PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY– M'SILA

FACULTY: TECHNOLOGY
DEPARTMENT: ELECTRONIC
No:

DOMAIN : SCIENCES AND TECHNOLOGY
SECTOR: ELECTRONIC
OPTION: ELECTRONIC OF EMBEDDED SYSTEMS

Thesis submitted for obtaining
Academic Master's degree
By:

BENABDELKRIM Nora

BENOUMHANI Nourelhoda

Entitled

Optimization And Energy Management In Embedded Systems

Defended before the jury composed of:

A. Bouchlagheme	University of M'Sila	Jury president
A. Benhamadouche	Univeristy of M'Sila	Supervisor
A. Ballouti	Univeristy of M'Sila	Examiner

Academic year: 2020/2021

Chapter 1: Embedded system generalities

Chapter 2: Power analysis and estimation

Chapter 3: Techniques for the management and optimization of energy consumption

General introduction

An embedded system is a processor-based system of hardware and software part, designed to perform dedicated functions. An embedded system interacts with mechanical, electrical, and chemical components along with a computer, hidden inside, to perform a dedicated purpose. There are more computers on this planet than there are people, and most of these computers are single-chip microcontrollers that are the brains of an embedded system. Embedded systems are a ubiquitous component of our everyday lives. We interact with hundreds of tiny computers every day that are embedded into our houses, our cars, our toys, and our work.

Specialized companies are continuing to push the boundaries on new features and functionality, all packed into portable, handheld, and battery powered devices. For such products, improving the battery life by minimizing power consumption is a huge differentiator and extremely important to their end users' applications. Improving the time it takes for a device to go from OFF/SLEEP state to ON/ACTIVE state is just as important, as the end user wants to have a seamless experience along with longer battery life.

Low power design is a collection of techniques and methodologies aimed at reducing the overall dynamic and static power consumption of an integrated circuit (IC). As companies, started packing more and more features and applications on the battery-operated devices (mobile/ handheld/ laptops), battery backup time became very important. Power consumption slowly became an increasingly important criterion from the customers. Due to this, all chip companies (having products for battery-operated devices) are focusing on lower power consumption. There are efforts to reduce dynamic as well as static power consumption. Companies started to reduce the nominal voltages inside the chip, however, this was also limited along with the technology. So a lot of low power design techniques started to get employed during the Chip Design process to reduce both static and dynamic power consumption.

In this work, we will study the different aspect of power consumption in embedded system, it involves power estimation technique, energy most consuming element in embedded system, and power management techniques dedicated for embedded system.

This document is organized as follow:

The first chapter is a general overview of embedded system and energy consumption. The second chapter presents techniques for analyzing energy consumption. The third chapter presents techniques for management and optimization of the energy consumption. Finally, a conclusion summarizes the results of this project.

1. Introduction

This chapter covers the background definition and technical information required in the succeeding chapters. The main items that we should discuss are power consumption in embedded systems, energy sources for autonomous systems and techniques for low power design. The aim of this chapter is to explain the different terminologies we use in this work.

2. What is an Embedded System?

An embedded system is a microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations.

Complexities range from a single microcontroller to a suite of processors with connected peripherals and networks; from no user interface to complex graphical user interfaces. The complexity of an embedded system varies significantly depending on the task for which it is designed. Embedded system applications range from digital watches and microwaves to hybrid vehicles and avionics. As much as 98 percent of all microprocessors manufactured are used in embedded systems.

2.1. How an Embedded System Works?

Embedded systems are managed by microcontrollers or digital signal processors (DSP), application-specific integrated circuits (ASIC), field-programmable gate arrays (FPGA), GPU technology, and gate arrays. These processing systems are integrated with components dedicated to handling electric and/or mechanical interfacing.

2.2. Basic Structure of an Embedded System

The basic structure of an embedded system includes the following components:

- Sensor: The sensor measures and converts the physical quantity to an electrical signal, which can then be read by an embedded systems engineer or any electronic instrument. A sensor stores the measured quantity to the memory,
- A-D Converter: An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal,

- Processor & ASICs: Processors assess the data to measure the output and store it to the memory,
- D-A Converter: A digital-to-analog converter changes the digital data fed by the processor to analog data,
- Actuator: An actuator compares the output given by the D-A Converter to the actual output stored and stores the approved output.

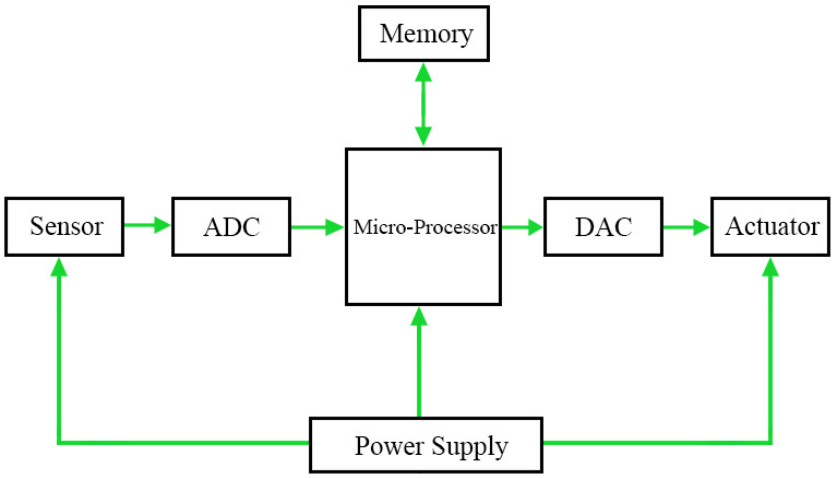


Figure 1 : basic Structure of an Embedded System

2.3. Future Trends in Embedded Systems

The industry for embedded systems is expected to continue growing rapidly, driven by the continued development of Artificial Intelligence (AI), Virtual Reality (VR) and Augmented Reality (AR), machine learning, deep learning, and the Internet of Things (IoT). The cognitive embedded system will be at the heart of such trends as: reduced energy consumption, improved security for embedded devices, cloud connectivity and mesh networking, deep learning applications, and visualization tools with real time data.

