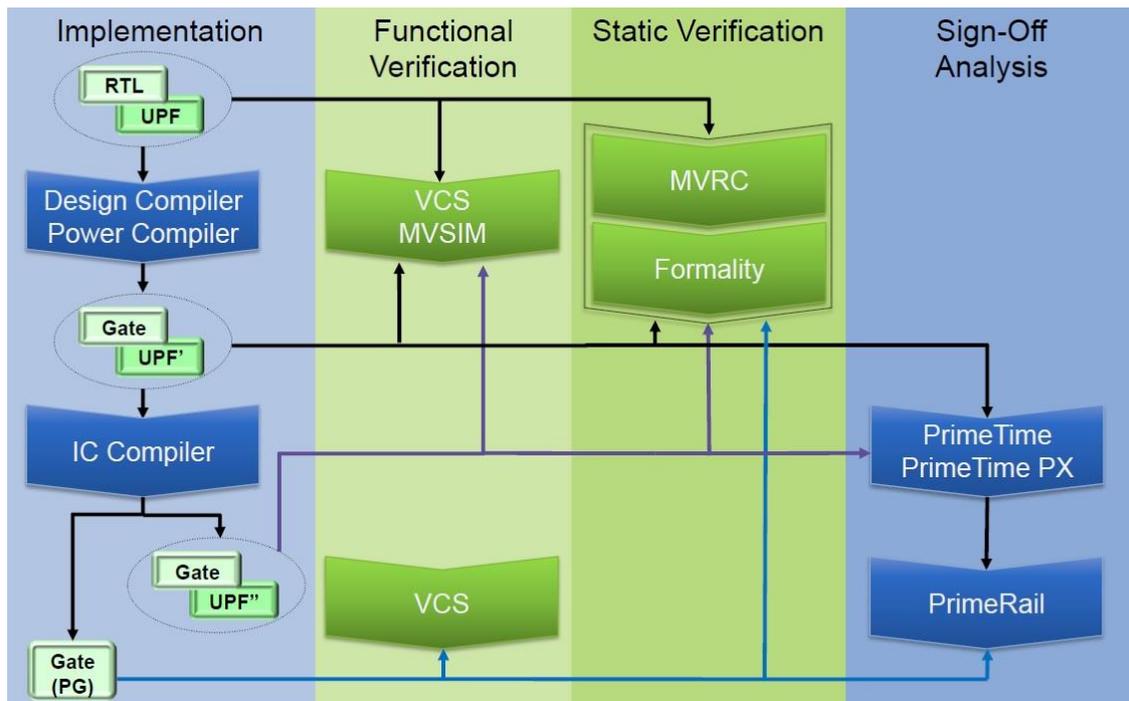


Figure 31. Synopsys UPF-enabled low power flow [149].

### 2.4.3. High-Level Power Estimation

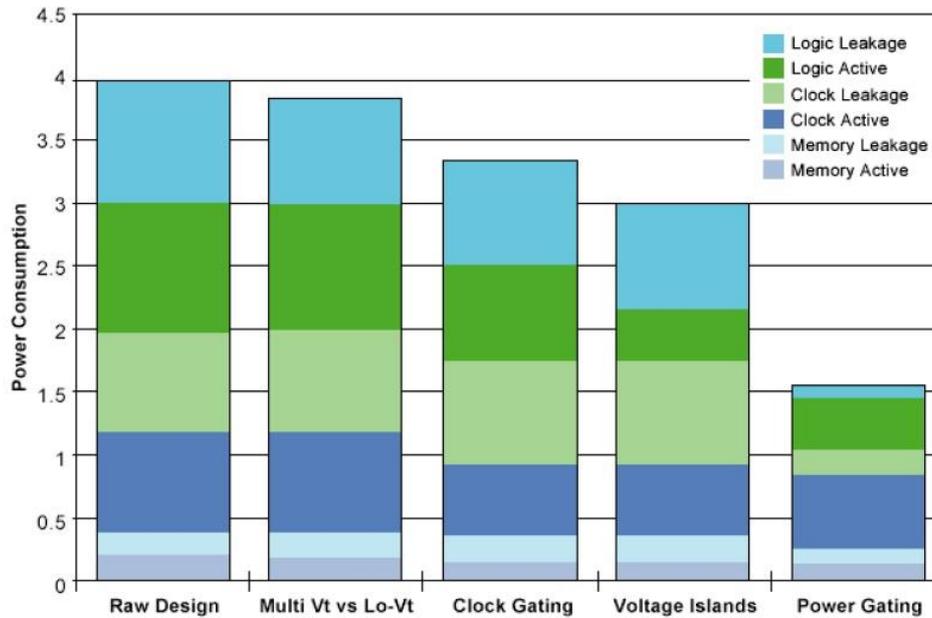
When designing a system, designers have to choose which power saving technique to apply. There is a variety of available power saving techniques that target both dynamic and static power. Each power saving technique has certain impact on the design. Choosing the optimal power saving strategy is one of the most critical design decisions. The energy efficiency of design will depend on the implemented power saving strategy and introduced overhead. A straightforward way to estimate power impact of different power-saving techniques is to apply each technique separately and obtain results at the late design flow stage. However, this approach is very time consuming, and it would require a large effort from designers. Therefore, there is a need for early power estimation.

The techniques for early power estimation can be applied at architecture-level, behavioral-level, instruction-level and system-level. Architecture-level power estimation uses either analytical or empirical methods to estimate the power. Analytical methods attempt to relate the power consumption of particular RTL description to fundamental quantities that describe the physical capacitance (complexity-based models) and activity of design (activity-based models). The complexity-based models relate the power of design to the estimated “gate equivalents” and corresponding power of a gate. The activity-based models relate the power to the amount of computational work a functional block performs. The advantage of analytical methods is that they require little information as input. The disadvantage is high inaccuracy due to difficulty to capture the power attributes of different functional blocks using only parameters such as gate equivalent count or activity. The empirical methods use the results of power measurements on existing implementations to build a high-level power estimation model. They use either fixed-activity models or activity-sensitive models for power estimation. The empirical methods are well suited for the IP-based designs. The behaviour-level power estimation relies on power estimation models that assume

some architectural style and related power consumption for given block behaviour. The behavioural methods use static or dynamic activity prediction to produce an estimate of the access frequency in hardware resources of behavioural block model. The instruction-level power estimation is used to estimate the power of processor by relating the power model to the estimated power of a single processor instruction. The system-level power estimation is used for rapid power prototyping using the empirical power estimation models based on the datasheet information of different components. The summary of some early works on high-level power estimation techniques is given in [150].

High-level power estimation for designs that implement advanced low power techniques is very difficult due to complex physical implementation, existence of multiple voltage and shutoff regions and inability to accurately model the costs of a particular low power implementation. The existing commercial solutions provide either IP-based system-level power estimation or RTL and gate-level power estimation. An example of system level design exploration tool is InCyte from Cadence [151]. InCyte allows fast architectural exploration of design under the presumption that the design is composed of the characterized IP components stored in the InCyte's IP library. This solution is well suited for fast architecture exploration of IP-based designs. The Sequence's Power Theater, now PowerArtist (after the acquisition of Sequence from Apache/Ansys's), is a tool that can perform early power estimation and architecture exploration at RTL level [152]. Power Artist relies on the activity information extracted from RTL simulations, and it uses the power and timing information from timing libraries to estimate the power. The tool also includes algorithms to identify clock gating candidates or to detect overhead in clock distribution. Power Artist supports CPF and UPF power intent formats, and it can give a rough estimation of power savings when different low power techniques are applied. However, the tool is not able to accurately account for the trade-off introduced by the implemented low power methodology. Also, a large problem of early RTL power estimation is providing representative scenarios of system activity [153]. Furthermore, in the case of long lasting RTL simulations, the saved circuit activity becomes too large to be efficiently processed by the tool. An example of early power estimation with the Power Theater tool for different power saving approaches is illustrated in Figure 32.

The tools for early system and RTL power analysis can be beneficial when designing a general purpose system-on-chip or processor which activity cannot be accurately predicted. Also, for a large systems-on-chip, it may help reduce the time to collect the power information related to the system architecture and clock distribution. However, when designing an application-specific integrated circuit, the decision on which power saving strategy to apply depends strongly on the application itself and the target semiconductor technology. When choosing power saving strategy for specific design, it is important to determine the features and limitations of the target technology that may affect the implementation of particular power saving technique and implementation costs. Furthermore, an accurate estimation of the activity profile of the target system architecture is required.



**Figure 32.** Comparison of different power saving techniques with Power Theater [153].

#### 2.4.4. Energy Overhead Modelling

The estimation of energy overhead introduced by the implementation of particular power saving technique is a critical task towards the accurate energy efficiency estimation of different low power implementations. The power and timing overhead of standard low power techniques is relatively small since no major changes in the design architecture and physical implementation are required. In multi- $V_{dd}$ , the power of a block is proportional to  $V_{dd}^2$ , and some power overhead is introduced in level shifters and in more complex power supply circuit. The dynamic multi-voltage and power gating techniques have significant impact on design architecture due to the insertion of special power cells and power control logic. These cells introduce significant overhead in power. Also, the creation of multiple power domains makes an impact on physical design increasing the total capacitance of power grid that contributes to the effects of power switching. For accurate power estimation, it is important to account for these effects. The impact of transition times when switching between different power modes has to be accounted as well.

The energy overhead in multi-voltage design is greatly influenced by the efficiency of power regulator and the duration of power transition times. An analytical model of DVFS energy overhead in a buck-type DC-DC converter applied to an ARM8-based prototype processor is presented in [154]. The model assumes that the processor is halted during the voltage/frequency transition time and that the energy is consumed both during voltage upscaling and downscaling. This is actually incorrect, since during downscaling, the charge stored in the bulk capacitor is discharged to the ground and there is no current flow from the power source. A more complex model of the energy overhead in DC-DC converter is presented in [155]. This model makes correct assumptions for the transition time energy and assumes that the processor is active during DVFS transitions. It accounts for the performance underdrive loss, inductor IR loss, PLL lock time, as well as for the charge transfer to and from the bulk capacitor. The model is applied to the Intel Core2 Duo E6850 processor and

LTC3733 DC–DC converter to calculate the energy overhead based on the simulation results. The models in [155] and [156] assume external DC-DC regulator. With an on-chip voltage regulator, the energy cost can be significantly reduced [53]. However, the impact of level shifters and power control logic has to be considered in the analysis of energy efficiency. Furthermore, there is some energy overhead due to the task scheduling that sometimes cannot be neglected [156].

The power gating energy overhead is mainly caused by the energy loss during power transitions. A simple transition model for the minimum idle time ( $t_{idle}$ ), at which power gating starts to benefit energy is presented in [157] and given by:

$$t_{idle} = \frac{E_{tr} - P_{leak\_n} \cdot (2t_{tr})}{(P_{leak} - P_{leak\_n})} \quad (2.12)$$

Where  $E_{tr}$  is the total transition energy,  $P_{leak}$  is the power wasted if no leakage reduction scheme is used,  $P_{leak\_n}$  is the resulting leakage power after the technique is applied, and  $2t_{tr}$  is the sum of *on* and *off* transition times. However, no methodology or models to calculate the members of the Equation (2.12) are presented. Another model for the estimation of the energy overhead in power-gated design is presented in [158]. The model calculates the number of clock cycles ( $N_{breakeven}$ ) until the breakeven point is reached as:

$$N_{breakeven} = 2 \frac{1}{L\alpha} \sqrt{\frac{mV_t W_H}{V_{dd} \cdot DIBL} \left(1 + 2 \frac{C_D}{C_S}\right)} \quad (2.13)$$

Where the leakage factor  $L$  is the leakage factor and presents the ratio of the average leakage and switching energy dissipated per cycle,  $\alpha$  is the average switching factor,  $V_t = kt/q \approx 25$  mV is the thermal voltage,  $m \approx 3$ ,  $W_H$  is the ratio of the total area of the header device to the area of the clock-gated macro,  $C_D$  is decoupling capacitance, and  $C_S$  is the total switching capacitance of all transistors in the macro. The model is applied to the fine-grain power gating in the execution units of a processor, where the idle times of the execution units are predetermined. In [14], the relative power saving of power gating (PRR) is expressed as:

$$PRR = \frac{\Delta P}{P_{npg}} \approx \frac{(1-d)(1-OQ)}{1 + \frac{d}{LQ}} \quad (2.14)$$

Where  $\Delta P = P_{npg} - P_{pg}$ , is the difference in power of the circuit block with power gating ( $P_{pg}$ ) and the circuit block without power gating ( $P_{npg}$ ),  $d$  is the duty cycle,  $OQ$  is the overhead quota defined as the ratio of the average activation power and the active mode leakage power, and  $LQ$  is the leakage factor defined as the ratio of the active mode leakage and the dynamic power. The model shows that for small leakage quota only a weak power reduction can be achieved. For larger values of the leakage quota, the power savings increase and become more dependent on the overhead quota. The increase in duty cycle reduces power saving capabilities. The RTL power gating models to be used in high-level synthesis are presented in [159].

The existing models for the estimation of power and energy efficiency of different low power implementations mostly relate to a single circuit unit or specific implementation. These models often miss a system perspective. As a system consist of a number of functional units, the implementation

of particular low power technique must be considered at both block and system level. Accordingly, the power estimation models have to be designed to account not only for the energy overhead of single block implementations, but also for the overhead of components shared on the system level, e.g., power control logic, voltage regulation, task scheduling, etc. Also, a practical approach is often not considered in the published literature. In this work, the power overhead modelling is considered at system level as part of the proposed low power methodology. It is illustrated in the design of a sensor node microcontroller. The methodology application and the microcontroller design are described later.

## **2.5. Sensor Node**

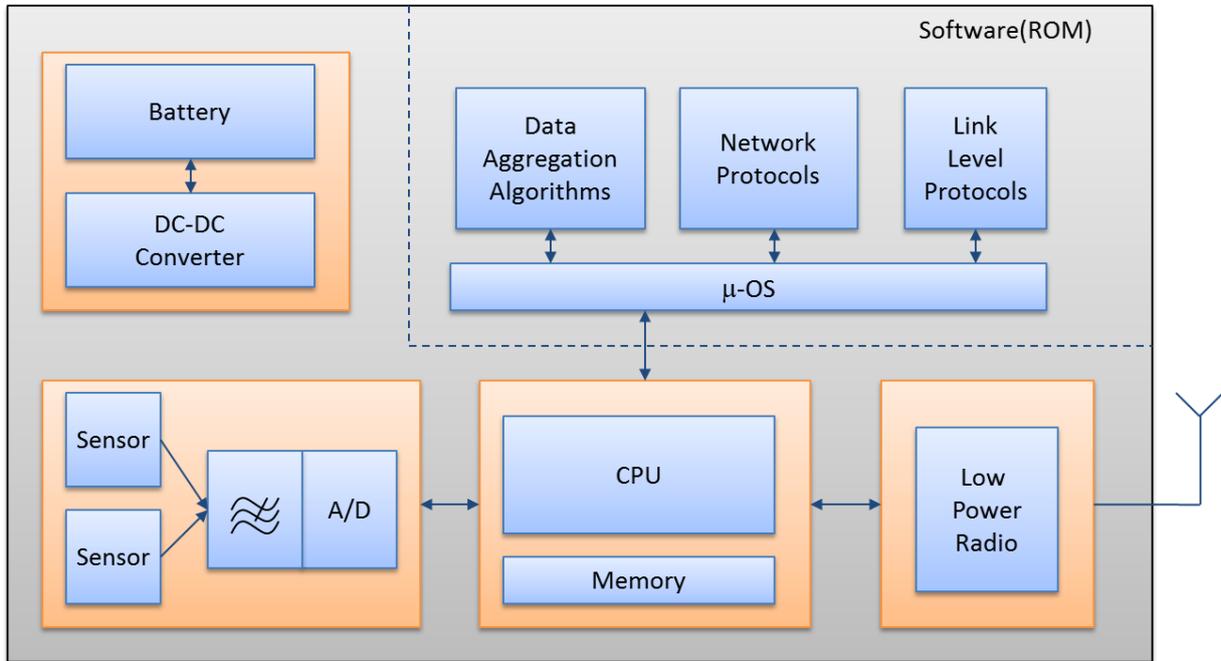
Sensor node devices combine computing, sensing and communication features within a tiny, battery-powered hardware system. Once deployed, sensor nodes are organized in a network where the data is seamlessly routed among all the nodes. The size of a network depends on specific application, and in some cases it might scale to hundreds or even thousands of nodes. Since the periodic recharge of battery is usually not maintainable, sensor nodes have to provide long-time operation with a very limited power budget. That makes power being the most important constraint when designing sensor node hardware.

The research on sensor node hardware systems is being focused in two directions. At one hand, at the development of generic sensor node hardware platforms built entirely from the COTS components covering wide range of sensor network applications, and, at the other hand, at the development of specialized sensor node platforms and processor architectures that may fully answer to the application-specific sensor network requirements.

### **2.5.1. Sensor Node Platforms**

Wireless sensor nodes can be used in wide range of different application scenarios [104]. In [105], authors noticed that sensor network systems require a hierarchy of nodes starting from low-level sensors having a specific purpose of monitoring and collecting of data and continuing up through high-level data aggregation, storage and analysis. Accordingly, the sensor node hardware platforms are divided into four different classes. At the lowest level are asset tags designed for specific sensing purpose. The asset tags have miniature size, limited bandwidth and low processing and storage capabilities. The second class are generic hardware platforms used for general-purpose sensing applications. The generic nodes offer more bandwidth and better processing and storage capabilities but at the cost of increased size and power consumption. The third class are sensor nodes used for high-bandwidth sensing applications (acoustic, vibration and video), and the fourth class are gateway nodes that perform data aggregation and link to the end user. Another classification [5] divides the sensor node platforms into three categories: a) adapted general-purpose computers such as PDAs and embedded PCs used for high-performance applications; b) embedded sensor modules built from the COTS components that have low cost and consume less power; and c) systems-on-chip that integrate micro electromechanical systems sensors, processor and wireless transceiver into a single application-specific integrated circuit.

In general, all sensor node hardware platforms share the same architecture. A typical sensor node hardware system consists of microprocessor, radio device, a number of sensors, memory-subsystem and power supply unit (Figure 33).



**Figure 33.** Standard architecture of sensor node.

The different classes of sensor nodes employ the hardware of different complexity. Low level sensor nodes rely on simple hardware components sufficient to perform basic functions when running at low duty cycle. They have small size, low cost and consume very low power. Advanced sensor nodes employ more powerful hardware components offering more performance when running at higher duty cycle. In order to efficiently utilize the available hardware resources, the software running on sensor nodes must be carefully designed. The low-level and generic hardware platforms usually employ a special operating system that can provide efficient utilization of underlying hardware. An example is an operating system called TinyOS (developed at the University of California, Berkley) that provides low-level hardware control through a built-in component model that eliminates layering [106]. The high-bandwidth and gateway nodes use however more complex operating systems.

An example of specialized low-level sensor node platform is Spec [107]. Spec relies on a custom-built 8-bit microprocessor integrated on the same chip with a simple RF transmitter and A/D converter. The integrated system has the size of 2.5 mm x 2.5 mm consuming 10-15 mA at 1.8 V and having the maximum frequency of 4-8 MHz. The Spec node can communicate only over short distances and it is not capable of receiving data. Also, Spec has very limited performance with radio bandwidth less than 50 kbps.

The most widely used hardware platforms for wireless sensor networks are the generic platforms built entirely from the COTS components. The most popular generic hardware platforms for sensor

applications are the family of Berkley motes, originally envisioned in the SmartDust project [108]. The early versions of Berkley motes [109] used a small 8-bit microcontroller (4 to 8 kB of flash, 512 bytes of RAM), a simple radio (OOK modulation at 4 kbps) and integrated sensors (magnetometers, accelerometers, temperature, pressure, etc.). Later designs (WeC [110], Rene, Rene2, and Dot) introduced the possibility of remote reprogramming and expansion slots for sensor interfacing. Mica [111] was the first general purpose sensor node platform from Berkley that was widely used in WSN research. It offered more memory, analogue and digital sensor interfaces, but it suffered from insufficient processing capabilities, short range radio and high idle power making it inappropriate for deployment. Its follower, Mica2 mote, solved many of the problems of its precedent establishing itself as the most popular sensor platform at the time. It is based on the Atmel's ATmega128 8-bit microcontroller consuming 60 mW in active state and 0.036 mW when sleeping. The radio device in Mica2 node is Chipcon CC1000 offering tuneable frequencies from 300 to 900 MHz and FSK modulation resilient to noise. Mica 2 platform, as well as its successor Mica-Z, was commercialized by the Crossbow Technologies. Mica-Z mote replaced the CC1000 radio with Zigbee-compliant Chipcon CC2420 supporting DSSS-O-QPSK modulation and data rates of up to 250 kbps when running at 2.4 GHz. The latest in the family of Berkley motes is Telos [112], which replaced the 8-bit microprocessor of Mica with more power-efficient 16-bit Texas Instruments MSP430 microcontroller. The radio on Telos node is also CC2420. The commercial version of the Telos node platform, called TelosB, includes a USB port for programming. A number of other sensor node platforms exist that are fairly similar to Berkley motes. For example, Tmote Sky from Motelv Corporation shares the same architecture with TelosB [113]. The more advanced low-bandwidth sensor platforms use a 32-bit processor core and more on-board memory. Sun Spot combines the powerful 32-bit ARM920T processor having 512 kB RAM and 4 MB of Flash with an IEEE 802.15.4 compatible radio. The Sun node includes different sensors and a USB port for programming. The node can be used in low-throughput applications requiring high processing power, such as security applications. Sun Spot relies on Java-based software that makes the node power inefficient as long as the code for it relies on Java virtual machine. A sensor platform developed by Intel, iMote2, includes powerful Xscale PXA271 processor with DVFS capability. iMote2 includes CC2420 radio to support high-performance, low-bandwidth applications at low power. The processor in iMote2 is running on frequencies up to 416 MHz, and it is supported by 32 MB of Flash and 32 MB of SDRAM integrated on chip.

The sensor node platforms based on an IEEE 802.15.4 radio device (such as CC2420) fail to handle data coming from complex sensors (audio, video, etc.). iMote, a sensor node developed by Intel Research, is designed for high-bandwidth sensing applications. The iMote node is based on a 32-bit processor core (ARM7) and consumes up to 60 mA at 3 V when active. iMote uses Bluetooth radio to support high-bandwidth applications. The application of both iMote and iMote2 nodes is presented in [114]. BT Node [115], having similar architecture as Mica-2, combines two radio devices on board, CC1000 for low-bandwidth applications and Bluetooth for high-bandwidth applications.

The Stargate platform is designed to serve as a gateway node for sensor network applications. The Stargate platform consists of two boards, one containing processor (Xscale PXA255), memory (64 MB of SDRAM, 32 MB of Flash) and communication slots (PCMCIA, Flash, SSP via Mica2 Connector), and the other, a daughter board, having additional communication interfaces (Ethernet, RS232, JTAG, USB). The platform is running under embedded Linux software and can be used for both high-bandwidth sensing and data aggregation. It uses a serial connection to access sensor network and provides bridge to wide-area networks through Ethernet or WLAN (PCMCIA). The

gateway functionality for sensor nodes can be also provided by embedded web server platforms such as CerfCube family from Intrinsync or by embedded computers such as those defined by PC104 standard [116]. The overview of the most important general-purpose sensor node platforms is given in Table 2-4.

**Table 2-4.** Overview of general-purpose sensor node platforms.

Platform	CPU	Radio	Data Bandwidth	Max Freq. (MHz)	Memory	CPU Power active sleep
<b>Spec</b>	8-bit custom	integrated transmitter	< 50 kbps	4	3kB SRAM	3mW@1.8V 3μW@1.8V
<b>Mica</b>	8-bit Atmel ATMega 103L	RF Monolithics TR1000	< 50 kbps	4	4kB SRAM 128kB Flash	16.5mW@3V 3μW@3V
<b>Mica2</b>	8-bit Atmel ATMega128L	Chipcon CC1000	< 76 kbps	7.3	4kB SRAM 128kB Flash	24mW@3V 45μW@3V
<b>Mica-Z</b>	8-bit Atmel ATMega128L	Chipcon CC2420	< 250 kbps	7.3	4kB SRAM 128k Flash	24mW@3V 45μW@3V
<b>TelosB</b>	16-bit TI MSP430F1611	Chipcon CC2420	< 250 kbps	8	10kB SRAM 46kB Flash	12mW@3V 1.5μW@3V
<b>SunSpot<sup>1</sup></b>	32-bit ARM920T	IEEE 802.15.4	< 250 kbps	180	512kB SRAM 4MB Flash	43.92mW@1.8V 936μW@1.8V
<b>iMote</b>	32-bit ARM7 TDMI	Bluetooth 1.1	< 500 kbps	48	64kB SRAM 512kB Flash	180mW@3V 300μW@3V
<b>iMote2</b>	32-bit Intel Xscale PXA271	CC2420	< 250 kbps	416	32MB SDRAM 32MB Flash	tens to hundreds mW ~1mW
<b>BTNode</b>	8-bit Atmel ATMega 128L	CC1000 + Bluetooth	< 500 kbps	7.3	4kB SRAM 4kB EEPROM 128kB Flash	24mW@3V 45μW@3V

<sup>1</sup>Power values are related to VDDCORE

### 2.5.2. Embedded Sensor Node Processors

The generic sensor node platforms offer good flexibility and software support and are successfully used by research community to develop applications and communication protocols for wireless sensor networks. However, in most cases, generic nodes cannot fully answer specific application requirements for performance and power. Also, the size and the cost of generic sensor nodes prevent their use in many application fields. To overcome those limitations, many researchers are focusing their work on developing novel architectures for embedded sensor node processors. The efforts are based on non-standard system and circuit techniques such as asynchronous design and subthreshold logic as well as on the application of advanced power saving methodologies and hardware accelerators in standard processor architectures.

One of the first embedded sensor node processors is designed for Spec platform [107]. Spec processor includes 3 kB of SRAM, an 8-bit RISC core (Atmel ATmega128 compatible), an UART and an SPI, a number of communication hardware accelerators, a 900 MHz transmitter and an AD converter. The communication accelerators support synchronisation, timing extraction, simple encryption and

data serialization. The chip is designed in 0.25  $\mu\text{m}$  process consuming 10-15 mA at 1.8 V running at 4 MHz and around 1  $\mu\text{W}$  in the sleep mode. Another simple processor for sensor applications is designed within the SmartDust project [117]. The processor is only 0.38  $\text{mm}^2$  in size consuming 5.9  $\mu\text{W}$  from a 1 V supply when running at 500 kHz. The architecture of an asynchronous sensor node processor called SNAP/LE is presented in [118]. SNAP/LE is based on an asynchronous 16-bit RISC core having its instruction set optimized for sensor-network applications. It includes hardware event queue and an event coprocessor that allows fast execution and helps reducing the overhead introduced by operating system (such as task scheduling and interrupt servicing). To enable synchronisation and timer functions, SNAP/LE includes a synchronous timer coprocessor consisting of three 24-bit timers. The communication with sensors and radio device is provided by the message coprocessor. The absence of complex clock module allows the processor to switch fast between active and sleep modes. The SNAP/LE architecture is event-based to some extent, but its computation engine is still a general-purpose microcontroller that remains active all the time. The implementation of SNAP/LE processor in 180 nm process claims the power efficiency of 24 pJ/instruction. The maximum reported performance is 28 MIPS when operating at 0.6 V. At 1.8 V, the power efficiency is 218 pJ/instruction, and the performance is 240 MIPS. It has to be noted that the power figures for SNAP/LE are extracted for a simple *blink* task. Although it promises low power operation and superior performance, SNAP/LE has been never built in silicon leaving many questions open, such as programmability, real-time performance, reliability, design complexity, etc. The later version of SNAP architecture called BitSNAP trades off the performance for extra power by replacing the parallel datapath of SNAP architecture with a bit-serial one [119]. The BitSNAP applies dynamic significance compression to the bit-serial datapath making it a length-adaptive datapath processor. However, the energy efficiency of such architecture when performing complex tasks is questionable. A group of authors from Harvard University proposed an event-based sensor node processor architecture that applies a number of hardware accelerators for handling of sensor network specific tasks [120]. The architecture includes a message coprocessor for data packet handling, timer subsystem consisting of four 16-bit timers, data filtering block and interfaces to radio and ADC. The interface between hardware accelerators and memory is provided by programmable event processor that has the functionality of a DMA engine. However, not all of the events can be handled by the event processor, so an additional microcontroller is present to handle so-called irregular events. The difference between event-based and time-based processor architectures is that in an event-based architecture, the event scheduler is a separate processing unit, usually a state-machine, which schedules the regular tasks to peripherals (accelerators, I/O). The irregular tasks that are not maintainable by accelerators are passed to a simple backup microcontroller unit. For some simple applications this might be an advantage, but if the processing requires an execution of many irregular tasks, then the scheduler might be a bottleneck. The implementation of the Harvard event-based processor is presented in [121]. The chip has been produced in a 130 nm process having nine different power domains and custom power gating elements for each domain. The control mechanism is implemented as part of the event processor. The chip size is 2 mm x 2 mm, and it runs with the maximum frequency of 12 MHz at 0.55 V. It integrates an open-source 8-bit Z80-based microcontroller and 4 kB of RAM. To underline the processor superiority to others, the authors extracted equivalent energy per instruction (0.44 pJ/instruction) by measuring the energy required to execute a simple *Sense and Transmit* task. However, the real impression of the processor performance and power efficiency can only be extracted by measuring the energy per task extracted for a real-life application. Charm is a sensor node protocol processor that implements a simple 8051-compatible microcontroller and hardware accelerators on chip. The hardware accelerators support

baseband and link layer functionality required by On-Off-Keyed (OOK) modulation [122]. The processor integrates a clock oscillator and power gating mechanism that can reduce the voltage of two power domains to the retention level (0.3 V - 0.5 V). The chip is designed in 130 nm process and has average power consumption of 150  $\mu$ W at 1 V and 8 MHz.

**Table 2-5.** Overview of embedded sensor node processors.

Processor	Architecture	Supply Frequency Memory	Process	Size	Accelerators and Peripherals	Extra Features
<b>Spec</b>	8-bit	1.8 V 4-8 MHz 3kB SRAM	250 nm	6.25 mm <sup>2</sup>	ADC, DMA, LFRS, RF transm.	n.a.
<b>SmartDust</b>	8-bit	1 V 500 kHz 3kB SRAM	250 nm	0.38 mm <sup>2</sup>	integrated oscillators	n.a.
<b>SNAP/LE</b>	16-bit asynchronous	0.6-1.8 V 23-200 MIPS 8kB SRAM	180 nm	n.a.	timer, message coprocessor	n.a.
<b>BitSNAP</b>	16-bit asynchronous bit-serial datapath	0.6-1.8 V 6-54 MIPS 8kB SRAM	180 nm	n.a.	timer, message coprocessor	compression
<b>Harvard</b>	8-bit event-driven	0.55-1.2 V 12.5 MHz 4kB SRAM	130 nm	4 mm <sup>2</sup>	timer, filter, message, event	power gating
<b>Subliminal</b>	8-bit subthreshold	0.2 V 833 kHz 2kB custom	130 nm	29817 $\mu$ m <sup>2</sup> core 55206 $\mu$ m <sup>2</sup> mem	n.a.	low-voltage
<b>Charm</b>	8-bit protocol processor	1 V 8 MHz 68kB SRAM	130 nm	7.29 mm <sup>2</sup>	oscillator, protocol	power scaling, retention
<b>Phoenix</b>	8-bit near-threshold	0.5 V 106 KHz 0.34kB RAM	180 nm	837225 $\mu$ m <sup>2</sup>	temp sensor, timers	low-voltage, compression, power gating, multi-V <sub>th</sub>

One of the first sensor node processors designed to run in subthreshold regime is Subliminal Processor from Michigan University [123], [124]. The processor operates in voltage range from 200 mV - 1.2 V, and it has maximum energy efficiency of 2.6pJ/instruction when running at 360 mV and 833 kHz. The processor is designed in triple-well 130 nm process supporting active bias control. The subthreshold operation of Subliminal Processor required the voltage and frequency of each die to be adjusted for the optimal energy efficiency operating point. Also, the 2 kB SRAM block had to be custom designed (mux-based) in order to reliably operate at subthreshold voltage. During design, all logic gates with fan-in higher than 2 had to be discarded from the library and the library itself had to be characterized for subthreshold operating corners. The measurements on the Subliminal processor show significant delay variability when operating in subthreshold regime. Phoenix is a simple 8-bit sensor node processor designed in a multi-V<sub>th</sub> 180 nm technology process [125]. The processor is

designed for low duty cycle sensor applications. It operates at near-threshold voltage of 0.5 V and consumes 297 nW at 106 kHz. The processor implements power gating and custom built SRAM blocks with retention functionality. The implemented SRAM cells were 9.1 times larger than standard six-transistor (6T) SRAM implementation claiming to deliver 62 times reduction in standby power. However, the write time of implemented SRAMs is quite slow due to the usage of high- $V_{th}$  devices. To increase the capacity of the built-in data RAM, the Phoenix processor includes a simple data compression block that could virtually improve the memory capacity up to 7.6 times. The embedded sensor node processors are summarized in Table 2-5.

### 2.5.3. Overview

The hardware solutions for sensor network applications are ranging from commodity-based general purpose sensor node platforms to custom low-power sensor node processors. The general purpose platforms are well suited for research activities in application and software development, but when it comes to large-scale deployments, they are quite limited in terms of size, power and performance. For specific application, sensor hardware has to provide sufficient computational resources and enough communication bandwidth. For small networks based on simple *sense and transmit* tasks, the tag assets based on simple 8-bit microcontrollers and simple low-bandwidth radios (< 50 kbps) might be sufficient (Spec, Mica). Often, the acquired data have to be processed before transmission. In that case, the processors having rich instruction set (Mica 2) are more efficient than those having simple instruction set. The medium bandwidth applications (< 250 kbps) require radio modules supporting more complex modulation schemes (QPSK) and frequencies such as those defined by the IEEE 802.15.4 standard (CC2420). The communication processing of such radio devices might be performed either in hardware or in software. Usually, the baseband processing and some time consuming functions of upper protocol layers are often integrated in the radio hardware. The software implementation of communication protocols requires sufficient memory resources and a capable processor. Advanced 8-bit microcontrollers with extended address bus (Atmel ATmega128L in MicaZ) or 16-bit microcontrollers (TI MSP430 in Telos) are able to provide the required functionality. To successfully transmit high-bandwidth data (audio, video, acoustic), a high-bandwidth radio such as Bluetooth is required (iMote). For more complex applications that require massive data processing, a 32-bit microcontroller is required (iMote2, SunSpot).

The application of advanced power saving methodologies and hardware accelerators in embedded sensor node processors may significantly improve the performance and power of sensor node systems. Power gating has been recognized as a suitable power-saving technique for low duty cycle sensor nodes [120], [122], [125]. Some of the contributions of this work discussed the issues related to the implementation of power gating in sensor node hardware systems [126], [127]. DVFS is also a promising power-saving technique, but it requires significant computation and data storage resources that are usually not available in most sensor node architectures, especially those relying on 8- and 16-bit microcontrollers [128]. Additionally, the implementation of DVFS requires a real-time operating system capable to support predictive task scheduling. Sensor node systems based on 32-bit microcontroller architecture are good candidates for DVFS implementation [129].

The existing event-based processors employ hardware accelerators that can process some sensor specific tasks more efficiently than a general-purpose microcontroller. The control in such systems is

provided by a dedicated event handler. However, the applicability of the presented event-based processors is defined by the frequency of so-called regular tasks that can be efficiently performed in implemented hardware accelerators. More complex operations would require frequent passing of irregular tasks to a simple back-up microcontroller that may prove as inefficient. The use of asynchronous logic in sensor processor design might simplify the clocking related issues, but it also introduces significant design overhead. The implementation of asynchronous logic is not straightforward and therefore, it is not supported by any industrial design flow. Additionally, the implementation of advanced leakage saving techniques might be difficult with asynchronous design. The processors running near or below threshold voltage promise ultra-low power operation, but their performance is greatly sacrificed for power. Additionally, sub-threshold logic is strongly dependant on PVT variations, and it requires implementation of specialized logic cells and memory blocks to provide correct operation in subthreshold regime. Also, the interfacing of subthreshold logic to common I/O and radio devices might be difficult.

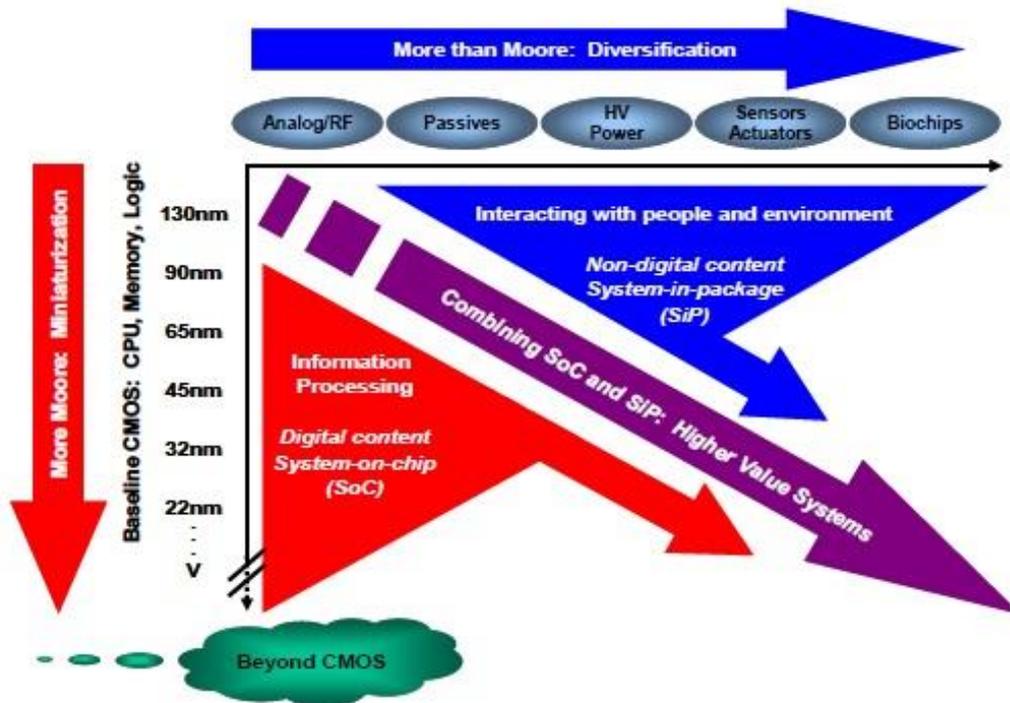
The application of advanced low power techniques in standard processor architectures combined with the application of hardware accelerators for specific WSN tasks might be the best compromise between performance, power and applicability. The Charm processor is an example of a system that applies advanced power saving methodology in standard 8-bit processor architecture enhanced with a communication accelerator. The application of hardware acceleration is also demonstrated by the Spec processor.

The design decisions leading to the implementation of specific sensor node hardware are often driven by designer experience and general system requirements. In published literature, very little or no attention is given analysing the impact of various low power hardware implementations in contrast to the energy efficiency of specific application scenarios. This problem is to be addressed by this work.

## **2.6. Future Trends in Low Power Design**

Far back in 1965, Gordon Moore published a paper, in which he stated that the number of components that could be incorporated per integrated circuit would double in approximately every two years [173]. Moore's observation, also called Moore's law, has proven to be accurate and is used by the semiconductor industry to guide long-term planning and to set targets for research and development. The Moore's law observed the trends in CMOS digital logic, e.g., processor, logic and memories. However, the ITRS predicts that many microelectronic products will have non-digital functionalities in the future (RF components, sensors, passives, etc.). This diversification of functional devices that contributes to the device miniaturization but not necessarily scales at the same rate as logic is designated as "More-than-Moore" trend [174]. The trend of the increase in performance based on device miniaturization is expect to continue, while performance can always be traded against power depending on the individual application ("More Moore" scaling).

The sensor node design is likely to follow the "More-than-Moore" trend. By combining non-digital functionalities to provide an interaction with users and environment and digital content on a single chip, the miniaturization trend of sensor node chips will lead to compact solutions driven by performance and power constraints.



**Figure 34.** The combined need for digital and non-digital functionalities in an integrated system is translated as a dual trend in the International Technology Roadmap for Semiconductors: miniaturization of the digital functions (“More Moore”) and functional diversification (“More-than-Moore”) [174].

The recent trends in process, device and material technology predict wider use of the emerging multi-gate and SOI technologies in future low power SoC designs. The trends predict superlinearly growing gap between available processing performance and processing requirements. This gap can potentially be solved by increasing the number of processing elements in the chip, subject to power and design effort constraints. Potential solutions include high-level hardware/software co-partitioning as well as automated interface technology from high-level design stages to implementation design stages (e.g., high-level synthesis).

Even though the future technologies will rely on high performance low power transistors, power will remain the critical challenge in design of SoC portable consumer chips. The power trend predictions (see Figure 2) and the global quest for “green” and energy-efficient products will lead to a power-centric design of future SoC devices. Some of the advanced low power techniques developed for the planar CMOS technologies dominated by leakage will have their impact in future technologies as well (power gating, DVFS, adaptive body biasing) [172]. However, future design innovations to reduce power will be more concentrated on the methods that apply at high level system design. The evolving role of system level design in overall system power minimization is illustrated in Figure 35.

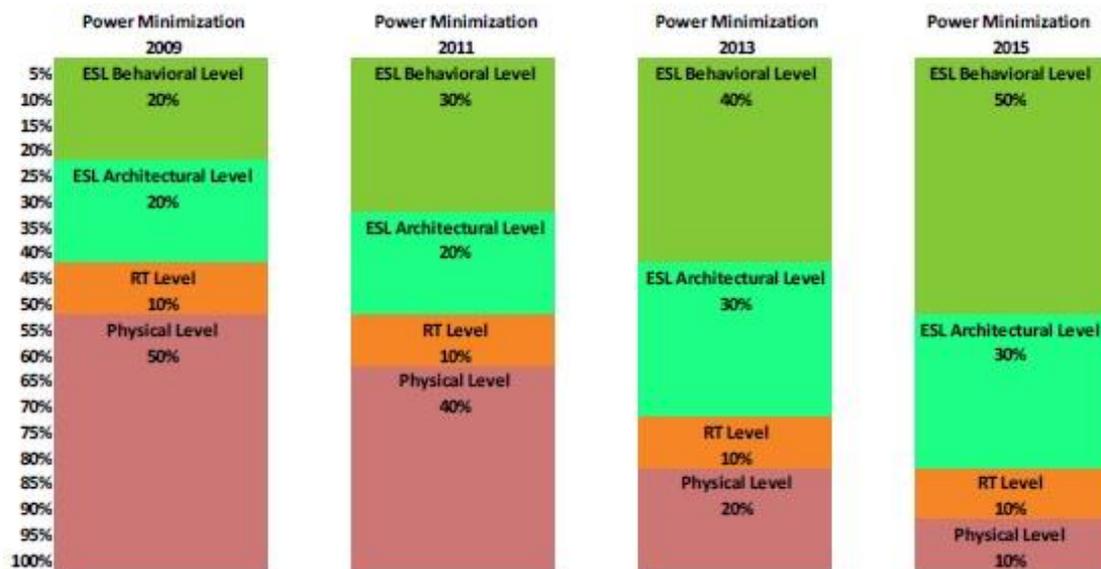


Figure 35. Evolving role of design phases in overall system power minimization [175].

According to ITRS, some of the most important design innovations expected to impact dynamic and static power in future designs are related to behavioral and architecture levels (Table 2-6).

Table 2-6. Low-power design technology improvements and impact on dynamic and static power [172].

Design Technology Improvement	Year	Dynamic Power Improvement (x)	Static Power Improvement (x)	Description of Improvements
Software Virtual Prototype	2011	1.23	1.20	Virtualization tools to allow the programmer to develop software prior to silicon
Frequency Islands	2013	1.26	1.00	Designing blocks that operate at different frequencies
Near-Threshold Computing	2015	1.23	0.80	Lowering $V_{dd}$ to 400 – 500 mV
Hardware/Software Co-Partitioning	2017	1.18	1.00	Hardware/software partitioning at the behavioral level based on power
Heterogeneous Parallel Processing	2019	1.18	1.00	Using multiple types of processors in a parallel computing architecture
Many Core Software Development Tools	2021	1.20	1.00	Using multiple types of processors in a parallel computing architecture
Power-Aware Software	2023	1.21	1.00	Developing software using power consumption as parameter
Asynchronous Design	2025	1.21	1.00	Non-clock driven design

The existing and future trends in low power design show that efficient power management is greatly determined by design decisions made in early phase of system design. Thus, the selection of power saving strategy becomes one of the most important challenges for designers. The related work in advanced low power design usually addresses a specific implementation problem without making connection to the system perspective and the impact of design decisions to system power. Also, the issues related to energy overhead of specific low power implementations are considered as isolated problems. At the other hand, the advances in embedded sensor node microcontroller design search to minimize power by reducing supply voltage close to threshold level or to apply some of the advanced low power techniques. However, none of the published works investigates the impact of the applied power saving strategies to the system power. Furthermore, none of the existing embedded sensor node designs is demonstrated in real-life application.

The methodology proposed in this work looks forward to future and approaches power early in design process. The methodology relies on the extraction of activity profiles of system components applied in the developed models for system energy estimation. The models account for the energy overhead introduced by specific low power implementation. The methodology is used to find an optimal power saving strategy for design. It is applied in the design of a sensor node microcontroller. The methodology is described in the next chapter.



### 3. A Methodology for Choosing Optimal Power Saving Strategy in SoC Design

#### 3.1. Methodology Overview

A SoC power saving strategy defines techniques and methods for power management in a power-constrained SoC design. In an advanced low power implementation, the multi-voltage and power-gating techniques are applied to the system. The advanced low power techniques rely on the concept of power domains that embrace selected system partitions. These system partitions usually comprise functional block units with similar design and functional properties. The identification of low power blocks, i.e., the blocks a specific power saving technique should apply to, is a critical step of particular low power implementation. For an early analysis of specific power saving strategy, it is required to identify potential low power block candidates and select those for which the low power implementation contributes the most to the reduction of system energy.

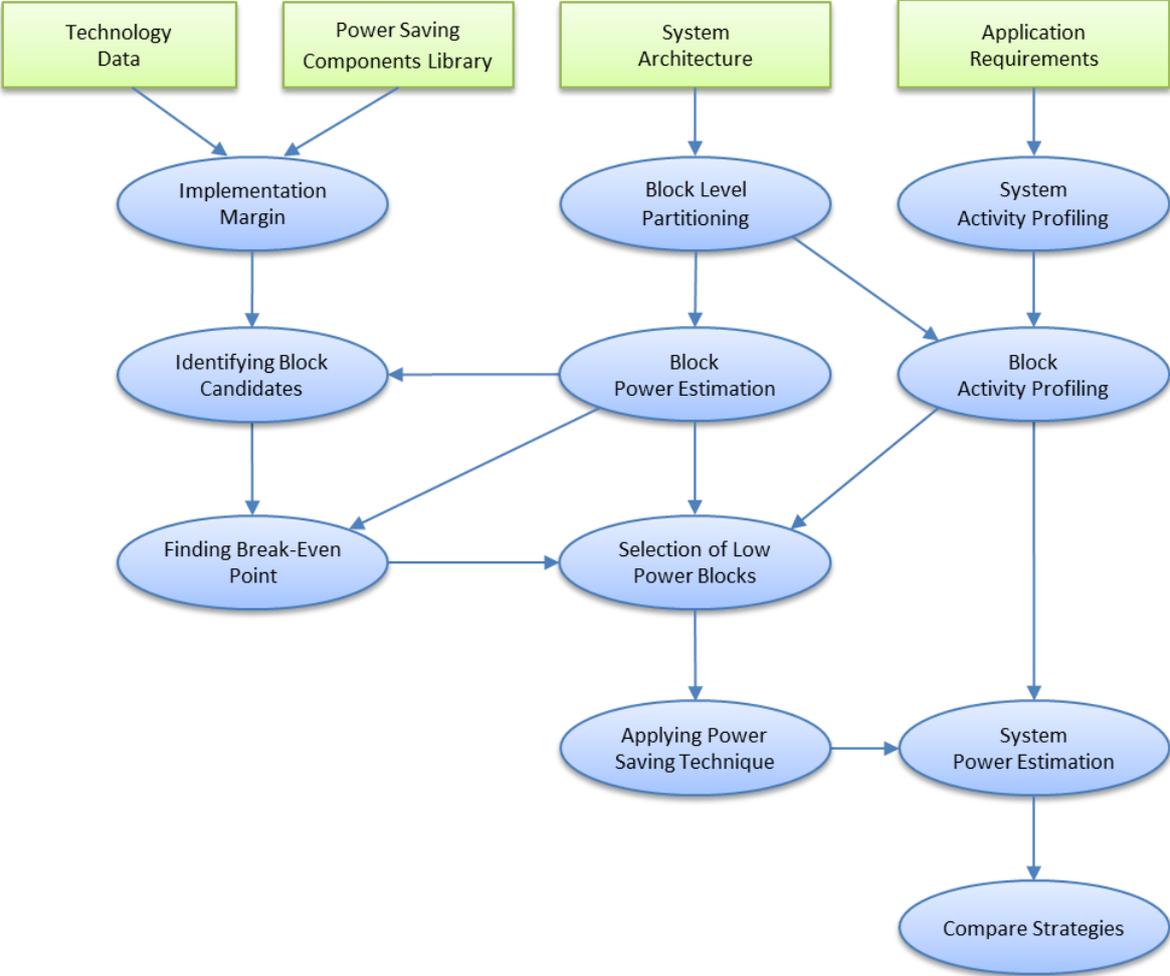


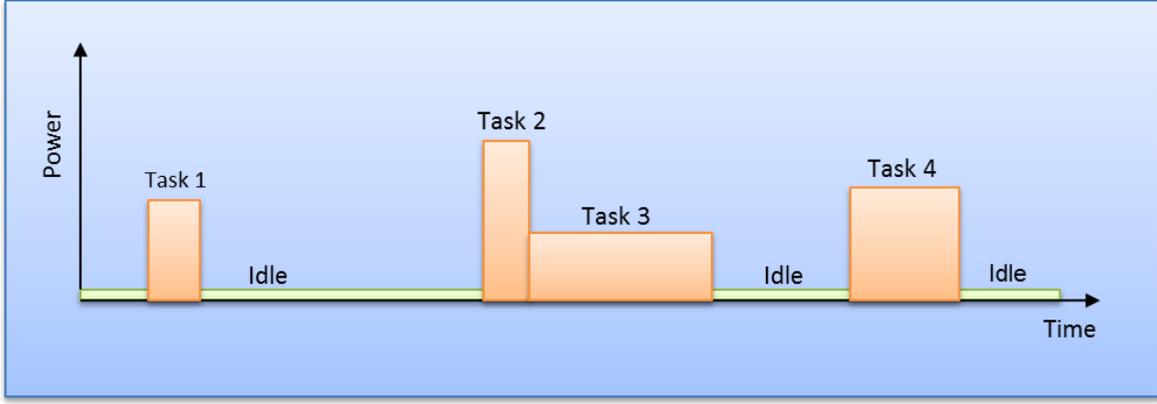
Figure 36. Methodology flow for power strategy analysis.

Figure 36 summarizes the steps to estimate the impact of particular power saving strategy when applied to the target system. The methodology is applied for all considered power saving techniques separately and the results are analysed and compared in order to find an optimal approach. The methodology considers the application requirements, the technology-driven implementation limits and the information from high-level system simulations, in order to find an optimal power saving strategy for design.

The methodology flow takes the technology data and the characterized library of specific power saving components as an input. Additionally, the system architecture and the application requirements must be specified as well. If specific power component (power switch, level shifter, power control logic, etc.) is not in the library, it has to be designed and characterized. The characterized data of power components are used to calculate the lower boundary of the implementation overhead, i.e., the implementation margin. The blocks having their power lower than the implementation margin are discarded as potential block candidates for particular low power implementation. The system partitioning is driven by the concept of voltage islands. Usually, the functional blocks, controllers, IO peripherals, processing units, etc. are considered as low power implementation candidates. Following the application requirements, the system activity is characterized and the activity profile is mapped to the activity of partitioned system blocks. The models for dynamic and leakage power estimation are applied to each block to estimate the power. The power results are compared to the established implementation margin, in order to identify block candidates with power saving potential. The break-even point, i.e., the point where specific low power implementation starts to benefit power, is calculated for each selected block. Analysing the block activity and the given break-even point for a block, the final selection of low power blocks is made. The power saving approach is applied to the system, and the energy efficiency of the implementation is estimated. The results are compared to alternative implementations in order to find the one with better energy efficiency.

### **3.2. Power Estimation Models**

During operation time, the workload of a system-on-chip is not distributed evenly. In a typical scenario, activity periods are usually followed by idle periods. Each task performed during the active operation time requires different amount of energy to complete. The energy represents the integral of power over the time. If the power consumption during processing of a single task is approximated with an average power consumption that is a constant value, then a simplified workload in the terms of power looks as in Figure 37. The areas coloured in tan represent the amount of energy needed for the completion of each task. It has to be noticed that the short tasks requiring high power make less impact on the battery life than the long tasks operating at low power. If the tasks represent different system operation modes, then the time a system spends in each of its operation modes determines the battery life. In other words, the total energy required for the completion of different system tasks during the lifetime is what determines the energy efficiency of a system. Therefore, the goal of the power saving strategy is to improve the system's energy efficiency.



**Figure 37.** Simplified workload in typical SoC.

The choice of power saving strategy is a critical step of system design. To estimate the impact of different power saving strategies, it is required to develop power estimation models, which will be used to calculate the system power in the terms of energy.

If a system consists of  $n$  components ( $i = 1$  to  $n$ ), then its power at the time  $t$  can be expressed as a sum of the power contributions of system components:

$$P(t) = \sum_i P_i(t) \quad (3.1)$$

Where  $P_i(t)$  is the power of  $i$ -th component of the system at the time  $t$ . The number of components is determined by the granularity of the analysis to be performed. The analysis can be done at block level, gate level or transistor level. Since common power-saving techniques apply at block level, further on, it is assumed that a system is composed of logical blocks. For a system consisting of  $n$  blocks, the total power consumption over a time period  $T$  can be expressed in the terms of energy as:

$$E = \sum_{i=1}^n \int_0^T P_i(t) dt \quad (3.2)$$

In a digital system, the power of a single component  $P_i$  can be approximated with discrete values, which represent the average power consumption of that component during different system operation modes. If a system operates in  $m$  different operation modes, its total energy can be expressed as a sum of the energy contributions related to each of its operation modes:

$$E = \sum_{j=1}^m \left( \sum_{i=1}^n \int_0^{t_j} P_i(t) dt \right)_j \quad (3.3)$$

Where  $t_j$  represents the total time a system spends in the operation mode  $j$ . If the component  $i$  can operate in  $k$  different power modes, then the total energy of system can be expressed as:

$$E = \sum_{j=1}^m \left( \sum_{i=1}^n P_{i,j} t_j \right), \quad P_{i,j} \in \{P_{i,k}\} \quad (3.4)$$

Where  $P_{i,j}$  is the power of the component  $i$  during the system operation mode  $j$ .  $\{P_{i,k}\}$  is a set of all existing power modes of the component  $i$ .

In a simple case, only two power modes per component exist, the active (on) and the idle (off) mode of operation ( $k = 2$ ). In the active state, the power of a component is approximated with its average dynamic power. In the sleep state, the power of a component is equal to its static power (leakage). Taking these approximations in the expression for the total system energy, Equation (3.2) becomes:

$$\begin{aligned} E &= \sum_{i=1}^n E_i = \sum_{i=1}^n \int_0^T P_i(t) dt = \sum_{i=1}^n \int_0^{t_i^{on}} P_i^{on}(t) dt + \sum_{i=1}^n \int_0^{t_i^{off}} P_i^{off}(t) dt \\ &= \sum_{i=1}^n P_i^{on} t_i^{on} + \sum_{i=1}^n P_i^{off} t_i^{off} = E_{on} + E_{off} \end{aligned} \quad (3.5)$$

Where  $P_i^{on}$  and  $P_i^{off}$  represent the average active and static power of the component  $i$ .  $E_{on}$  and  $E_{off}$  are the total dynamic and static energy of system during *on* and *off* state.  $T = t_i^{on} + t_i^{off}$  is the time until the battery gets empty.

The last expression shows that the total energy of system is equal to the sum of active and static power contributions of all system components during time  $T$ . Therefore, to calculate the system energy, it is required to partition a system to  $n$  components and to calculate the energy of each component  $i$ . By defining the probability for the component  $i$  to be active during a time period  $T$  with a certain duty cycle  $k_i$ , the energy  $E_i$  of the component  $i$  can be expressed as:

$$E_i = P_i^{on} t_i^{on} + P_i^{off} t_i^{off} = k_i P_i^{on} T + (1 - k_i) P_i^{off} T \quad (3.6)$$

Where  $k_i$  ( $0 < k_i < 1$ ) represents the duty cycle of the component  $i$ .

From the last equation it is clear that for the estimation of system energy, it is required to calculate the average active and static power of all system components and their duty cycles. The dynamic and static power is approximated with the expressions presented in Chapter 2.1. The models for dynamic and static power estimation are incorporated in RTL and gate level power estimation tools (see Chapter 2.4.3). A practical approach to extract the terms for power consumption of system components is by performing block-level after-synthesis power analysis. Obtaining the duty cycle coefficients is more complex, and it depends on specific application and the activity profiling methodology. The activity profiling methodology is discussed later.

### 3.3. Implementation Margin

For a specific technology, it is possible to roughly estimate the applicability region of certain power saving method. Each power saving technique introduces the minimum trade-off in power that has to be considered when planning a power saving strategy for design. If the trade-off in specific

scenario is too high, then the power saving strategy should be reconsidered. The identification of parameters that affect the implementation of specific power saving technique to design is critical.

### 3.3.1. Technology Limits

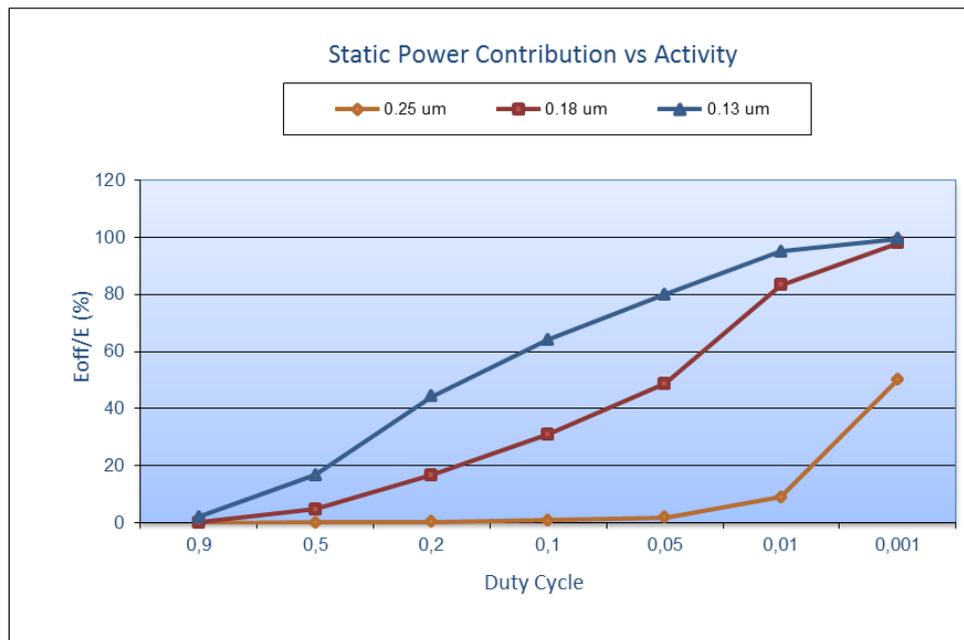
Advanced power-saving techniques differentiate in the aggressiveness when targeting dynamic or static power. At smaller geometries, the static power contribution is significant and has to be maintained more aggressively. At larger geometries, dynamic power dominates. However, the impact of dynamic and static power to system energy efficiency depends strongly on the block activity, i.e., the duty cycle. As already shown, the total energy of digital block as a function of duty cycle  $k$  is given by:

$$E = E^{on} + E^{off} = P^{on}t^{on} + P^{off}t^{off} = k P^{on}T + (1-k)P^{off}T \quad (3.7)$$

From the last equation, the contribution of static energy in the total energy budget of digital block can be expressed as:

$$\frac{E^{off}}{E} = \frac{1}{1 + \frac{k}{1-k} \frac{P^{on}}{P^{off}}} \quad (3.8)$$

In the last equation, the ratio between on-state and off-state power can be roughly approximated with a constant value. By inserting approximated values to the equation, it is possible to calculate static power contribution in specific technology as a function of duty cycle.

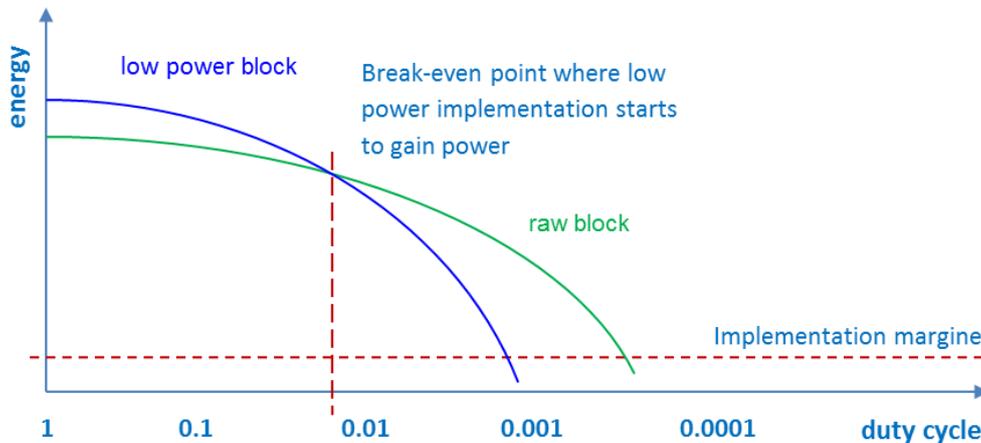


**Figure 38.** Static power contribution as a function of duty cycle.

The observation shows that even for large scale geometries, if duty cycle is low enough, the contribution of static power in the total power budget becomes significant. For example, in a 0.25  $\mu\text{m}$  technology process, the contribution of static power is 50% when duty cycle is 1:1000 (Figure 38).

### 3.3.2. Break-Even Point

For each technology node, it is possible to estimate the minimum design impact on power when implementing specific power saving technique. The implementation cost is related to the energy overhead introduced by the implementation of low power components in design. For example, in the case of power gating implementation, the energy overhead is contributed by power gating controller, isolation logic, and power switches. By finding the minimal energy overhead, it is possible to find the implementation margin for specific low power implementation. A relation to the implementation margin can be determined either for the minimal block size or the minimum leakage or active power of a block, at which specific low power implementation has potential to benefit power. If the energy overhead defined by the implementation margin is higher than expected power savings for a block, the block is discarded from consideration. For blocks with expected power savings higher than that of the overhead logic, it is required to determine expected duty cycles. By knowing the block duty cycle, a break-even point for a block can be found. This is illustrated in Figure 39.



**Figure 39.** Prototyping of low power candidates.

The estimation of the energy overhead for a specific low power implementation requires the estimation of dynamic and leakage power of introduced low power components. In multi-voltage design, power terms for different supply voltages have to be calculated. Also, the power contribution of voltage shifters, voltage regulator and power manager must be taken into account. The DVFS approach is far more complex since the frequency and voltage pairs may change dynamically. This makes the power estimation much more complicated because the voltage scaling depends on typical

task scheduling for each operation task. For a known application, this information can be obtained from high-level simulation models.

### 3.4. Implementation Overhead

The implementation of specific power saving technique will impact the power and activity terms in Equation (3.3), and it will introduce additional trade-off members in the equation. The goal of power saving implementation is to reduce the energy of raw system  $E_{raw}$  to a power efficient level  $E_{lp}$ , so that:

$$E'_{lp} < E_{raw} \quad (3.9)$$

Where  $E'_{lp}$  is the reduced energy of raw system without calculated implementation overhead. However, the implementation of low power techniques requires implementation of additional logic that consumes extra power. If the energy costs due to the implementation overhead are  $E_{oh}$ , then the total energy of system becomes:

$$E_{lp} = E'_{lp} + E_{oh} \quad (3.10)$$

To justify low power implementation, it is required that a system having implemented low power technique has a better energy efficiency than the raw system:

$$E_{lp} < E_{raw} \quad (3.11)$$

In other words, the power gain ( $\Delta E$ ) has to be larger than the introduced overhead. Otherwise, the implementation is not effective:

$$\Delta E = (E_{raw} - E'_{lp}) > E_{oh} \quad (3.12)$$

The accurate estimation of energy overhead is critical for the analysis.

#### 3.4.1. Power Gating Overhead

When implementing power gating on specific component, the off-state power of the component will be dramatically reduced. However, new power terms will be introduced to Equation (3.5) due to the addition of isolation logic, power gating controller and power switches. The energy of a power-gated system is then described as a sum of the energy contributions of  $n$  system components  $E_{sys}$  and the energy overhead  $E_{oh}$  made by additional  $m$  components introduced by power gating implementation:

$$E_{pg} = E_{sys} + E_{oh} = \left( \sum_{i=1}^n P_i^{on} t_i^{on} + \sum_{i=1}^n P_i^{off} t_i^{off} \right) + \left( \sum_{j=1}^m P_j^{on} t_j^{on} + \sum_{j=1}^m P_j^{off} t_j^{off} \right) + E_{tr} \quad (3.13)$$

In the last equation,  $E_{tr}$  is the energy loss during power transitions. In a typical power-gated system with no retention, there are three additional sources of energy consumption ( $m=3$ ): the

energy of power gating controller ( $E_{pgc}$ ), the energy of isolation logic ( $E_{iso}$ ), and the energy of power switches ( $E_{psw}$ ). The total energy overhead of a power-gated system is then given by:

$$\begin{aligned} E_{oh} &= E_{pgc} + E_{iso} + E_{psw} = \\ &= \left( E_{pgc}^{on} + E_{pgc}^{off} + E_{pgc}^{tr} \right) + \left( E_{iso}^{on} + E_{iso}^{off} + E_{iso}^{tr} \right) + \left( E_{psw}^{on} + E_{psw}^{off} + E_{psw}^{tr} \right) \end{aligned} \quad (3.14)$$

Where, the *on* and *off* power terms represent the energy contributions of power gating components in active and idle state, respectively, and the *tr* power terms reflect the energy loss during power transitions. The last equation gives the total energy overhead of power gating implementation related to the energy consumption of all low power components in the system. However, it is more practical to calculate the total energy overhead as a sum of the energy overhead contributions introduced by each power-gated block in the system. In that case, Equation (3.14) becomes:

$$E_{oh} = \sum_{i=1}^k (E_{oh})_i = \sum_{i=1}^k (E_{pgc} + E_{iso} + E_{sw})_i \quad (3.15)$$

Where  $k$  is the total number of power-gated blocks in the system. To calculate the energy overhead per block, it is required to calculate the energy contributions of low power components during active and idle state of block operation as well as the energy loss during power transitions. The transition energy loss is related to the charge of capacitive load in a block during power-up and the energy a controller consumes when initiating power switching operations.

In a typical power-gated system, isolation cells are implemented as combinatorial logic. The active power of isolation cells is approximated with the average dynamic power of all isolation cells  $P_{iso}^{dyn}$ . The static power consumption  $P_{iso}^{stat}$  is equal to the cumulative leakage of all isolation cells. The energy overhead of the isolation logic is expressed as:

$$E_{iso} = E_{iso}^{on} + E_{iso}^{off} + E_{iso}^{tr} = P_{iso}^{dyn} \cdot t_{iso}^{on} + P_{iso}^{stat} \cdot (t_{iso}^{off} + t_{iso}^{tr} + t_{iso}^{on}) \quad (3.16)$$

The power consumption during power transitions is considered to be the same as for the idle mode. The isolation logic is active when a block to which the isolation is applied is active. The switching activity of isolation cells depends on how often the block updates its outputs. When the block is powered-off, the isolation logic is in idle state and contributes to the energy overhead only with its leakage power. The number of isolation cells per block is equal to the number of the block outputs. If the isolation cells are implemented as AND gates and their number is  $n_i$ , then the energy overhead of the isolation can be expressed as:

$$E_{iso} \approx n_i \cdot (P_{and}^{on} \cdot t_{iso}^{on} + P_{and}^{off} \cdot t_{iso}^{off}) \quad (3.17)$$

Where  $P_{and}^{on}$  and  $P_{and}^{off}$  represent the average dynamic and static power of an AND gate, and  $t_{iso}^{off} \gg t_{iso}^{tr}$ .

Power gating controller is usually constructed as an always-on block. The power of power gating controller can be approximated with its average dynamic power  $P_{pgc}^{dyn}$  and its static power  $P_{pgc}^{stat}$ . The

active energy contribution of power gating controller is associated with the initiation of power state transitions in a block. It can be expressed as:

$$E_{pgc}^{tr} = P_{pgc}^{dyn} \cdot t_{pgc}^{tr} = n_{tr} \cdot P_{pgc}^{dyn} \cdot t_{tr} \quad (3.18)$$

Where  $n_{tr}$  is the total number of power transitions per block and  $t_{tr}$  is the time duration of a single transition. The estimation of  $P_{pgc}^{on}$  has to consider a relatively low switching activity factor of power controller. The complexity of power gating controller will increase with the increase in the number of power-gated blocks in a system.

The energy overhead of power switches includes the power consumption in *on* state (a block is powered up), the power consumption when a block is powered-off, and the energy loss during power transitions. In the on-state, the power of a switch is determined by its on-resistance and the average active current for a block. In the off-state, the power of a switch is related to its leakage current. To reduce the leakage, it is desirable for power switches to have as small as possible on-resistance  $R_{on}$  and as large as possible off-resistance  $R_{off}$ . The *on* and *off* power of a power switch is given as:

$$P_{sw}^{on} = I_{on} \cdot (V_{dd} - V_{vddg}) = I_{on}^2 \cdot R_{on} \quad (3.19)$$

$$P_{sw}^{off} = V_{dd} \cdot I_{off} \quad (3.20)$$

Where  $I_{on}$  is the active current through the switch,  $I_{off}$  is the leakage current of switch, and  $V_{vddg}$  is a voltage level on the virtual power rail.  $V_{vddg}$  is slightly lower than  $V_{dd}$  due to IR drop on the switch.

When calculating the transition power of power switches, it is required to consider the related effects during cut-off and wake-up transitions. The typical mode transitions during power gating are illustrated in Figure 40.

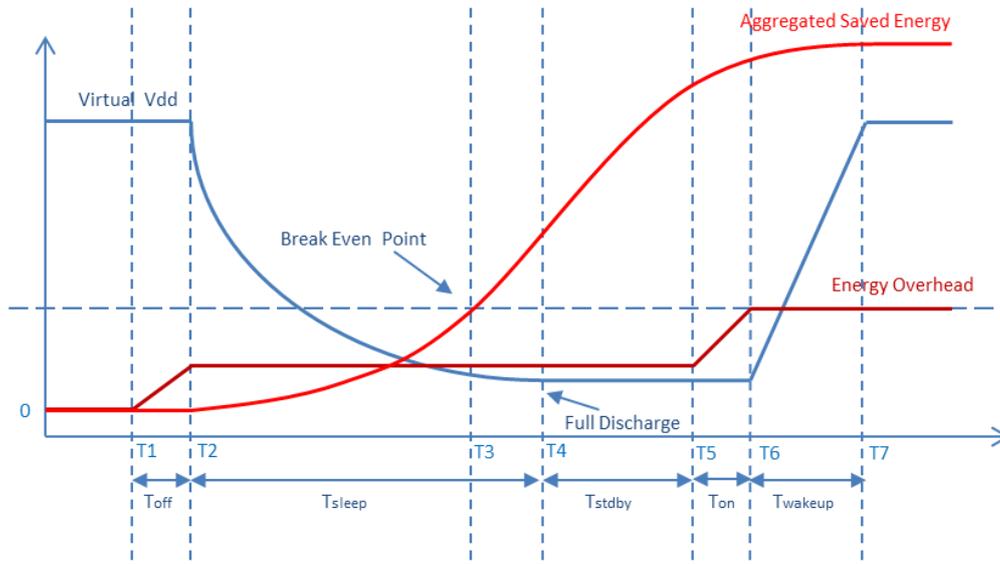


Figure 40. Power-gating mode transitions.

When a block is selected for power gating, the controller initiates a procedure to cut-off the power ( $T_1$ ). During the time  $T_{off} = T_2 - T_1$ , the power control signals are propagated and the power is turned off. The voltage level at the virtual rail starts to drop decreasing the leakage current of the block. After the time  $T_{sleep}$ , the capacitive load in the block gets fully discharged and the energy saving starts to aggregate linearly. At some point of time, the aggregated saved energy exceeds the overhead energy. This break-even point can occur before or after the full discharge, depending on the technology and the block size. When the block is selected for wake-up, the control signals are propagated to power switches and the power is turned on ( $T_{on}$ ). After the time  $T_{wakeup}$ , the voltage level on virtual rail reaches its maximum and the block enters the active mode of operation.

The energy impact of transition power depends on the effective load capacitance of the gated logic. Under the assumption that half of CMOS devices in a block are at logic one before the power is turned on, the transition energy of the power-on transition can be expressed as:

$$E_{sw}^{tr_{on}} = \left( \frac{1}{2} C_S + C_D \right) \cdot V_{dd} \cdot V_{ddg} \quad (3.21)$$

Where  $C_s$  is the total switching capacitance of all transistors and wires in a block,  $C_D$  is the capacitance of the virtual power grid and decoupling capacitors, and  $V_{ddg}$  is the voltage level on virtual rail for which a block is considered to be powered-on. Usually, it is assumed that a block is powered-on when  $V_{ddg}$  reaches 90% of  $V_{dd}$ . During the power-off transition, the charge stored in the internal capacitances is discharged to the ground and no energy is consumed at the source:

$$E_{sw}^{tr_{off}} \approx 0 \quad (3.22)$$

In  $V_{dd}$ -gating, each time the power is turned-on, the potential on virtual rail will rise rapidly causing the transient switching of internal nodes and high in-rush current. This effect generates undesired noise and contributes to the increased IR drop in the active logic. The in-rush current depends on the node capacitance and strength of the driving gate. The in-rush current can be controlled by turning the switches on in groups or one-by-one. The maximum in-rush current  $I_{rush}^{max}$  is determined by the number of power switches  $n_{sw}$  and the saturation current of power switch  $I_{dsat}$ . If all switches in a block switch at the same time, the maximum in-rush current will be:

$$I_{rush}^{max} = n_{sw} \cdot I_{dsat} \quad (3.23)$$

Where the minimal number of the required power switches per block can be estimated as:

$$n_{sw} = \frac{R_{on} \cdot I_{on}^{max}}{\Delta V_{vdd}^{max}} \quad (3.24)$$

In the last equation  $I_{on}^{max}$  represents the maximum power consumption for a block domain and  $\Delta V_{vdd}^{max}$  is the maximum allowed voltage drop.  $R_{on}$  is the switch on-resistance. While the large number of power switches increases the total leakage and area, the insufficient number of power switches increases IR drop and degrades performance. Modern EDA tools allow fast prototyping of the required number of power switches per block.

The turn-on time by the simultaneous switching of all power switches in a block is given as:

$$t_{on} = \frac{\left(\frac{1}{2}C_S + C_D\right) \cdot 0.9V_{dd}}{I_{rush}^{max}} \quad (3.25)$$

The insertion of delay between power switch cells will increase the turn-on time, but it will also reduce the maximum in-rush current. By using the estimated turn-on time from the last equation, the transition energy overhead can be expressed as a function of the average transition power consumption of a switch:

$$E_{sw}^{tr} = n_{sw} \cdot P_{sw}^{tr} \cdot t_{on} \quad (3.26)$$

For a delayed buffer chain, it is required to find an average turn-on time per switch. If a buffer delay in the switch is  $\Delta t$ , then the electrical charge that is transferred from the source during the turn-on time  $t'_{on}$  is:

$$Q = C_T \cdot V = \int_0^{t'_{on}} I(t) dt = I_{dsat} \cdot t'_{on} + I_{dsat} \cdot (t'_{on} - \Delta t) + \dots + I_{dsat} \cdot (t'_{on} - (n_{sw} - 1)\Delta t) \quad (3.27)$$

Where  $C_T$  is the total capacitance ( $C_T = 1/2C_S + C_D$ ) and  $V$  is  $0.9V_{dd}$ . From the last equation, the turn-on time for delayed buffer chain can be expressed as:

$$t'_{on} = \frac{\left(\frac{1}{2}C_S + C_D\right) \cdot 0.9V_{dd}}{n_{sw} I_{dsat}} + \frac{n_{sw} - 1}{2} \Delta t = t_{on} + \frac{n_{sw} - 1}{2} \Delta t \quad (3.28)$$

The second term in the last equation represents the increase in turn-on time when the buffer delay  $\Delta t$  is introduced between the switches. The average turn-on time per switch is given by:

$$t'_{on\_av} = \frac{t'_{on} + (t'_{on} - n_{sw}\Delta t)}{2} \quad (3.29)$$

The total energy overhead introduced by a block can be now expressed as a function of the average switch transition power and energy contributions of power gating controller and isolation logic:

$$E_{oh} = n_{sw} \cdot P_{sw}^{tr} \cdot t'_{on\_av} + P_{pgc}^{dyn} \cdot t_{tr} + P_{iso}^{stat} \cdot t_{tr} \quad (3.30)$$

For a known leakage power of a block  $P_{leak}$ , it is now possible to determine the breakeven point as:

$$t_{be} = \frac{E_{oh}}{P_{leak} - P_{oh\_leak}} \quad (3.31)$$

Where  $P_{oh\_leak}$  is the total leakage power of overhead components. The longer a block spends sleeping the larger are the power savings due to power gating.

### 3.4.2. Multi-Voltage Overhead

In static multi-voltage scaling, the energy overhead is determined by the overhead introduced by the voltage source circuit and voltage shifters. If voltage domains operate at different frequencies, then the frequency generator and synchronisation logic make an additional overhead to the system energy. The total energy of a multi- $V_{dd}$  system composed of  $n$  blocks can be expressed as:

$$E = \sum_{i=1}^n (E_i) + E_{ls} + E_{dc} + E_{sync} \quad (3.32)$$

Where  $E_{ls}$  is the total power of all voltage-level shifters,  $E_{dc}$  is the power of an on-chip voltage source, and  $E_{sync}$  is the power of clock synchronisation logic. The power consumption of the voltage generation circuit depends on its architecture. The supply voltage circuit is active all the time and its power can be estimated in analogue simulations. The voltage level shifters have to be inserted on the crossings between voltage domains. There are two types of voltage level shifters, high-to-low and low-to-high shifters. Usually, the level shifters are designed as special library cells. The power of voltage shifters depends on the difference in input and output voltage levels. The larger the voltage swing is the higher is the area and power consumption of voltage shifters. The power impact of synchronizers and frequency dividers can be prototyped by standard trial synthesis methods.

### 3.4.3. DVFS Overhead

In DVFS, the voltage and frequency of specific power domains change dynamically. However, the most DVFS implementations provide the finite number of power states for each domain, determined by voltage/frequency pairs. The energy of a block belonging to a single power domain can be expressed as:

$$E_{pd} = \sum_{i=1}^n E_{pd}^i = \sum_{i=1}^n P_{pd}^i t_{pd}^i = \sum_{i=1}^n (V_{pd}^i)^2 C_{pd} f_{pd}^i \alpha_{pd}^i + \sum_{i=1}^n E_i^{off} \quad (3.33)$$

Where  $V_{pd}^i$  and  $f_{pd}^i$  represent the voltage/frequency pairs for  $i$  ( $i = 1$  to  $n$ ) power states.  $\alpha_{pd}^i$  is duty cycle for each power state.  $E_i^{off}$  is the off-state leakage component for different sleep mode states. Usually, a DVFS system is constructed to have single sleep state mode determined by a single voltage level.

The energy overhead of DVFS is the overhead due to voltage regulation circuit, voltage shifters, power controller, and synchronisation logic.

$$E_{oh} = E_{dc} + E_{ls} + E_{pgc} + E_{sync} \quad (3.34)$$

The voltage regulation circuit has two power components, the power consumption in the quiescent mode and the power of switching. The power efficiency of voltage regulator depends on its architecture and the frequency of switching. In a circuit with multiple power regions that are dynamically scaled, it is required to have one voltage regulator for each domain. Since this can be

fairly inefficient, DVFS is usually implemented to scale the voltage and frequency of a single power domain that usually involves complete chip logic.

The voltage shifters in a DVFS design are inserted at power domain boundaries. Since voltage shifting can introduce disturbance in signal levels, the level shifters for DVFS include isolation logic to keep the signals stable until the voltage level is stabilized. In designs with large voltage swing, these cells can introduce significant power overhead. The voltage shifters are designed and characterized as library cells.

The power control circuit is required to control voltage transitions in DVFS circuit. The controller makes part of a general DVFS scheme, where processor keeps the block signals electrically stable until the voltage/frequency transition is safely completed. The power overhead of control and synchronization logic can be estimated by standard prototyping methods, e.g., by trial synthesis power estimation.

To estimate the power of DVFS design, it is required to extract the behaviour of system activity. Since the power in DVFS systems is usually software controlled, it is required to perform system level simulations of typical task scheduling in a given application in order to profile the system activity. Additionally, to enable early power estimation, the timing libraries have to be characterized for all voltage levels provided by a DVFS implementation.

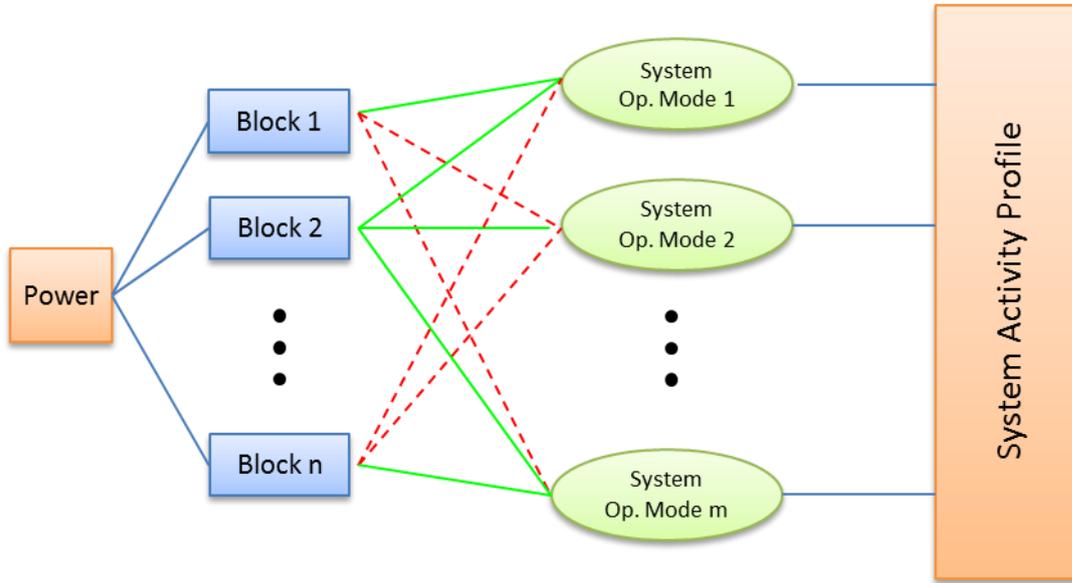
### **3.5. Activity Profiling**

The estimation of energy overhead is critical for finding the break-even points of particular low power block implementations. However, to determine if a specific implementation will benefit power, it is required to find the activity profile of the target system component. The advanced low power techniques apply typically at block level. Therefore, the system activity profiling is supposed to provide the information on specific block activity in a given application scenario. The proposed method for the extraction of required block activities is briefly described.