3. Design of embedded systems

To cope with the rapidly growing complexity of embedded systems, designers must work at higher levels of abstraction. Depending on the abstraction layer, the level of detail used to describe the system, designers can address different concerns. The key is to model the system at each abstraction layer with as little detail as possible and then collect performance metrics that help the development team make sound engineering decisions.

Among the many metrics used to characterize the quality of an embedded system design, we will go through the following metrics:

- Execution-time
- Power consumption
- Non-Recurring Engineering (NRE)
- Size
- Flexibility
- Unit cost
- Time-to-Market
- Maintainability

Power consumption is a major concern for portable or battery-operated devices. Power issues, such as how long the device needs to run and whether the batteries can be recharged, need to be thought out ahead of time. In some systems, replacing a battery in a device can be a big expense. This means the system must be conscious of the amount of power it uses and take appropriate steps to conserve battery life. There are several methods to conserve power in an embedded system, including clock control, power-sensitive processors, low-voltage ICs, and circuit shutdown. Some of these techniques must be addressed by the hardware designer in his selection of the different system ICs. Some power-saving techniques are under software control.

4. Energy in embedded systems

4.1. What is energy and what is power?

In computing, the terms energy and power are often used interchangeably, but they are not exactly the same. Energy is the ability to do work, and devices like batteries or capacitors can store a fixed amount of it. Power is the rate at which energy is used, and so power and energy are related by time

$$P = \frac{\Delta E}{\Delta T} \qquad (1)$$

where P is average power, ΔE is the energy used and ΔT is the amount of time the energy use takes. For devices with a fixed supply voltage like the processors examined in this thesis, average power is linearly related to average current I as given by

$$P = I \times V_{dd} \qquad (2)$$

Since V_{dd} is fixed, knowing average current means power is known, and energy consumption is given simply as

$$\Delta E = \Delta T * I * V_{dd} \qquad (3)$$

The terms are used interchangeably because they give related information - knowing how much average power a device uses means it is known how much energy it will consume in a given amount of time. In this thesis, the term power will be used to refer to average power and also when discussing current, and the term energy when discussing consumption during a fixed period of time, such as during the execution of a program, instruction, or clock cycle.

4.2. Energy sources in embedded systems

Embedded systems must have a power supply in order to provide energy necessary for its operation. There are different types of energetic resources. Choose one, depends on the size (size, location of the node, etc.), autonomy and functionality required by the application. It is also possible to combine several sources, in the same embedded application to cover several design requirement and constraint.

4.2.1. Battery

This form of power supply is the most used in sensor networks without son. It stores electrical energy through a reaction chemical. The main interest of batteries is their available energy density. They are characterized by low leakage currents for long storage term. They also have the advantage of providing voltage stable power supply allowing them to be connected directly to electronics (Figure 2).

Figure 2 : Different types of batteries for embedded application

The batteries may or may not be rechargeable. Non-rechargeable ones, or batteries, are disposable. They are characterized by energy densities bigger. However, the design of the node should provide for a simple replacement once this source is exhausted. On the other hand, a rechargeable battery, or accumulator, can be recharged from another energy source. An electronic control must be put in place. Note that the density energy from accumulators decreases over time (discharge, aging). Li-Ion batteries and thin-film batteries are the most used in wireless sensor networks. They differ from each other in form, energy density, charging circuit, leakage currents and prices.

4.2.2. Super capacitor

Super-capacities store energy in the form of a field electrostatic. They are a compromise between rechargeable batteries and standard capacities. They have the advantage of having a longer lifespan long and shorter charging time. Their major drawback lies in very large leakage currents. They are then used for short-term storage; their use also requires source energy and control electronics to recharge them.

Figure 3 : Examples of compact supercapacitor

4.2.3. Photovoltaic

A photovoltaic cell converts light into electrical energy. This operation depends on three factors: aging, lighting (spectral composition, angle of incidence and intensity) and temperature. Photovoltaic panel with a backup battery is effective power source for an embedded system. Thus, solar energy is intermittent source, it means that it not disponible all the day, that why we have to store the extending energy in an appropriate battery.

This mode of energy supplying is suited for isolated site, or for space object like satellite for example. Actual photovoltaic cells are highly efficient, reliable, and the lifetime of photovoltaic modules is usually specified by manufactures to be approx. 25 years.