The profiling method assumes that the designer can predict or simulate the global system behaviour for a given system architecture and application requirements. The basic idea is to describe system behaviour with a finite number of system operation modes and relate specific block activity to each of the system operation modes. The system global behaviour is modelled by a sequence of system tasks, where each task corresponds to a single operation mode of a system. By finding the frequency of specific system tasks for known task completion time, it is possible to calculate the total time a system spends in each of its operation modes, i.e., to profile the system activity. The system blocks are assumed to operate in multiple operation modes, where each block operation mode is determined by the average power consumption of a block, i.e., the power mode of a block. By mapping the system activity to specific block operation modes, it is possible to profile the block activity and estimate the energy efficiency of specific low power implementation.

In a simple case, each block operates in two power modes, active and idle, where the power consumption of the active mode equals to the average dynamic power of a block and the power consumption of the idle mode equals to the leakage power of a block. The activity mapping is

illustrated in Figure 41. The power budget is shared among all system blocks. The green and dashed red lines show if for a specific system operation mode a block is active or not.



**Figure 41.** Simplified mapping of block activity to specific system operation modes.

A formal description of the proposed method describes a system  $N$  as a set of  $n$  functional blocks  $b_i$ , where each block  $b_i$  can operate in  $k_i$  different operation modes  $p_i$ .

$$b_i \in N, \quad N = \{b_1, b_2, \dots, b_n\} \quad (3.35)$$

$$p_i \Rightarrow b_i(p_i), \quad p_i \in P_i, \quad P_i = \{p_i(1), p_i(2), \dots, p_i(k_i)\} \quad (3.36)$$

The system activity  $S$  is described as a sequence of  $n$  system tasks  $t_a$ , where each task represents one of the  $m$  system operation modes  $s_{op}$ .

$$S = (t_a)_{a=1}^n = (t_1, t_2, \dots, t_n), \quad t_a = s_{op}, \quad s_{op} \in S_{op}, \quad S_{op} = \{s_1, s_2, \dots, s_m\}, \quad n \geq m \quad (3.37)$$

A particular block operation mode  $p_{op}$  is mapped to each of the system operation modes  $s_{op}$ .

$$b_i(p_{op}) \Rightarrow s_{op}, \quad p_{op} \in P_i \quad (3.38)$$

Accordingly, the time related to the completion of system task  $s_{op}$  is equal to the time a block  $b_i$  operates in the operation mode  $p_{op}$ .

$$t(s_{op}) = t(b_i(p_{op})), \quad p_{op} \in P_i \quad (3.39)$$

By predicting the system activity related to each of its operation modes, it is possible to extract the total block activity and calculate the energy contribution of each component.

Extracting the system behaviour from a high-level simulation might be a tricky task. Often, the system level simulations are performed at high level of abstraction and they are not time based.

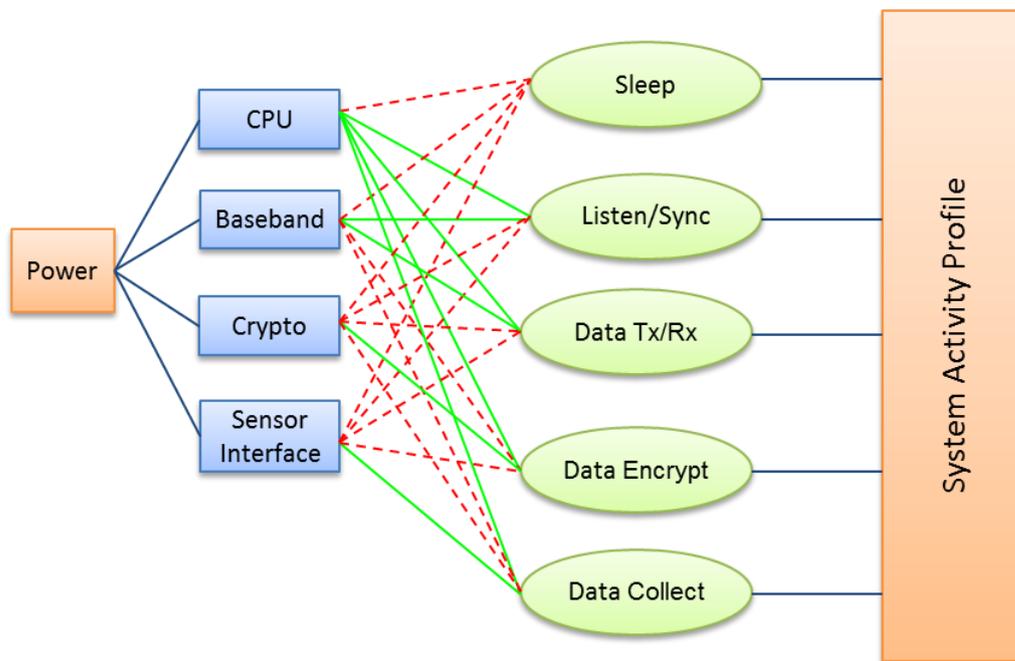
In a time-based simulation the system model includes the information on the required time to complete a single system task. The simulator can estimate the total duration for each of the system tasks during a given simulation time. However, the time-based simulators are sometimes incapable to deal with complex simulation scenarios.

In an event-based simulator, it is not possible to extract the time-related information directly. However, it is possible to extract the frequency of single system tasks. For many applications, the duration of specific system tasks can be predicted or calculated from RTL simulations. By mapping the task frequency to the estimated task duration, it is possible to calculate the system activity profile in time domain and extract the activity of system blocks.

The activity profiling methodology for WSN specific tasks is presented in the next section.

### 3.6. Activity Profiling in Sensor Nodes

The basic architecture of a sensor node consists of three main parts: processing, radio and sensing. In an application-specific architecture, a sensor node microcontroller may include hardware accelerators for communication protocols, a baseband processing unit, crypto cores and different IO peripherals.

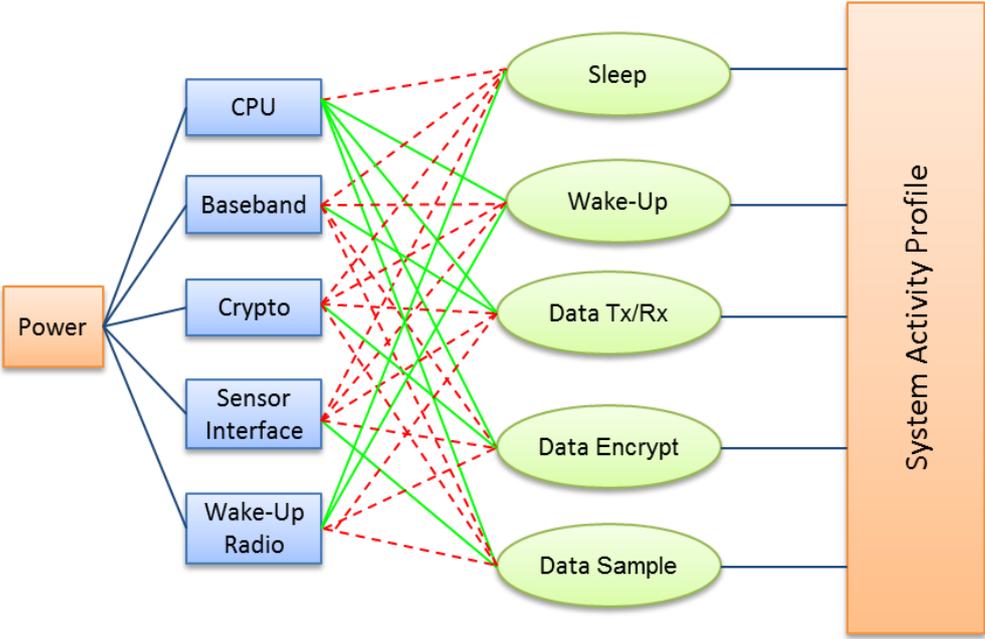


**Figure 42.** Example of sensor node activity mapping in WSN application relying on schedule-based communication protocols.

To profile the activity of different sensor node system components, a sensor node must be partitioned to block candidates relevant for the considered low power implementation. A common

way is to partition the system to its functional logic blocks (processor, peripheral, hardware accelerator, etc.). In some cases, it is possible to go higher or deeper in the hierarchy and estimate power for each partition of interest. Furthermore, it is required to identify relevant operation modes of a sensor node. The selection of operation modes depends on application requirements and network topology. Typical sensor node operations include sleeping, idle listening, data receiving, data transmitting, data encryption, data collection, etc. In a low duty cycle multi-hop network, nodes must synchronize their wake-up times to enable communication. That is, both the sender and the receiver must be awake at the same time, referred to as rendezvous. Wireless sensor network with such schedules is referred as a schedule-based WSN. An example of the sensor node activity mapping in a schedule-based WSN is shown in Figure 42.

In a single-hop network or in a network based on the wake-up radio control, the implicit synchronization between nodes is not required. In the former, the base station or the cluster head listens for potential data all the time. Therefore, nodes can send data to it at any time and do not need extra wake-up synchronization solutions. With wake-up radios, nodes sending a wake-up signal will immediately wake the neighbour and no other synchronization steps are needed. In these both cases, a node wakes up occasionally to sample and transmit the data. The activity mapping in a wake-up controlled sensor node is illustrated in Figure 43.



**Figure 43.** Example of sensor node activity mapping in WSN application relying on wake-up control.

In order to extract the block activity of a sensor node implementation, a specific system activity defined by different system operation modes is mapped to the activity of the selected system components. The extraction of block activities is required to determine if the break-even point defined for a block is reached, and also, to calculate the energy efficiency of applied power saving approach. The application requirements define type of the implemented sensor devices and frequency of data sampling. Accordingly, depending on the target network size and topology,

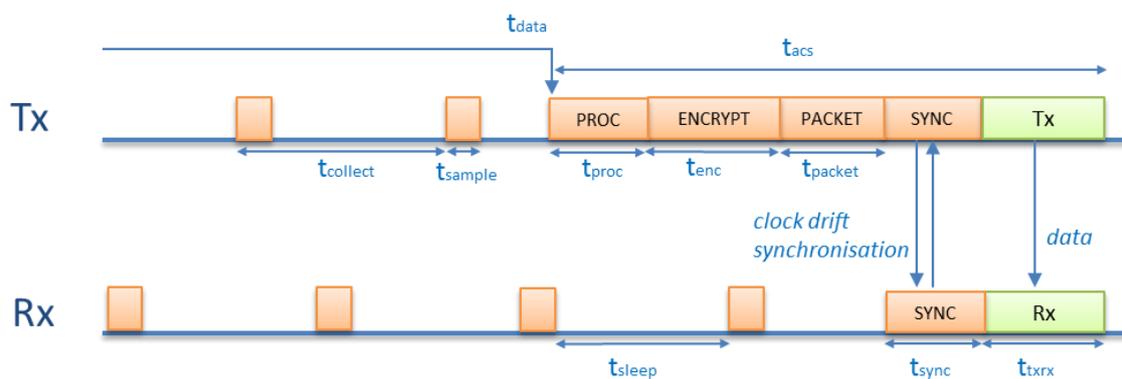
communication architecture is defined as well. The activity is then profiled for the given target application and network environment.

Various network simulators may provide activity profiling for single sensor nodes. As discussed before, in an event-based simulator that is typical for wireless sensor networks, it is possible to profile the activity by modelling the frequency of predefined system tasks during simulation time. Depending on time parameters defined in the target application and implemented communication protocols, it is possible to extract the average time a sensor node spends in each of its operation modes. This information can be used to extract the block activity to be applied in the models for energy estimation. It must be considered that nodes in a network do not have identical workload. Therefore, it is required to extract the activity data for a number of representative nodes and find a relevant activity profile for typical node behaviour. For small-scale networks, if all communication and application parameters are known, the activity can be profiled by analytical methods.

Examples of activity profiling in typical sensor node scenarios are presented.

### 3.6.1. Example of Activity Profiling in Schedule-Based WSN

In a schedule-based WSN application, nodes wake-up periodically to synchronize their clocks or to exchange the data. There are a number of communication protocols that define specific behaviour of a sensor node in a schedule-based WSN (TDMA, CSMA-CA, etc.). Nevertheless, the basic activity pattern of a sensor node is very similar to most of them. Each node in a schedule-based WSN has scheduled activity slots reserved for data transmission and data reception as well as for communication specific tasks. Also, there is a time slot reserved for data sampling that is defined by the application. Depending on the implemented communication protocol, data can be either sampled in the same active phase reserved for data transmission or a node can collect data outside the scheduled timing slot. An example of a schedule-based WSN application is shown in Figure 44.



**Figure 44.** Example of a scheduled-based WSN application.

In the given example, nodes wake-up after a scheduled time  $t_{data}$  to exchange data and synchronize their clocks. A node remains active for an active scheduled time  $t_{acs}$ , afterward it returns

to sleep. In the time between two active phases, a node wakes-up  $n_{sample}$  times to sample data. The time required for a single data sampling is  $t_{sample}$ . The period between two data sampling tasks,  $t_{collect}$ , is defined by the application. During the active scheduled time ( $t_{acs}$ ), the transmit node prepares one or more packet payloads from the collected data ( $t_{proc}$ ), makes the data payload encryption ( $t_{enc}$ ), assembles the packets ( $t_{packet}$ ), exchanges the synchronisation frames with the receiver ( $t_{sync}$ ) and transmits the packet to the receiver ( $t_{txrx}$ ). If it is assumed that the scheduled wake-up time period  $t_{data}$  is much longer than the period between two data samplings ( $t_{data} \gg t_{collect}$ ), all node activity can be derived for the time  $t_{data}$ , where:

$$t_{data} = t'_{sleep} + t'_{active} \quad (3.40)$$

In the last equation,  $t'_{sleep}$  and  $t'_{active}$  are the total sleep and active time of a node during the time  $t_{data}$ , respectively. The node active scheduled time  $t_{acs}$  is given by:

$$t_{acs} = t_{proc} + t_{enc} + t_{packet} + t_{sync} + t_{txrx} \quad (3.41)$$

If the number of forwarded packets during the time  $t_{data}$  is given by  $n_{fw}$  and if it is assumed that the packet forwarding is done within the scheduled timing slot, then the last equation becomes:

$$t_{acs} = t_{proc} + t_{enc} + t_{packet} + (1 + 2n_{fw})(t_{sync} + t_{txrx}) \quad (3.42)$$

In a non-ideal network the packets will be retransmitted with some retransmission rate  $r$  that can be expressed as:

$$r = \frac{n_r(t)}{t} \quad (3.43)$$

Where  $n_r(t)$  is the number of retransmissions during the time  $t$ . When derived to the time  $t_{data}$ , the retransmission rate can be expressed as a function of packet error rate (PER), which represents the percentage of the lost packets due to transmission errors:

$$r' = \frac{PER(\%)}{100} \quad (3.44)$$

The active scheduled time during the time  $t_{data}$  becomes:

$$t'_{acs} = t_{proc} + t_{enc} + t_{packet} + (1 + r')(1 + 2n_{fw})(t_{sync} + t_{txrx}) \quad (3.45)$$

The total active time during the time  $t_{data}$  can be expressed as the total time of the scheduled active phase and data collection:

$$t'_{active} = t'_{acs} + t'_{sample} = n_{sample} \cdot t_{sample} + t_{proc} + t_{enc} + t_{packet} + (1 + r')(1 + 2n_{fw})(t_{sync} + t_{txrx}) \quad (3.46)$$

The total sleep time during the time  $t_{data}$  is now:

$$t'_{sleep} = n_{sample} \cdot t_{sleep} = t_{data} - t'_{active} \quad (3.47)$$

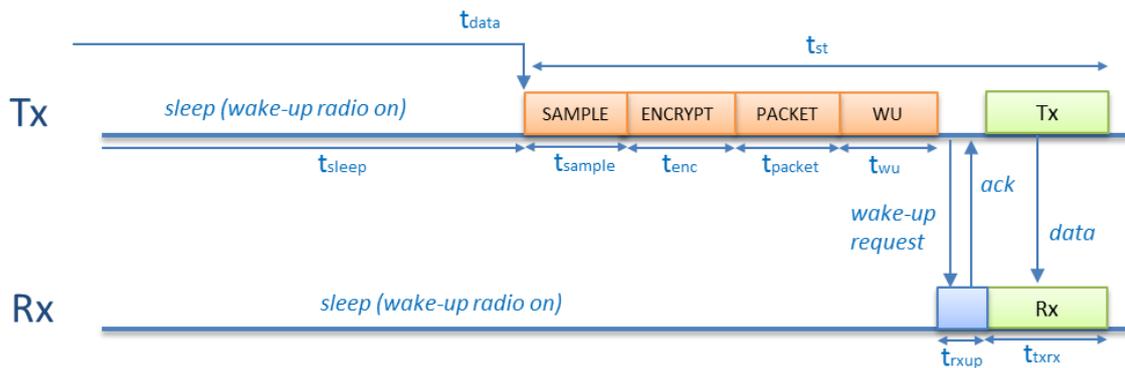
Where  $t_{sleep}$  is the sleep time between two data sampling tasks. The time terms from Equation (3.45) can be estimated in RTL simulations. If the required number of clock cycles to perform each operation is  $n_{clk}(i)$ , then for a given clock period  $T$ , the required time for the operation  $i$  is given by:

$$t_i = n_{clk}(i) \cdot T, \quad t_i \in \{t_{collect}, t_{proc}, t_{enc}, t_{packet}, t_{sync}, t_{txrx}\} \quad (3.48)$$

By knowing the time required to complete each of the system tasks and the number of task occurrences during the time  $t_{data}$  and by applying the block activity mapping to specific tasks, it is possible to extract the activity of each block and calculate the system energy.

### 3.6.2. Example of Activity Profiling in Wake-Up Controlled WSN

An example of WSN application based on wake-up radios is presented in Figure 45. The wake-up radio is a low power radio device that enables “on-request” communication between two neighbouring nodes. In a typical application, a node samples data periodically ( $t_{collect}$ ) and sends the data to its neighbour after a time period  $t_{data}$ , defined by the application. However, this example assumes a simple ‘sense and transmit’ approach, where the readings are sent immediately after sampling ( $t_{data} = t_{collect}$ ). The time required to make a single data reading is  $t_{sample}$ . The sampled data is encrypted ( $t_{enc}$ ), and the packet is prepared for sending ( $t_{packet}$ ). Each time a node has a data to send, it transmits a signal request to wake receiver ( $t_{wu}$ ). The wake-up radio of the receiver detects the request signal and wakes the receiver node up from the sleep mode. After the time  $t_{rxup}$ , the receiver (Rx) is ready to receive the data, and it sends an acknowledgment signal (ack) to the transmitter (Tx). Similar to a single-hop network, in a wake-up based network, the synchronisation of wake-up times between two nodes is not required. A node sleeps for the time  $t_{sleep}$  before it wakes-up to sample the data and send it to the receiver.



**Figure 45.** Example of WSN application powered by wake-up radio control.

In a low duty cycle application the time between two data samplings ( $t_{data}$ ) is much longer than the time a node is active ( $t_{st}$ ). Therefore, it makes sense to extract the system activity over the time  $t_{data}$ . In the given example,  $t_{data}$  can be expressed as:

$$t_{data} = t'_{sleep} + t'_{active} \quad (3.49)$$

Where  $t'_{sleep}$  and  $t'_{active}$  are the total active and sleep time of a node during the time  $t_{data}$ , respectively. The active time period related to the execution of a single 'sample and transmit' task can be expressed as:

$$t_{st} = t_{sample} + t_{enc} + t_{packet} + t_{wu} + t_{txidle} + t_{txrx} \quad (3.50)$$

Where  $t_{txidle} = t_{rxup}$  is the idle time a transmit node waits for the receiver to wake-up. During the time  $t_{data}$ , a node performs one 'sample and transmit' task and forwards  $n_{fw}$  packets. Each time a packet is sent to a node, the node has to respond to the wake-up signal, process the data and forward the packet to the next node or the sink. The time required for data forwarding is:

$$t'_{fw} = n_{fw}(t_{rxup} + t_{txidle} + t_{wu} + 2t_{txrx}) \quad (3.51)$$

If the retransmission rate due to the packet loss ( $r'$ ) is defined as the number of retransmissions over the time  $t_{data}$ , the last expression becomes:

$$t'_{fw} = n_{fw}(t_{rxup} + t_{txidle} + t_{wu} + 2(1+r')t_{txrx}) \quad (3.52)$$

The retransmission rate can be expressed as a function of packet error rate (PER):

$$r' = \frac{PER(\%)}{100} \quad (3.53)$$

Where  $PER$  is the estimated percentage of lost packets. The total active time over the time  $t_{data}$  is now:

$$t'_{active} = t_{sample} + t_{enc} + t_{packet} + t_{wu} + t_{txidle} + n_{fw}(2(1+r')t_{txrx} + t_{rxup} + t_{wu} + t_{txidle}) \quad (3.54)$$

The total sleep time of node during the time  $t_{data}$  is given by:

$$t'_{sleep} = t_{data} - t'_{active} \quad (3.55)$$

The time terms from Equation (3.54) can be estimated from RTL simulations by finding the required number of clock cycles  $n_{clk}(i)$  to perform each operation  $i$ . For a given clock period  $T$ , the required time to complete the operation  $i$  is given by:

$$t_i = n_{clk}(i) \cdot T, \quad t_i \in \{t_{sample}, t_{enc}, t_{packet}, t_{wu}, t_{rxup}\} \quad (3.56)$$

Same as in the previous example, by knowing the time required to complete each of the system tasks and the number of task occurrences during the time  $t_{data}$ , and by using the block activity mapping to specific tasks, it is possible to extract the activity of each block and calculate the energy.

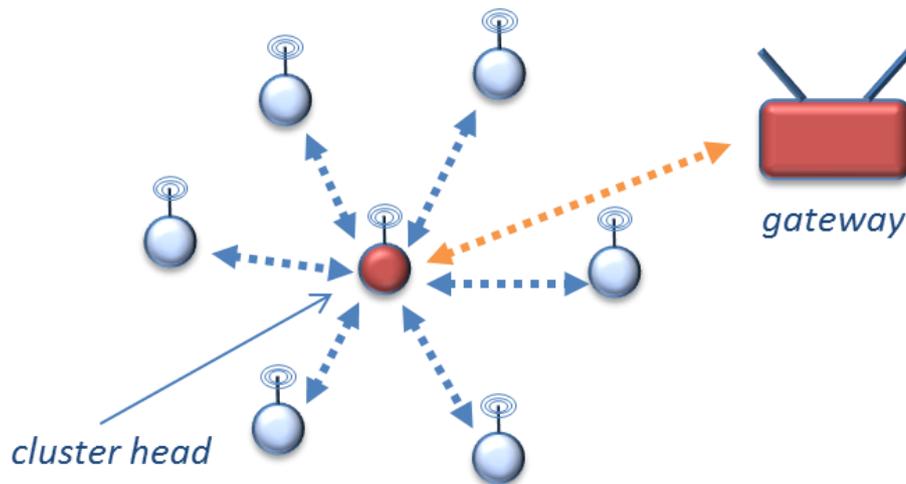
### 3.6.3. Impact of Network Topology

A single sensor node acts both as transmitter and receiver. A node can be programmed to perform both data sampling and packet forwarding, or it can do only one of the two tasks. Depending on the network topology, implemented routing protocols and application, the number of forwarded packets during the time  $t_{data}$  will be different for different sensor nodes in a network. Therefore, it is required to find a representative node model for a given network topology.

#### *Clustered WSN Topology*

In a single-hop clustered network, under assumption that the cluster head has unlimited power, the implicit synchronization is not required since all nodes communicate directly to the sink, e.g., to the cluster head (Figure 46). Therefore, the number of forwarded packets  $n_{fw} = 0$  and  $t'_{listen} = 0$ .

The number of forwarded packets in the cluster head during the time  $t_{data}$  will be equal to the number of cluster members  $n_c$  ( $n_{fw} = n_c$ ). In a multi-hop clustered network, the cluster heads will aggregate the packets from neighbouring clusters or gateway nodes. This will increase the number of transmission tasks in cluster heads accordingly.



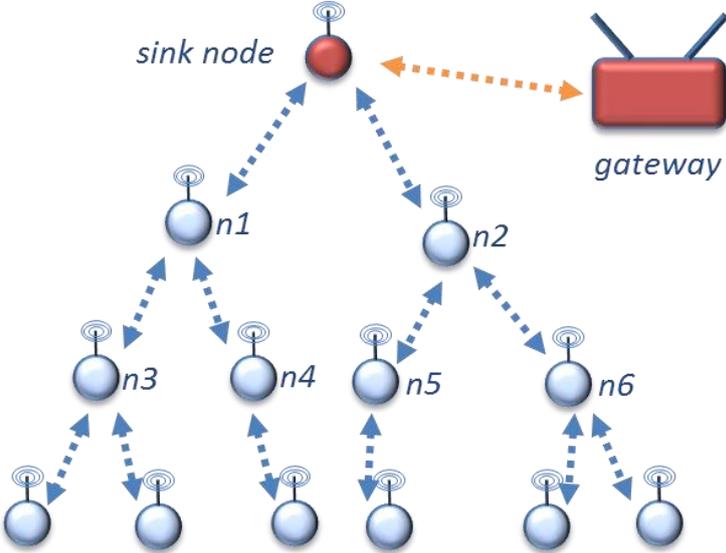
**Figure 46.** Single-hop cluster topology of WSN.

#### *Tree WSN Topology*

In a multi-hop tree topology, the sensor nodes at the bottom of the tree will make no packet forwarding ( $n_{fw} = 0$ ), but they will only transmit sampled data. The sensor nodes in the middle and at the top of the tree will aggregate and forward the data from the bottom branches of the tree.

In an example from Figure 47, assuming a single packet transmission per node during the time  $t_{data}$ , the nodes  $n3$  and  $n6$  will forward two packets each, and the nodes  $n4$  and  $n6$  will forward a single packet each during the time  $t_{data}$ . The node  $n1$  will forward the data from the nodes  $n3$  and  $n4$ , and the node  $n2$  will forward the data from the nodes  $n5$  and  $n6$ . If all nodes are configured to

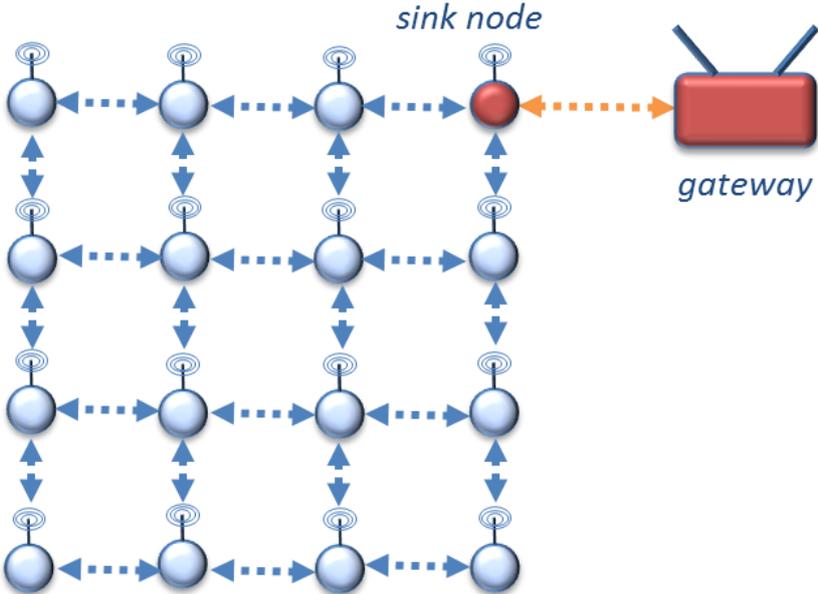
sample and forward data, then the number of forwarded packets of the nodes  $n1$  and  $n2$  during the time  $t_{data}$  will be 5. For the profiling of node activity, the designer may decide to calculate either the average or the maximum number of forwarded packets per node.



**Figure 47.** Tree topology of WSN.

*Mesh WSN Topology*

In a mesh topology, each node can communicate with all neighbours within its reach (Figure 48).



**Figure 48.** Mesh topology of WSN.

The forwarded packet rate of a mesh network will depend on the implemented routing protocol and the size of a mesh. For small-scale networks, knowing the routing paths, it is possible to analytically calculate the average or the maximum number of forwarded packets per node during the given time  $t_{data}$ . In large scale networks, considering the average number of neighbours, distance to the sink, etc. the results can also be obtained analytically. However, more accurate results can be delivered by WSN simulation tools.

The application of the presented methodology to a sensor node microcontroller design is demonstrated later in this work. The developed design flow incorporating proposed methodology is described in the following chapter.



# 4. Power-Driven Design Flow

## 4.1. Overview

Low power design requires a holistic approach that considers power in all steps of design flow. A typical design flow includes four basic steps of design: system level design, RTL design, implementation and verification (Figure 49). In a power-driven design flow, each step has to consider the design impact on system power. Therefore, the actions related to each of design flow steps are determined by power constraints. Often, to satisfy design and power requirements, several iterations through the design flow are required.

The actions related to system-level design include design planning and exploration as well as hardware/software partitioning driven by performance and power. The goal of system-level design is the definition of final system architecture. The power methodology presented in Chapter 3 applies at system-level. However, some actions related to the RTL design and to implementation are required by the methodology application.

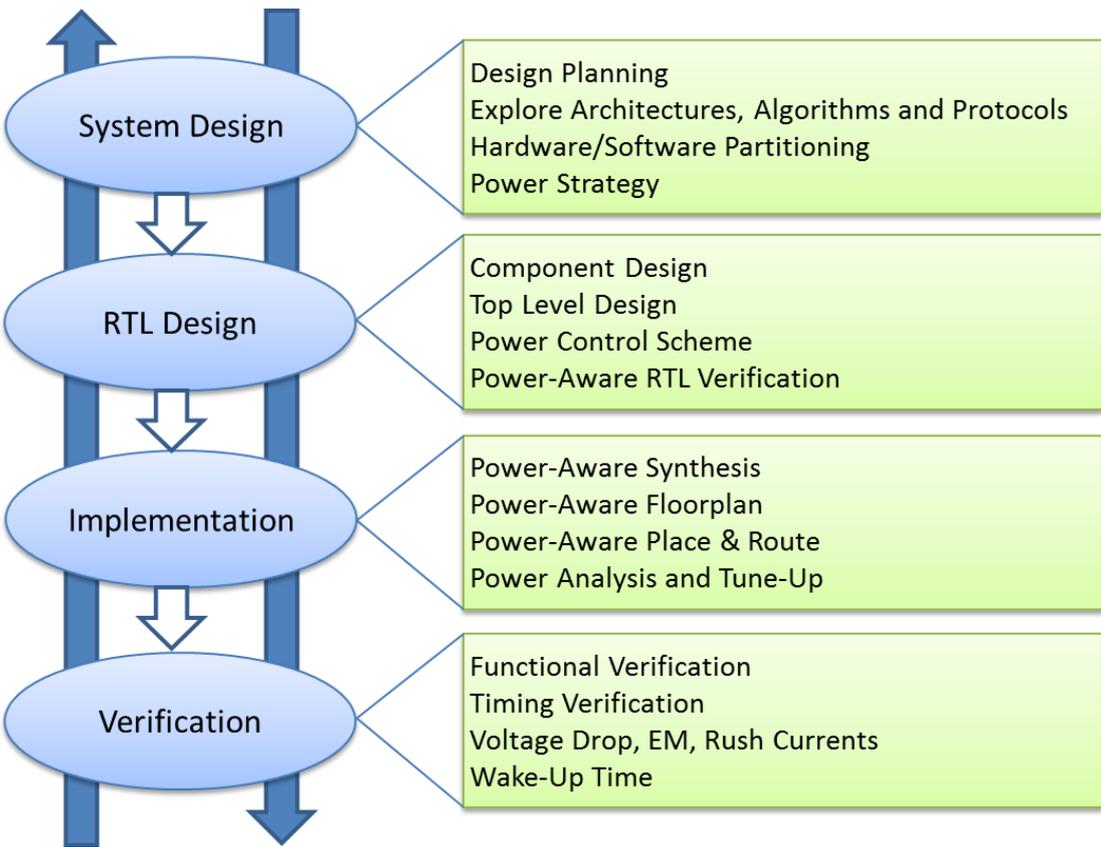


Figure 49. Basic steps in power-aware design flow.

RTL design includes actions for HDL design and verification of system components and top-level system. At this stage, control logic for selected power scheme is designed and integrated in top-level

RTL description. Power-aware RTL design includes power-aware functional verification of RTL system components and system itself.

Power-aware implementation includes logical synthesis and physical design of a system driven by power constraints. In power-aware synthesis, the RTL system description is mapped to logical gates and the isolation cells and level shifters are inserted. A design is also proved for timing and power. In physical implementation, the layout of a chip is created. The layout design includes floorplanning, place and route, and the timing and physical sign-off. The power analysis and insertion of power switches are part of power-aware implementation flow.

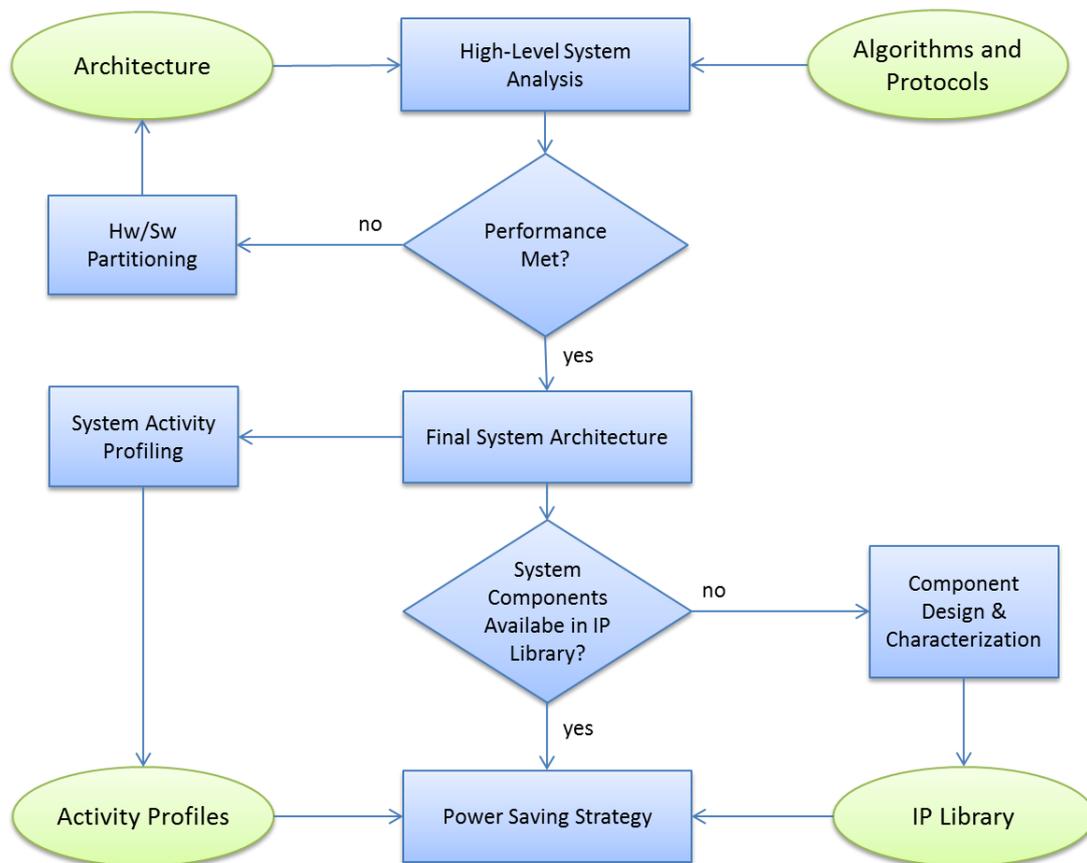
The verification flow includes the steps to verify design for timing, power, functionality and physical correctness. The verification steps apply after each main step of the design flow. Power-aware verification proves for the effects of voltage drop and electromigration as well as for the rush currents and wake-up time of a power gating implementation.

#### **4.2. Power-Aware System-Level Design**

The basic steps of the power-aware system-level design flow incorporating proposed methodology are illustrated in Figure 50. The system-level design starts with the definition of general system architecture considering required functionality and design constraints. Usually, a high-level system model is constructed and analysed under predefined conditions. The system level analysis inspects the performance of different algorithms and protocols for a given system architecture. In case of wireless sensor networks, the system performance is analysed for different network topologies, network sizes and underlying communication protocols.

In processor-based systems, the system performance is often defined by the performance of processor and available storage capability. If processor is not able to satisfy the performance requirements of certain algorithm, the designer has to consider the implementation of hardware acceleration for that specific task. This process is known as hardware/software co-partitioning. It is also possible to perform hardware/software partitioning based on power constraints. If the energy required by the processor to complete a specific task is found to be too high, the designer may implement a dedicated low power coprocessor instead.

For selected system architecture, it is required to define power saving strategy of design. To find a power saving strategy for a general-purpose system, some of the available tools for high-level design exploration can be applied (see Chapter 2.4.3). However, for application-specific designs, it is required to apply more sophisticated methods. The methodology that is proposed in this work is general and can be applied to any design as long as an accurate system activity profiling is possible to perform. An example of the activity profiling in typical sensor node systems is presented earlier in this work (see Chapter 3.5.1). Additional to activity profiling, in order to estimate the system energy, it is required to calculate an average power consumption of system components during active and sleep modes. For many system components that already exist in local IP library, the power information is provided. This also applies to low power components such as power switches, level shifters, etc. The new components have to be designed in RTL, synthesized, and characterized for power. The designed component is stored in IP library and the characterized data are available for usage in future designs.

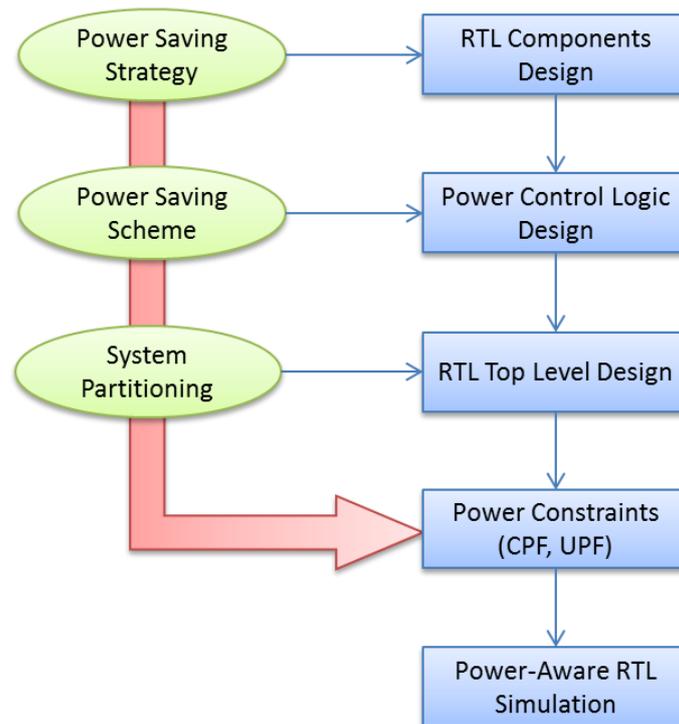


**Figure 50.** Power-aware system level design.

### 4.3. Power-Aware RTL Design

RTL design includes HDL (hardware description language) design of system components and HDL design of top-level system architecture. The components that are not part of available IP libraries have to be designed and verified in RTL. For a defined power saving strategy, it is required to create control logic to support selected power saving scheme. Accordingly, the design of power gating controller must consider all issues related to power transition times, state retention, control of in-rush currents, etc. The tasks of power-aware RTL design are illustrated in Figure 51.

When designing the RTL description of top-level system architecture, each power domain is created as single level of RTL hierarchy. This is required by the implementation tools in order to support the creation of power islands and the automatic insertion of special power cells (power switches, level shifters, and isolation). Additionally, power constraints are specified in the UPF or CPF format depending on selected design tool flow. Finally, the top-level RTL design is simulated for correct functionality. The simulations have to verify correct behaviour of power control logic as well. Simulation tools have to be able to understand the power intent specified in a CPF or UPF file. If supported by the tools, an early power analysis can be performed at RTL level as well.

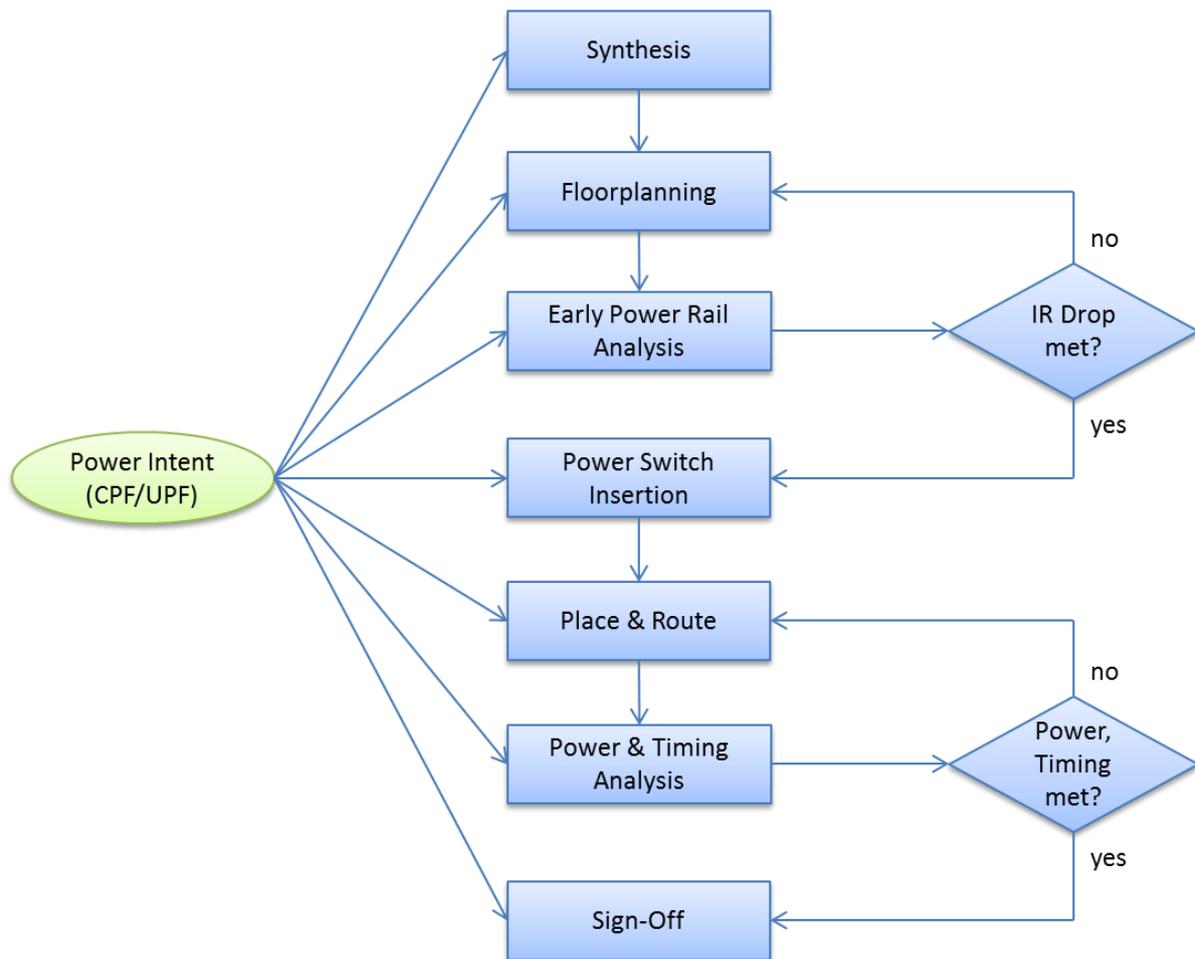


**Figure 51.** Power-Aware RTL design.

#### 4.4. Power-Aware Implementation

The implementation flow (Figure 52) includes the tasks required to transform the RTL description of design to physical implementation. Each action in the implementation flow is followed by an optimization step to fix timing and design violations. Power-aware physical implementation uses information from a power intent file to complete the tasks. The flow starts with the synthesis of RTL design to gates. The synthesis constraints must ensure that design hierarchy is not changed during optimization process. In power-aware synthesis, isolation cells, level shifters and retention flip-flops are inserted in design. A synthesized design is then verified for power and timing. Timing is analysed for multiple delay corners. Finally, a back-annotation is performed to check for correct functionality of the synthesized design. Optionally, the insertion of scan chains is also part of the synthesis flow. The actions related to logical synthesis are referred as the frontend implementation flow.

The backend implementation flow starts with the creation of floorplan. The floorplanning is performed manually using a layout design tool (for example, SoC Encounter from Cadence). During floorplanning, the power supply grids for an entire chip are created. In multi-voltage or power gated design, it is required to create a separate power grid for each power domain. Additionally, the placement of macro cells and IO pads as well as the definition of placement and routing blockages are the tasks of floorplanning. Once the power grid network is created, an early power rail analysis is performed to estimate the IR drop. The analysis results are then used to tune-up the size of power grid.



**Figure 52.** Power-Aware Implementation.

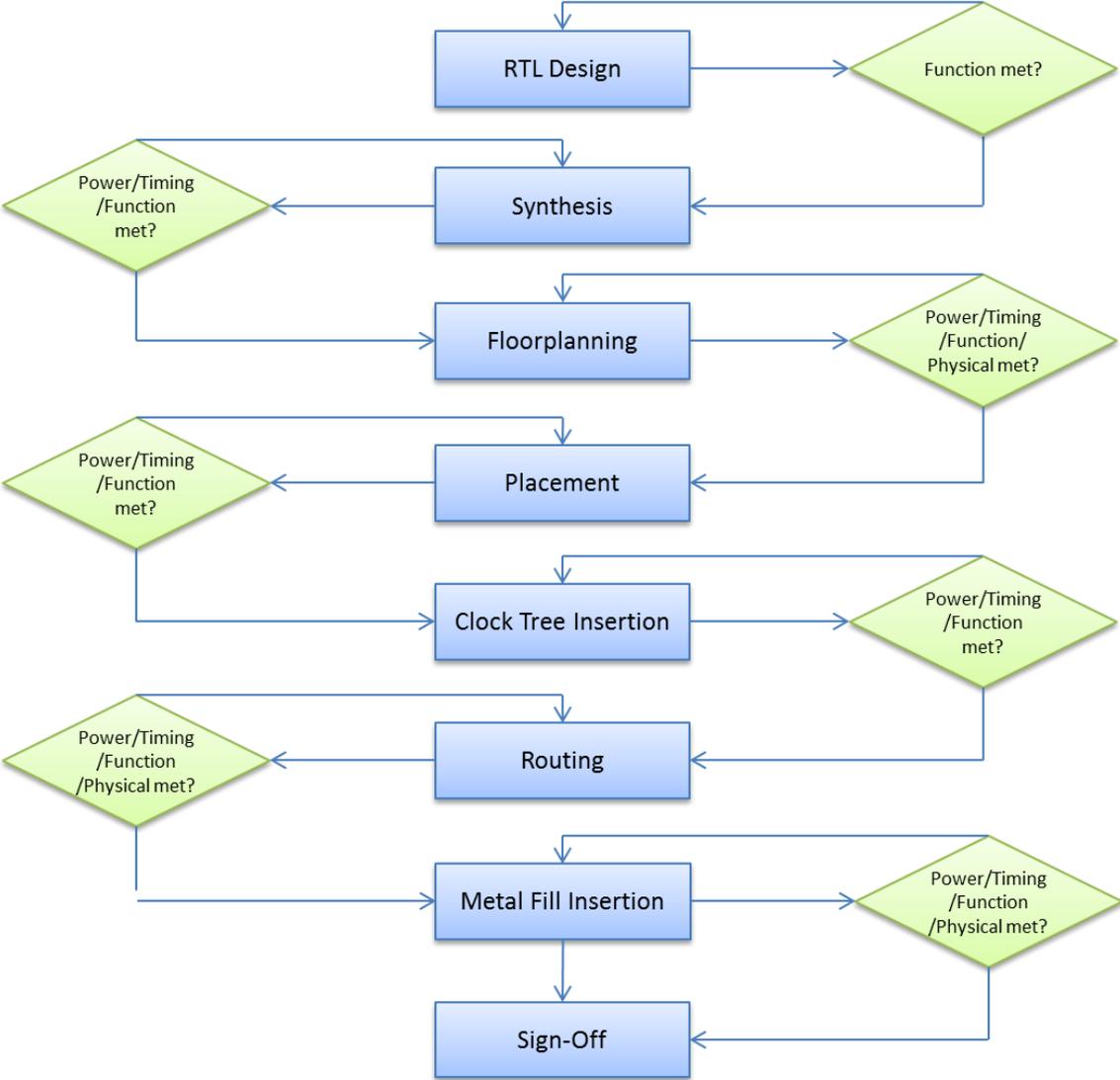
After IR drop is met, the power switch insertion is performed. Power switches are inserted either inside a power domain (column-based style) or around a power domain (ring-based style). The number of inserted power switches is optimized according to user specified constraints. Also, the sleep signals are distributed to power switches.

The insertion of power switches is followed by cell placement, clock tree synthesis and global routing. The static timing analysis and low power checks are performed after each of these steps. If required, the in-place timing optimization is initiated to improve timing. During placement, scan chains are reordered. The results of power analysis on a routed design are used for fine tune-up of the power switch numbers and rush currents. Once, the timing and low power checks are in place and the physical design requirements are met, the design is ready for sign-off.

#### 4.5. Power-Aware Verification

The actions related to design verification apply after each main step in design flow (Figure 53). The design is verified for timing, power, physical correctness and functional behaviour. Power analysis includes analysis of leakage and dynamic power, voltage drop check, analysis of

electromigration and signal integrity, and analysis of rush currents and turn-on time. Timing verification proves if the setup and hold time constraints for design are met. Physical verification searches for violations in design geometry and connectivity. Finally, functional verification proves the system logic behaviour.



**Figure 53.** Functional, timing, power and physical verification.

Timing verification includes the steps for static timing analysis of the setup and hold times for all defined delay corners. In a multi-voltage design, the timing is analysed for each power mode and different delay corners. This type of analysis is called multi-mode-multi-corner analysis (MMMC).

The most important power analysis checks are done after creation of power supply network and insertion of power gating cells in design. Early power checks analyse for the voltage drop on power rails. After insertion of power switches, the turn-on time and rush currents are analysed as well. The results are used for optimization of the total number of power switches in design and for optimization of the power control unit.

Physical verification proves for geometry and connectivity errors in design. The initial physical verification of power and ground connections is performed after the floorplanning step. Physical analysis of all wires and geometries in design is done after the place and route step is completed and metal fill is inserted.

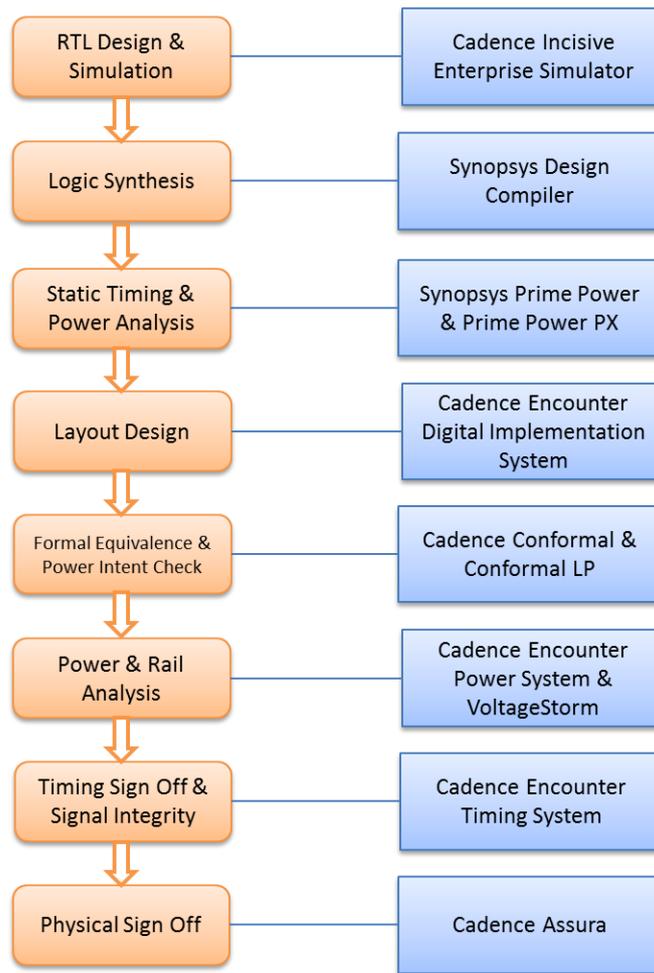
Functional verification is performed after each step of design flow that changes the logical structure of design. However, for large designs, the frequent timing simulations can be time consuming. Therefore, designers often rely on the results of static timing analysis and formal equivalence checking. Formal equivalence checking proves formally if two representations of a circuit exhibit exactly the same behaviour. Therefore, each time design is altered the netlist representation of the design is generated and proved for its logical equivalence with a golden reference model, usually a RTL model described in some hardware description language (VHDL or Verilog).

#### **4.6. Tools**

The major steps of each design flow rely on the design tool support. All major EDA tool vendors offer their solutions to support chip design automation. Generally, the choice of design tools is often driven by the cost factor and tool efficiency. In this work, the tools supporting proposed design flow are chosen according to the tool availability and designer experience working with different tools over the years. The simplified illustration of the presented design flow and the related design toolkit is presented in Figure 54.

The tools used in the flow are standard design tools from the industry leaders Cadence and Synopsys. Synopsys tools are used in the frontend flow for logic synthesis (Design Compiler) and for static timing and power analysis of the synthesized design (Prime Power & Prime Power PX). The CPF-based backend flow is enabled by the Encounter Digital Implementation System (EDI) toolset from Cadence. Beside the layout design tool support, the EDI System environment includes tools for formal equivalence checking (Conformal), power intent checking (Conformal LP), power and rail analysis (Encounter Power System, VoltageStorm) and timing signoff and signal integrity analysis (Encounter Timing System). The physical signoff that checks for DRC (design rule check) and LVS (layout versus schematic) errors is performed with the Cadence Assura Physical Verification tool. Functional verification and back-annotation is done with the Cadence Incisive Enterprise Simulator (IES). The IES tool suite, also referred as NCsim, contains the tools for compilation and elaboration of VHDL and Verilog files (NCvhdl, NCvlog and NCellab) as well as HDL simulator (NCsim) and waveform viewer (SimVision). The IES toolset supports low power verification based on CPF.

The proposed design flow and supporting tools are used to design a low power sensor node microcontroller chip. The design and implementation process is described in the next chapter.



**Figure 54.** Design toolkit for the presented power-driven design flow.

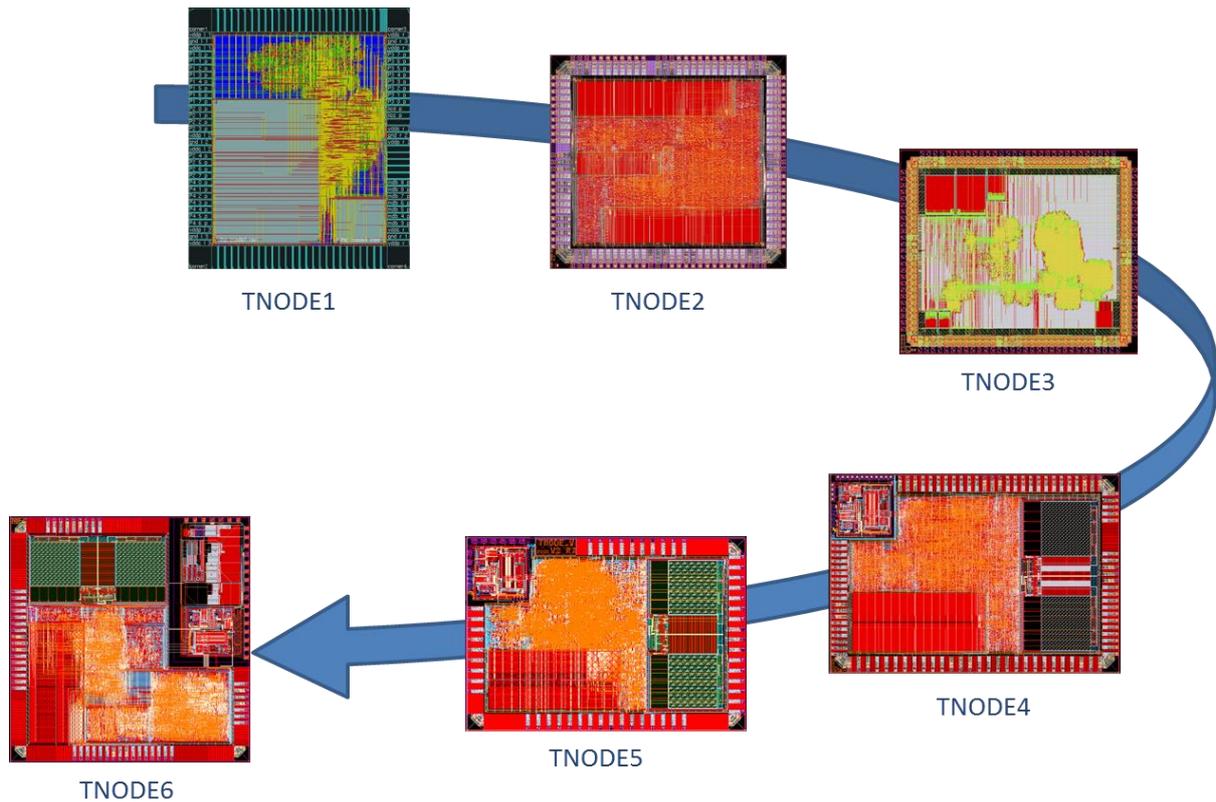
## **5. Design of Embedded Sensor Node Microcontroller**

### **5.1. Introduction**

The presented low power methodology has been validated through the design of an embedded sensor node microcontroller system-on-chip. The microcontroller has been designed for sensor network applications with strong security demands. The designed chip is a complex power-gated system that implements a processor core, a number of IO peripherals, hardware accelerators for security and communication tasks, Flash memory and an analogue-to-digital converter. The chip development followed the design of several sensor node systems, each of them having different components required by specific project requirements. All developed systems are based on a low power 16-bit RISC processor core, compatible to the Texas Instrument (TI) MSP430x specification. Along with the sensor node system design, the components required by the implementation of advanced low power methodologies have been developed as well.

### **5.2. Development of Sensor Node Microcontroller**

A sensor node microcontroller has been initially designed and developed for the Tandem project [160]. The envisioned microcontroller has been named TNODE (Tandem NODE), and six successive releases of this microcontroller (TNODE1 to TNODE6) have been fabricated in the IHP CMOS and BiCMOS technologies. The main goal of the Tandem project was to define a low power multiprocessor architecture for specific sensor network applications. The envisioned solution combined a general purpose RISC processor core and hardware accelerators in a power efficient, single-chip, sensor node microcontroller design. The TNODE design has been continually improved during the projects Smart [161], Matrix [162], and StrokeBack [164]. The Smart project investigated the requirements of sensor node implementations in security demanding WSNs. The primary goal of the Matrix and the StrokeBack projects was to develop a telemedicine system to be used in patient monitoring programs. The sensitive nature of the monitored data requires a high level of data confidentiality, which can be provided by the cryptographic means implemented in TNODE chip. The latest version of the chip has been developed as part of the Telediagnostik project [163]. In the Telediagnostik project, sensor nodes are used to monitor the biochemical parameters of bioreactors. These sensor nodes must be featured with a similar cryptographic mean to achieve the demanded security according the sensitivity of the monitored data. In total, six different versions of TNODE chip have been developed, including the one that is in the focus of this work, the TNODE6 (Figure 55). All TNODE chips include the IPMS430x processor, a quasi-asynchronous clone of the TI MSP430x architecture, developed by Fraunhofer IPMS [136].



**Figure 55.** Development of a sensor node microcontroller. Six releases of the TNODE.

The TNODE development started with the design of basic microcontroller unit supported by a number of peripherals, data memory and communication accelerator, called TNODE1 [130]. The second version of the chip, TNODE2, was the first TNODE chip designed for security applications [131]. TNODE2 included on-chip crypto and communication accelerators, and it was designed for the IHP 0.25  $\mu\text{m}$  CMOS technology process. The TNODE2 chip used an external flash memory accessed through a parallel data interface for its program memory. This interface raised a large number of pads, which makes the chip design pad-limited. Furthermore, it induces significant area overhead and increased power consumption in the pad cells. The third version of TNODE (TNODE3) was designed for the IHP 0.13  $\mu\text{m}$  CMOS technology process. This chip required an external program memory as well, but contained four crypto accelerators and a communication accelerator to support link layer operations defined by the IEEE 802.15.4 standard [132]. TNODE4 was the first TNODE chip to implement an embedded Flash memory and an analogue-to-digital convertor on chip [133]. This chip was designed for the IHP 0.25  $\mu\text{m}$  BiCMOS process, and it runs with the maximum frequency of 11.4 MHz. Driven by research constraint, the high speed parallel data interface is still maintained in TNODE4. By keeping this interface, it was possible to use an external program memory as well as externally connected peripherals. TNODE5 reduced the number of pads to 64, and it introduced an on-chip digitally controlled oscillator (DCO). This enabled additional power savings in the pads, and the on-chip DCO made the chip more flexible to application demands. The current version of TNODE, so called TNODE6, includes additionally to crypto and communication accelerators, complex clock control logic and a preamplifier for ADC. Furthermore, the TNODE6 chip implements on-chip power gating. The overview of the main features of different TNODE microcontrollers is given in Table 4-1.

**Table 4-1.** Features of the six TNODE releases.

Chip Version	Process	Number of Pads	Max Frequency	Memory	Chip Area (mm <sup>2</sup> )	Peripherals	Security	Power Strategy
TNODE1	0.25 μm	115	20 MHz	5 kB RAM	12.6	4x IO ports, timer, SPI, baseband	n.a.	clock-gating
TNODE2	0.25 μm	104	20 MHz	18 kB RAM	15.7	3x IO ports, timer, SPI, 2x UART, baseband	AES, SHA-1, ECC	clock-gating, hw acc
TNODE3	0.13 μm	112	20 MHz	18 kB RAM	8.9	3x IO ports, timer, SPI, 2x UART, baseband, MAC	AES, SHA-1, ECC, RSA	clock-gating, hw acc
TNODE4	0.25 μm	127	11.4 MHz	16 kB RAM 64 kB Flash	24.7	3x IO ports, 2x timer, 2x SPI, 2x UART, ADC	AES, SHA-1, ECC	clock-gating, hw acc
TNODE5	0.25 μm	64	11.4 MHz	16 kB RAM 64 kB Flash	23.96	4x IO ports, timer16, timer32, 2x UART, 2x SPI, ADC, DCO, baseband	AES, SHA-1, ECC	clock-gating, hw acc
TNODE6	0.25 μm	71	11.4MHz	16 kB RAM 64 kB Flash	31.1	4x IO ports, timer16, timer32, 2x UART, 2x SPI, ADC+BMS, DCO, baseband, clock ctrl	AES, SHA-1, ECC	clock-gating, hw acc, power-gating

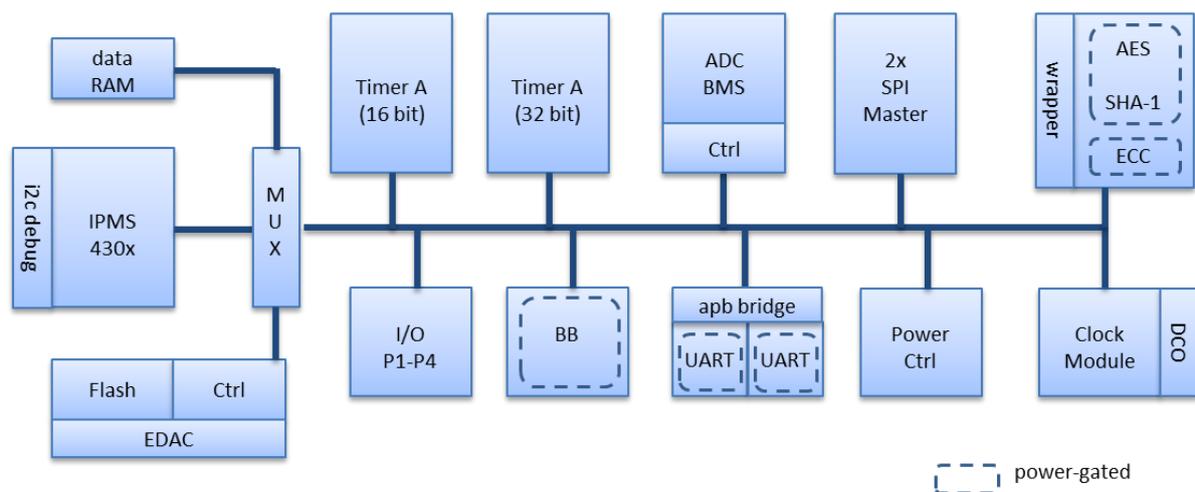
The TNODE6 is the first TNODE chip designed by applying the described low power methodology and power-driven design flow (see Chapters 3 and Chapter 4). The complete design process of the TNODE6 microcontroller is described in next subsections.

### 5.3. Requirements

At the very beginning of TNODE development, it had to be decided on the target technology and basic system requirements. For practical reasons, the choice of the technology process felt between two in-house available processes, the IHP BiCMOS 0.25 μm and the IHP BiCMOS 0.13 μm. The availability of the Flash process and different analogue components made the IHP 0.25 μm technology being preferred solution for the most of the designed sensor node microcontrollers. The application area for the target sensor node was basically a wireless sensor network with strong security demands. This required the development of system architecture that can support the execution of complex crypto algorithms used in security protocols. The choice of processor core had to meet the requirements for low power, high performance and good software and debug support. The decision on processor felt on a 16-bit RISC core compatible to the TI MSP430 family of processors, used in the popular Telos and Tmote Sky sensor node platforms. These platforms have been used for software development in different in-house projects. Thus, the developed software could be easily adapted to the designed embedded system. The peripheral components of each sensor node chip are chosen according to specific project requirements. The developed sensor nodes targeted extremely low duty cycle applications, where a node is considered to sleep most of the time. Therefore, a power saving strategy that targets both dynamic and static power consumption in chip had to be applied.

## 5.4. Sensor Node Architecture

The initial constraints for the design of TNODE microcontroller demand for low power operation at low bandwidth and strong security support. The execution of complex crypto algorithms in software has proven as highly inefficient, especially with 8- or 16-bit microcontrollers, which have limited storage and computation resources. Therefore, the implementation of hardware support for required crypto operations into the system is required. Furthermore, to improve the chip performance and to increase security, the chip implements a baseband (BB) controller supporting direct sequence signal spreading (DSSS). The DSSS improves resilience to bit manipulation, interference, jamming and eavesdropping. The core of the system is the 16-bit IPMS430x RISC core. The IPMS430x is a poly-phased clock, latch-based, ultra-low power processor core. An asynchronous nature of the core requires a non-standard approach for peripheral interfacing. The chip includes a number of dedicated peripherals, designed to support communication with external devices such as sensors, radio modules, led diodes, etc. Among designed peripherals are digital IO ports, timers, serial ports as well as controllers for Flash and ADC. The clock distribution has been provided through the integrated DCO and two external sources, a slow and a fast clock oscillator. The implemented clock controller allows for an individual setup of each peripheral clock. Finally, a power gating on selected blocks has been implemented. The final architecture of the system is illustrated in Figure 56.

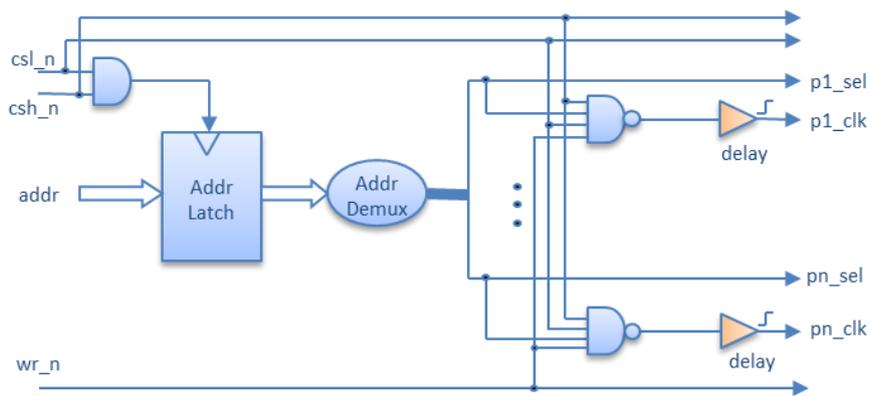


**Figure 56.** Architecture of the power-gated TNODE6.

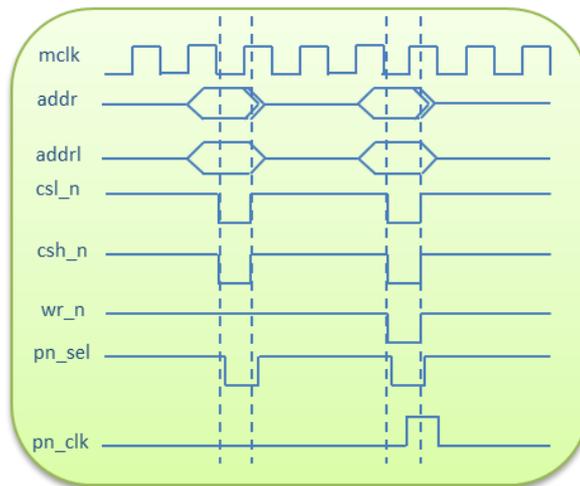
Some of the components used in the target system have been contributed by other projects (crypto accelerators, baseband, SPI, flash controller, and EDAC) or they have been taken from available IP libraries (processor, UART, Flash, ADC, and DCO). The contribution of this work is the design of timers, digital IO ports, power management unit, clock module, ADC controller, system bus controller and power-gating cells. Additionally, the interface of each peripheral component has been adapted to support the handshake with the asynchronous processor. Furthermore, a 16-to-32 bit interface to crypto cores block has been designed, as well as a bridge logic to the APB (AMBA peripheral bus) interface of UARTs. The top-level system design, implementation and verification have been also contributed by this work.

### 5.4.1. Processor Core and Peripheral Interface

The IPMS430x processor is fully compatible with the TI MSP430x architecture. It is a latch-based design controlled by two internally generated poly-phased clocks. The implemented processor architecture provides a very small footprint and ultra-low power operation without sacrifice in performance. However, it has been required to design a dedicated peripheral interface between the asynchronous processor and synchronous peripherals (Figure 57). This is due to the latch-based architecture of the processor that generates non-synchronized control output signals to the input clock.



(a)



(b)

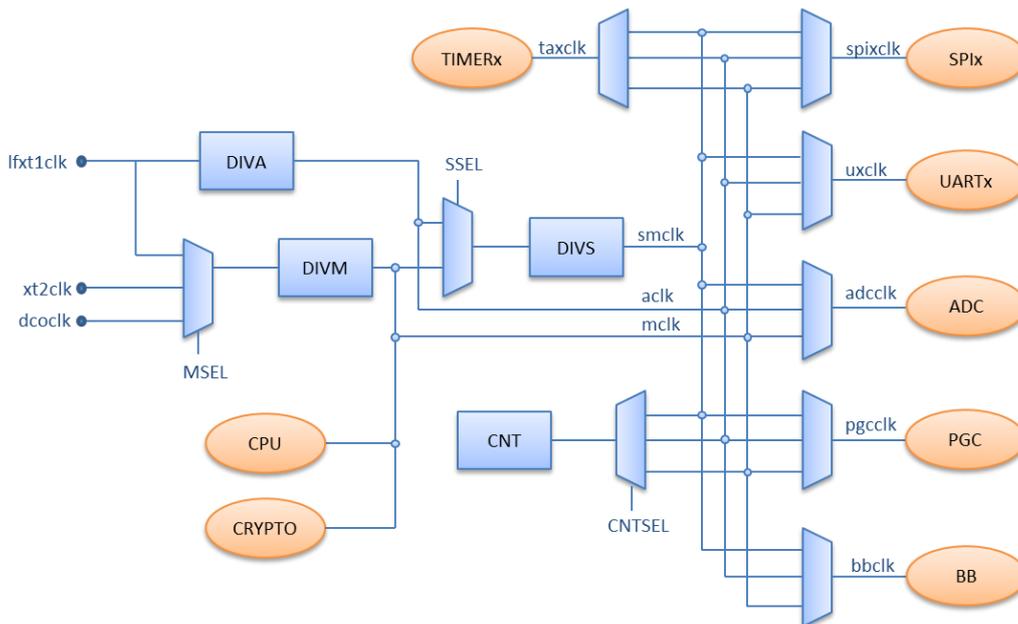
**Figure 57.** Interface between asynchronous processor and peripherals: (a) interface logic, (b) handshake flow.

The IPMS430x generates control signals for low byte select (csl\_n), high byte select (csh\_n) and write strobe (wr\_n) that are actually derived from the input clock (mclk). This property of IPMS430x makes the clock synchronization difficult when interfacing the processor to standard synchronous components. To solve this problem, the designed interface logic (Figure 57a) makes an implicit signal

synchronization and enables error free data flow (Figure 57b). The interface allows an access to peripheral control registers without disturbing peripheral function. The peripherals are selected by the generated peripheral select signals (px\_sel). The address signals from the processor (addr) have been latched (addrl) in order to ensure that the address remains stable whenever the control signals are active. The designed interface also provides dedicated clocks to peripherals (px\_clk) that apply only when the processor is writing to peripheral control registers. In standard operation mode, all peripherals run with clock signals provided by the clock module. To avoid conflicts, the processor can stall a peripheral's main clock until the write operation has been finished. The peripherals communicate with the processor over their interrupts. The interrupts have been connected through a prioritized interrupt chain. For the time an interrupt from a high-priority peripheral is pending, an incoming interrupt from a low-priority peripheral is being stalled. The debugging features have been provided through a dedicated I2C interface. The I2C debugging interface has been also used for the programming of Flash memory.

#### 5.4.2. Clock Module and DCO

The clock module unit provides clock distribution to system components (Figure 58). It takes clock signals from three different sources (DCO, low-frequency oscillator and high-frequency oscillator) and generates internal clock signals used by the processor and peripheral components. The internal clocks are selectable from the available clock sources, and they can be divided by the factor of 1, 2, 4 or 8. All multiplexers used for the clock source and clock ratio selection implement synchronization logic that prevents generation of glitches and possible malfunction. There is a dedicated clock implemented for each peripheral except for the crypto block, which runs at the same fast clock as the processor. The clock module implements a 16-bit clock counter that increments with each clock cycle. The counter is used to determine the duration of software modules.



**Figure 58.** Clock distribution in sensor node microcontroller chip.

The implemented DCO supports 64 different frequencies ranging from 5.8 MHz to 13.9 MHz. The target frequency of DCO is programmed by an 8-bit control register, where only 6-bits are effectively in use. The DCO consists of comparator, two transmission gates, output driver, and voltage divider used to provide different reference voltages. It has the size of 0.024 mm<sup>2</sup> and consumes around 265  $\mu$ W when active. When disabled, its standby power is around 2.5  $\mu$ W. More details on the implemented DCO are given in [139].

### **5.4.3. Digital IO**

The sensor node microcontroller implements 4 digital IO ports (P1-P4). The IO ports are fully compatible to the TI digital IO specification. The ports P1 and P2 have interrupt capability and can be used to trigger events on either rising or falling edge of the signal. The ports P3 and P4 are simple digital IO ports. All IO ports can be configured to connect to external signals or to share internally defined input or output signals. All external ports of the TNODE related to timers and serial controllers have been shared over IO ports.

### **5.4.4. Timers**

The sensor node chip implements two timers, a 16-bit timer and a 32-bit timer. The 16-bit timer is fully compatible with “Timer A” specification of TI. It includes three capture/compare registers and supports multiple captures, pulse width modulated outputs and interval timing. The timer also provides several interrupt capabilities, where the interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers. The capture unit supports both synchronous and asynchronous capture events, which may be triggered on rising, falling or both edges. The timer clock is selected between the system internal clocks and a dedicated external clock input. The timer clock can be divided by the factors of 2, 4 and 8. The 32-bit timer has the same features as the 16-bit timer, except its counter register is extended to 32-bits. This enables long counting modes that are desired in low duty cycle sensor node systems.

### **5.4.5. Serial Ports**

The connectivity to external devices (sensors, radio control, etc.) is provided by two UART controllers and two SPI master controllers. The UARTs are compatible to the TI 16550 specification, and they support a variable data word size of 5 up to 8 data bits. Furthermore, an optional parity bit and one or two stop bits are also supported. Each UART includes a programmable 12-bit clock divider to control data bit-rate. The hardware flow-control of an UART is supported via RTS/CTS handshake signals. The UART receiver is able to detect parity, framing, break and overrun errors.

The SPI master controllers provide simple serial connectivity to one or more slaves. The implemented SPI cores support active edge selection and burst mode operation allowing exchange of larger bit frames. The control of slave select signal in the burst mode is provided through an IO port. The control of SPI bitrate is regulated by an integrated clock divider that supports clock division ratios from 1:4 to 1:32.

#### **5.4.6. Memory Subsystem**

The sensor node microcontroller implements 16 kB of RAM and 64 kB of non-volatile Flash. The processor in the TNODE is based on von-Neumann architecture, so both RAM and Flash can be used for data and program storage. The system boot address is mapped to the Flash address space. The implemented Flash is a sector-erasable device having 16k x 44-bit architecture. The additional twelve bits in the Flash data word have been used for the implementation of error detection and correction logic (EDAC). The implemented EDAC can handle 1-bit errors and detect 2-bit errors. The Flash controller is designed to support Flash programming in debug mode. Alternatively, the Flash can be programmed by the processor, during active mode of operation. The sector-based programming feature enables on-the-fly replacement of the software modules stored in Flash. The size of the implemented Flash block is around 5 mm<sup>2</sup>.

#### **5.4.7. Crypto Accelerators**

Sensor nodes are vulnerable to a number of possible security attacks. The security attacks can be launched at all layers of sensor network protocol stack. An effective countermeasure to deal with the security threats in WSN is to use cryptography. There are two families of crypto algorithms that are commonly in use: symmetric (shared-key cryptography) and asymmetric (public-key cryptography). In symmetric cryptography two parties are using the same shared key to encrypt and decrypt data. The symmetric algorithms are generally faster and less computationally expensive. In combination with cryptographic hash functions, symmetric algorithms can be used to generate “fingerprints”, which can guarantee data integrity. However, the problem of key distribution restricts the usage of symmetric cryptography in WSN [140]. Asymmetric crypto algorithms use a pair of mathematically-related keys for encryption and decryption. Asymmetric crypto algorithms are computationally expensive and usually seen as inappropriate for application in wireless sensor networks. In our envisioned scenario, asymmetric cryptography is used for the key exchange and symmetric algorithms are used for data encryption. Accordingly, the required infrastructure to support both symmetric and asymmetric crypto operations as well as hash generation is integrated in the chip. The designed chip includes hardware accelerators for symmetric advanced encryption standard (AES), elliptic curve cryptography (ECC) and secure hash algorithm (SHA-1). All crypto cores have 32-bit data registers and operate with 32-bit data. The cores are organized in a single crypto block sharing the same interrupt address. The implemented 16-to-32 bit controller provides the interface between the processor and crypto peripherals.

### *AES*

The AES is a symmetric block cipher algorithm, based on the Rijndael cipher [141]. It has a fixed block size of 128-bit and supports three different key lengths 128, 192, and 256 bits. The designed chip implements an AES hardware accelerator that supports 128-bit key length only. The implemented accelerator core takes 66 clock cycles to accomplish encrypt or decrypt operation. This is substantially less compared to software implementation on TI MSP430F1611 microcontroller, which takes 6600 cycles for encryption and 8400 cycles for decryption [142]. More details on the implemented AES core are given in [143].

### *ECC*

The ECC is an asymmetric cipher algorithm that uses algebraic operations on elliptic curves over finite fields. The security of ECC is based on the complexity of finding the discrete logarithm of a random elliptic curve element. Due to the higher difficulty of calculating a discrete logarithm for points of elliptic curves, the algorithm is able to provide very high level of security even with relatively short key sizes. The implemented ECC accelerator core operates with key lengths of 233 bits. The elliptic curve point multiplication (ECPM) is performed by the Lopez-Dahab algorithm [137], and the multiplier is an Iterative Karatsuba multiplier [138], which requires 9 cycles for each 233 bit operation. The ECC is the largest single digital block in the system.

### *SHA-1*

The SHA-1 is one of the most widely used algorithms for secure hash calculation. It generates a 160-bit message digest of 512-bit input block. The implemented SHA-1 core requires 160 clock cycles to generate a 160-bit hash message. The basic function of SHA-1 is to ensure data integrity. Additionally, when combined with ECC, it can be used to provide digital signatures using the elliptic curve digital signature algorithm (ECDSA). Its programmable initial value register makes it also suitable for generation of pseudo random numbers, which are often required by various cryptographic operations.

#### **5.4.8. Communication Accelerator**

The processing of communication protocols in software is often time consuming and requires substantial power to complete, especially when complex coding techniques are used in signal processing. To support fast signal processing and to improve communication, a baseband hardware accelerator that complies with the DIN EN 13757-4 standard has been designed and implemented in the chip. The DIN EN 13757-4 is communication standard for meters and remote reading of meters specified for short range devices. It defines frequencies of 868-870 MHz with data rates ranging from 2.4 to 66.6 Kbit/s. The baseband supports 3-out-of-6 and Manchester coding defined by the standard. To improve communication in highly jammed environments, the baseband functionality is extended to support direct sequence spread spectrum (Barker 7 and Barker 11 modulation). The DSSS uses the spreading code for converting the narrow band information into wideband

information. In DSSS, each of the information bit is XOR-ed with the barker code at the point of transmission. As result, the whole information is converted into a very wide band, and the information signal remains the same. The spread spectrum improves resilience to interference and narrow-band jamming attacks, and it allows the recovery of signal when less than fifty percent of the data bits have been damaged during transmission. To save power, the baseband core implements clock gating and wake-up support. It has been designed to tolerate a clock offset higher than 2%, which is compensated during the frame reception. The implemented clock prescaler allows programming of various bitrates derived for different input clock frequencies. A detailed description of the implemented baseband is given in [165].