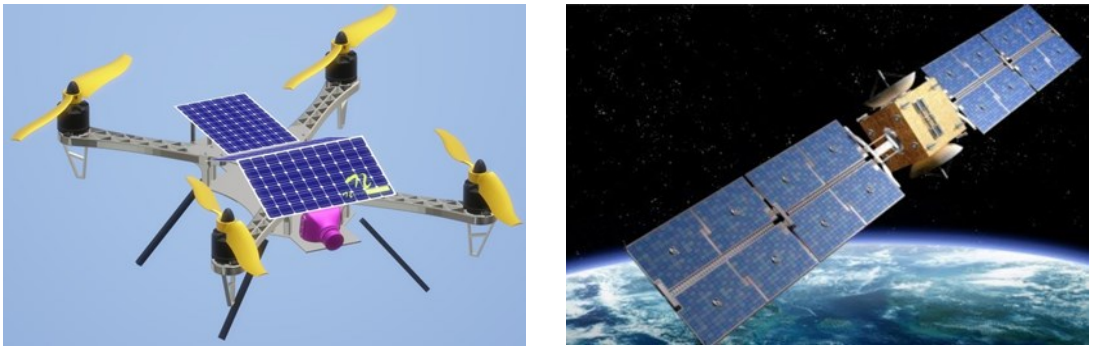


Figure 4 : Example for embedded system supplied with PV panels

4.2.4. Energy recovery

Another solution to power embedded systems is to recover energy from its own environment, like human body vibration or the environment temperature. It reduces dependence batteries as well as maintenance costs (replacing batteries, etc.). It also protects the environment (chemical waste).

5. Optimization of energy consumption

The optimization of energy consumption seeks to reduce waste and unnecessary consumption, it is therefore also an important element of performance environmental in some cases energy saving can even improve the quality of service. The optimization of energy consumption also aims to reduce costs (direct and ecological, economic and social effects of production, transport and energy consumption, it helps reduce the environmental footprint (by the energy footprint and sometimes the carbon footprint), it increases energy security, and

adaptation to climate change and the fight against greenhouse gas energy transition, it is of the five pillars the industrial revolution

6. Low Power Design

Power dissipation is a very critical parameter that has to be taken into account during the design of embedded system. If in the early years of system design the main concern was related to performance and size, sub-micrometer and nanometer technologies have brought power consumption to a main role. Some of the related problems are the heating in high performance systems that leads to the necessity of an adequate cooling system, battery lifetime in portable devices or in wireless sensor networks (WSN) where one of the most critical issues is represented by the limited availability of energy on network nodes sensor, and the high-power budget (e.g., the cost of a data center is determined solely by the monthly power bill, not by the cost of hardware or maintenance).

In embedded system, the most energy-intensive component is the processor (we intend by processor, every element used for data processing including FPGA, ASIC, DSP…etc). In this section we will focus the attention to CMOS (Complementary metal oxide semi-conductor) devices, this technology being the first brick for electronic design, and we should understand where drawing energy is consumed.

6.1. CMOS Power Consumption

Power consumption in CMOS circuits is a function of switching activity, capacitance, voltage, and the transistor structure itself. It can be divided into two different categories called dynamic (proportional to activity) and static (independent to activity) components. Hence a careful balancing between the two is one of the subtleties of advanced low-power design.

In general, the total consumption in a CMOS circuit can be expressed as:

$$Power = P_{dyn} + P_{leak} \qquad (4)$$

6.1.1. Dynamic Power

The charging and discharging of capacitance is the main source of dynamic power dissipation. In fact, these operations are at the core of what constitutes MOS digital circuit design. Other contributions are parasitic effects such as short-circuits currents and dynamic hazards or glitches (fast spikes usually unwanted).

A. Charging and discharging capacitors

For simplicity, we consider a CMOS inverter where the PMOS and NMOS transistors form the resistive charge and discharge networks and the total capacitance of the network is lumped into the output capacitance C_L of the gate as shown in Figure 1. As the input switches from high to low, the NMOS pull-down network is cut off and PMOS pull-up network is activated, charging load capacitance C_L up to $V_{dd.}$ This charging process takes from the supply an amount of energy equal to $C_L V_{dd}^2$. Half of this is stored on the capacitor and the other half is dissipated as heat in the resistance of the charging network. Then, when the input returns to V_{dd}, the process is reversed and the capacitance is discharged, its energy being dissipated in the NMOS network.

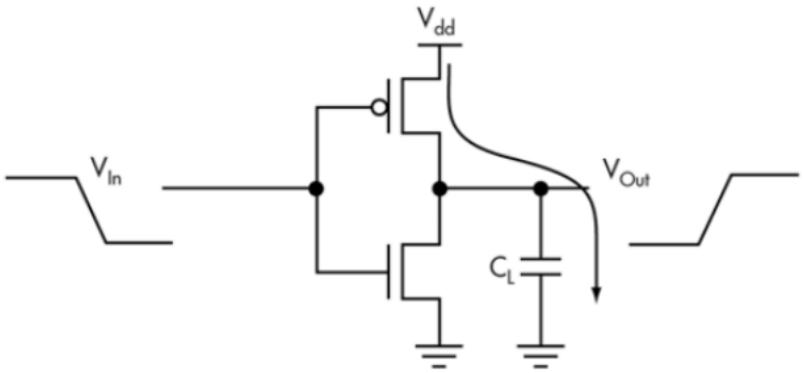


Figure 5: Dynamic charging in a common CMOS inverter

To convert the derived energy per operation into power, it must be multiplied with the frequency of "power-consuming" transitions f0→1 of the output node, i.e., the switching probability p0→1 of the output transition multiplied by the clock frequency of the circuit f. This probability is the switching activity factor α, function of the circuit topology and the activity of the input signals, and his good knowledge has high influence on the accuracy of the power estimation. The average capacitive power is expressed as:

$$P_{dyn} = C_L V_{dd}^2 \alpha f \qquad where \quad 0 \leq \alpha \leq 1 \qquad (5)$$

Here αC_L is called the effective capacitance of the module, and equals the average amount of capacitance that is being charged in the module every clock cycles.

B. Glitch

An additional source of dynamic power dissipation (i.e., proportional to the clock frequency) is the occurrence of events known as glitches. They occur in CMOS circuits when differential delay at the inputs of a gate is greater than inertial delay, which results into

increased gate switching and hence notable amount of power consumption. Such a mismatch in signal timing is typically the result of different path lengths with respect to primary inputs of the network.