#### **5.4.9. Analogue-to-Digital Converter**

The connectivity to analogue sensors has been provided through an integrated analogue-to-digital convertor. The implemented ADC is a fully differential 100 kSps 12-bit successive approximation register (SAR) ADC with self-calibration capabilities. The successive approximation ADCs are generally used in data acquisition, meteorology, high speed and high resolution sensor measurements for medical, industrial and metering applications. The ADC operates with nominal supply voltage of 2.5 V. It is also possible to operate ADC at 2 V or to increase the sampling rate to 1 MSps at nominal supply voltage, both at the cost of accuracy. The differential ports of ADC have been modified to single-ended in order to reduce the total number of pads in the chip. The four channels of ADC are internally attached to a preamplifier block performing sensor signal transformation and amplification. The implementation of preamplifier block has been required by biomedical sensors (BMS) used in Telediagnostik project. Therefore, the preamplifier block is referred as the BMS block. In BMS, two ADC channels are connected to a DC-coupled amplifier that amplifies the input signal ranging from -0.5 V to 0.5 V with the factor of 2.5. These channels are used for potentiometric measurements. One ADC channel is connected to a circuit consisting of transimpedance amplifier and differential amplifier. It is used for the measurements from sensors which act as current sources. The current of -100 nA gives 2.5 V at the output. The fourth ADC channel connects to an equalizer followed by a DC amplifier that amplifies the signal with the factor of 5. It is used for measurements of AC input signals. The circuit has the transimpedance of 25 kOhms. The remaining four ADC channels have been connected directly to the output ports. The BMS preamplifier consumes significant power when active. However, in the standby mode its power drops to several microwatts.

#### **5.5. Power Consideration**

The power consumption of TNODE is addressed at different levels. The power has been considered when defining system architecture, choosing system components, and applying low power techniques to the system. The system components and the target system architecture have been defined according to the specific project requirements and the used technology process. The chosen architecture relies on a low power processor supported by a number of hardware accelerators performing application specific tasks. The designed system is a single-master, multiple-slave architecture, where all system operations are initiated by the processor. The peripherals have been organized in logic blocks initiating their requests to the processor through a prioritized interrupt

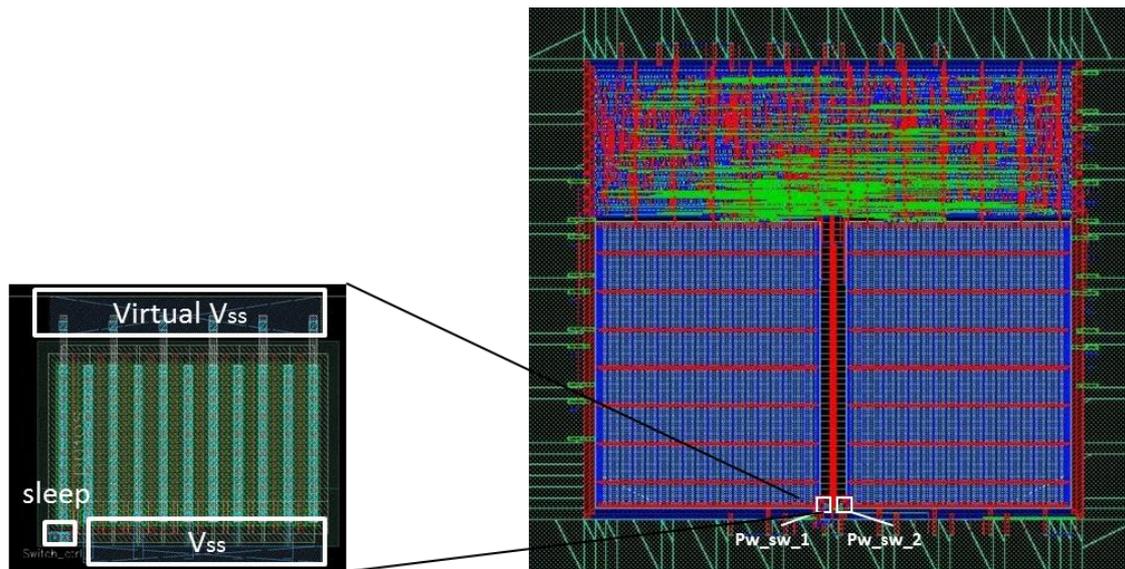
chain. The block-based system structure allows good control of activity and power consumption of each individual block. The TNODE chip implements global clock gating on each block. Thus, the processor is able to switch off the clock of an inactive peripheral. The processor itself can also enter the low power standby mode by disabling its internal clocks. The return to active state is initiated by an interrupt generated by timer or some other peripheral. The analogue components have been also designed to support the standby mode, which can be entered on the processor's request. To gain additional power savings, the implementation of advanced low power techniques that target static power consumption in idle state has been considered. The considered advanced low power techniques are DVFS, static voltage scaling and power gating. The implementation of DVFS in TNODE is found as infeasible due to the lack of processing and storage resources required by the DVFS task scheduling. Additionally, the implementation of voltage and frequency generators and voltage shifters at block level would introduce large overhead to the total power of the system. Static voltage and frequency scaling could simplify design effort, but the overhead due to additional voltage control circuits and voltage shifters does not justify the effort. Finally, the power gating methodology has been chosen as an effective measure to increase power savings in the system by controlling the power of each system component independently, without introducing significant overhead to the system performance. The presented methodology for power strategy selection has been applied in order to find an efficient power gating implementation for the developed system. To support the power gating implementation, all required power gating components have been developed and integrated into the existing design kit. Furthermore, the power-driven design flow has been established supporting step-by-step implementation of the target system. Finally, the sensor node microcontroller chip has been designed, produced and evaluated.

## **5.6. Design of Power-Gating Components**

To incorporate power gating in TNODE, basic components (power switches, isolation cells, and power control logic) have been developed.

### **5.6.1. Power Switches**

When the work on the TNODE development started, an automatic power gating insertion was still not fully supported by the available implementation tools. Therefore, early versions of power switching cells have been custom designed. The early power switches are designed as hard macro cells to be integrated manually in the design flow. The functionality of designed power switching cells is evaluated through the design of a test circuit consisting of two power-gated blocks (Figure 59). Each block in the test circuit is switched with a single custom switch transistor. The implemented power switches are large p-type MOS transistors used in the footer configuration to disconnect  $V_{SS}$  rail of block from ground. The switches are designed to have voltage drop of maximum 0.1 V for the estimated peak power consumption of block. The chip measurements have confirmed the expected difference in power consumption of power-gated block and block with only clock gating applied [126].

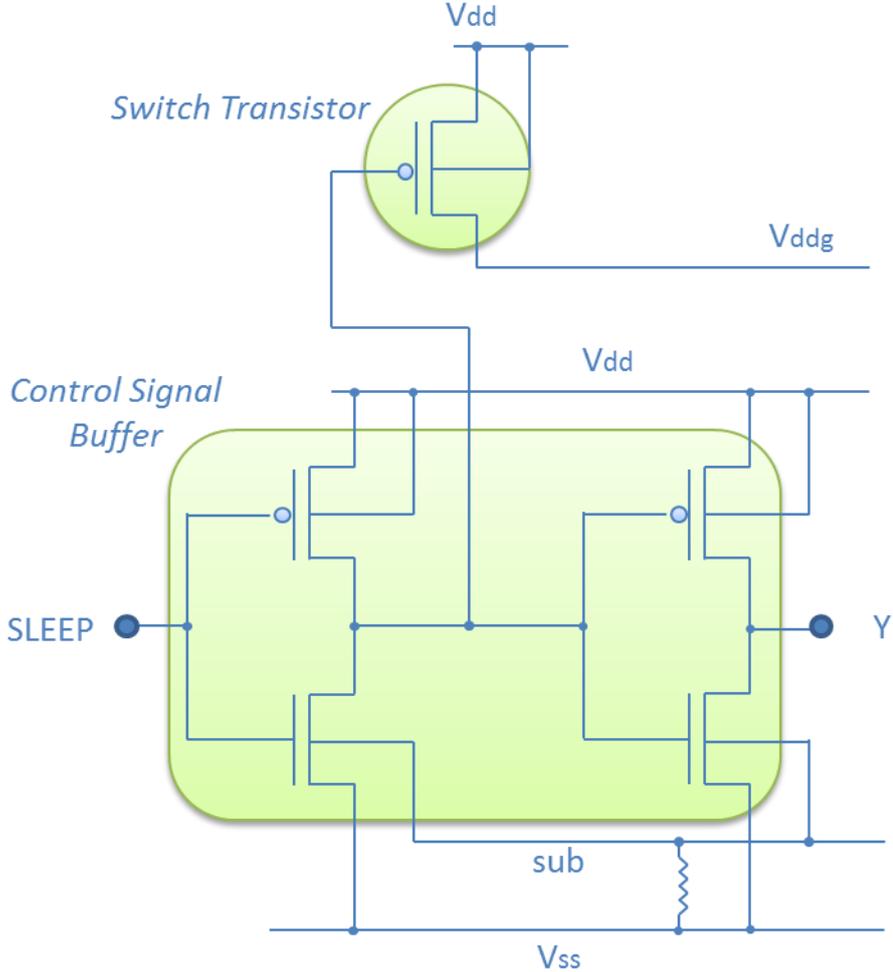


**Figure 59.** Test chip for custom power switch design.

The manual insertion of the custom-designed power switching cells has been shown as suboptimal, since an accurate characterization of the required switch sizes and the effects of power switching are difficult. Furthermore, the power-gating implementation based on a single large power switch per block is prone to high in-rush current that can cause large IR drop in the active circuit.

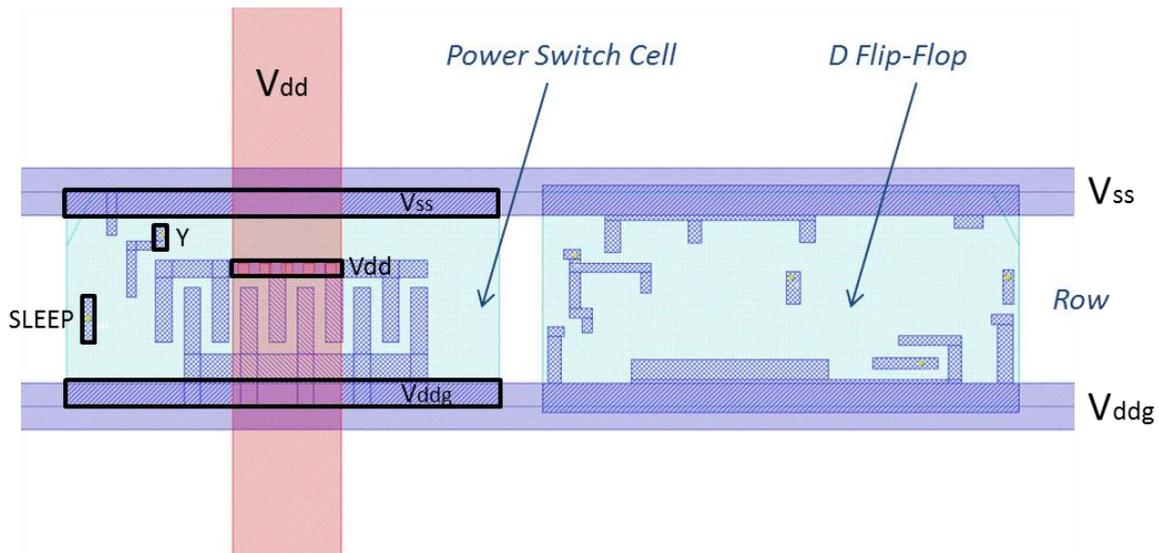
The introduction of support for automatic power gating insertion in industrial design flows (see Section 0) initiated the design of new power gating cells supported by these flows. In an automatized power gating insertion flow, it is required that the power switching cells are designed as special library cells. The design and characterization of power switching library cells is not straightforward. Since no industrial design flow for the design of power switching cells exists, these cells have to be designed in custom way. When designing power switching library cells, it has to be decided which type of the switching transistors to use, a NMOS transistor in footer configuration or a PMOS in header configuration. The target IHP technology is a p-substrate based technology having PMOS devices integrated within n-well regions and NMOS devices integrated directly in the substrate. The design of an NMOS power switch in a single well p-substrate technology would require its implicit electrical isolation from other NMOS devices placed in the same or in subsequent rows. A solution for this problem is to create an isolation trench around a NMOS transistor, but it would consequently increase the size of the switching cell, and it would make implementation quite complicated. Therefore, the PMOS power switching cell is designed. In the target technology, a PMOS header switch is naturally isolated from the other cells by its n-well. The input parameters for the design of the power switching cell have been chosen to support the maximum peak power of 300  $\mu$ A with the voltage drop not larger than 55 mV. This corresponds to the switch on-resistance  $R_{on} = 183.33$  Ohm. The transistor feature size has been analysed in relation to the off-state leakage current and the given input parameters. The analysis has found that the minimum off-state leakage is achieved for the transistor length of 260 nm and the transistor width of 20  $\mu$ m. In order to reduce the effect of the in-rush current during wake-up, a buffer is integrated on the sleep control signal of power switch. In a typical configuration, where a series of switches provide power gating of a block, the buffers introduce the delay in control signal making switches to turn on sequentially and not simultaneously.

This helps reducing the maximum rush current during the power-on. A schematic view of the designed power-switching cell is presented in Figure 60.



**Figure 60.** Schematic of designed power switching cell.

The layout of the power switching cell is designed to satisfy design rules specified for the existing digital library. Once the cell has been designed and tested, an abstract view of the cell in LEF format is generated. The cell abstract view of the cell is used in an automatic design flow to support floorplanning, automatic power switch insertion and routing. The snapshot taken from the chip layout shows the placement of the designed power switch cell inside a cell row (Figure 61). Figure 61 also shows the connection of the switch power pins to the internal power and ground rails of power domain ( $V_{ddg}$  and  $V_{ss}$ ). The connection between the switch and the vertical stripe connected to global power supply network ( $V_{dd}$ ) can be seen as well. From Figure 61, it is possible to notice the relative size of the designed power switch cell compared to a D-type flip-flop.



**Figure 61.** Power switch cell placed in a row.

The timing characterization of the designed power switch cell has been performed in two steps. In the first step, an automatic characterization tool from Cadence (ECL) has been used to characterize the timing characteristic of the control signal buffer. The characterization tool has been able to recognize the inserted buffer and to generate related timing tables for it in standard timing library file format (*lib*). The second step is to characterize the switching transistor itself, since the characterization of power switching cells is not supported by the characterization tool. The switching transistor has been characterized with its I-V characteristic. Accordingly, the simulation setup to extract the required I-V data has been designed. The obtained simulation results are used to update the previously created *lib* files. The information from *lib* files is used by the implementation tools to optimize the number of power switches and to complete the timing closure on design.

### 5.6.2. Isolation Cells

The isolation cells are used to electrically isolate the active from the power-gated regions on chip. The isolation cells can be constructed as special library cells that act as pull-up or pull-down gates, or they may have the functionality of simple AND or OR gate. At early development stage, some custom isolation cells have been designed that could be programmed to set the signal value at logic "0" or at logic "1". However, those cells have been never implemented in a system. On contrary, the already characterized library equivalents for isolation cells have been used in test implementations. Those cells have been inserted directly in RTL code. Consequently, it has to be ensured that the active isolation cells are not placed within the powered-down regions during implementation flow. Therefore, the synthesis and layout scripts have been updated with the constraints, which prevent isolation cells to be moved outside the predefined design boundaries.

### 5.6.3. Power-Gating Controller

The design of power gating controller has to satisfy the requirements for low power operation and fail-safe switching of power states. A simple power gating controller is a finite state machine that controls basic power-on and power-off operations. The complexity of the controller usually depends on the complexity of the system and specific user requirements. For example, a time-based system requires an active timer to control the duration of sleep states. This timer can be designed as part of the controller, or the controller can rely on the system timer if exists. Early test circuits designed in the scope of this work, included a programmable timer in power control unit. The timer is used to control the sleep mode of power-gated blocks. However, in the latest TNODE design, the timer is omitted from the controller. The designed system rather relies on two general-purpose system timers that are used for power gating time control.

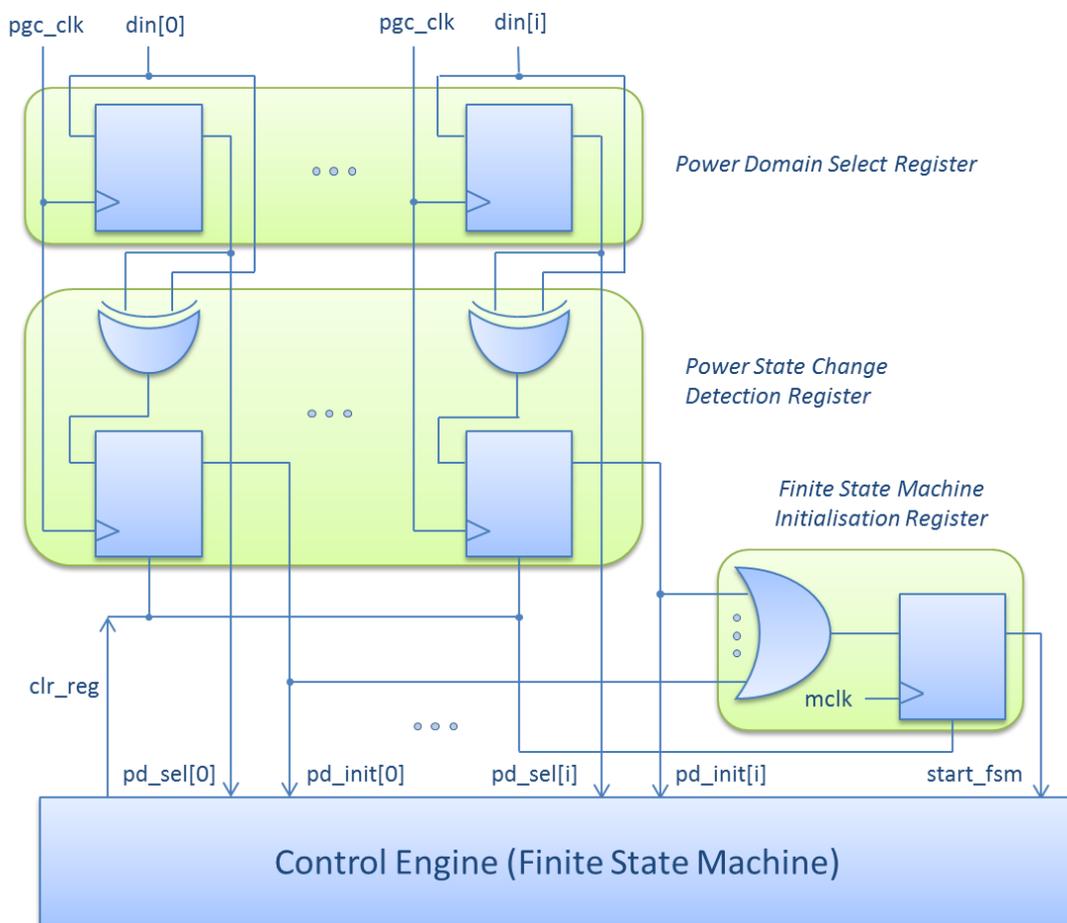
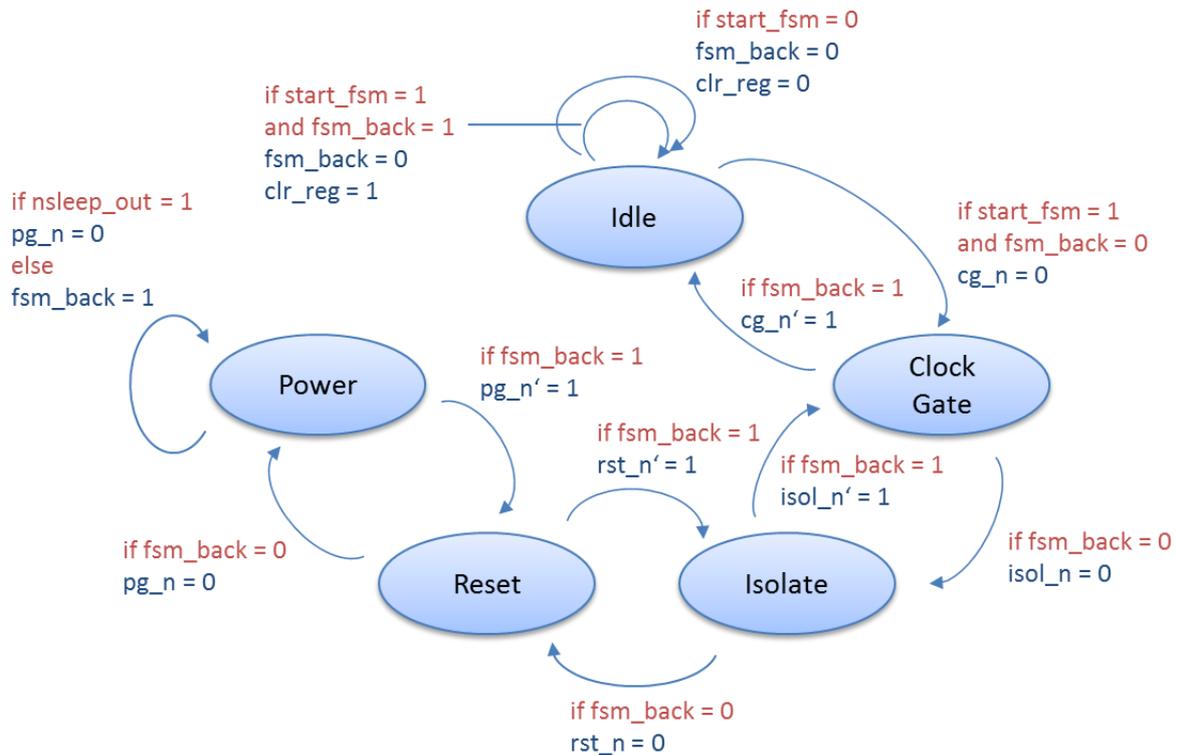


Figure 62. Power gating controller.

The implemented power gating controller is a highly efficient low power control unit capable to switch between power states in just few clock cycles. The controller consists of control engine (finite state machine), power domain select register and the circuit for the detection of power state changes in the select register (Figure 62). Each change in the power domain select register is detected, and the start of the control engine is triggered. After the power gating control register has

been initialised, the control engine needs ten clock cycles to switch off and on the power of all selected blocks. When the control engine completes power switching for selected power domains, it generates a signal (*clr\_reg*) to reset the states of state-change detection register and finite state machine initialisation register. The simplified finite state machine diagram of the control engine is illustrated in Figure 63.

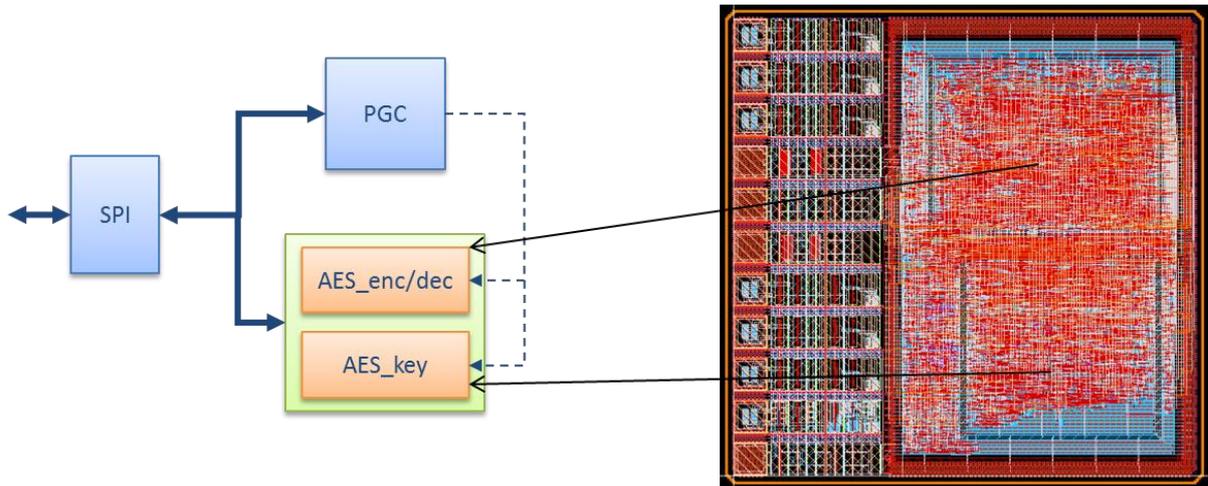


**Figure 63.** Finite state machine diagram of the control engine.

When initialised (*start\_fsm = 1*), the finite state machine of the control engine starts procedure to switch on and off the power of selected blocks. The power-down sequence starts with the gating of the clock inputs of selected blocks (*cg\_n = 0*). This is followed by the isolation of selected block output signals (*isol\_n*), the activation of reset signals (*rst\_n = 0*) and the turn-off of power (*pg\_n = 0*). The control sleep signal that propagates through the power switch chain is fed back to the controller to acknowledge the completion of power-down transition (*nsleep\_out = 1*). Before it returns to the idle state, the controller checks if some blocks have been selected for wake-up. The control signal *fsm\_back* is set, and the power-on procedure is activated. During power-on, the state machine goes back through control states turning on power (*pg\_n' = 1*), removing reset (*rst\_n' = 1*) and isolation (*isol\_n = 1'*), and enabling clock (*cg\_n = 1'*) to the blocks that are selected for wake-up. If no blocks have been selected, the controller simply goes through all states back until it reaches the idle state. In the idle state, the *fsm\_back* signal is reset, and a signal to clear the state of power state detection and fsm initialisation registers is set (*clr\_reg = 1*). This way, a single write to the state register initiates power switching of multiple blocks in a single forth and back loop of the state machine. It is also possible to selectively switch the blocks on/off by software.

#### 5.6.4. AES Test Circuit

The functionality of designed power gating components, and their integration in the established design flow, has been evaluated with the design of a test chip. The test chip contains an AES block partitioned into two power-gated blocks, a power gating controller (PGC) and a SPI interface used for chip programming (Figure 64).



**Figure 64.** AES test chip.

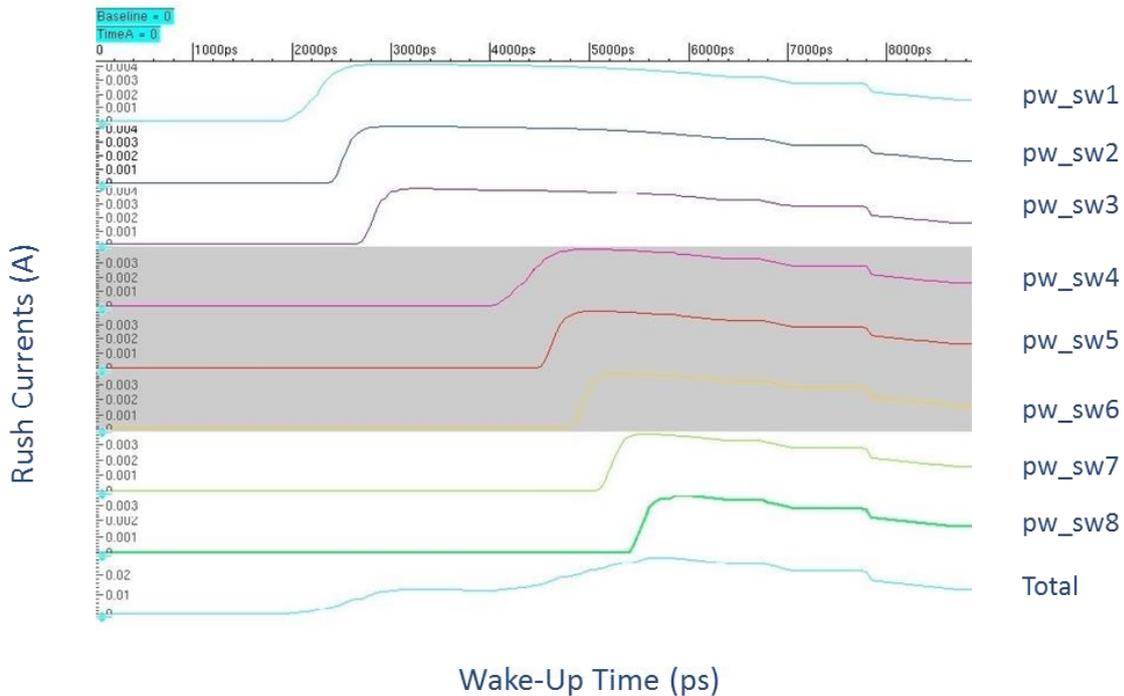
The AES block is partitioned to encryption/decryption block and key management block. This architecture enables the retention of the stored key, when the rest of the AES block is switched-off from power. The integrated power gating controller includes a 32-bit counter, and it can operate in two modes, timer mode and external mode. In the timer mode, the duration of sleep time is controlled by the time stamp set in the time stamp register of the controller. Once the timer reaches the programmed value, the controller starts procedure to wake up selected blocks. The external mode of operation relies on the external control of sleep duration. In this mode, the duration of sleep state is controlled over 1-bit power enable register. When the external control mode has been selected, the counter is deactivated. By setting the power enable register, the controller starts turning off power of the selected blocks. Once the power enable register has been cleared, the wake-up procedure is started. The selection between two operation modes is performed by writing a 1-bit mode select register of the controller. The size of the designed test chip is 1.17 mm<sup>2</sup>. The chip includes 12 IO pads, 4 power/ground pads and 8 signal pads. The functionality of the chip is verified by on-wafer tests.

During the test circuit development, the power-related parameters of power switches have been characterized in order to generate the library files required by the power analysis tools integrated in the Cadence design tool kit. The generated libraries have been later reused for the TNODE chip development. The power switch characterization process has extracted drain saturation current ( $I_{dsat}$ ), leakage current ( $I_{leakage}$ ) and on-resistance ( $R_{on}$ ) for different levels of the control input signal. The extracted values are presented in Table 5-1.

**Table 5-1.** Electrical parameters of a power switching cell.

Control Voltage (V)	$I_{dsat}$ (A)	$I_{leakage}$ (pA)	$R_{on}$ (Ohm)
1.25	0.0009724	9.495	410.7
1.75	0.002434	11.84	263.4
2.25	0.004241	14.66	207.2
2.5	0.005233	16.29	190.3
3	0.009587	20.1	167.2
3.5	0.009587	24.92	152

The estimated on-resistance of 190.3 Ohm for the input level signal of 2.5 V correlates good to the input parameters for the switch design ( $R_{on} = 183.33$  Ohm). The power analysis is performed to estimate wake-up times and to calculate the rush currents in power gated blocks.



**Figure 65.** Analysis of rush current.

Figure 65 presents the plotted results of the rush currents analysis for the power domain belonging to the crypto engine block of the AES chip. The plot diagram shows the rush currents of daisy-chained power switches and the total rush current. The maximum measured rush current is 29 mA, and the measured wake-up time is 9.7 ns. The measured peak currents of single switches in the chain are in the range from 3.6 mA (pw\_sw8) to 4.18 mA (pw\_sw1). The peak current levels in power switches are below the maximum measured saturation level of 5.2 mA (see Table 5-1). The wake-up time of switch depends on its position in the chain. The measurements have shown that the first from eight switches turns-on at 1.44 ns and the last at 4.73 ns. This gives approximately 0.41 ns

delay between two adjacent switches. The results from the conducted analysis give a good indication of the impact of rush currents in a power-gated block in relation to the required number of power switches. The increased number of power switches would reduce the wake-up time, but they would increase the maximum rush current as well. For the conducted analysis the average power of a power switch is equal to:

$$P_{sw} = I_{sw} \cdot V_{sw} = \int_{t_{on}}^{t_{wakeup}} I(t) dt \cdot \int_{t_{on}}^{t_{wakeup}} V(t) dt \quad (5.1)$$

Where  $I_{sw}$  and  $V_{sw}$  represent the average current and voltage levels during the wake-up transition.  $t_{on}$  is the time required for control signal to arrive to the first switch in the chain, and  $t_{wakeup}$  is the time at which the voltage level on virtual rail becomes stable. If the change of current and voltage levels during power transition time is assumed to be linear, then expression (5.1) can be approximated with the average values as:

$$P_{sw} = \frac{(I_{on} + I_{wakeup}) \cdot (V_{wakeup} + V_{on})}{4} \approx \frac{I_{on} \cdot V_{dd}}{4} \quad (5.2)$$

In the last equation  $I_{on}$  is equal to  $I_{dsat} = 4$  mA,  $V_{on} = 0.05$  V,  $V_{wakeup} = V_{on} = 2.45$  V, and  $I_{wakeup}$  is the average leakage current per switch ( $I_{wakeup} \ll I_{on}$ ). From the last equation, the average switching power of switch is estimated to 2.5 mW.

The total rush energy of wake-up transition depends on the average dynamic power of switch ( $P_{sw}$ ), average transition time per switch ( $t_{sw}$ ), and the number of switches ( $n_{sw}$ ). It can be calculated as:

$$E_{rush} = \sum_{n_{sw}} P_{sw} t_{sw} \quad (5.3)$$

Where  $t_{sw}$  is the wake up time of a switch. The average wake-up time per switch is the mean value of the shortest and the longest wake-up period, i.e., the wake-up time period of the first and the last switch in the chain. If the delay between two switches is approximated with  $t_{del} = 0.4$  ns and the total wake-up time is  $t_{wakeup} = 10$  ns, then, for a chain of 8 switches, the average wake-up time can be expressed as (Equation (3.29)):

$$t_{sw\_av} = \frac{t_{wakeup} + (t_{wakeup} - n_{sw} t_{del})}{2} = 8.4 \text{ ns} \quad (5.4)$$

The total energy consumption per single power transition due to rush currents is a function of the average wake-up time of switch ( $t_{sw\_av}$ ), the optimal number of switches in block ( $n_{sw}$ ), and the average dynamic power consumption of switch ( $P_{sw}$ ) (Equation (3.30)):

$$E_{rush} = P_{sw} \cdot t_{sw\_av} \cdot n_{sw} = 2.5 \text{ mW} \cdot 8.4 \text{ ns} \cdot 8 = 168 \text{ pJ} \quad (5.5)$$

## 5.7. Application of Power Saving Methodology to Sensor Node Design

The application of the methodology presented in Chapter 3 is demonstrated in the design of the sensor node chip, which architecture is described in Section 5.4. The goal of the methodology is to find an efficient power-gating partitioning for the selected design architecture and the envisioned application scenario.

### 5.7.1. Selection of Power-Gating Candidates

The first step of the energy analysis is to create the database of average dynamic and static power consumptions of main system components and power gating elements. The system is partitioned to its functional block units as illustrated in Figure 56. The Table 5-2 shows the estimated power values for sensor node system components.

**Table 5-2.** Characterization of dynamic, static and internal (clock-network) power of sensor node components.

Chip component	Dynamic Power (mW)	Static Power ( $\mu$ W)	Clock-Network Internal Power (mW)
ECC	6.2289	16.6977	6.0940
Baseband	0.5459	5.5354	0.4564
SHA-1	1.8459	3.9574	1.7140
CPU	2.2569	3.7135	0.9131
AES	0.8789	2.7203	0.8348
UART	1.0892	1.1634	1.0470
Flash Controller	0.7165	1.1329	0.5511
Timer (32 bit)	0.1610	1.1167	0.0933
Clock Module	0.1659	0.8777	0.1032
Timer (16 bit)	0.1233	0.7817	0.0913
P1/P2	0.1685	0.2589	0.1021
SPI	0.1237	0.1657	0.0993
P3/P4	0.0808	0.1261	0.0467
Bus Control	0.0258	0.1220	0.0069
<b>Total</b>	<b>15.8734</b>	<b>40.0835</b>	<b>13.4483</b>

The estimation assumed the static probability for the gate inputs to be 0.5 (50% of the signals are at “logic 1”) and the toggle rate of  $0.1 \cdot \text{clk}$  (the signals are switching once every ten clock cycles). The results are obtained for the worst case operating conditions ( $T = 125 \text{ C}$ ,  $V_{\text{dd}} = 2.25 \text{ V}$ ), and the clock frequency of 20 MHz. The table shows estimated dynamic and static power of system components. The static power is power of an inactive block whose gates are not switching. It is caused by the cell leakage power. The dynamic power is power dissipated by an active block. The estimated dynamic power consists of switching and internal power. The switching power is power dissipated during charging and discharging of the load capacitances at cell outputs. The internal power is the power dissipated within the cell boundaries. Those two components of dynamic power are not considered

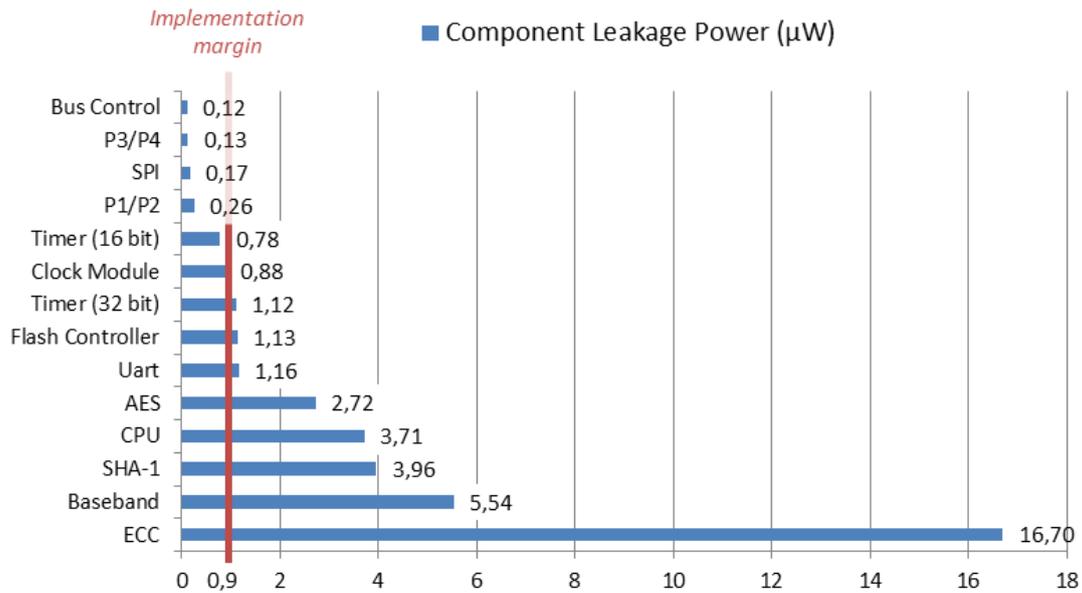
separately. However, the internal power of the block clock network has been estimated. The estimated clock-network internal power represents dynamic power of a block in idle state whit active clock. The values of clock-network internal power are used to estimate impact of the clock gating on the system energy.

To analyse the impact of power gating to specific block, it is required to analyse the contributions of power gating components to the total energy consumption of the component. The total dynamic power of power gating controller is estimated to 34.67  $\mu\text{W}$  at 20 MHz. Its cell leakage power is estimated to 70 nW. To estimate the power contribution of isolation cells, it has been assumed that each block contains 20 output ports requiring isolation (16 data output ports and four control signals). Thus, the power contribution of the isolation per block is found to be equal to the power of 20 AND gate equivalents. The dynamic power of isolation logic depends on the toggle rate of the block output signals. The default toggle rate of 0.1 used for dynamic power calculation of system blocks is found as too pessimistic for the estimation of isolation logic power. For example, in the case of ECC, the outputs are updated every 14000 clock cycles. The AES updates its outputs in more than 100 clock cycles (66 for encryption/decryption and at least 40 more to move the data to AES). The baseband and UART blocks have their toggle rates lower than 0.001. This makes dynamic power contribution of the isolation logic very low. Therefore, the dynamic power of isolation logic is estimated for the toggle rate of 0.01. The dynamic power of power switch is characterized by the average power consumption during the wake-up time. The same approach has been used as in previous section (Equation (5.2)). Table 5-3 shows the results of power characterization for different power gating components.

**Table 5-3.** Characterization of dynamic and static power of power gating components.

Chip component	Dynamic Power ( $\mu\text{W}$ )	Static Power (nW)
Power Controller	34.6467	70.300
Isolation Cells per Block	0.522	17.500
Power Switch	2500	0.016

The implementation margin for low power block candidates is now defined as  $10 \times$  (total cell leakage power of characterized power-gating components), and it is estimated to 0.9  $\mu\text{W}$ . All blocks having their leakage power less than the implementation margin are discarded as power-gating candidates. High duty cycle components, the processor, the Flash controller and the 32-bit timer, are not considered for power gating implementation. The block candidates for the implementation of power gating have been selected by comparing the estimated leakage power of sensor node system components to the established implementation margin. Finally, the selected blocks identified as good power-gating candidates are crypto blocks, baseband and UARTs. All these blocks have their leakage power higher than the implementation margin (Figure 66).



**Figure 66.** Prototyping of power gating block candidates.

### 5.7.2. Determination of Break-Even Point

To determine the break-even points of power gating candidates, it is required to estimate the energy overhead introduced by power gating implementation on selected blocks. The energy overhead of the power gating implementation is determined by the required number of power switches per block and related turn-on time. The required number of switches depends on peak active current of block. The peak active current of block has been estimated in static power analysis for the clock period of 50 ns (20 MHz). The turn-on time is determined by the effective load capacitance of block and the estimated capacitance of its power grid. To estimate the block grid capacitance, an initial floorplan has been created for each block and analysed in the layout tool. Alternatively, it is also possible to create an experience-based relation between the grid capacitance and the estimated block size. In this analysis, the power switch prototyping is done using the design tool support. The prototyping results give the optimal number of power switches per block and the related turn-on time. The obtained data are stored in the IP database, remaining available for reuse in future designs. The results of power switch prototyping for selected power-gating candidates are presented in Table 5-4. The performed analysis has assumed following switch parameters:  $I_{dsat} = 0.005$  A,  $R_{on} = 190$  Ohm, and  $I_{leak} = 20$  pA. Also, it is assumed in the analysis that all switches turn-on at the same time.

**Table 5-4.** Optimal number of power switches and turn-on time of components.

Chip component	Optimal Number of Power Switches	Turn-On Time (ns)	Peak Rush Current (mA)
Baseband	4	35.0	20
ECC	38	18.2	190
UART	2	23.3	10
AES	5	22.5	25
SHA-1	5	25.7	25
AES+SHA-1	12	21.7	60

By using Equations (3.29) and (3.30), it is possible to calculate the average turn-on time per switch and the energy contribution of rush current during the wake-up for each block. To calculate the dynamic power contribution of power gating controller, the controller is considered to take 10 clock cycles for power-up and 10 clock cycles for power down. Therefore, its energy contribution to single power transition is equal to 3.445 pJ ( $f = 20$  MHz,  $P_{dyn} = 34.64$  mW). This value has been added to the estimated energy contributions of rush current in order to calculate the break-even points. The dynamic power contribution of isolation cells during power up is minimal, and it does not make large impact on the total transition energy overhead. From Equation (3.31), the break-even point is now calculated as:

$$t_{breakeven} = \frac{E_{rush} + E_{pgc}}{P_{cleak} - P_{ohleak}} \quad (5.6)$$

Where  $E_{rush}$  is the energy contribution due to rush currents,  $E_{pgc}$  is the overhead introduced by the power gating controller,  $P_{cleak}$  is the total cell leakage power of the power-gated block, and  $P_{ohleak}$  is the leakage overhead of the power gating implementation (0.09  $\mu$ W). The estimated break-even time for the selected components is given in Table 5-5.

**Table 5-5.** Energy contributions of rush current and break-even points of the selected system components.

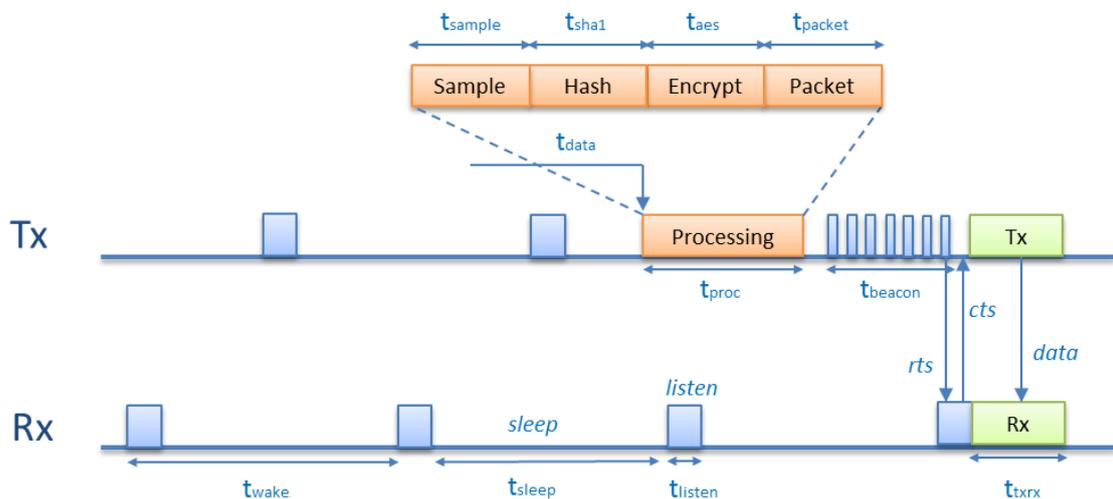
Chip component	Energy Contribution of Rush Current (pJ)	Break-Even Point (us)
Baseband	350	64.90
ECC	1729	104.31
UART	116.5	111.74
AES	281.25	108.23
SHA-1	321.25	83.95
AES+SHA-1	651	99.34

To find the energy efficiency of a particular low power implementation, it is required to define the target application and perform the profiling of system and block activity. Generally, to benefit the

power, the sleep time intervals of selected block candidates must be larger than the estimated break-even time. The longer are sleep intervals the higher are energy savings.

### 5.7.3. Definition of Target Application

The definition of the TNODE target application followed the scenario defined by the Telediagnostik project, where sensor nodes are placed within the biochemical reactors and programmed to collect data in predefined time intervals. In this scenario, each sensor node samples data after the time  $t_{data}$  and transmits the data to the neighbouring node or directly to the sink. To maintain the synchronisation, the nodes wake up after the predefined wake-up time interval  $t_{wake}$  and listen for the time  $t_{listen}$  if the data transmission has been initiated. If no data is available during the time  $t_{listen}$ , a node returns to sleep. The time between two synchronisation intervals  $t_{sleep}$ , is the time a node spends sleeping. The data transmission is initiated by periodical transmission of beacons. The transmit node sends one or a series of beacons and waits for an acknowledgment from the receiver. In the case of missing receiver, the transmit node waits until the synchronisation time expires and discards the sampled data. Otherwise, the communication between two nodes is established and the processed data is transmitted. The data processing during the time  $t_{proc}$  includes the time required for data sampling ( $t_{sample}$ ), the time needed for data hashing ( $t_{sha1}$ ) and encryption ( $t_{aes}$ ), and the time required to assemble the data packet ( $t_{packet}$ ). A typical communication pattern between two nodes, the one that transmits the data (Tx) and the other that receives the data (Rx), is illustrated in Figure 67.



**Figure 67.** Application scenario showing a typical behaviour of the transmitting (Tx) and receiving node (Rx).

In the target application, the time  $t_{data}$  is considered to be much longer than the time  $t_{wake}$ . Therefore, the time terms of task activities have been derived for the predefined data sampling time of sensor node,  $t_{data}$ .

In the time between two data samplings, each node transmits data at least once and wakes up  $n_{wake}$  times to listen for the incoming traffic, where:

$$n_{wake} = \frac{t_{data}}{t_{wake}} \quad (5.7)$$

The total time a node spends in the idle listening state during the time  $t_{data}$  is given by:

$$t'_{listen} = n_{wake} \cdot t_{listen} \quad (5.8)$$

The total sleep time during the time  $t_{data}$  is defined as:

$$t'_{sleep} = n_{wake} \cdot t_{sleep} = n_{wake} \cdot (t_{wake} - t_{listen}) \quad (5.9)$$

The time for the beacon transmission is in the worst case equal to  $t_{wake}$ . In average, the beacon sending time can be approximated with:

$$t_{beacon} = \frac{t_{wake}}{2} \quad (5.10)$$

During this time, node sends a number of beacons ( $n_{beacon}$ ), where the transmission time of single beacon is  $t_{sb}$ . Since a single beacon transmission period is followed by the period a node listens for acknowledgment, the number of transmitted beacons during the time  $t_{beacon}$  can be expressed as:

$$n_{beacon} = \frac{t_{beacon}}{2t_{sb}} \quad (5.11)$$

The time  $t_{txrx}$  is the time required for data transmission at the sender node, i.e., the data reception at the receiver node. The total time a node spends transmitting and receiving data during the time  $t_{data}$  is given as:

$$t'_{txrx} = (n_{tx} + n_{rx}) \cdot t_{txrx} \quad (5.12)$$

In the last equation,  $n_{tx}$  and  $n_{rx}$  represent the total number of packet data transmissions and packet data receptions during the time  $t_{data}$ , respectively. These numbers depend on specific network topology and packet loss rate. In a real-life application, a node will encounter some packet loss due to the channel imperfection. The packet loss requires data retransmission that can be defined by the retransmission factor  $r$ :

$$r = \frac{n_r(t)}{t} \quad (5.13)$$

The  $n_r(t)$  is the number of retransmissions during the time  $t$ . In the time  $t_{data}$ , where only a single packet is transmitted by a sensor node, the retransmission factor is equivalent to the packet error rate  $PER$ :

$$r' = \frac{n_r(t_{data})}{t_{data}} = \frac{PER(\%)}{100} \quad (5.14)$$

If the number of forwarded packets during the time  $t_{data}$  is  $n_{fw}$ , then the expression (5.12) can be written as:

$$t'_{txrx} = (1 + r') \cdot (1 + 2n_{fw}) \cdot t_{txrx} \quad (5.15)$$

The member  $2n_{fw}$  in the last equation is due to the fact that each packet forwarding action takes  $t_{txrx}$  time for single packet transmission and an equal time for its reception. Each transmission of forwarded packet requires new beacons sequence, so the time for beacon transmission during the time  $t_{data}$  can be expressed as:

$$t'_{beacon} = (1 + n_{fw}) \cdot t_{beacon} \quad (5.16)$$

The data processing time  $t_{proc}$  is a sum of the data sampling time, data encryption time, data hashing time, and time required for packet assembly and calculation of route:

$$t_{proc} = t_{sample} + t_{shal} + t_{aes} + t_{packet} \quad (5.17)$$

In the last equation,  $t_{sample}$  is the time required to sample the data from an external sensor attached to a serial or ADC port,  $t_{shal}$  is the time required to perform a single hash function with SHA1 accelerator,  $t_{aes}$  is the time required to encrypt the sampled data in AES block, and  $t_{packet}$  is the time the processor takes to assemble the data packet and to calculate the route. The distribution of public key is performed only once during the network initialisation process. After the initialisation phase, the ECC block remains inactive until the public key update request is initialised by the end user. In the target application, the UARTs are considered to be inactive all the time. The data sampling and packet processing is done only once during the time  $t_{data}$ , so the total processing time during the time  $t_{data}$  is:

$$t'_{proc} = t_{proc} \quad (5.18)$$

Now, the total activity time of sensor node during the time  $t_{data}$  can be approximated as:

$$t_{data} \cong t'_{sample} + t'_{shal} + t'_{aes} + t'_{packet} + t'_{listen} + t'_{txrx} + t'_{beacon} + t'_{sleep} \quad (5.19)$$

#### 5.7.4. Extraction of Activity Profiles

The time terms in Equation (5.19) represent the accumulated time a sensor node spends in each of its operation modes, where operation modes are defined as *data sample*, *hash calculation*, *data encryption*, *packet processing*, *idle listening*, *data transmission*, *beacon transmission* and *sleep mode*. The total accumulated time the system spends in a single operation mode can be expressed as the product of task duration and number of task occurrences during the time  $t_{data}$ :

$$t'_{task} = n_{task} t_{task}, \quad t_{task} = \{t_{sample}, t_{shal}, t_{aes}, t_{packet}, t_{listen}, t_{txrx}, t_{beacon}, t_{sleep}\} \quad (5.20)$$

The duration of each task is defined by the clock frequency and the number of clock cycles required for the task completion. In this analysis, the target clock frequency of the processor is 1 MHz. This corresponds to the clock period of 1  $\mu$ s. The defined data sampling rate is  $t_{data} = 10$  s.

For the given sampling rate, it is required to define the duration of the idle listening period  $t_{listen}$  and the scheduled wake-up period  $t_{wake}$ . The idle listening period  $t_{listen}$  is defined by the time required to transmit a single beacon, increased for some time  $\Delta t$  to ensure that the receiver will be alert when the beacon is sent. The data transmission time  $t_{txrx}$  is determined by the selected data rate and the size of the packet. If the data have been transmitted with the maximum data rate of 66 kbps, and the packet size is defined as 256-bit (128-bit payload, 64-bit header, 64-bit preamble), the data transmission time for the given data rate is  $t_{txrx} = 3.87$  ms. A single beacon is a zero-payload packet of 128 bits that takes 1.93 ms to transmit. The idle listening period  $t_{idle}$  is now defined by the single beacon transmission period ( $t_{sb}$ ) increased for  $\Delta t = 0.07$  ms defined by the user, so that  $t_{idle} = 2$  ms. In the given application,  $t_{beacon}$  is the function of  $t_{wake}$  (Equation 5.10). By increasing  $t_{wake}$ ,  $t_{beacon}$  will increase as well. Therefore it is required to find an optimal time  $t_{wake}$  for which the total accumulated time  $t_{acc}$  the system spends in idle listening mode and beacon transmission mode is minimal. If the total time  $t_{acc}$  is given as the function of  $t_{wake}$ , finding its minimum gives the optimal schedule time for the given application. From Equations (5.7), (5.8) and (5.10), the time  $t_{acc}$  is:

$$t_{acc} = f(t_{wake}) = t'_{listen} + t_{beacon} = n_{wake} t_{listen} + \frac{t_{wake}}{2} = \frac{t_{data}}{t_{wake}} t_{listen} + \frac{t_{wake}}{2} \quad (5.21)$$

The last expression can be seen as the function  $f(x)$  given by:

$$f(x) = \frac{a}{x} + bx \quad (5.22)$$

The minimal point of the function  $f(x)$  can be calculated from:

$$\frac{\partial f(x)}{\partial x} = 0 \quad (5.23)$$

From the last equation, the variable  $x$  becomes:

$$-\frac{a}{x^2} + b = 0 \Rightarrow x = \sqrt{\frac{a}{b}} \quad (5.24)$$

Coming back to Equation (5.21), the optimal wake-up time for the given application is:

$$t_{wake} = \sqrt{2t_{data}t_{listen}} = 200 \text{ ms} \quad (5.25)$$

The total data sampling period with ADC takes 20 clock cycles. Ten clock cycles are required by the ADC processing and additional ten clock cycles are required to configure the ADC, move the sampled data to memory and put the ADC to standby mode. The hash calculation in the SHA1 block requires 64 clock cycles to copy a 512-bit data block to SHA-1, 160 clock cycles to perform the hash operation and 20 clock cycles to read the hashed data. For the data encryption with AES, 16 clock cycles are required to move the 128-bit key to AES and 16 clock cycles to move the 128-bit data block. The encryption takes 66 clock cycles, and the data reading takes additional 16 clock cycles. The processing of the packet is done by the CPU. The time to assemble the packet from the encrypted 128-bit data payload is estimated to 100 clock cycles. The beacon transmission time  $t_{beacon}$  is determined by the scheduled wake-up time (Equation (5.10)). For the  $t_{wake} = 200$  ms, the beacon time  $t_{beacon}$  is 100 ms. During this time a node repeatedly sends beacons and listens for acknowledgement,

where the single beacon transmission time and the time waiting for an acknowledgment are the same,  $t_{sb} = 1.93$  ms. A node sends approximately 25 beacons during the 100 ms beacon time. The sleep time  $t_{sleep}$  is given by Equation (5.9), and it equals to 198 ms. The total number of scheduled wake-up events during the time  $t_{data}$  is  $n_{wake} = t_{data}/t_{wake} = 50$ . The duration time of different system tasks and the active blocks involved in the task processing are summarized in Table 5-6.

**Table 5-6.** Duration time and block activity of system tasks.

TASK	Duration Time ( $\mu$ s)	Active Blocks
data sample ( $t_{sample}$ )	20	CPU, Flash, RAM, ADC+BMS
hash calculation ( $t_{sha1}$ )	244	CPU, Flash, RAM, SHA1
data encryption ( $t_{aes}$ )	114	CPU, Flash, RAM, AES
packet processing ( $t_{packet}$ )	100	CPU, Flash, RAM
idle listening ( $t_{listen}$ )	2000	CPU, Flash, RAM, BB
data transmission ( $t_{txrx}$ )	3870	CPU, Flash, RAM, BB
beacon transmission ( $t_{beacon}$ )	100000	CPU, Flash, RAM, BB
sleep ( $t_{sleep}$ )	198000	CPU, Flash, RAM, Timer32

The processor and memories are active during the initialisation of each task. In the time periods related to data processing ( $t_{sample}$ ,  $t_{sha1}$ ,  $t_{aes}$ , and  $t_{packet}$ ), the processor and memories are active during the entire task duration. In the time periods  $t_{listen}$  and  $t_{txrx}$ , the processor and memories are active only for a short time period to initialise the baseband. After the initialisation, the processor and memories enter the standby mode and wait for an interrupt from the baseband. During the time  $t_{beacon}$ , the processor wakes up 50 times (each 1.93 ms) to initialise the baseband either for the beacon transmission or for waiting an acknowledgment. During the sleep time, the processor and memories are active only for a short time required to initialise the timer.

In the given application, the estimated sleep time of system components is much longer than the calculated break-even point (see Table 5-5). Therefore, the implementation of power gating on the selected block candidates is shown as justified.

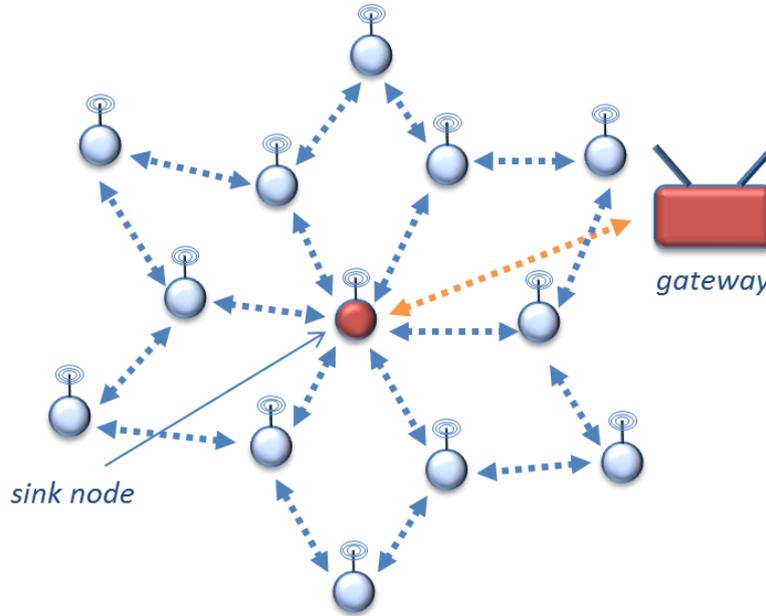
### 5.7.5. Target Network Topology and Energy Estimation

To estimate the system energy over the time period  $t_{data}$ , it is required to define the number of forwarded packets and the retransmission rate for the target network topology. The project requirements define two network topologies, a single-hop network topology (see Figure 46) as the default topology, and an alternative network topology that is applied when not all nodes are within the reach of the sink (Figure 68). The second case is taken as the target case for the analysis.

The target network topology is a two-hop network where all nodes placed at one-hop distance from the sink can forward data from maximum two peripheral nodes ( $n_{fw} = 2$ ). The two forwarded packets per one data sample are considered as the target case for the system energy analysis. The retransmission rate is usually very low, and it does not significantly impact the results of the estimation. For example, if PER is 1% (one retransmission each 100 packets), the retransmission rate

over the time  $t_{data}$  is  $r' = 0.01$ . For the conducted analysis, the impact of retransmission is ignored ( $r' = 0$ ). The energy of the system over the time  $t_{data}$  is now given by:

$$E_{data} = E'_{sample} + E'_{shal} + E'_{aes} + E'_{packet} + E'_{listen} + E'_{txrx} + E'_{beacon} + E'_{sleep} \quad (5.26)$$



**Figure 68.** Network topology defined by the application.

Using the values for the estimated dynamic and static power of system components (Table 5-2) and the estimated values for the time duration of specific system tasks (Table 5-6), it is possible to calculate the energy contribution of different system tasks and the energy of the system. The power of the implemented hard macro cells for the clock frequency of 1 MHz is given in Table 5-7.

**Table 5-7.** Power of hard macro blocks of the TNODE chip.

Hard Macro	Active Power (mW)	Standby Power ( $\mu$ W)
Flash	7.6	200
RAM	0.4	0.2
ADC+BMS	44	200
DCO	0.265	2.5

The energy contribution of the Flash memory and analogue blocks has been expected to be high, since these blocks have been still under development as the design on the latest TNODE chip started. Furthermore, these blocks have not been specifically designed for low power. The state-of-the-art low power memories and analogue circuits would make the contribution to the system energy significantly lower.

The energy contribution of a system task for the time  $t_{data}$  is calculated as the sum of the total active energy and the total leakage energy of a single task execution multiplied by the number of task occurrences ( $n_{task}$ ) during the time  $t_{data}$ :

$$E'_{task} = n_{task} \left( \left( \sum_i P_i^{dyn} + P_{leak} \right) t_{task} + E_{cpu}^{init} \right) \quad (5.27)$$

Where  $P_i^{dyn}$  is the average dynamic power of the active component  $i$  during the task execution time  $t_{task}$ .  $P_{leak}$  is the total leakage power of the system, and  $E_{cpu}^{init}$  is the energy contribution of active components during the task initialization time. This estimation assumes the clock of inactive components to be gated during the task execution.

To calculate the task initialisation energy, it is assumed that the processor takes 200 clock cycles for task initialisation. This is the time required to process an interrupt generated after the task completion, and time required to initialise the start of a new task. The task initialisation energy is given by:

$$E_{cpu}^{init} = P_{cpu}^{init} t_{cpu}^{init} = (P_{cpusys}^{dyn} + P_{leak}) t_{cpu}^{init} \quad (5.28)$$

Where total power ( $P_{cpu}^{init}$ ) during initialisation time ( $t_{cpu}^{init}$ ) is given as a sum of the dynamic power of active components during the CPU processing  $P_{cpusys}^{dyn}$  (processor, flash, flash controller, bus controller, RAM, and clock module) and total leakage power of system components  $P_{leak}$ .

The estimated dynamic power of the active system components during CPU processing is given by:

$$P_{cpusys}^{dyn} = P_{cpu}^{dyn} + P_{flash}^{dyn} + P_{fctrl}^{dyn} + P_{bctrl}^{dyn} + P_{ram}^{dyn} + P_{cm}^{dyn} = 0.5383mW \quad (5.29)$$

In the last equation, dynamic power terms of the active system components are extracted from Table 5-2.

The total leakage power  $P_{leak}$  comprises leakage power of all digital and analogue design blocks:

$$P_{leak} = P_{leak}^{dig} + P_{leak}^{an} = 40.0835\mu W + 402.7\mu W = 442.7835\mu W \quad (5.30)$$

For the given clock frequency of 1 MHz and the initialisation time of 200  $\mu s$  (200 clock cycles), the calculated initialisation energy is:

$$E_{cpu}^{init} = P_{cpu}^{init} t_{cpu}^{init} = 981.0835\mu W \cdot 200\mu s = 196.2167nJ \quad (5.31)$$

Now, the total energy contribution of system tasks accumulated over the time  $t_{data}$  is calculated.

The energy of the data sampling mode is given as:

$$\begin{aligned} E'_{sample} &= (P_{cpusys}^{dyn} + P_{leak} + P_{adc}^{dyn}) t_{sample} + E_{cpu}^{init} \\ &= (538.3\mu W + 442.7835\mu W + 44mW) \cdot 20\mu s + 196.2167nJ \\ &= 10.766nJ + 8.8557nJ + 880nJ + 196.2167nJ = 1.0958\mu J \end{aligned} \quad (5.32)$$

The energy of the hash calculation mode is given as:

$$\begin{aligned}
E'_{shal} &= (P_{cpusys}^{dyn} + P_{leak} + P_{shal}^{dyn})t_{shal} + E_{cpu}^{init} \\
&= (538.3\mu W + 442.7835\mu W + 922.95\mu W) \cdot 244\mu s + 196.2167nJ \\
&= 131.3452nJ + 108.0392nJ + 225.1998nJ + 196.2167nJ = 660.8009nJ
\end{aligned} \tag{5.33}$$

The energy of the data encryption mode is given as:

$$\begin{aligned}
E'_{aes} &= (P_{cpusys}^{dyn} + P_{leak} + P_{aes}^{dyn})t_{aes} + E_{cpu}^{init} \\
&= (538.3\mu W + 442.7835\mu W + 439.45\mu W) \cdot 114\mu s + 196.2167nJ \\
&= 61.3662nJ + 50.4773nJ + 50.0973nJ + 196.2167nJ = 358.1575nJ
\end{aligned} \tag{5.34}$$

The energy of the data packet processing mode is given as:

$$\begin{aligned}
E'_{packet} &= (P_{cpusys}^{dyn} + P_{leak})t_{packet} + E_{cpu}^{init} \\
&= (538.3\mu W + 442.7835\mu W) \cdot 100\mu s + 196.2167nJ \\
&= 53.830nJ + 44.2783nJ + 196.2167nJ = 294.325nJ
\end{aligned} \tag{5.35}$$

The energy of the idle listening mode is given as:

$$\begin{aligned}
E'_{listen} &= n_{wake}(P_{bb}^{dyn} + P_{leak})t_{listen} + n_{wake}E_{cpu}^{init} \\
&= 50 \cdot (27.297\mu W + 442.7835\mu W) \cdot 2ms + 50 \cdot 196.2167nJ \\
&= 2.7297\mu J + 44.2783\mu J + 9.8108\mu J = 56.8188\mu J
\end{aligned} \tag{5.36}$$

The energy of the data transmission mode is given as:

$$\begin{aligned}
E'_{txrx} &= (1 + 2n_{fw})(P_{bb}^{dyn} + P_{leak})t_{txrx} + (1 + 2n_{fw})E_{cpu}^{init} \\
&= 5 \cdot (27.297\mu W + 442.7835\mu W) \cdot 3.87ms + 5 \cdot 196.2167nJ \\
&= 0.5282\mu J + 8.5679\mu J + 0.9811\mu J = 10.0772\mu J
\end{aligned} \tag{5.37}$$

The energy of the beacon transmission mode is given as:

$$\begin{aligned}
E'_{beacon} &= (1 + 2n_{fw})(P_{bb}^{on} + P_{leak})t_{beacon} + (1 + 2n_{fw})2n_{beacon}E_{cpu}^{init} \\
&= 5 \cdot (27.297\mu W + 442.7835\mu W) \cdot 100ms + 5 \cdot 50 \cdot 196.2167nJ \\
&= 13.6485\mu J + 221.3917\mu J + 49.0542\mu J = 284.0944\mu J
\end{aligned} \tag{5.38}$$

Finally, the energy of the sleep mode is given by:

$$\begin{aligned}
E'_{sleep} &= n_{wake}(P_{ta32}^{on} + P_{leak})t_{sleep} + n_{wake}E_{cpu}^{init} \\
&= 50 \cdot (8.05\mu W + 442.7835\mu W) \cdot 198ms + 50 \cdot 196.2167nJ \\
&= 79.695\mu J + 4383.5566\mu J + 9.8108\mu J = 4473.0624\mu J
\end{aligned} \tag{5.39}$$

The results are summarized in Table 5-8.

**Table 5-8.** Estimated energy contribution of system tasks for the time  $t_{data}$ . The results indicate a high portion of leakage energy in the total system energy (the clock of all inactive blocks is gated).

TASK	Initialisation Energy ( $\mu$ J)	Dynamic Energy ( $\mu$ J)	Leakage Energy ( $\mu$ J)	Total Energy ( $\mu$ J)
data sampling	0.1962	0.8908	0.0088	1.0958
hash calculation	0.1962	0.3566	0.1080	0.6608
data encryption	0.1962	0.1115	0.0505	0.3582
packet processing	0.1962	0.0538	0.0443	0.2943
idle listening	9.8108	2.7297	44.2783	56.8188
data transmission	0.9811	0.5282	8.5679	10.0772
beacon transmission	49.0542	13.6485	221.3917	284.0944
sleep	9.8108	79.695	4383.5566	4473.0624
<b>TOTAL</b>	<b>70.4177</b>	<b>98.0141</b>	<b>4658.0061</b>	<b>4826.4619</b>

The estimated results for the accumulated energy of different system tasks during the time  $t_{data}$  show very high contribution of leakage energy in the total energy of the system. This is mostly due to high standby power of Flash and ADC devices that dominate the total leakage power of the system. The analysis shows that the largest portion of energy is consumed during the sleep mode. Also, the tasks related to data processing are shown to have minor impact on the system energy compared to the long-lasting communication-related tasks. Therefore, the implementation of voltage scaling techniques for the given application would have much less effect on the system power than power gating which targets specifically for the aggressive reduction of leakage power.

### 5.7.6. Selection of Power Saving Strategy

In order to select a power saving strategy for design, it is required to compare the energy efficiency of different low power approaches implemented on raw system. The raw system is the system implementation with no low power techniques applied. The specific architecture of the presented sensor node microcontroller design already includes global clock gating on all blocks. However, to emphasize the impact of clock gating on dynamic power reduction, an analysis of the raw system energy is made.

The energy of the clock-gated sensor node system, estimated in the previous section, is dominated by very high standby power of available ADC and Flash components. The energy contribution of those high-leakage components is likely to mask the impact of low power implementations on the system energy. Therefore, to make the results of the analysis fairly comparable and visible, it is assumed that both Flash and combined ADC/BMS block have their leakage power close to the industry standards (approximately 2.5  $\mu$ W per block). This assumption is applied to the energy estimation of all considered low power system implementations.

In the analysis, the raw system energy is compared with the clock gating implementation and two power gating implementations, one with all power-gating block candidates implemented as separate power islands, and the other with the AES and SHA1 blocks sharing the same power

domain. The results of the analysis are used to select a target low power implementation of the sensor node microcontroller.

#### Approach 1: Raw System

The energy of the raw system is estimated under the assumption that the idle power of an inactive block during the task execution corresponds to the estimated clock-network internal power of the block (see Table 5-2). The expression for the task energy estimation is given by:

$$E'_{task} = n_{task} \left( \left( \sum_i P_i^{dyn} + \sum_j P_j^{clk} + P_{leakage} \right) t_{task} + E_{cpu}^{init} \right) \quad (5.40)$$

Where  $P_j^{clk}$  represents the idle power of inactive block  $j$ . The idle block power corresponds to the total internal power of block registers caused by the switching in clock network. Taking into account the assumed standby power of Flash and combined ADC/BMS block, the total leakage power of all system components becomes  $P_{leak} = 47.7835 \mu W$ . The task initialisation energy of the raw system is now given by:

$$E_{cpu}^{init} = (P_{cpusys}^{dyn} + P_{idle}^{init} + P_{leak}) t_{cpu}^{init} = (538.3 \mu W + 593.7 \mu W + 47.7835 \mu W) \cdot 200 \mu s = 235.9567 nJ \quad (5.41)$$

Where  $P_{idle}^{init}$  is the idle power of all inactive components during the task initialization time. The estimated results for the energy contribution of different system tasks are presented in Table 5-9.

**Table 5-9.** Estimated energy contribution of system tasks for the raw system with no power saving techniques implemented. The standby power of Flash and ADC+BMS is assumed to be 2.5  $\mu W$  each.

TASK	Initialisation Energy ( $\mu J$ )	Dynamic Energy ( $\mu J$ )	Leakage Energy ( $\mu J$ )	Total Energy ( $\mu J$ )
data sampling	0.2360	0.9026	0.0009	1.1395
hash calculation	0.2360	0.4805	0.0117	0.7282
data encryption	0.2360	0.1744	0.0055	0.4159
packet processing	0.2360	0.1132	0.0048	0.3540
idle listening	11.7978	67.6892	4.7783	84.2653
data transmission	1.1798	13.0979	0.9246	15.2023
beacon transmission	58.9892	336.4460	23.8917	419.3269
sleep	11.7978	6690.4200	473.0566	7175.2744
<b>TOTAL</b>	<b>84.7086</b>	<b>7109.3238</b>	<b>502.6741</b>	<b>7696.7065</b>

The energy estimation of the raw system shows high contribution of dynamic energy to the overall system energy. The application of clock gating is expected to reduce the dynamic energy contribution.

### Approach 2: System with Clock Gating

The first considered power saving approach is the application of standard low power techniques to the raw system, i.e., the implementation of global clock gating on all system blocks. The estimated task initialisation energy is given by:

$$E_{cpu}^{init} = (P_{cpusys}^{dyn} + P_{leak})t_{cpu}^{init} = (583.8\mu W + 47.7835\mu W) \cdot 200\mu s = 126.2167nJ \quad (5.42)$$

By using Equations (5.31) to (5.38) and taking the updated values for the leakage power of ADC, BMS and Flash, the energy of the clock-gated system (presented in Table 5-8) is recalculated. The estimated results are presented in Table 5-10.

**Table 5-10.** Estimated energy contribution of system tasks for the system with global clock gating.

TASK	Initialisation Energy ( $\mu$ J)	Dynamic Energy ( $\mu$ J)	Leakage Energy ( $\mu$ J)	Total Energy ( $\mu$ J)
data sampling	0.1262	0.8908	0.0009	1.0179
hash calculation	0.1262	0.3565	0.0117	0.4944
data encryption	0.1262	0.1115	0.0055	0.2432
packet processing	0.1262	0.0538	0.0048	0.1848
idle listening	6.3108	2.7297	4.7783	13.8188
data transmission	0.6311	0.5282	0.9246	2.0839
beacon transmission	31.5542	13.6485	23.8917	69.0944
sleep	6.3108	79.6950	473.0566	559.0624
<b>TOTAL</b>	<b>45.3117</b>	<b>98.0140</b>	<b>502.6741</b>	<b>645.9998</b>

The estimated results show large impact of clock gating to the dynamic energy of the system. In the given application scenario, dominated by long sleep periods, the dynamic energy contribution is reduced 72 times by clock gating. The task initialisation energy is also reduced by the factor of 1.86. The results for overall system energy indicate that around 83% of the total power consumption in the clock-gated system is due to the leakage (passive and active). Around 78% of the total power consumption is the leakage power of the sleep mode (passive leakage). The dynamic energy due to the switching power of digital logic is 15.2%. To reduce further the overall system energy and minimize the impact of the leakage energy contribution, two power-gating approaches are considered.

### Approach 3: Power-Gating Implementation with Six Power Domains

The second analysed power saving approach is the power gating implementation on all identified power-gating block candidates (see Section 5.7.1). This approach assumes that the global clock gating is implemented on all always-on blocks. The energy estimation considers the reduction in leakage power due to power gating implementation, and it accounts for the introduced energy overhead.

The estimated leakage overhead of power-gating implementation is given by:

$$P_{leak}^{oh} = P_{leak}^{pgc} + n_{pd} P_{leak}^{isol} + n_{sw} P_{leak}^{sw} = 0.0703 \mu W + 6 \cdot 0.0175 \mu W + 56 \cdot 0.000016 \mu W = 0.1753 \mu W \quad (5.43)$$

Where  $P_{leak}^{pgc}$  is the leakage power of power gating controller,  $n_{pd}$  is the total number of switchable power domains in the system,  $P_{leak}^{isol}$  is the leakage power of isolation per single power domain,  $n_{sw}$  is the total number of power switches for all power domains, and  $P_{leak}^{sw}$  is the leakage power of a single power switch. The total leakage power of the system when all components are in the sleep mode is given by:

$$P'_{leak} = P_{leak}^{dig} + P_{leak}^{an} + P_{leak}^{oh} = 8.8459 \mu W + 7.7 \mu W + 0.1753 \mu W = 16.7212 \mu W \quad (5.44)$$

Where  $P_{leak}^{dig}$  is the leakage power of all always-on digital components in the system, and  $P_{leak}^{an}$  is the leakage power of analogue components and memories. The initialisation energy is given as:

$$E_{cpu}^{init} = (P_{cpusys}^{dyn} + P'_{leak}) t_{cpu}^{init} = (583.8 \mu W + 16.7212 \mu W) \cdot 200 \mu s = 120.1042 nJ \quad (5.45)$$

The accumulated task energy is calculated as:

$$E'_{task} = n_{task} \left( \left( \sum_i P_i^{dyn} + P_{leak}^{task} \right) t_{task} + E_{rush}^{task} + E_{cpu}^{init} \right) \quad (5.46)$$

Where  $E_{rush}^{task}$  is the overhead energy of rush current for specific power-gated block activation (see Table 5-5).  $P_{leak}^{task}$  is the total leakage power during the task execution time given by:

$$P_{leak}^{task} = P'_{leak} + \sum_k P_{leak}^k \quad (5.47)$$

Where  $P_{leak}^k$  is the leakage power of  $k$  power-gated components that are active during the task execution time  $t_{task}$ . The energy contributions of different system operation modes are calculated and summarized in Table 5-11.

The results show that the system with power gating implementation consumes 2 times less energy than the system with clock gating implementation. The implementation of power gating makes considerable impact on the total leakage energy of the system, reducing it 2.8 times. The impact of rush currents to the total system energy is minor, and for the given application scenario, it makes only 0.006% of the total system energy. Due to the reduction in leakage power of inactive components, the task initialisation energy is slightly reduced.

**Table 5-11.** Estimated energy contribution of system tasks for the system with six power-gated blocks. The total system energy is reduced more than twice compared to the energy of raw system.

TASK	Initialisation Energy ( $\mu$ J)	Dynamic Energy ( $\mu$ J)	Leakage Energy ( $\mu$ J)	Rush Energy ( $\mu$ J)	Total Energy ( $\mu$ J)
data sampling	0.1201	0.8908	0.0003	0	1.0112
hash calculation	0.1201	0.3565	0.0050	0.0003	0.4819
data encryption	0.1201	0.1115	0.0022	0.0003	0.2341
packet processing	0.1201	0.0538	0.0017	0	0.1756
idle listening	6.0052	2.7297	2.2257	0.0175	10.9781
data transmission	0.6005	0.5282	0.4307	0.0017	1.5611
beacon transmission	30.0260	13.6485	11.1283	0.0017	54.8045
sleep	6.0052	79.6950	165.5399	0	251.2401
<b>TOTAL</b>	<b>43.1173</b>	<b>98.0140</b>	<b>179.3338</b>	<b>0.0215</b>	<b>320.4866</b>

The conducted analysis has clearly shown the benefit of the power gating implementation in the target system. However, a large number of power domains implies significant design effort and contributes to the increase in the chip area. The following approach considers the power gating implementation with the reduced number of power domains.

#### Approach 4: Power-Gating Implementation with Five Power Domains

The increase in the number of power domains introduces additional area overhead and affects design complexity. In the given application scenario, the data hashing mode and the encryption mode always execute successively. Since the system spends relatively short time in both modes, it makes sense to consider the energy of the system implementation, where AES and SHA1 blocks share the same power domain. Wrapped in a single block, the two blocks can share the same output port, reducing the number of required isolation cells. The goal of the analysis is to determine, how the reduced number of power domains affects the energy of the system. The estimated leakage overhead for the implementation is given by:

$$P_{leak}^{oh} = P_{leak}^{pgc} + n_{pd} P_{leak}^{isol} + n_{sw} P_{leak}^{sw} = 0.0703 \mu W + 5 \cdot 0.0175 \mu W + 58 \cdot 0.000016 \mu W = 0.1587 \mu W \quad (5.48)$$

The total leakage power of the system is:

$$P'_{leak} = P_{leak}^{dig} + P_{leak}^{an} + P_{leak}^{oh} = 8.8459 \mu W + 7.7 \mu W + 0.1587 \mu W = 16.7046 \mu W \quad (5.49)$$

The initialisation energy per task is given by:

$$E_{cpu}^{init} = (P_{cpusys}^{dyn} + P'_{leak}) t_{cpu}^{init} = (583.8 \mu W + 16.7046 \mu W) \cdot 200 \mu s = 120.1009 nJ \quad (5.50)$$

By using Equations (5.43) and (5.44), the accumulated task energy and the system energy are calculated. The energy of the combined data hashing and encryption tasks is calculated as:

$$E'_{aes\_shal} = E'_{shal} + E'_{aes} \quad (5.51)$$

$$E'_{shal} = (P_{cpusys}^{dyn} + P'_{leak} + P_{leak}^{shal} + P_{leak}^{aes} + P_{shal}^{dyn})t_{shal} + E_{cpu}^{init} + E_{rush}^{aes\_shal} / 2 \quad (5.52)$$

$$E'_{aes} = (P_{cpusys}^{dyn} + P'_{leak} + P_{leak}^{shal} + P_{leak}^{aes} + P_{aes}^{dyn})t_{aes} + E_{cpu}^{init} + E_{rush}^{aes\_shal} / 2 \quad (5.53)$$

Where  $E_{rush}^{aes\_shal}$  is the rush energy due to power switching in the shared AES/SHA1 domain. If the data hashing and encryption modes are considered as separate tasks, then a half of the rush energy can be assigned to each task. The results of the energy estimation are summarized in Table 5-12.

**Table 5-12.** Estimated energy contribution of system tasks for the system with reduced number of power domains. The total system energy is slightly reduced due to reduction of the total leakage energy of the isolation logic.

TASK	Initialisation Energy ( $\mu$ J)	Dynamic Energy ( $\mu$ J)	Leakage Energy ( $\mu$ J)	Rush Energy ( $\mu$ J)	Total Energy ( $\mu$ J)
data sampling	0.1201	0.8908	0.0003	0	1.0112
hash calculation	0.1201	0.3565	0.0057	0.0003	0.4826
data encryption	0.1201	0.1115	0.0027	0.0003	0.2346
packet processing	0.1201	0.0538	0.0017	0	0.1756
idle listening	6.0050	2.7297	2.2240	0.0175	10.9762
data transmission	0.6005	0.5282	0.4303	0.0017	1.5607
beacon transmission	30.0252	13.6485	11.1200	0.0017	54.7954
sleep	6.0050	79.6950	165.3755	0	251.0755
<b>TOTAL</b>	<b>43.1161</b>	<b>98.0140</b>	<b>179.1602</b>	<b>0.0215</b>	<b>320.3118</b>

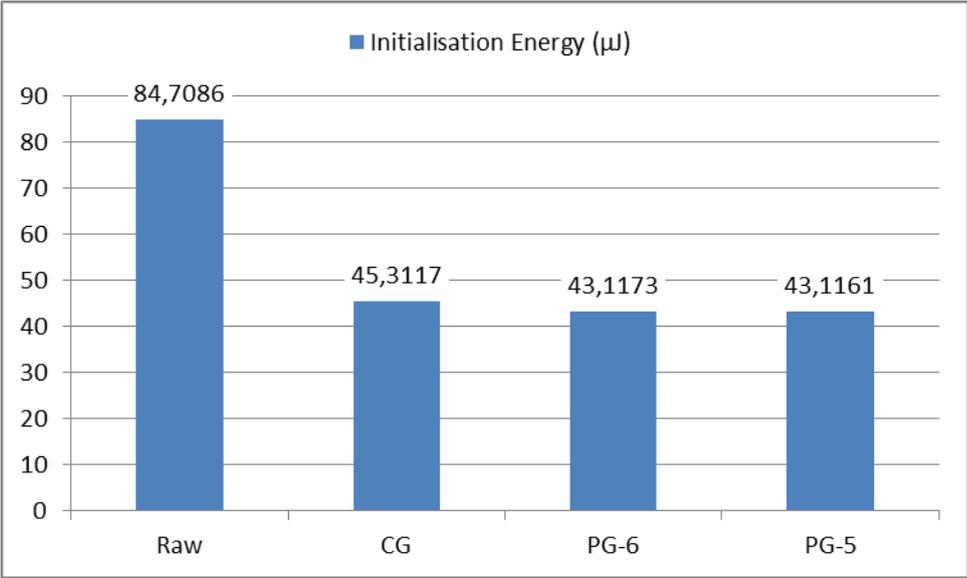
The system with the reduced number of power domains shows slightly better energy efficiency compared to the system with six power domains. As expected, the energies of hash and encryption tasks are slightly increased due to the fact that both AES and SHA1 are powered-on. However, this is compensated by the leakage energy reduction, resulting from the reduction in the number of isolation cells.