Consider Figure 2, in the circuit we can see the unbalanced arrival times of the inputs due to the inverter circuit in the lower input path of the NAND gate. Thus, the differential delay of the NAND gate is 2 units. This differential delay makes the NAND gate to switch 2 times more than the required functioning forming spurious transitions at the output which consume some dynamic power called as glitch power.

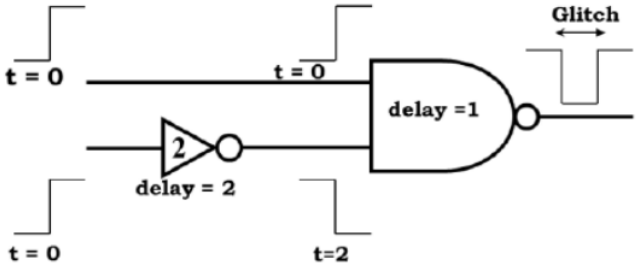


Figure 6: Basic circuit showing the formation of glitches

The glitch power is becoming more prominent in lower technology nodes and is a dominant component in long chains of gates. The solution is balancing data path, for example with the introduction of buffers at the input of the logic gate.

C. Short-circuit currents

The other parasitic effect mentioned above are the short circuit currents. The mechanism of short-circuit power dissipation is depicted in Figure 3 for a CMOS inverter. During an input transition, there will be a time period in which both the NMOS and PMOS will conduct, causing short-circuit current to flow from supply to ground. This current flows within a time window, where input voltage is higher than a threshold voltage of NMOS V_{Tn} (keeping NMOS on), and lower than a threshold voltage of PMOS $V_{T\,p}$ below V_{dd} (keeping PMOS on).

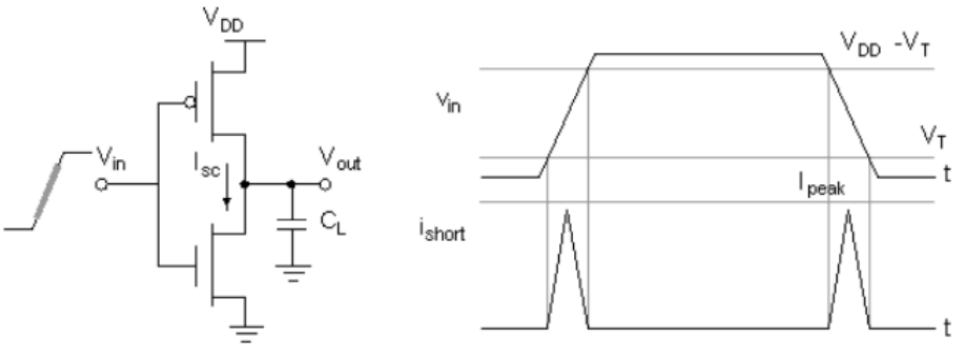


Figure 7: Short-circuit current in CMOS inverter

The peak value of these short circuit or crowbar currents depends on switching activity, similar to the capacitive power dissipation P_{dyn} and also on the ratio between the slopes of the input and the output signals. The latter relationship is illustrated in Figure 4, where the best case (left side) and the worst case (rigth side) are considered.

In the best case the capacitance is very large and the output fall time is significantly larger than the input rise time because of the slower charging of capacitive node. The input moves through the transient region, where PMOS and NMOS are contemporely on, before the output starts to change. Hence V_{DS} is approximately zero and so the PMOS shuts off without delivering any current. Considering the

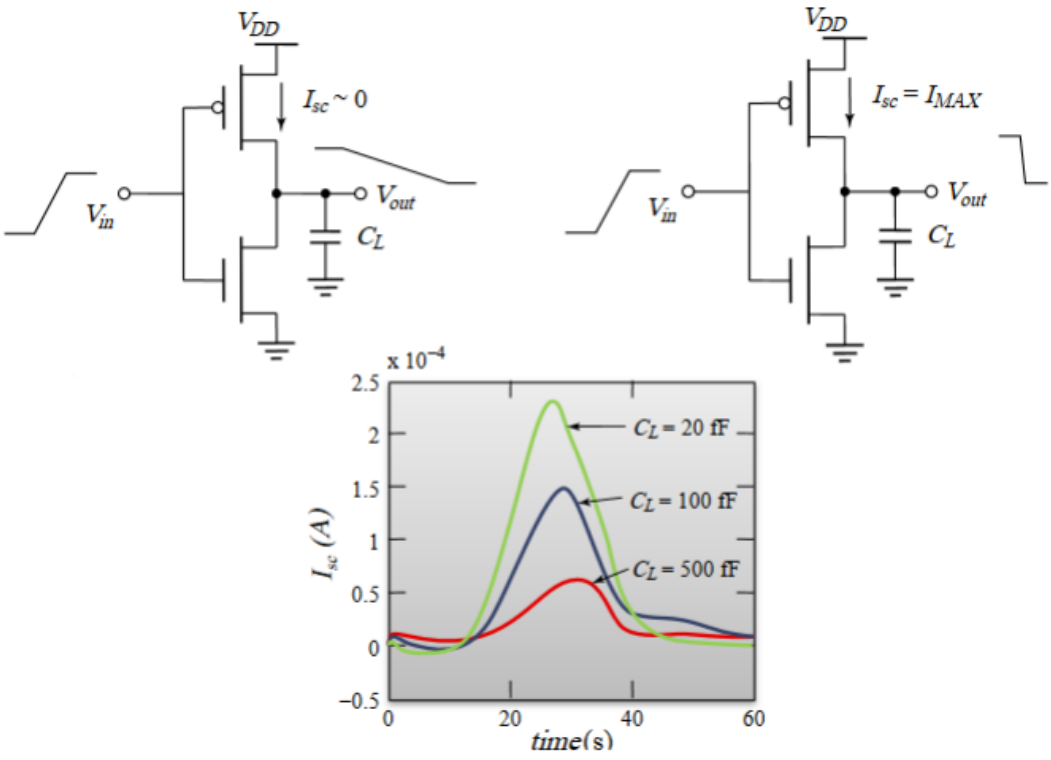


Figure 8: Short-circuit current function of input/output signals' slopes

worst case, having a small capacitance, the fall time of the output is smaller than the fall time of the input and V_{DS} is close to V_{dd} causing a maximal I_{SC}.