### Analysis of Results

The conducted analysis compares the energy efficiency of three different power approaches and the energy of the raw system. The application of clock gating is shown to have large impact on dynamic energy contribution. However, in the application with long sleep periods, the leakage energy of the clock-gated system remains significant. The application of power gating has contributed to the reduction of leakage energy. The comparison between two power gating implementations, one with six power domains and other with five power domains, has shown minor difference in energy efficiency, but the reduction in area and design complexity gives clear advantage

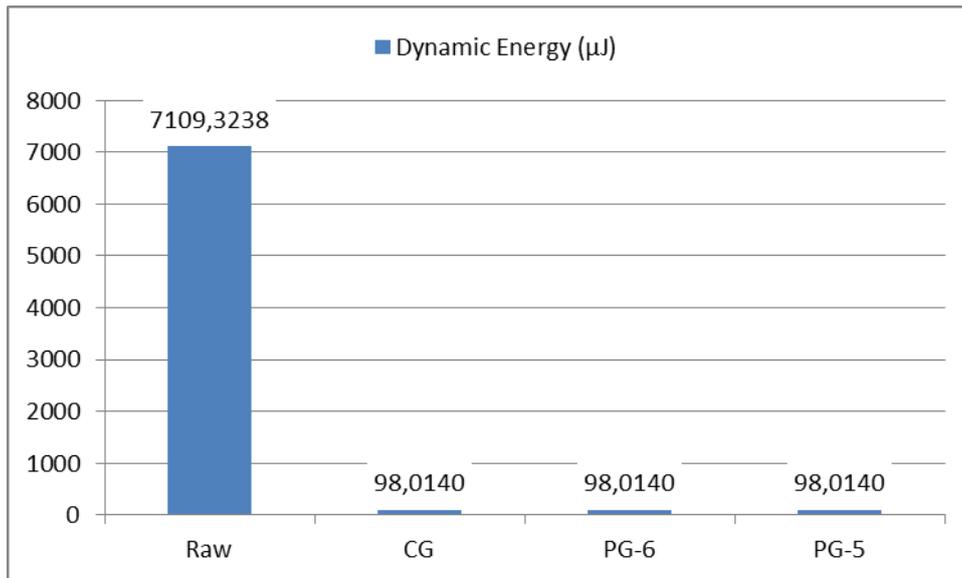
to the latter approach. The impact of the considered saving approaches on the initialization, leakage and dynamic energy is discussed in more details.

The analysis of the task initialization energy shows large impact of clock gating to the initialization energy of the raw system (Figure 69). The clock gating implementation has reduced the initialisation energy 1.8 times. The impact of power gating to the initialisation energy is minimal. This is due to short duration of initialisation tasks.



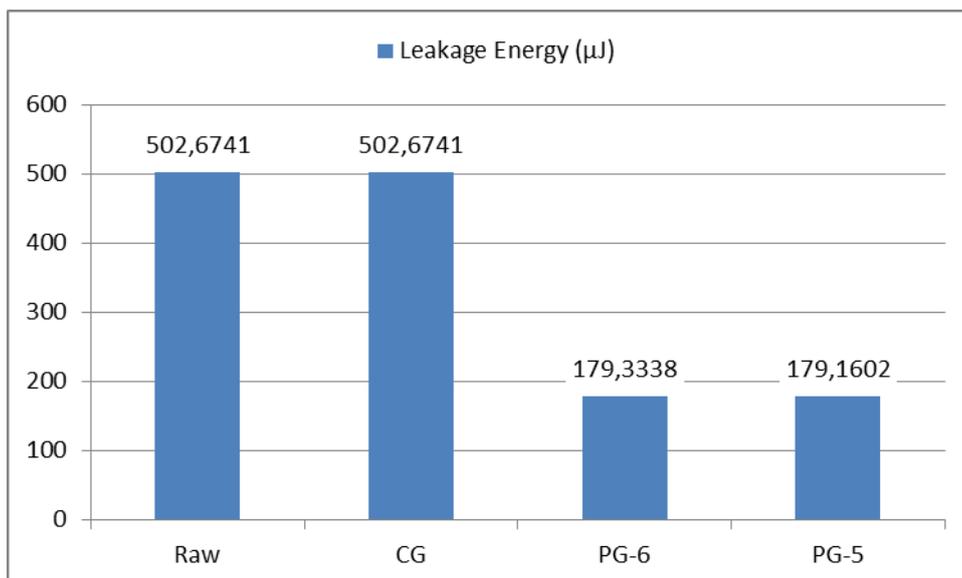
**Figure 69.** Comparison of the task initialisation energy for different power saving approaches: Approach 1 - raw system (Raw), Approach 2 - clock-gated system (CG), Approach 3 -power gating implementation with six power domains (PG-6) and Approach 4 - power gating implementation with five power domains (PG-5).

The dynamic energy contribution dominates the total energy of the raw system (Figure 70). This has been expected since the energy of the raw system is estimated for the case, where all clocks are considered to be active all the time. The implementation of clock gating reduces the dynamic energy of the system 72 times, and it reduces the total energy of the raw system 11.9 times. The implementation of power gating makes no visible impact to the dynamic energy of the system. However, some minor impact is made by the overhead energy due to rush currents. This impact is negligible in the observed application scenario.



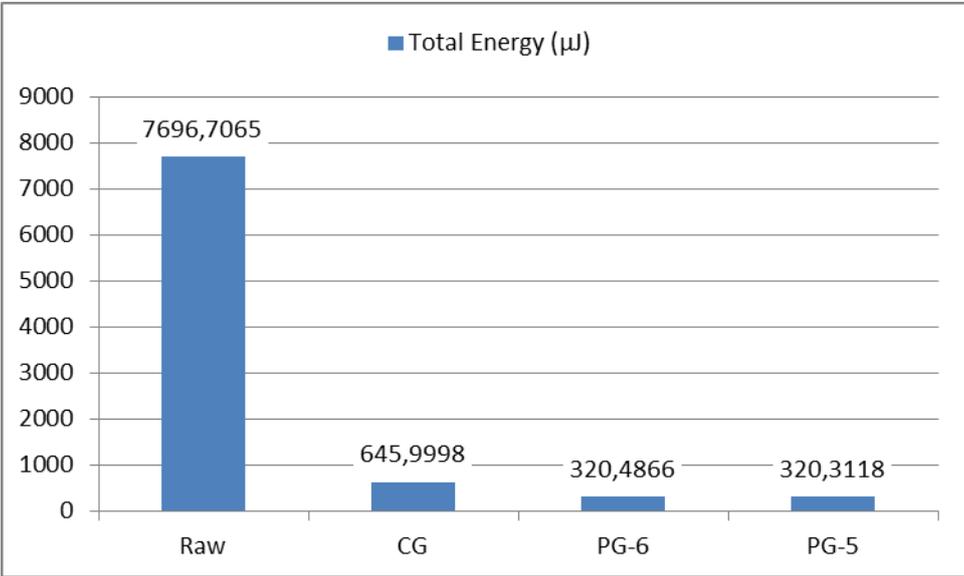
**Figure 70.** Comparison of dynamic energy contribution of different power saving approaches. The clock gating reduces dynamic energy of the raw system. Power gating makes no visible impact on dynamic energy.

The power gating implementation makes considerable reduction of the total leakage energy. In the given application scenario, the power gating implementation reduces the leakage energy 2.8 times, and it reduces the total system energy for the factor of two (Figure 71). The clock gating implementation makes no impact on the leakage energy of the system.



**Figure 71.** Comparison of leakage energy contribution of different power saving approaches. Power gating reduces leakage energy for a factor of two.

The power gating implementation with the reduced number of power domains shows similar energy efficiency as the one with six power domains. The reduction in the number of required isolation cells compensates the increase in the energy of the encryption related tasks. The comparison in the total energy of alternative system implementations is given in Figure 72.



**Figure 72.** Comparison of the total energy of different power saving approaches. A combination of clock and power gating gives the best results for the selected system implementation. The reduction in number of power domains reduces the system complexity and design effort.

The performed analysis has shown that the concurrent implementation of clock and power gating can significantly reduce the energy of the proposed sensor node microcontroller design. As the clock gating implementation makes significant reduction of dynamic energy in the system, the power gating implementation gains additional savings in the leakage energy. The application of proposed low power methodology made it possible to quantify the expected energy savings and to choose the most efficient of all considered low power implementations. The power-gating approach having five power domains, where AES and SHA1 share a single power domain, along with global clock gating implemented on all inactive blocks, is selected for the implementation in the design of the TNODE sensor node system.

## 6. Implementation and Evaluation

### 6.1. System Implementation

The system implementation flow started with the creation of power intent file for the design. The design power intent has been described in CPF format required by the Cadence implementation tools. The design power intent describes five switchable power domains and a single always-on base domain. Since each of the five switchable domains can be separately turned on and off from power, there exist  $2^5$  possible power modes for the system. However, it has been sufficient to define seven different power modes in CPF. The defined power modes are: “all blocks powered on”, “all blocks powered off”, and a single power mode for “each single block powered off”. Furthermore, power connections and power switching rules have been defined in CPF for all power domains. The CPF power intent file is given in Appendix. The synthesis is performed in a bottom-up style using the Design Compiler tool from Synopsys. Each component has been synthesized separately before an incremental synthesis is performed at the top level. The top-level synthesis is designed to prevent the isolation cells to be moved across the block level boundaries. Before designing the layout, the synthesized design is back-annotated for the correct functionality. The layout of design is created with the Encounter tool from Cadence, following the CPF flow established for low-power designs. The implemented power gating allows controlling each power domain independently. The power islands are created for the baseband unit, ECC, a block consisting of AES and SHA-1, and for each of the UART controllers. The floorplanning and distribution of power network are created manually. This has been followed by the power switch insertion and cell placement. The cell placement had to ensure that all core cells are placed inside their design and power boundaries. The power-aware clock tree insertion is applied to distribute clock buffers through the chip core area. Finally, the design has been proved for correct functionality and compliance to electrical and design rule constraints, and it has been sent for production.

#### 6.1.1. Synthesis

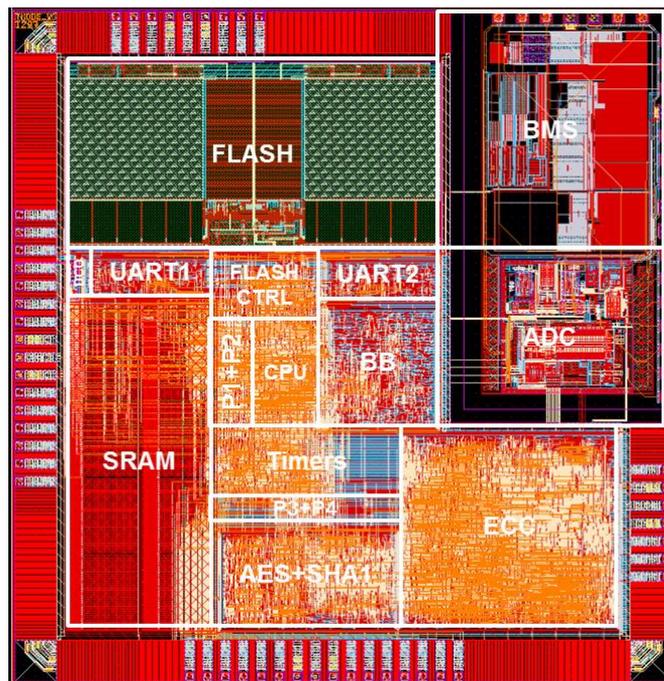
The sensor node microcontroller chip is designed for the IHP 0.25  $\mu\text{m}$  BiCMOS technology process. The synthesis has been performed for the maximum clock frequency of 20 MHz. The total cell area of the synthesized chip without pads and power switches is estimated to 20  $\text{mm}^2$ . The results for the cell area of different system components are summarized in Table 6-1. The analogue components and memories make more than 80% of the total cell area. The ECC block is the largest single digital component making 7.2% of the total cell area. The other two crypto cores, AES and SHA-1, take 3% of the total cell area. The baseband occupies around 2.6% of the total cell area. The overhead in the total cell area due to the insertion of crypto cores and baseband is estimated to 12.8%. The overhead introduced by insertion of power gating controller makes only 0.1% of the total cell area.

**Table6-1.** Cell area of sensor node components.

Chip component	Cell area (mm <sup>2</sup> )	Percent (%)
ADC+BMS	7.970	39.6
Flash	5.000	25.0
RAM	3.300	16.5
ECC	1.450	7.2
AES + SHA1	0.600	3.0
Baseband	0.530	2.6
Uart1 + Uart2	0.400	2.0
CPU	0.300	1.5
Flash Controller	0.120	0.6
Timer (32 bit)	0.110	0.6
Timer (16 bit)	0.075	0.4
Clock Module incl. DCO	0.056	0.2
SPI1 + SPI2	0.048	0.2
P1+P2	0.048	0.2
P3+P4	0.022	0.2
Bus Control	0.018	0.1
Power Controller	0.013	0.1
TNODE	20	100

### 6.1.2. Layout

The layout phase has started with creation of power islands for the power-gated blocks. This has been followed by placement of pads, hard macros and standard cells in the core area.



**Figure 73.** TNODE chip layout.

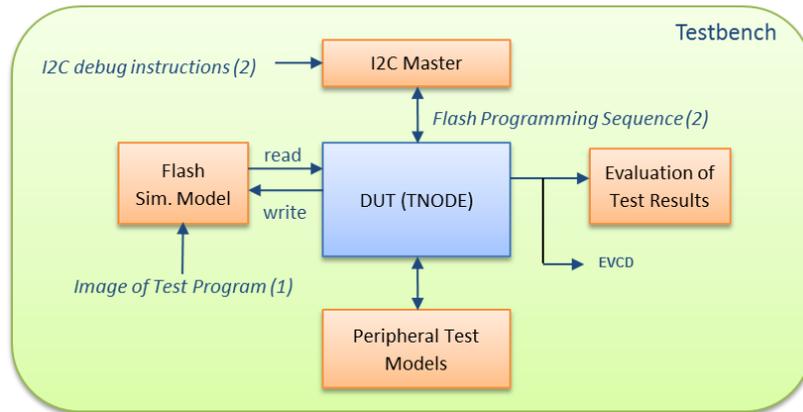
The area of the final chip is  $31 \mu\text{m}^2$ . More than a half of the chip area is occupied by hard macro blocks ( $16.27 \mu\text{m}^2$ ). The total number of power switch instances is 61, having the total size of  $0.0086 \mu\text{m}^2$ . The power switches make minor impact on the total chip area. The total number of pads is 71 (41 digital, 12 power, and 18 analogue pads). The chip design makes possible to leave some of the IO pads unconnected (for example ADC inputs) and do packaging in a standard 64-pin package frame. The chip layout is shown in Figure 73.

## **6.2. Verification and Measurements**

The verification of the sensor node chip includes the functional and power verification during design process and the testing of the produced chip. The functional verification includes simulations of digital and analogue parts and system level simulations. The simulation tests are designed to prove the overall system functionality and functionality of peripherals. The created test suite includes a set of more than 30 test programs. During power verification, the energy contribution and turn-on time of power gating implementations as well as the chip power consumption have been measured. The production tests have confirmed the correct functionality of fabricated chips and the compliance to the specified electrical and system requirements (power and performance). The final evaluation of the chip is to be performed on a dedicated hardware evaluation platform.

### **6.2.1. System Level Tests**

The TNODE simulations include a set of test programs, which execute from the Flash memory. The tests are designed to verify the peripheral functionality and the functionality of the system. In a simulation test, the image of test program is preloaded into a functional model of the Flash and applied to the system. The results of test operations are consequently written to external ports and evaluated in the testbench. The traces of signal changes during simulations are written to *evcd* (extended value change dump) files that are used as inputs in the production tests. To test the Flash functionality, a simulation model of I2C debug device (I2C master) has been designed and used to program the Flash. The created I2C Flash programming sequences verify common Flash operations, e.g., erase, write, read and sector-based operations. The same approach is used to test the complete instruction set of the processor and to generate the Flash-based test patterns required in production tests. The applied simulation approach is illustrated in Figure 74. The functionality of analogue components has been tested in analogue simulations. In digital simulations, only the functionality of ADC and DCO control logic is tested.



**Figure 74.** TNODE testbench: (1) tests executed from Flash; (2) tests executed via I2C debug port.

### 6.2.2. Turn-On Time and Rush Currents

The turn-on time of the power-gated blocks has been measured on the post-layout chip design using the VoltageStorm tool from Cadence. Table 6-2 shows the implemented number of switch cells per block and the measured turn-on times and peak rush currents.

**Table 6-2.** Measured turn-on time and peak rush current of power-gated blocks in the TNODE chip.

Chip component	Implemented Number of Switches	Turn-On Time (ns)	Peak Rush Current (mA)
Baseband	4	34.51	16.91
ECC	39	28.71	117.13
U1	2	22.56	8.74
U2	4	11.10	16.48
AES+SHA-1	12	22.90	47.92

**Table 6-3.** Measured energy contribution of the rush currents and break-even points of the power-gated blocks in the TNODE chip.

Chip component	Average Wake-Up Time per Switch (ns)	Energy Contribution of Rush Current (pJ)	Break-Even Point ( $\mu$ s)
Baseband	33.71	337	62.51
ECC	20.91	2038	122.24
U1	22.16	110.8	98.00
U2	10.30	103	91.49
AES+SHA-1	20.50	615	92.70

Table 6-3 shows the measured energy contribution due to power switching and the break-even points of the power-gated blocks in the chip.

The measured results correlate quite well with the estimated results from Table 5-4 and Table 5-5. The peak rush-currents are reduced due to insertion of buffers on the sleep control signals in power switches.

### 6.2.3. Production Tests

The production tests of the fabricated TNODE chips have been performed by wafer testing on Agilent Verigy 9300 industrial tester. The Verigy 9300 is a cycle-based industrial tester for digital circuits that supports both wafer and package testing. The tester operates with the data extracted from the *evcd* file generated in simulation. The *evcd* file keeps the value change information for all input and output signals, which are changing during the simulation test. In standard testing procedure, the input stimuli extracted from an *evcd* file are applied to chip. The output signals of the chip under test are sampled by the tester and compared to expected output values. The TNODE chip could not be driven with a test sequence externally, since it contains only an internal program memory (Flash). For that reason, the stimuli data must be written and executed from the Flash. A set of initial I2C program sequences is created and used to write the Flash with the required test program data. Each programming sequence contains the I2C instructions to erase the Flash, to write test program to the Flash, and to read and prove the written data from the Flash. The start of the test program stored in the Flash is initiated by the processor boot procedure. The boot procedure executes the test program and the sampled output data are verified by the tester. Following this approach, a set of test programs is designed and executed in an automatized test procedure. The execution of test program in each test procedure follows the Flash programming operation created for that specific test. The analogue components, ADC and DCO, could not be thoroughly tested in the automatized test procedure. However, the tests to analyse the conversion of predefined DC levels from the ADC analogue inputs has been performed. The measured ADC conversion data are compared to the expected values obtained from analogue simulations. More detailed tests of ADC have been performed on a dedicated test board using the packaged TNODE chips.

### 6.2.4. Power Consumption

The power of the chip is estimated for the post-layout design and measured on the fabricated chips. The post-layout power estimation is made with the Encounter Power System tool that is the part of the Cadence Encounter design toolset. The power estimation is done for typical operating conditions ( $T = 25^{\circ}\text{C}$ ,  $V_{\text{core}} = 2.5\text{ V}$ ,  $V_{\text{pad}} = 3.3\text{ V}$ ) and the operating frequency of 1 MHz. The performed dynamic-based power estimation has used the activity profiles extracted from the post-layout simulations. The power is estimated for different crypto operations, which exploit the functionality of the integrated crypto cores. In addition, the power during transmission and reception of the data processed by the baseband core has been estimated as well. Also, the power consumption during the data exchange over SPI is estimated. In all tests, ADC, BMS and Flash operated in active state. The results of power estimation are summarized in Table 6-4. The table shows power consumptions of chip logic for different operations.

**Table 6-4.** Post-layout power estimation results.

Chip Function	Chip Logic (mW)	ADC+BMS (mW)	Flash (mW)	Total (mW)
SHA-1 Calculation	1.15	44	7.6	52.75
AES Decryption	1.09	44	7.6	52.69
AES Encryption	1.25	44	7.6	52.85
ECC Point Multiplication	3.85	44	7.6	55.45
ECC First Point Inversion	1.94	44	7.6	53.54
ECC Second Point Inversion	2.06	44	7.6	53.66
Transmit Mode	0.82	44	7.6	52.42
Receive Mode	0.89	44	7.6	52.49
SPI	0.93	44	7.6	52.53

The power estimation results show that analogue components and the Flash memory consume most of active power. The BMS preamplifier is the largest active power consumer, but in the target application it remains off most of the time. In the standby mode, the power of ADC and BMS drops to 200  $\mu$ W. Flash, at the other hand, is active during read operations that occur constantly whenever the processor is active. However, the Flash controller is designed to put Flash into standby mode whenever it's not accessed by the processor. The standby power of the Flash is 200  $\mu$ W. As expected, the maximum estimated power consumption in digital logic has been estimated for ECC operations.

Early power estimation of the fabricated chips has been performed during production tests. The TNODE chip includes separate power supplies for core (VDDCORE), pads (VDDPAD), and analogue and mixed signal parts (VDDA and VDDM). The VDDCORE provides the power for standard logic cells, Flash, digital part of ADC, and pads. The average power consumption in different tests is measured for all voltage sources. Table 6-5 gives the results of power measurement for the selected production tests. The power is measured for nominal supply voltage levels (VDDCORE = 2.5 V, VDDPAD = 3.3 V, VDDA = 2.5 V, VDDM = 2.5V). The power results are extracted for the operating clock frequency of 1 MHz. The ADC and BMS were active during the tests.

**Table 6-5.** Results of power measurements on the fabricated chips.

Test @ 1 MHz	P <sub>VDDCORE</sub> (mW)	P <sub>VDDPAD</sub> (mW)	P <sub>VDDA</sub> (mW)	P <sub>VDDM</sub> (mW)	P <sub>TOTAL</sub> (mW)
Rx	7.17	3.90	39.25	2.575	52.895
Tx	6.86	2.50	39.40	2.575	51.335
Crypto	7.37	2.97	39.40	2.575	52.315
SPI	5.93	4.29	39.38	2.572	52.172
UART	6.44	3.75	39.39	2.572	52.152

The results in Table 6-5 present average power consumptions extracted from the measurements on 118 chips. The power measurement results show a good agreement with the estimated power results (see Table 6-4). The measured results for VDDCORE indicate the highest power consumption during crypto test. The crypto test includes the tests of all crypto blocks, and the measured power is the average power consumption for all tests. The power consumption in the pads is higher for SPI,

UART and baseband tests. This is due to the increased activity on external ports during these tests. The testing of different power-gating modes is also conducted. The correct chip behaviour has been confirmed in all test cases. Due to the relatively short duration and high activity of conducted tests, the power measurements of different power-gating modes couldn't show the real effect of power gating. Additionally, the inability of the tester to accurately measure small power changes at desired time prevents from precise chip power analysis during industrial tests. However, the power measurements of different low power modes will be measured at the TNODE evaluation platform.

### 6.2.5. Performance

The TNODE chip has been designed for the target frequency of 20 MHz. However, due to the limitations in the Flash access time, the maximum measured operating frequency is 11.4 MHz. The ECC performance of the designed chip has been compared to the results for different ECC software implementations (Table 6-6). The assumed average power consumption of the TNODE during ECC operations is approximately 10 mW at 1 MHz (ADC and BMS in standby). The results show the TNODE superiority in the effective power consumption over the software-based solutions.

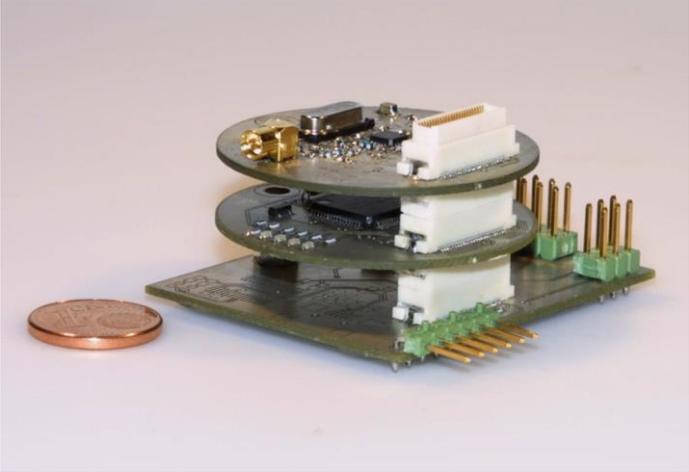
**Table 6-6.** Comparison of power consumption of several ECC implementations including TNODE.

CPU	Key Size (bit)	Frequency (MHz)	Energy (mWs)	Reference
MSP430	163	1	2790.0	[166]
MSP430	283	1	6815.0	[166]
MSP430 (periph. sleep)	163	1	261.6	[166]
MSP430 (periph. sleep)	283	1	638.9	[166]
MSP430/FPGA	571	16	0.831	[166]
MIPS R4000	163	33	7.0	[167]
Atmel ATmega128	163	7.4	120	[168]
Motorola Dragonball	163	16	135	[169]
StrongARM	163	206	3.6	[170]
Pentium II	163	400	90	[141]
MIPS 4kEP	233	33	16.49	[171]
TNODE @ 1 MHz	233	1	0.131	this work

### 6.2.6. Evaluation Platform

The further testing of the chip includes the chip evaluation on the custom designed evaluation platform [134]. The platform has been successfully used to test the previous versions of the TNODE chips. The platform provides the stack-based connection of different sensor node components into a modular system. Each component is assembled on a single printed circuit board (PCB). These PCBs are connected via a well-defined connector. The platform modularity enables the connection of the TNODE chip to different radio and sensor devices. The designed PCBs include radio transceiver, storage modules, sensors and different power supply as well as special debugging boards. The "lego-like" design of the platform enables easy replacement of sensor node components with other compatible devices. Figure 75 shows an assembled sensor node with a debugging board, the TNODE

board and a CC1101 transceiver board. The debugging board provides the power supply and the connection to the debug and control unit. The debug and control unit is actually a remote personal computer (PC) running debugging software. The PC connects to the platform via USB-to-I2C/UART converter cable.



**Figure 75.** TNode evaluation platform.

The previous TNode chips have been tested with the Chipcon CC1101 radio module and different sensors connected directly to the available IO pins of the TNode board. The platform is to be used to test the TNode6 functionality in a network environment and to measure the power in different power-saving modes. At the moment of writing of this thesis, the dedicated platform board to carry the TNode6 chip has been under development.

## 7. Conclusion

A low power methodology for selecting the most suitable power saving strategy for a specified design is presented in this thesis. The methodology extracts a system activity profile based on single block activities in order to estimate the energy efficiency of a particular implementation. The developed power estimation models account for the energy overhead introduced by implementation of the advanced low power techniques. The models consider system perspective of energy estimation, which is often omitted in literature and practice. The highest attention is paid to power gating that is considered to be the most promising technique for saving power in low duty cycle systems (defined by long sleep periods) such as sensor nodes. Power gating implies complex power estimation models, due to effects of large number of power domains and overhead circuits. The presented methodology is shown to be well suited for sensor node systems, whose activity can be precisely estimated for a given application. The activity profiles and equations are derived for a few typical WSN applications. The presented methodology is integrated in the power-driven design flow and used to design a low power sensor node microcontroller.

The energy efficiency of clock-gating implementation and two alternative power-gating implementations is compared to the energy efficiency of the non-gated system. Candidates for power-gating are identified among the system components and break-even points are calculated for each selected component. The system activity is analysed for a given application scenario and the extracted activity profile is annotated to the selected components. The results of activity profiling are used in the analysis of alternative low power system implementations. The clock-gating implementation is shown to have large impact on system dynamic energy. The implementation of power gating brought additional savings of energy. A system architecture that includes five power-gating islands and global clock gating has been selected to illustrate the proposed power saving methodology.

Custom power-gating cells have been designed and characterized as well. A test chip, having power gating implemented in the AES block, is used for cell characterization and analysis of rush currents. The same test chip is also used to verify the proposed design flow.

Several sensor node systems-on-chip (TNODE1 to TNODE6) have been designed and implemented. The latest designed chip, presented in the thesis, includes hardware accelerators for communication and security tasks, on-chip Flash memory, ADC enhanced with a preamplifier block, and clock distribution logic. The chip implements several low power techniques: power-gating, clock-gating, frequency islands, asynchronous logic as well as hardware acceleration based on power-driven hardware/software co-partitioning. The integrated power controller allows individual power control of each power-gated block. The chip implementation process has been based on the presented methodology. The microcontroller has been successfully fabricated and tested.

Finally, the power evaluation results are presented showing good correlation between estimated and measured power data for the chip. The chip will be also tested on a dedicated evaluation board. The packaged chip is to be integrated on a custom sensor board and used in field measurements.

Future work may try to improve the proposed methodology with more detailed models for energy estimation. The improved models should include the estimated power of signal wires and pads as well as the power of clock tree buffers. As the power of pads can be relatively well

characterized by using the data from pad cell libraries, the power of clock buffers has to be approximated, for example, by using more accurate wire-load models in trial synthesis. The power of clock buffers can be estimated in tools by running the trial clock tree insertion on the placed design, or it can be extracted by using experience-based methods. In this work, dynamic power of system components has been characterized by using statistical approach for power estimation. However, more accurate data for dynamic power could be obtained in dynamic power analysis. This requires development of representative simulation tests for all characterized system components.

To provide further support and easy use of the methodology in design of large-scale sensor node systems, a sensor network simulator might be extended with accurate activity profiling. It is possible to update the sensor node model of a WSN simulation tool with the information on specific system tasks. A WSN simulator might try to extract the frequency of specific system tasks during the simulation time for all network nodes. The analysis of the acquired data should give the average activity profile of a sensor node. Furthermore, it would be possible to extend the sensor node models with the information on typical duration time of specific sensor node system tasks and therewith extract the system energy terms directly from simulation data.

The implementation of power gating in future designs should consider switching off the processor from power as well. This would require implementation of retention registers in the processor register file that could save the critical data in power-down mode. The functionality of power gating controller could be also improved by adding an intelligent power mode switching based on activity monitoring. In a well-defined system, it would be possible to design a prediction-based power control mechanism that could control the power of a system without need for software support.

## References

- [1] Kim, N. S.; Austin, T. M.; Blaauw, D.; Mudge, T. N.; Flautner, K.; Hu, J. S.; Irwin, M. J.; Kandemir, M. T. & Vijaykrishnan, N. (2003), "Leakage Current: Moore's Law Meets Static Power.", *IEEE Computer* 36 (12) , 68-75.
- [2] The International technology roadmap for semiconductors, "ITRS 2005 Edition", Tech. Rep (2005).
- [3] [http://www.umc.com/english/pdf/design\\_support\\_Brochure.pdf](http://www.umc.com/english/pdf/design_support_Brochure.pdf)
- [4] Arampatzis, Th, J. Lygeros, and S. Manesis. "A survey of applications of wireless sensors and wireless sensor networks." *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation.* IEEE, 2005.
- [5] Zhao, Feng, and Leonidas J. Guibas. *Wireless sensor networks: an information processing approach.* Morgan Kaufmann, 2004.
- [6] Frank, David J. "Power-constrained CMOS scaling limits." *IBM Journal of Research and Development* 46.2.3 (2002): 235-244.
- [7] Roy, Kaushik, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits." *Proceedings of the IEEE* 91.2 (2003): 305-327.
- [8] Chandrakasan, Anantha P., Samuel Sheng, and Robert W. Brodersen. "Low-power CMOS digital design." *IEICE Transactions on Electronics* 75.4 (1992): 371-382.
- [9] Usami, Kimiyoshi, and Mark Horowitz. "Clustered voltage scaling technique for low-power design." *Proceedings of the 1995 international symposium on Low power design.* ACM, 1995.
- [10] Mutoh, Shin'ichiro, et al. "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS." *Solid-State Circuits, IEEE Journal of* 30.8 (1995): 847-854.
- [11] Wei, Liqiong, et al. "Design and optimization of dual-threshold circuits for low-voltage low-power applications." *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 7.1 (1999): 16-24.
- [12] Wu, Qing, Massoud Pedram, and Xunwei Wu. "Clock-gating and its application to low power design of sequential circuits." *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on* 47.3 (2000): 415-420.
- [13] Donno, Monica, et al. "Clock-tree power optimization based on RTL clock-gating." *Design Automation Conference, 2003. Proceedings.* IEEE, 2003.
- [14] Henzler, Stephan. *Power management of digital circuits in deep sub-micron CMOS technologies.* Vol. 25. Springer, 2007.
- [15] Emmett, Frank, and Mark Biegel. "Power reduction through RTL clock gating." *SNUG, San Jose* (2000).
- [16] Münch, Michael, et al. "Automating RT-level operand isolation to minimize power consumption in datapaths." *Proceedings of the conference on Design, automation and test in Europe.* ACM, 2000.
- [17] Kapadia, Hema, Luca Benini, and Giovanni De Micheli. "Reducing switching activity on datapath buses with control-signal gating." *Solid-State Circuits, IEEE Journal of* 34.3 (1999): 405-414.
- [18] Chen, Benjamin, and Ivailo Nedelchev. "Power compiler: a gate-level power optimization and synthesis system." *Computer Design: VLSI in Computers and Processors, 1997. ICCD'97. Proceedings., 1997 IEEE International Conference on.* IEEE, 1997.
- [19] Weste, Neil HE, and Kamran Eshraghian. "Principles of CMOS VLSI design: a systems perspective." *NASA STI/Recon Technical Report A 85* (1985): 47028.

- [20] Chang, Chih-Wei, et al. "Fast post-placement rewiring using easily detectable functional symmetries." Proceedings of the 37th Annual Design Automation Conference. ACM, 2000.
- [21] L. Benini and G. D. Micheli, "Logic Synthesis for Low Power," in Logic Synthesis and Verification, S. Hassoun and T. Sasao, Eds. Kluwer Academic Publishers, 2002, pp. 197–223.
- [22] De, Vivek, and Shekhar Borkar. "Technology and design challenges for low power and high performance." Proceedings of the 1999 international symposium on Low power electronics and design. ACM, 1999.
- [23] Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits Roy, Kaushik, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. ". " Proceedings of the IEEE 91.2 (2003): 305-327.
- [24] Sirisantana, Naran, Liqiong Wei, and Kaushik Roy. "High-performance low-power CMOS circuits using multiple channel length and multiple oxide thickness." Computer Design, 2000. Proceedings. 2000 International Conference on. IEEE, 2000.
- [25] J.-P. Colinge. Silicon-on-insulator technology: materials to VLSI. Kluwer Academic Pub, 1997.
- [26] Celler, G. K., and Sorin Cristoloveanu. "Frontiers of silicon-on-insulator." Journal of Applied Physics 93.9 (2003): 4955-4978.
- [27] Vitale, Steven A., et al. "FDSOI process technology for subthreshold-operation ultralow-power electronics." Proceedings of the IEEE 98.2 (2010): 333-342.
- [28] Shahidi, Ghavam G. "SOI technology for the GHz era." IBM journal of Research and Development 46.2.3 (2002): 121-131.
- [29] Lu, Pong-Fei, et al. "Floating-body effects in partially depleted SOI CMOS circuits." Solid-State Circuits, IEEE Journal of 32.8 (1997): 1241-1253.
- [30] Choi, J-Y., and Jerry G. Fossum. "Analysis and control of floating-body bipolar effects in fully depleted submicrometer SOI MOSFET's." Electron Devices, IEEE Transactions on 38.6 (1991): 1384-1391.
- [31] Simonen, Piia, et al. "Comparison of bulk and SOI CMOS Technologies in a DSP Processor Circuit Implementation." Microelectronics, 2001. ICM 2001 Proceedings. The 13th International Conference on. IEEE, 2001.
- [32] Zamdmer, N., et al. "A 0.13- $\mu\text{m}$  SOI CMOS technology for low-power digital and RF applications." VLSI Technology, 2001. Digest of Technical Papers. 2001 Symposium on. IEEE, 2001.
- [33] Colinge, Jean-Pierre. "Multiple-gate soi mosfets." Solid-State Electronics 48.6 (2004): 897-905.
- [34] Hisamoto, Digh, et al. "FinFET-a self-aligned double-gate MOSFET scalable to 20 nm." Electron Devices, IEEE Transactions on 47.12 (2000): 2320-2325.
- [35] Chiarella, Thomas, et al. "Benchmarking SOI and bulk FinFET alternatives for PLANAR CMOS scaling succession." Solid-State Electronics 54.9 (2010): 855-860.
- [36] Lackey, David E., et al. "Managing power and performance for System-on-Chip designs using Voltage Islands." Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on. IEEE, 2002.
- [37] Kursun, Volkan, and Eby G. Friedman. Multi-voltage CMOS circuit design. Wiley. com, 2006.
- [38] Borkar, Shekhar. "Low power design challenges for the decade (invited talk)." Proceedings of the 2001 Asia and South Pacific Design Automation Conference. ACM, 2001.
- [39] Weiser, Mark, et al. "Scheduling for reduced CPU energy." Mobile Computing. Springer US, 1996. 449-471.
- [40] Pillai, Padmanabhan, and Kang G. Shin. "Real-time dynamic voltage scaling for low-power embedded operating systems." ACM SIGOPS Operating Systems Review. Vol. 35. No. 5. ACM, 2001.

- [41] Zhai, Bo, et al. "Theoretical and practical limits of dynamic voltage scaling." Proceedings of the 41st annual Design Automation Conference. ACM, 2004.
- [42] Zhai, Bo, et al. "Analysis and mitigation of variability in subthreshold design." Proceedings of the 2005 international symposium on Low power electronics and design. ACM, 2005.
- [43] Nowka, Kevin J., et al. "A 32-bit PowerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling." Solid-State Circuits, IEEE Journal of 37.11 (2002): 1441-1447.
- [44] Nakai, Masakatsu, et al. "Dynamic voltage and frequency management for a low-power embedded microprocessor." Solid-State Circuits, IEEE Journal of 40.1 (2005): 28-35.
- [45] McGowen, Rich, et al. "Power and temperature control on a 90-nm Itanium family processor." Solid-State Circuits, IEEE Journal of 41.1 (2006): 229-237.
- [46] Le Sueur, Etienne, and Gernot Heiser. "Dynamic voltage and frequency scaling: The laws of diminishing returns." Proceedings of the 2010 international conference on Power aware computing and systems. USENIX Association, 2010.
- [47] Chowdhury, Princey, and Chaitali Chakrabarti. "Static task-scheduling algorithms for battery-powered DVS systems." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 13.2 (2005): 226-237.
- [48] Gruian, Flavius. "Hard real-time scheduling for low-energy using stochastic data and DVS processors." Proceedings of the 2001 international symposium on Low power electronics and design. ACM, 2001.
- [49] Zhuo, Jianli, and Chaitali Chakrabarti. "System-level energy-efficient dynamic task scheduling." Design Automation Conference, 2005. Proceedings. 42nd. IEEE, 2005.
- [50] Nielsen, Lars Skovby, et al. "Low-power operation using self-timed circuits and adaptive scaling of the supply voltage." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 2.4 (1994): 391-397.
- [51] Elgebaly, Mohamed, and Manoj Sachdev. "Efficient adaptive voltage scaling system through on-chip critical path emulation." Proceedings of the 2004 international symposium on Low power electronics and design. ACM, 2004.
- [52] Elgebaly, Mohamed, and Manoj Sachdev. "Variation-aware adaptive voltage scaling system." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 15.5 (2007): 560-571.
- [53] Kim, Wonyoung, et al. "System level analysis of fast, per-core DVFS using on-chip switching regulators." High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on. IEEE, 2008.
- [54] Hazucha, Peter, et al. "Area-efficient linear regulator with ultra-fast load regulation." Solid-State Circuits, IEEE Journal of 40.4 (2005): 933-940.
- [55] Leung, Ka Nang, and Philip KT Mok. "A capacitor-free CMOS low-dropout regulator with damping-factor-control frequency compensation." Solid-State Circuits, IEEE Journal of 38.10 (2003): 1691-1702.
- [56] Hoon, S. K., et al. "A low noise, high power supply rejection low dropout regulator for wireless system-on-chip applications." Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005. IEEE, 2005.
- [57] Milliken, Robert J., Jose Silva-Martínez, and Edgar Sanchez-Sinencio. "Full on-chip CMOS low-dropout voltage regulator." Circuits and Systems I: Regular Papers, IEEE Transactions on 54.9 (2007): 1879-1890.