The short-circuit power can be modeled as a capacitor C_{SC} function of the input τ_{in} and output τ_{out} transition times:

$$C_{SC} = K\left(a * \frac{\tau_{in}}{\tau_{out}} + b\right) \qquad (6)$$

where:

- a, b = technology parameters
- k = function of supply and threshold voltages, and transistor sizes Therefore the short-circuit power will be expressed by:

$$P_{sc} = C_{sc} V_{dd}^2 f \qquad (7)$$

This analysis may lead to the faulty conclusion that the short- circuit dissipation is minimized by making the output rise/fall time substantially larger than the input rise/fall time but a more practical rule that optimizes the power consumption in a global way is matching the rise/fall times of the input and output signals. This limits the short-circuit power P_{SC} to 10-15% of the dynamic dissipation P_{dyn}.

6.1.2. Static Power

Although dynamic power traditionally have dominated the power budget, static power has become an increasing concern when scaling below 100nm and essentially consists of the power used when the transistor is not in the process of switching. With the scaling of devices, gate oxide thicknesses decrease and there is increased probability of tunneling, resulting in larger and larger leakage currents. As shown in Figure 5, leakage can be divided into sub-threshold leakage, junction leakage and gate leakage.

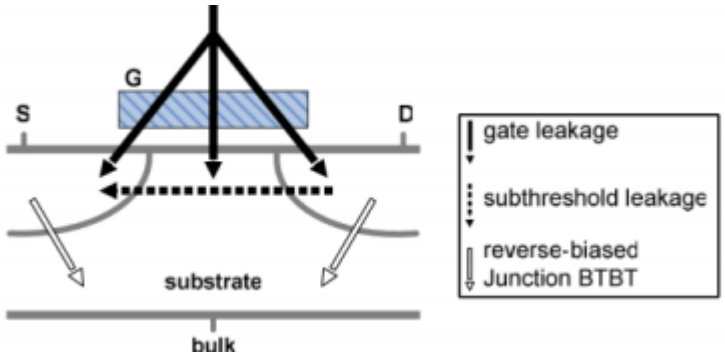


Figure 9: Leakage currents in MOS transistor

A. Sub-threshold leakage

Sub-threshold current between source and drain in an MOS transistor occurs when gate voltage is below V_{Th}. In fact, the current does not drop abruptly to 0 at $V_{GS} = V_{TH}$ but the MOS transistor is already partially conducting. This effect is called sub threshold or weak inversion conduction. With the scaling of transistor size, threshold voltage decreases forced by the lowering of the supply voltage and the drain-source leakage increases. In Figure 6 we can observe such dependence: the leakage voltage at $V_{GS} = 0$ goes up exponentially with a linear reduction in threshold voltage. An additional factor that affects the off-current is the impact of drain-induced barrier lowering (DIBL) effect. Increasing the drain voltage, the potential barries in

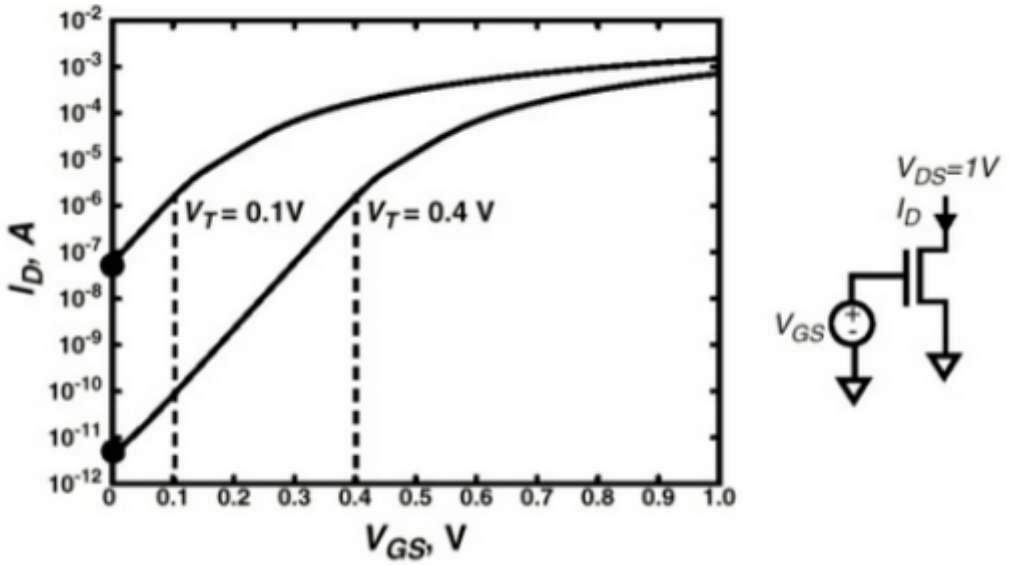


Figure 10: Impact of reduced threshold voltage on leakage

the channel decreases lowering the threshold voltage. Such reduction is approximately linear with V_{DS} as shown by the following expression:

$$V_{GS} = V_{TH0} - \lambda_d V_{DS} \qquad (8)$$

Therefore, the sub-threshold current depends exponentially upon both V_{Th0} and V_{DS}:

$$I_{leak} \propto 10^{-\frac{V_{TH0}+\lambda V_{DS}}{s}} \qquad (9)$$

where S is called slope factor and measure how much V_{GS} has to be reduced for the drain current to drop by a factor 10. It is the slope of MOSFET's current-voltage characteristic in the sub-threshold region and its value at room temperature is 60 mV/decade. Furthermore, the dependence of sub-threshold leakage on temperature is not negligible. In

fact, with an increase in temperature, we have the reduction of the threshold voltage. This means that the sub-threshold leakage is exponentially dependent on temperature.

B. Gate leakage

Another leakage effect that's becoming significant in the sub 100 nm era is the gate leakage. One of the attractive properties of the MOS transistor has always been its very high (not infinite) input resistance but the scaling, leads to gate-oxide thickness of a couple of molecules that cause a reduction in the gate resistance of the transistor, as current starts to leak through the dielectric.

The gate oxide is scaled to increase the process transconductance parameter $K = \mu C_g$ and consequently, to maintain the current drive of the short channel transistor where the saturation of velocity of carrier leads to reduction of mobility. As shown in Figure 7 a way to keep gate leakage under control is obtain the increase of the transconductance k' replacing $Si0_2$ with materials with a higher permittivity, so-called high-k dielectrics.

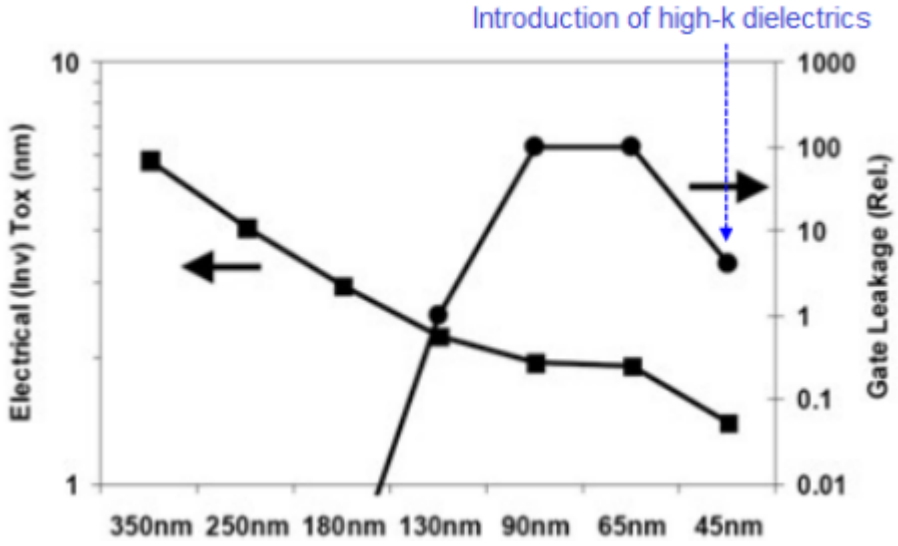


Figure 11: Evolution of the gate thickness and the gate leakage over various technology nodes

The effectiveness of that kind of dielectric is measured by the equivalent oxide thickness (EOT), which equals $Tg(\epsilon_{ox}/\epsilon_g)$ and the advantages are the reduction of gate leakage and faster transistors.

C. Junction leakage

Another leakage contribution is junction leakage substantially smaller than the previously mentioned. These currents are due to the diffusion of minority carriers through the reverse-biased junctions formed by the substrate and source and drain regions.

7. Low power Techniques

From a chip-engineering perspective, effective energy management for an embedded systems must be built into the design starting at the architecture stage and low-power techniques need to be employed at every stage of the design. For this reason, it is important to plan, at the first stages of the design, a low-power design methodology which is capable of covering the following issues:

- Power characterization and modeling (each cell must have a power model in addition to the usual functional and timing models),
- Power analysis method (when and how often to analyze, which modes of the chip to check, how to use the obtained data for optimization),
- Power reduction efforts (which are the power targets and which priority they have, also with respect to other design parameters like die size, performance, design time, etc...),
- Power integrity (analysis of limits due to the power delivery network like electro migration, instantaneous peaks, IR drops).

7.1. At functional block level

7.1.1. Clock Gating

Clock gating is the most popular technique to reduce dynamic power dissipation in synchronous circuits where the clock net is responsible for significant part of power dissipation (up to 40%). It reduces power by disabling the switching of clock tree net in the parts of circuit that are at particular time inactive.

As shown in Figure 12, a clock gating element could be built as a simple AND gate; one input is clock while the second input is an enable signal used to control the output: the clock toggles only when the enable signal is true, and is held steady when the enable signal is false.

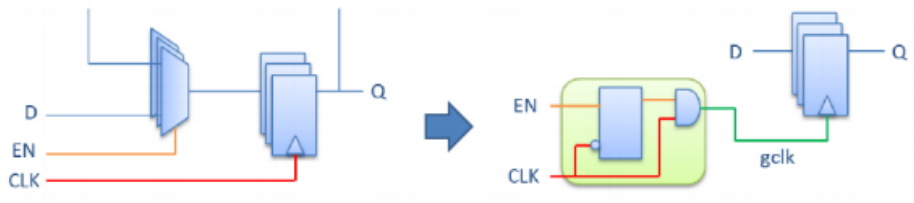


Figure 12: Clock gating implementation

Two different flavors of clock gating are commonly use: the local clock gating which involves gating individual register, or banks of registers, whereas the global clock gating is used to gate all the registers within a block of logic.