- [58] Su, Feng, Wing-Hung Ki, and Chi-Ying Tsui. "Ultra fast fixed-frequency hysteretic buck converter with maximum charging current control and adaptive delay compensation for DVS applications." *Solid-State Circuits, IEEE Journal of* 43.4 (2008): 815-822.
- [59] Ramadass, Yogesh K., and Anantha P. Chandrakasan. "Voltage scalable switched capacitor DC-DC converter for ultra-low-power on-chip applications." *Power Electronics Specialists Conference, 2007. PESC 2007. IEEE. IEEE, 2007.*
- [60] Villar-Pique, Gerard, H. Bergveld, and Eduard Alarcon. "Survey and benchmark of Fully Integrated Switching Power Converters: Switched-Capacitor vs Inductive approach." (2013): 1-1.
- [61] Kim, Wonyoung, David Brooks, and Gu-Yeon Wei. "A fully-integrated 3-level DC-DC converter for nanosecond-scale DVFS." *Solid-State Circuits, IEEE Journal of* 47.1 (2012): 206-219.
- [62] Patounakis, George, Yee William Li, and Kenneth L. Shepard. "A fully integrated on-chip DC-DC conversion and power management system." *Solid-State Circuits, IEEE Journal of* 39.3 (2004): 443-451.
- [63] Burd, Thomas D., and Robert W. Brodersen. "Design issues for dynamic voltage scaling." *Low Power Electronics and Design, 2000. ISLPED'00. Proceedings of the 2000 International Symposium on.* IEEE, 2000.
- [64] Tran, Canh Q., Hiroshi Kawaguchi, and Takayasu Sakurai. "Low-power high-speed level shifter design for block-level dynamic voltage scaling environment." *Integrated Circuit Design and Technology, 2005. ICICDT 2005. 2005 International Conference on.* IEEE, 2005.
- [65] Garg, Rajesh, Gagandeep Mallarapu, and Sunil P. Khatri. "A single-supply true voltage level shifter." *Proceedings of the conference on Design, automation and test in Europe.* ACM, 2008.
- [66] Horiguchi, Masashi, Takeshi Sakata, and Kiyoo Itoh. "Switched-source-impedance CMOS circuit for low standby subthreshold current giga-scale LSI's." *Solid-State Circuits, IEEE Journal of* 28.11 (1993): 1131-1135.
- [67] Shin, Youngsoo, et al. "Power gating: Circuits, design methodologies, and best practice for standard-cell vlsi designs." *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 15.4 (2010): 28.
- [68] Min, Kyeong-Sik, Hiroshi Kawaguchi, and Takayasu Sakurai. "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: an alternative to clock-gating scheme in leakage dominant era." *Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International.* IEEE, 2003.
- [69] Calhoun, Benton H., Frank A. Honore, and Anantha Chandrakasan. "Design methodology for fine-grained leakage control in MTCMOS." *Proceedings of the 2003 international symposium on Low power electronics and design.* ACM, 2003.
- [70] Khandelwal, Vishal, and Ankur Srivastava. "Leakage control through fine-grained placement and sizing of sleep transistors." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 26.7 (2007): 1246-1255.
- [71] Usami, Kimiyoshi, and Naoaki Ohkubo. "A design approach for fine-grained run-time power gating using locally extracted sleep signals." *Computer Design, 2006. ICCD 2006. International Conference on.* IEEE, 2007.
- [72] Pakbaznia, Ehsan, and Massoud Pedram. "Coarse-grain MTCMOS sleep transistor sizing using delay budgeting." *Design, Automation and Test in Europe, 2008. DATE'08.* IEEE, 2008.
- [73] Keating, Michael. *Low Power Methodology Manual: For System on Chip Design.* Springer, 2007.
- [74] Shi, Kaijian, and David Howard. "Sleep transistor design and implementation-simple concepts yet challenges to be optimum." *VLSI Design, Automation and Test, 2006 International Symposium on.* IEEE, 2006.

- [75] Kao, James, Anantha Chandrakasan, and Dimitri Antoniadis. "Transistor sizing issues and tool for multi-threshold CMOS technology." Proceedings of the 34th annual Design Automation Conference. ACM, 1997.
- [76] Khandelwal, Vishal, and Ankur Srivastava. "Leakage control through fine-grained placement and sizing of sleep transistors." Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 26.7 (2007): 1246-1255.
- [77] Sani, Mehdi Hamidi, Gregory A. Uvieghara, and John DeJaco. "Regulation of crowbar current in circuits employing footswitches/headswitches." U.S. Patent Application 10/155,956.
- [78] Shigematsu, Satoshi, et al. "A 1-V high-speed MTCMOS circuit scheme for power-down application circuits." Solid-State Circuits, IEEE Journal of 32.6 (1997): 861-869.
- [79] Akamatsu, Hironori, et al. "A low power data holding circuit with an intermittent power supply scheme for sub-1V MT-CMOS LSIs." VLSI Circuits, 1996. Digest of Technical Papers., 1996 Symposium on. IEEE, 1996.
- [80] Kao, James, and Anantha Chandrakasan. "MTCMOS sequential circuits." Solid-State Circuits Conference, 2001. ESSCIRC 2001. Proceedings of the 27th European. IEEE, 2001.
- [81] Mahmoodi-Meimand, Hamid, and Kaushik Roy. "Data-retention flip-flops for power-down applications." Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on. Vol. 2. IEEE, 2004.
- [82] Zyuban, Victor, and Stephen V. Kosonocky. "Low power integrated scan-retention mechanism." Proceedings of the 2002 international symposium on Low power electronics and design. ACM, 2002.
- [83] Idgunji, Sachin. "Case study of a low power MTCMOS based ARM926 SoC: Design, analysis and test challenges." Test Conference, 2007. ITC 2007. IEEE International. IEEE, 2007.
- [84] Powell, Michael, et al. "Gated-V dd: a circuit technique to reduce leakage in deep-submicron cache memories." Proceedings of the 2000 international symposium on Low power electronics and design. ACM, 2000.
- [85] Hardee, K., et al. "A 0.6 V 205MHz 19.5 ns tRC 16Mb embedded DRAM." Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International. IEEE, 2004.
- [86] Qin, Hulfang, et al. "SRAM leakage suppression by minimizing standby supply voltage." Quality Electronic Design, 2004. Proceedings. 5th International Symposium on. IEEE, 2004.
- [87] Hua, Chung-Hsien, Tung-Shuan Cheng, and Wei Hwang. "Distributed data-retention power gating techniques for column and row co-controlled embedded SRAM." Memory Technology, Design, and Testing, 2005. MTDT 2005. 2005 IEEE International Workshop on. IEEE, 2005.
- [88] Borkar, Shekhar, et al. "Parameter variations and impact on circuits and microarchitecture." Proceedings of the 40th annual Design Automation Conference. ACM, 2003.
- [89] Tschanz, James W., et al. "System using body-biased sleep transistors to reduce leakage power while minimizing performance penalties and noise." U.S. Patent No. 6,744,301. 1 Jun. 2004.
- [90] Tschanz, James W., et al. "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage." Solid-State Circuits, IEEE Journal of 37.11 (2002): 1396-1402.
- [91] Martin, Steven M., et al. "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads." Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design. ACM, 2002.
- [92] Keshavarzi, Ali, et al. "Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS ICs." Low Power Electronics and Design, International Symposium on, 2001.. IEEE, 2001.

- [93] V. De et al., "Techniques for leakage power reduction," in Design of High-Performance Microprocessor Circuits, A. Chandrakasan, W. J. Bowhill, and F. Fox, Eds. Piscataway, NJ: IEEE Press, 2001, ch. 3, pp. 52–55.
- [94] Mukhopadhyay, Saibal, et al. "Gate leakage reduction for scaled devices using transistor stacking." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 11.4 (2003): 716-730.
- [95] Narendra, Siva, et al. "Scaling of stack effect and its application for leakage reduction." Proceedings of the 2001 international symposium on Low power electronics and design. ACM, 2001.
- [96] Piguet, C. Low-Power Electronics Design. CRC Press, 2005.
- [97] Soeleman, Hendrawan, and Kaushik Roy. "Ultra-low power digital subthreshold logic circuits." Proceedings of the 1999 international symposium on Low power electronics and design. ACM, 1999.
- [98] Soeleman, Hendrawan, and Kaushik Roy. "Digital CMOS logic operation in the sub-threshold region." Proceedings of the 10th Great Lakes symposium on VLSI. ACM, 2000.
- [99] Soeleman, Hendrawan, Kaushik Roy, and Bipul Paul. "Robust ultra-low power sub-threshold DT MOS logic." Proceedings of the 2000 international symposium on Low power electronics and design. ACM, 2000.
- [100] Wang, Alice, Benton H. Calhoun, and Anantha P. Chandrakasan. Sub-threshold design for ultra low-power systems. Springer, 2006.
- [101] Benini, Luca, Alessandro Bogliolo, and Giovanni De Micheli. "A survey of design techniques for system-level dynamic power management." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 8.3 (2000): 299-316.
- [102] Lu, Yung-Hsiang, and Giovanni De Micheli. "Comparing system level power management policies." Design & Test of Computers, IEEE 18.2 (2001): 10-19.
- [103] Brock, Bishop, and Karthick Rajamani. "Dynamic power management for embedded systems [SOC design]." SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip]. IEEE, 2003.
- [104] Arampatzis, Th, J. Lygeros, and S. Manesis. "A survey of applications of wireless sensors and wireless sensor networks." Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation. IEEE, 2005.
- [105] Hill, Jason, et al. "The platforms enabling wireless sensor networks." Communications of the ACM 47.6 (2004): 41-46.
- [106] Levis, Philip, et al. "TinyOS: An operating system for sensor networks." Ambient intelligence. Springer Berlin Heidelberg, 2005. 115-148.
- [107] Hill, Jason Lester. System architecture for wireless sensor networks. Diss. University of California, 2003.
- [108] Kahn, Joseph M., Randy H. Katz, and Kristofer SJ Pister. "Next century challenges: mobile networking for "Smart Dust"." Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking. ACM, 1999.
- [109] Hollar, Seth Edward-Austin. "Cots dust.", Master's Thesis, University of California, (2000).
- [110] McLurkin, James D. Algorithms for distributed sensor networks. Diss. Department of Electrical Engineering and Computer Sciences, University of California, 1999.
- [111] Hill, Jason L., and David E. Culler. "Mica: A wireless platform for deeply embedded networks." Micro, IEEE 22.6 (2002): 12-24.

- [112] Polastre, Joseph, Robert Szewczyk, and David Culler. "Telos: enabling ultra-low power wireless research." *Information Processing in Sensor Networks*, 2005. IPSN 2005. Fourth International Symposium on. IEEE, 2005.
- [113] Werner-Allen, Geoffrey, et al. "Deploying a wireless sensor network on an active volcano." *Internet Computing*, IEEE 10.2 (2006): 18-25.
- [114] Choudhury, Tanzeem, et al. "The mobile sensing platform: An embedded activity recognition system." *Pervasive Computing*, IEEE 7.2 (2008): 32-41.
- [115] Beute, J. "Fast-prototyping using the BNode platform." *Design, Automation and Test in Europe*, 2006. DATE'06. Proceedings. Vol. 1. IEEE, 2006.
- [116] PC/104 Embedded Consortium. "PC/104 Specification." Online:[[http://www. pc104.org/pc104\\_specs. php](http://www.pc104.org/pc104_specs.php)] (2003).
- [117] Warneke, Brett A., and Kristofer SJ Pister. "An ultra-low energy microcontroller for smart dust wireless sensor networks." *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*. IEEE, 2004.
- [118] Ekanayake, Virantha, Clinton Kelly IV, and Rajit Manohar. "An ultra low-power processor for sensor networks." *ACM SIGOPS Operating Systems Review* 38.5 (2004): 27-36.
- [119] Ekanayake, Virantha N., Clinton Kelly IV, and Rajit Manohar. "BitSNAP: Dynamic significance compression for a low-energy sensor network asynchronous processor." *Asynchronous Circuits and Systems*, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on. IEEE, 2005.
- [120] Hempstead, Mark, et al. "An ultra low power system architecture for sensor network applications." *ACM SIGARCH Computer Architecture News*. Vol. 33. No. 2. IEEE Computer Society, 2005.
- [121] Hempstead, Mark, David Brooks, and G. Wei. "An accelerator-based wireless sensor network processor in 130 nm CMOS." *Emerging and Selected Topics in Circuits and Systems*, *IEEE Journal on* 1.2 (2011): 193-202.
- [122] Sheets, M., et al. "A power-managed protocol processor for wireless sensor networks." *VLSI circuits*, 2006. Digest of technical papers. 2006 symposium on. IEEE, 2006.
- [123] Zhai, Bo, et al. "A 2.60 pJ/Inst subthreshold sensor processor for optimal energy efficiency." *VLSI Circuits*, 2006. Digest of Technical Papers. 2006 Symposium on. IEEE, 2006.
- [124] Zhai, Bo, et al. "Energy-efficient subthreshold processor design." *Very Large Scale Integration (VLSI) Systems*, *IEEE Transactions on* 17.8 (2009): 1127-1137.
- [125] Hanson, Scott, et al. "A low-voltage processor for sensing applications with picowatt standby mode." *Solid-State Circuits*, *IEEE Journal of* 44.4 (2009): 1145-1155.
- [126] Panic, G., Z. Stamenkovic, and R. Kraemer. "Power gating in wireless sensor networks." *Wireless Pervasive Computing*, 2008. ISWPC 2008. 3rd International Symposium on. IEEE, 2008.
- [127] Panic, Goran, Daniel Dietterle, and Zoran Stamenkovic. "Architecture of a power-gated wireless sensor node." *Digital System Design Architectures, Methods and Tools*, 2008. DSD'08. 11th EUROMICRO Conference on. IEEE, 2008.
- [128] Dargie, Walteneagus. "Dynamic power management in wireless sensor networks: State-of-the-art." *Sensors Journal*, IEEE 12.5 (2012): 1518-1528.
- [129] Sinha, Amit, and Anantha Chandrakasan. "Dynamic power management in wireless sensor networks." *Design & Test of Computers*, IEEE 18.2 (2001): 62-74.
- [130] Panic, Goran, et al. "Low power sensor node processor architecture." *Electronics, Circuits, and Systems (ICECS)*, 2010 17th IEEE International Conference on. IEEE, 2010.
- [131] Panic, Goran, et al. "Sensor node processor for security applications." *Electronics, Circuits and Systems (ICECS)*, 2011 18th IEEE International Conference on. IEEE, 2011.

- [132] Panic, Goran, et al. "Design of a sensor node crypto processor for IEEE 802.15. 4 applications." SOC Conference (SOCC), 2012 IEEE International. IEEE, 2012.
- [133] Panic, Goran, et al. "TNODE: A low power sensor node processor for secure wireless networks." System on Chip (SoC), 2013 International Symposium on. IEEE, 2013.
- [134] Stecklina, Oliver, Dieter Genschow, and Christian Goltz. "TandemStack-A Flexible and Customizable Sensor Node Platform for Low Power Applications." SENSORNETS. 2012.
- [135] Chandrakasan, Anantha P., William J. Bowhill, and Frank Fox. Design of high-performance microprocessor circuits. Wiley-IEEE press, 2000.
- [136] IMPS430X, Fraunhofer IPMS, <http://www.ipms.fraunhofer.de/>
- [137] López, Julio, and Ricardo Dahab. "Fast multiplication on elliptic curves over GF (2m) without precomputation." Cryptographic Hardware and Embedded Systems, Springer Berlin Heidelberg, 1999.
- [138] Dyka Zoya and Peter Langendoerfer. "Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsuba's method." Design, Automation and Test in Europe, 2005. Proceedings. IEEE, 2005.
- [139] Schrape, Oliver, and Frank Vater. "Embedded low power clock generator for sensor nodes." NORCHIP, 2012. IEEE, 2012.
- [140] Chen, Chi-Yuan, and Han-Chieh Chao. "A survey of key distribution in wireless sensor networks." Security and communication networks (2011).
- [141] Daemen, Joan, and Vincent Rijmen. "AES proposal: Rijndael." First Advanced Encryption Standard (AES) Conference. 1998.
- [142] Uli Kretzschmar, "AES128 – A C Implementation for Encryption and Decryption," TI - White Paper, 2009
- [143] F. Vater and P. Langendörfer, "An Area Efficient Realisation of AES for Wireless Devices," *Information Technology* 2007, 188-193
- [144] Sorin Dobre, Pete Hardee, Colin Holehouse, Minh Chau and Rolf Lagerquist, "Power Intent Formats: Light at the End of the Tunnel?", *EETimes*, 23/04/2012
- [145] Si2 CPF specification version 2.0,  
[http://www.si2.org/cgi-bin/openeda.si2.org/download?group\\_id=51&file\\_id=1626&filename=si2\\_cpf\\_v2.0\\_14-feb-2011\\_with\\_Errata\\_14-mar-2011.pdf](http://www.si2.org/cgi-bin/openeda.si2.org/download?group_id=51&file_id=1626&filename=si2_cpf_v2.0_14-feb-2011_with_Errata_14-mar-2011.pdf)
- [146] IEEE 1801-2009 specification,  
[http://www.techstreet.com/standards/ieee/1801\\_2009?product\\_id=1744966](http://www.techstreet.com/standards/ieee/1801_2009?product_id=1744966)
- [147] Qi Wang, The Evolution of Power Format Standards: A Cadence Viewpoint,  
[http://www.cadence.com/rl/Resources/white\\_papers/power\\_format\\_wp.pdf](http://www.cadence.com/rl/Resources/white_papers/power_format_wp.pdf)
- [148] Anis Jarrar and John Dalbey, Low-Power Implementation on Freescale Kinetis Family, Technical Paper, [http://www.cadence.com/rl/Resources/technical\\_papers/Freescale\\_tp.pdf](http://www.cadence.com/rl/Resources/technical_papers/Freescale_tp.pdf)
- [149] Synopsys®, "Synopsys® Low-Power Flow User Guide. Version B-2008.09.", September 2008
- [150] Landman, Paul. "High-level power estimation." Low Power Electronics and Design, 1996., International Symposium on. IEEE, 1996.
- [151] Cadence, InCyte Chip Estimate, <http://www.chipestimate.com/>
- [152] Ansys/Apache, Power Artist, <https://www.apache-da.com/products/powerartist>
- [153] Preeti Gupta. Be Early with Power. Chip Design Magazine.  
<http://www.chipdesignmag.com/display.php?articleId=613>

- [154] Burd, Thomas D., and Robert W. Brodersen. "Design issues for dynamic voltage scaling." *Low Power Electronics and Design*, 2000. ISLPED'00. Proceedings of the 2000 International Symposium on. IEEE, 2000.
- [155] Park, Jaehyun, et al. "Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors." *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*. ACM, 2010.
- [156] Bambagini, Mario, et al. "On the Impact of Runtime Overhead on Energy-Aware Scheduling." *Workshop on Power, Energy, and Temperature Aware Realtime Systems (PETARS 2013)*, Philadelphia, USA. 2013.
- [157] Duarte, David, et al. "Evaluating run-time techniques for leakage power reduction." *Proceedings of the 2002 Asia and South Pacific Design Automation Conference*. IEEE Computer Society, 2002.
- [158] Hu, Zhigang, et al. "Microarchitectural techniques for power gating of execution units." *Proceedings of the 2004 international symposium on Low power electronics and design*. ACM, 2004.
- [159] Rosinger, Sven. *RT-Level power-gating models optimizing dynamic leakage-management*. Diss. Universität Oldenburg, 2012.
- [160] Tandem Project, <http://www.tandem-projekt.de>
- [161] Smart Project, <http://www.artemis-smart.eu>
- [162] Matrix Project, <http://www.ihp-microelectronics.com/de/forschung/drahtlose-systeme-und-anwendungen/projects/matrix.html>
- [163] Telediagnostik Project, <http://www.theradiagnostik.de/>
- [164] Strokeback Project, <http://www.strokeback.eu>
- [165] Lukasz Lopacinski, "Implementation and verification of an ultra low power baseband processor", Master Thesis, West Pomeranian University of Technology, Szczecin, 2009
- [166] J. Portilla, J. Andrés Otero, E. de la Torre, T. Riesgo, O. Stecklina, S. Peter, P. Langendörfer, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors," *International Journal of Distributed Sensor Networks*, Vol. 2010
- [167] Scott, Michael. "Mira—multiprecision integer and rational arithmetic c/c++ library." Shamus Software Ltd, Dublin, Ireland (2003).
- [168] Liu, An, and Peng Ning. "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks." *Information Processing in Sensor Networks*, 2008. IPSN'08. International Conference on. IEEE, 2008.
- [169] Weimerskirch, André, Christof Paar, and Sheueling Chang Shantz. "Elliptic curve cryptography on a Palm OS device." *Information Security and Privacy*. Springer Berlin Heidelberg, 2001.
- [170] Riedel, Ingo, and Ing Christof Paar. "Security in ad-hoc networks: Protocols and elliptic curve cryptography on an embedded platform." Master's thesis, Ruhr-Universitaet Bochum (2003).
- [171] S. Peter, "Evaluation of Design Alternatives for Flexible Elliptic Curve Hardware Accelerators," Diploma Thesis, BTU Cottbus, 2006.
- [172] *International Technology Roadmap for Semiconductors*, 2011 Edition, [www.itrs.net](http://www.itrs.net)
- [173] Moore G.E., "Cramming more Components onto Integrated Circuits", *Electronics*, 38 (8) (April 19, 1965); reproduced in *Proc. IEEE*, 86, 82 (1998).
- [174] Wolfgang Arden, Michel Brillouët, Patrick Coge, Mart Graef, Bert Huizing, Reinhard Mahnkopf, „More-than-Moore“, White Paper, ITRS 2010.
- [175] *International Technology Roadmap for Semiconductors*, 2009 Edition, [www.itrs.net](http://www.itrs.net)



## Appendix

### CPF Power Intent Description of Sensor Node Microcontroller Design

```
set_cpf_version 1.1
```

```
#####
```

```
# Technology Part
```

```
#####
```

```
define_library_set -name setl_wc -libraries
```

```
{/ihpsoft/designkits/sgb25tm1tm2/encounter/TLF/SESAME-IO-IHP_0.25um_SS_3.0_125.lib  
/ihpsoft/designkits/sgb25tm1tm2/encounter/TLF/SESAME-LP2_IHP_0.25um_SS_2.25_125.lib  
/home/panic/sgb25v/SpRAM_Generator/SpRAM_8192x8/SpRAM_8192x8.wc.lib  
/home/panic/design/Flash/16K44CE/Flash_16k44ce.lib  
/home/panic/design/IHP_TNODE_v6/lib/ADC_BMS/ADC_BMS_ss.lib  
/home/panic/design/PwrSw/CG_PSwitch_250u.lib}
```

```
define_library_set -name setl_bc -libraries
```

```
{/ihpsoft/designkits/sgb25tm1tm2/encounter/TLF/SESAME-IO-IHP_0.25um_FF_3.6_-40.lib  
/ihpsoft/designkits/sgb25tm1tm2/encounter/TLF/SESAME-LP2_IHP_0.25um_FF_2.75_-40.lib  
/home/panic/sgb25v/SpRAM_Generator/SpRAM_8192x8/SpRAM_8192x8.bc.lib  
/home/panic/design/Flash/16K44CE/Flash_16k44ce.lib  
/home/panic/design/IHP_TNODE_v6/lib/ADC_BMS/ADC_BMS_ff.lib  
/home/panic/design/PwrSw/CG_PSwitch_250u.lib}
```

```
define_power_switch_cell -cells {hswitch_300u_50m} -stage_1_enable {!SLEEP} -stage_1_output  
{Y} -type header -power vdd! -power_switchable vddg
```

```
#####
```

```
# Design Part
```

```
#####
```

```
set_design ihp_tnode_v6_io
```

```
create_power_domain -name PD1 -default
```

```
create_power_domain -name PD2 -instances {i_ihp_tnode_v6.i_bb868_wrap.i_bb868}
```

```
-shutoff_condition {!i_ihp_tnode_v6.i_pmu.shut_down[0]} -base_domains PD1
```

```
create_power_domain -name PD3 -instances {i_ihp_tnode_v6.i_uart_wrapper.i_uart1}
```

```
-shutoff_condition {!i_ihp_tnode_v6.i_pmu.shut_down[1]} -base_domains PD1
```

```
create_power_domain -name PD4 -instances {i_ihp_tnode_v6.i_uart_wrapper.i_uart2}
```

```
-shutoff_condition {!i_ihp_tnode_v6.i_pmu.shut_down[2]} -base_domains PD1
```

```
create_power_domain -name PD5 -instances {i_ihp_tnode_v6.i_crypto_wrapper.i_aes_sha1}
```

```
-shutoff_condition {!i_ihp_tnode_v6.i_pmu.shut_down[3]} -base_domains PD1
```

```
create_power_domain -name PD6 -instances {i_ihp_tnode_v6.i_crypto_wrapper.i_ecc}
```

```

-shutoff_condition {!i_ihp_tnode_v6.i_pmu.shut_down[4]} -base_domains PD1

create_nominal_condition -name off -voltage 0
create_nominal_condition -name on -voltage 2.5

create_power_mode -name PM1 -domain_conditions {PD1@on PD2@on PD3@on PD4@on
PD5@on PD6@on} -default
create_power_mode -name PM2 -domain_conditions {PD1@on PD2@on PD3@off PD4@off
PD5@off PD6@off}
create_power_mode -name PM3 -domain_conditions {PD1@on PD2@off PD3@on PD4@off
PD5@off PD6@off}
create_power_mode -name PM4 -domain_conditions {PD1@on PD2@off PD3@off PD4@on
PD5@off PD6@off}
create_power_mode -name PM5 -domain_conditions {PD1@on PD2@off PD3@off PD4@off
PD5@on PD6@off}
create_power_mode -name PM6 -domain_conditions {PD1@on PD2@off PD3@off PD4@off
PD5@off PD6@on}
create_power_mode -name PM7 -domain_conditions {PD1@on PD2@off PD3@off PD4@off
PD5@off PD6@off}

update_nominal_condition -name on -library_set setl_wc

#####
# Synthesis Related
#####

update_power_mode -name PM1 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}
update_power_mode -name PM2 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}
update_power_mode -name PM3 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}
update_power_mode -name PM4 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}
update_power_mode -name PM5 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}
update_power_mode -name PM6 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}
update_power_mode -name PM7 -sdc_files
{/home/panic/design/LayoutFE/IHP_TNODE_v6/input/ihp_tnode_v6_io.sdc}

#####
# Layout Related
#####

create_power_nets -nets vdd! -voltage 2.5

```

```
create_power_nets -nets {VDD_BB VDD_U1 VDD_U2 VDD_AES VDD_ECC} -internal
create_ground_nets -nets gnd! -voltage 0
```

```
create_global_connection -domain PD2 -net VDD_BB -pins vddg
create_global_connection -domain PD3 -net VDD_U1 -pins vddg
create_global_connection -domain PD4 -net VDD_U2 -pins vddg
create_global_connection -domain PD5 -net VDD_AES -pins vddg
create_global_connection -domain PD6 -net VDD_AES -pins vddg
```

```
create_power_switch_rule -name SW_BB -domain PD2 -external_power_net vdd!
create_power_switch_rule -name SW_U1 -domain PD3 -external_power_net vdd!
create_power_switch_rule -name SW_U2 -domain PD4 -external_power_net vdd!
create_power_switch_rule -name SW_AES -domain PD5 -external_power_net vdd!
create_power_switch_rule -name SW_ECC -domain PD6 -external_power_net vdd!
```

```
update_power_switch_rule -name SW_BB -cells hswitch_300u_50m -prefix CDN_
update_power_switch_rule -name SW_U1 -cells hswitch_300u_50m -prefix CDN_
update_power_switch_rule -name SW_U2 -cells hswitch_300u_50m -prefix CDN_
update_power_switch_rule -name SW_AES -cells hswitch_300u_50m -prefix CDN_
update_power_switch_rule -name SW_ECC -cells hswitch_300u_50m -prefix CDN_
```

```
update_power_domain -name PD1 -primary_power_net vdd! -primary_ground_net gnd!
update_power_domain -name PD2 -primary_power_net VDD_BB -primary_ground_net gnd!
update_power_domain -name PD3 -primary_power_net VDD_U1 -primary_ground_net gnd!
update_power_domain -name PD4 -primary_power_net VDD_U2 -primary_ground_net gnd!
update_power_domain -name PD5 -primary_power_net VDD_AES -primary_ground_net gnd!
update_power_domain -name PD6 -primary_power_net VDD_ECC -primary_ground_net gnd!
```

```
create_operating_corner -name BC -process 1 -temperature -40 -voltage 2.75 -library_set setl_bc
create_operating_corner -name WC -process 1 -temperature 125 -voltage 2.25 -library_set
setl_wc
```

```
create_analysis_view -name AV_PM1_bc -mode PM1 -domain_corners {PD1@BC PD2@BC
PD3@BC PD4@BC PD5@BC PD6@BC}
create_analysis_view -name AV_PM1_wc -mode PM1 -domain_corners {PD1@WC PD2@WC
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
create_analysis_view -name AV_PM2_bc -mode PM2 -domain_corners {PD1@BC PD2@BC
PD3@BC PD4@BC PD5@BC PD6@BC}
create_analysis_view -name AV_PM2_wc -mode PM2 -domain_corners {PD1@WC PD2@WC
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
create_analysis_view -name AV_PM3_bc -mode PM3 -domain_corners {PD1@BC PD2@BC
PD3@BC PD4@BC PD5@BC PD6@BC}
create_analysis_view -name AV_PM3_wc -mode PM3 -domain_corners {PD1@WC PD2@WC
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
create_analysis_view -name AV_PM4_bc -mode PM4 -domain_corners {PD1@BC PD2@BC  
PD3@BC PD4@BC PD5@BC PD6@BC}
```

```
create_analysis_view -name AV_PM4_wc -mode PM4 -domain_corners {PD1@WC PD2@WC  
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
create_analysis_view -name AV_PM5_bc -mode PM5 -domain_corners {PD1@BC PD2@BC  
PD3@BC PD4@BC PD5@BC PD6@BC}
```

```
create_analysis_view -name AV_PM5_wc -mode PM5 -domain_corners {PD1@WC PD2@WC  
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
create_analysis_view -name AV_PM6_bc -mode PM6 -domain_corners {PD1@BC PD2@BC  
PD3@BC PD4@BC PD5@BC PD6@BC}
```

```
create_analysis_view -name AV_PM6_wc -mode PM6 -domain_corners {PD1@WC PD2@WC  
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
create_analysis_view -name AV_PM7_bc -mode PM7 -domain_corners {PD1@BC PD2@BC  
PD3@BC PD4@BC PD5@BC PD6@BC}
```

```
create_analysis_view -name AV_PM7_wc -mode PM7 -domain_corners {PD1@WC PD2@WC  
PD3@WC PD4@WC PD5@WC PD6@WC}
```

```
end_design
```

## List of Figures

Figure 1. Projected power trends vs power requirements in planar bulk-CMOS [2].	5
Figure 2. Projected power trends in future CMOS. The projection accounts for the impact of future technologies such as SOI and MG-CMOS [172].	6
Figure 3. Wireless sensor network.	7
Figure 4. Power consumption trends for different technology nodes [3].	11
Figure 5. Power vs. Energy. Two approaches have different power but same energy.	12
Figure 6. Switching power in CMOS: a) 0-to-1 transition, b) 1-to-0 transition.	13
Figure 7. Internal power.	14
Figure 8. Static leakage currents: a) gate at logic level 0, b) gate at logic level 1.	15
Figure 9. Multi-voltage architecture.	17
Figure 10. $V_{th}$ -dependency of transistor leakage current and transistor delay.	18
Figure 11. Clock gating: a) no clock gating implemented, b) clock gating implemented.	19
Figure 12. Example of operand isolation applied to simple ALU block: a) no operand isolation applied, b) operand isolation applied.	20
Figure 13. Different aspects of well engineering.	22
Figure 14. Bulk-CMOS (a) vs SOI (b) technology.	22
Figure 15. Bulk-CMOS vs PDSOI vs FDSOI.	23
Figure 16. Tri-gate FinFET.	24
Figure 17. Multi-voltage design techniques: a) static voltage scaling, b) multi voltage scaling, c) dynamic frequency and voltage scaling, and d) adaptive frequency and voltage scaling.	25
Figure 18. Example of frequency and voltage scaling. The frequency scaling with fixed supply voltage does not save the energy (blue). In the case of variable supply voltage, the energy consumption is reduced (dashed).	26
Figure 19. Basic concept of power gating. PMOS (header) and NMOS (footer) power switches disconnect a block from the global power supply network. The block outputs to always-on logic are electrically isolated to prevent crowbar currents. The power sequencing is controlled by power gating controller.	29
Figure 20. Activity profiles: clock gating vs power gating.	30
Figure 21. Isolation cell.	31
Figure 22. Basic architecture of retention flip-flop.	31
Figure 23. Example of power switching sequencing in a power-gated design: a) basic steps of power-on and power-off procedures, b) control signals from power gating controller.	32
Figure 24. Architecture of a complex power gating controller.	33
Figure 25. Examples of coarse-grain power switches.	35
Figure 26. Examples of power switching networks.	35
Figure 27. Body Biasing.	37
Figure 28. Stacked 3-input NAND Gate.	38
Figure 29. Overview of Cadence technologies enabled by CPF [147].	41
Figure 30. SoC low power flow with Cadence tools [148].	42
Figure 31. Synopsys UPF-enabled low power flow [149].	43
Figure 32. Comparison of different power saving techniques with Power Theater [153].	45
Figure 33. Standard architecture of sensor node.	48
Figure 34. The combined need for digital and non-digital functionalities in an integrated system is translated as a dual trend in the International Technology Roadmap for Semiconductors:	

miniaturization of the digital functions (“More Moore”) and functional diversification (“More-than-Moore”) [174].	55
Figure 35. Evolving role of design phases in overall system power minimization [175].	56
Figure 36. Methodology flow for power strategy analysis.	59
Figure 37. Simplified workload in typical SoC.	61
Figure 38. Static power contribution as a function of duty cycle.	63
Figure 39. Prototyping of low power candidates.	64
Figure 40. Power-gating mode transitions.	67
Figure 41. Simplified mapping of block activity to specific system operation modes.	72
Figure 42. Example of sensor node activity mapping in WSN application relying on schedule-based communication protocols.	73
Figure 43. Example of sensor node activity mapping in WSN application relying on wake-up control.	74
Figure 44. Example of a scheduled-based WSN application.	75
Figure 45. Example of WSN application powered by wake-up radio control.	77
Figure 46. Single-hop cluster topology of WSN.	79
Figure 47. Tree topology of WSN.	80
Figure 48. Mesh topology of WSN.	80
Figure 49. Basic steps in power-aware design flow.	83
Figure 50. Power-aware system level design.	85
Figure 51. Power-Aware RTL design.	86
Figure 52. Power-Aware Implementation.	87
Figure 53. Functional, timing, power and physical verification.	88
Figure 54. Design toolkit for the presented power-driven design flow.	90
Figure 55. Development of a sensor node microcontroller. Six releases of the TNODE.	92
Figure 56. Architecture of the power-gated TNODE6.	94
Figure 57. Interface between asynchronous processor and peripherals: (a) interface logic, (b) handshake flow.	95
Figure 58. Clock distribution in sensor node microcontroller chip.	96
Figure 59. Test chip for custom power switch design.	102
Figure 60. Schematic of designed power switching cell.	103
Figure 61. Power switch cell placed in a row.	104
Figure 62. Power gating controller.	105
Figure 63. Finite state machine diagram of the control engine.	106
Figure 64. AES test chip.	107
Figure 65. Analysis of rush current.	108
Figure 66. Prototyping of power gating block candidates.	112
Figure 67. Application scenario showing a typical behaviour of the transmitting (Tx) and receiving node (Rx).	114
Figure 68. Network topology defined by the application.	119
Figure 69. Comparison of the task initialisation energy for different power saving approaches: Approach 1 - raw system (Raw), Approach 2 - clock-gated system (CG), Approach 3 -power gating implementation with six power domains (PG-6) and Approach 4 - power gating implementation with five power domains (PG-5).	128

Figure 70. Comparison of dynamic energy contribution of different power saving approaches. The clock gating reduces dynamic energy of the raw system. Power gating makes no visible impact on dynamic energy. .... 129

Figure 71. Comparison of leakage energy contribution of different power saving approaches. Power gating reduces leakage energy for a factor of two..... 129

Figure 72. Comparison of the total energy of different power saving approaches. A combination of clock and power gating gives the best results for the selected system implementation. The reduction in number of power domains reduces the system complexity and design effort..... 130

Figure 73. TNODE chip layout..... 132

Figure 74. TNODE testbench: (1) tests executed from Flash; (2) tests executed via I2C debug port. 134

Figure 75. TNODE evaluation platform. .... 138



## List of Tables

Table 2-1. Power increase projection for different technology process nodes [73].	12
Table 2-2. Overview of standard low power techniques.	21
Table 2-3. Summary of advanced low power techniques.	40
Table 2-4. Overview of general-purpose sensor node platforms.	50
Table 2-5. Overview of embedded sensor node processors.	52
Table 2-6. Low-power design technology improvements and impact on dynamic and static power [172].	56
Table 4-1. Features of the six TNODE releases.	93
Table 5-1. Electrical parameters of a power switching cell.	108
Table 5-2. Characterization of dynamic, static and internal (clock-network) power of sensor node components.	110
Table 5-3. Characterization of dynamic and static power of power gating components.	111
Table 5-4. Optimal number of power switches and turn-on time of components.	113
Table 5-5. Energy contributions of rush current and break-even points of the selected system components.	113
Table 5-6. Duration time and block activity of system tasks.	118
Table 5-7. Power of hard macro blocks of the TNODE chip.	119
Table 5-8. Estimated energy contribution of system tasks for the time $t_{data}$ . The results indicate a high portion of leakage energy in the total system energy (the clock of all inactive blocks is gated).	122
Table 5-9. Estimated energy contribution of system tasks for the raw system with no power saving techniques implemented. The standby power of Flash and ADC+BMS is assumed to be 2.5 $\mu$ W each.	123
Table 5-10. Estimated energy contribution of system tasks for the system with global clock gating.	124
Table 5-11. Estimated energy contribution of system tasks for the system with six power-gated blocks. The total system energy is reduced more than twice compared to the energy of raw system.	126
Table 5-12. Estimated energy contribution of system tasks for the system with reduced number of power domains. The total system energy is slightly reduced due to reduction of the total leakage energy of the isolation logic.	127
Table 6-1. Cell area of sensor node components.	132
Table 6-2. Measured turn-on time and peak rush current of power-gated blocks in the TNODE chip.	134
Table 6-3. Measured energy contribution of the rush currents and break-even points of the power-gated blocks in the TNODE chip.	134
Table 6-4. Post-layout power estimation results.	136
Table 6-5. Results of power measurements on the fabricated chips.	136
Table 6-6. Comparison of power consumption of several ECC implementations including TNODE...	137



## List of Abbreviations

ABB – Adaptive Body Biasing  
ADC – Analogue-to-Digital Converter  
AES – Advanced Encryption Standard  
ALU – Arithmetic and Logic Unit  
AMBA – Advanced Microcontroller Bus Architecture  
APB – Advanced Peripheral Bus  
ASIC – Application Specific Integrated Circuit  
AVFS – Adaptive Voltage and Frequency Scaling  
BB – Baseband  
BiCMOS – Bipolar Complementary Metal-Oxide-Semiconductor  
BMS – Bio-Mechanical Sensor  
CMOS – Complementary Metal-Oxide-Semiconductor  
COTS – Commodity-Of-The-Shelf  
CPF – Common Power Format  
CPU – Central Processing Unit  
CSMA-CA – Carrier Sense Multiple Access-Collision Avoidance  
DC – Direct Current  
DCO – Digitally-Controlled Oscillator  
DRAM – Dynamic Random Access Memory  
DSSS – Direct Spread Spectrum Sequence  
DUT – Design Under Test  
DVFS – Dynamic Voltage and Frequency Scaling  
DVS – Dynamic Voltage Scaling  
ECC – Elliptic Curve Cryptography  
ECDSA – Elliptic Curve Digital Signature Algorithm  
ECPM – Elliptic Curve Point Multiplication  
EDA – Electronic Design Automation  
EVCD – Extended Value Change Dump  
FBB – Forward Body Biasing  
FDSOI – Fully Depleted Silicon-On-Insulator  
FIFO – First In First Out  
FinFET – Fin Field-Effect-Transistor  
FSM – Finite State Machine  
GIDL – Gate Induced Drain Leakage  
HDL – Hardware Description Language  
IEEE – Institute of Electrical and Electronics Engineers  
I2C – Inter-Integrated Circuit  
I/O – Input/Output  
IP – Intellectual Property  
ITRS – International Technology Roadmap for Semiconductors  
LDO – Low-Dropout Regulator  
LEF – Library Exchange Format  
MEMS – Micro Electro-Mechanical System  
MG-CMOS – Multi-Gate Complementary Metal-Oxide-Semiconductor  
MMMC – Multi Mode Multi Corner  
MOSFET – Metal-Oxide-Semiconductor Field-Effect-Transistor  
MTCMOS – Multi-Threshold Complementary Metal-Oxide-Semiconductor  
MVFS – Multi Voltage and Frequency Scaling  
MVS – Multi Voltage Scaling  
NMOS – N-type Metal-Oxide-Semiconductor

OOK – On-Off Keyed  
PC – Personal Computer  
PCB – Printed Circuit Board  
PDA – Personal Digital Assistant  
PDSOI – Partially Depleted Silicon-On-Insulator  
PER – Packet Error Rate  
PGC – Power Gating Controller  
PMOS - P-type Metal-Oxide-Semiconductor  
PSO – Power Shut-Off  
PVT – Process, Voltage and Temperature  
QPSK – Quadrature Phase-Shift Keying  
RAM – Random Access Memory  
RBB – Reverse Body Biasing  
RF – Radio Frequency  
SAR – Successive Approximation Register  
RISC – Reduced Instruction Set Computer  
RTL – Register Transfer Level  
SC – Switched Capacitance  
SHA – Secure Hash Algorithm  
SiGe – Silicon Germanium  
SoC – System on Chip  
SOI - Silicon-On-Insulator  
SDRAM - Synchronous Dynamic Random Access Memory  
SPI – Serial Peripheral Interface  
SRAM – Static Random Access Memory  
SSI – Switched-Source Impedance  
SVFS – Static Voltage and Frequency Scaling  
SVS – Static Voltage Scaling  
SVFS – Static Voltage and Frequency Scaling  
TDMA – Time Division Multiple Access  
UART – Universal Asynchronous Receiver-Transmitter  
UPF – Unified Power Format  
USB – Universal Serial Bus  
WLAN – Wireless Local Area Network  
WSN – Wireless Sensor Network