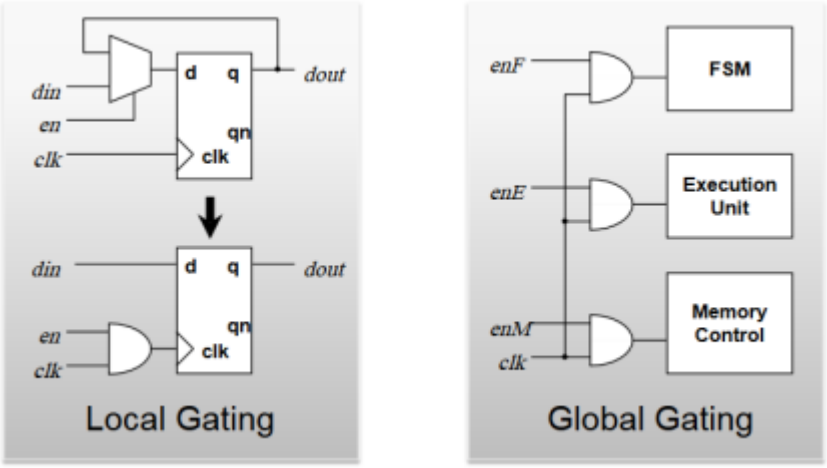


Figure 13: Local and Global clock gating

The modern electronic design automation (EDA) tools support automatic insertion of clock gating that goes deep into design hierarchy.

7.1.2. Operand isolation

Designs which do not fully utilize their arithmetic Datapath components typically exhibit a significant overhead in power consumption. Whenever a module performs an operation, whose result is not used in the downstream circuit, it unnecessarily consumes power. Hence the idea of operand isolation is to identify such operations called redundant, and minimize their power overhead by selectively blocking the propagation of switching activity through the circuit.

A basic application of the operand isolation technique is shown in Figure 10.

If the output C of the adder a_0 will be not stored in registers r_0 and r_1 because of a certain configuration of multiplexer and register enable signals, a_0 will continue to compute a new output whenever there is switching activity at its inputs A and B, therefore consuming power by executing redundant computations.

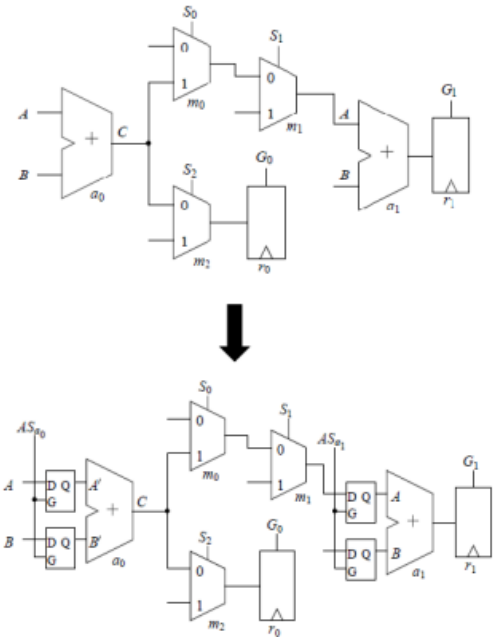


Figure 14: Design with operand isolation

In such situation, to minimize the power overhead the concept of operand isolation is applied using for example a transparent latch that "freeze" the inputs of a_0 effectively preventing the propagation of switching activity into the module.

7.1.3. Pin swapping

The pin swapping technique consists in the reduction of dynamic power consumption assigning a higher switching rate net to a lower capacitance pin. In fact some cells can have input pins that are symmetric with respect to the logic function but have different capacitance values. For example, we can consider 4 input NAND gate with different capacitance value at the pin as show in the Figure 11. The high activity net is connected to the pin no. 4 that is pin "d" which has the maximum input capacitance. Hence, the pin swapping is done between the pins "a" and "d" so that the high activity net is connected to the pin with minimum input capacitance.

Figure 15: Pin swapping applied on a NAND gate

7.2. At system level

The power gating or power shut-off (PSO) is the most effective technique for reducing leakage power. It consists in the adding of a sleep transistor between actual ground rail and circuit ground called virtual ground as illustrated in Figure 12. In the sleep mode the transistor is turned-off and the leakage path is cutted-off with a substantial reduction in leakage. In fact, the virtual ground rail changes up to a steady state value close to V_{DD}.

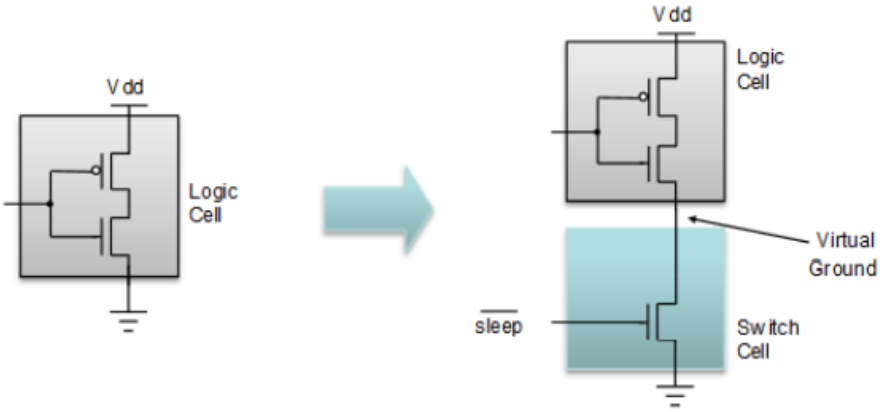


Figure 16: Power gating implementation

However, it also has a drawback that while switching back to the active mode from the sleep mode, the virtual ground rail takes a long time to discharge through the sleep transistor. This result in a significant wake up latency and limits overall leakage savings by limiting how often a logic block can go in and out of the sleep mode.

7.2.1. Multi V_{DD} Design

Multi-voltage design techniques are based on reduction of dynamic power by reducing the V_{DD}^2 term. The idea is to have a design with different power domains, also called voltage islands or power islands that have their separate supply voltage and clock frequency. Several variants are possible and some of them.

8. Conclusion

In this chapter we present an overview of the different thematic related to embedded system design. An extensive part was dedicated to energy source and utilization in embedded system. Therefore, we presented energy sources for embedded system, power consumption and power dissipation phenomena in integrated electronic device.

1. Introduction

An energy profiling is the representation of the system's energy consumption. Because of the general usage, a profiling may highlight some key factors of energy consumption of the system while abstract away some others. Energy profiling can be based on the real measurements or the estimation models.

In this chapter we review the evolution and state-of-the-art in processor's power hardware measurement and estimation models that rely on the running software. For estimation technique, two main abstraction levels are considered. The low-level power modeling and estimation techniques cover the circuit-level, gate-level, Register Transfer (RT)-level and the micro-architecture level. The high-level techniques can be divided into two categories the Instruction Level Power Analysis (ILPA) and the Functional Level Power Analysis (FLPA).

2. Power Analysis

Analysis of power consumption includes various measurement, experimentation and estimation methodologies and techniques that are used to evaluate and validate the power consumption in a system. Power analysis is used to model the power consumption of the various components of a computer system. These components include processors, memory units and other vital parts of the computer system.

This analysis is related to:

Energy consumption of low-power devices is difficult to measure; we can globally divide measurement and estimation methods into three distinct categories: **hardware based**, **software based** and **simulation based**.

2.1. Hardware measurement

Concerning hardware measurement, most of the techniques rely on measuring the voltage drop over a shunt resistor (high precision resistor) in series with the target device to profile.

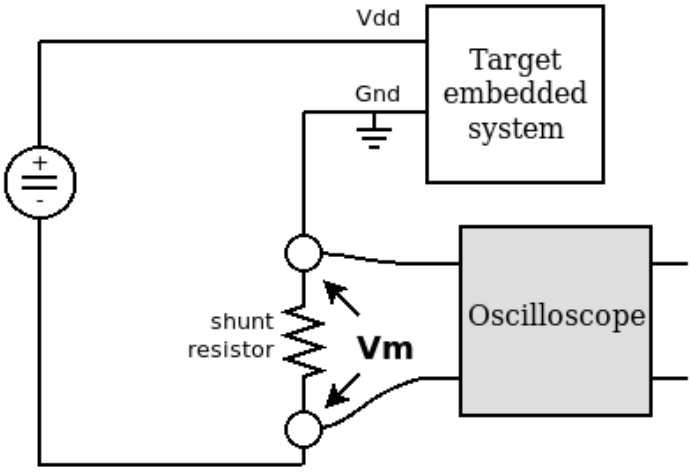


Figure 1: measuring power _embedded systems

Based on that, the data acquisition is performed using a digital oscilloscope or a dedicated board as it is the case for **Flock Lab**, which is a **testbed** for timing and ***power consumption profiling*** which uses the on board **GPIO** of target devices to record logical events and correlate them to the energy drain curve. Alternative hardware methods of using shunt resistor exist. One of them use special capacitors, called **GoldCaps** with very large capacities (1 Farad) for powering the target device instead of using a traditional battery. This enables short term experiments permitting a prediction of the device overall lifetime.

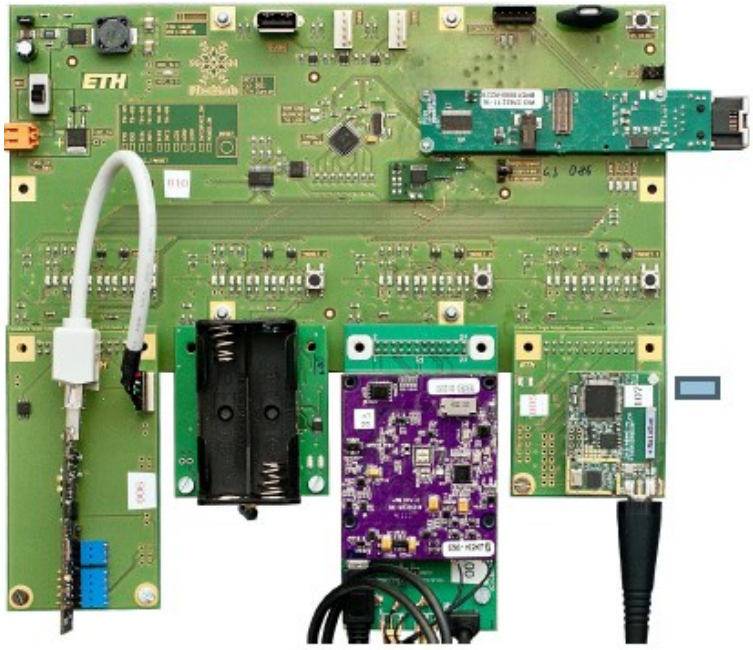


Figure 2: FlockLab observer connected via interface boards

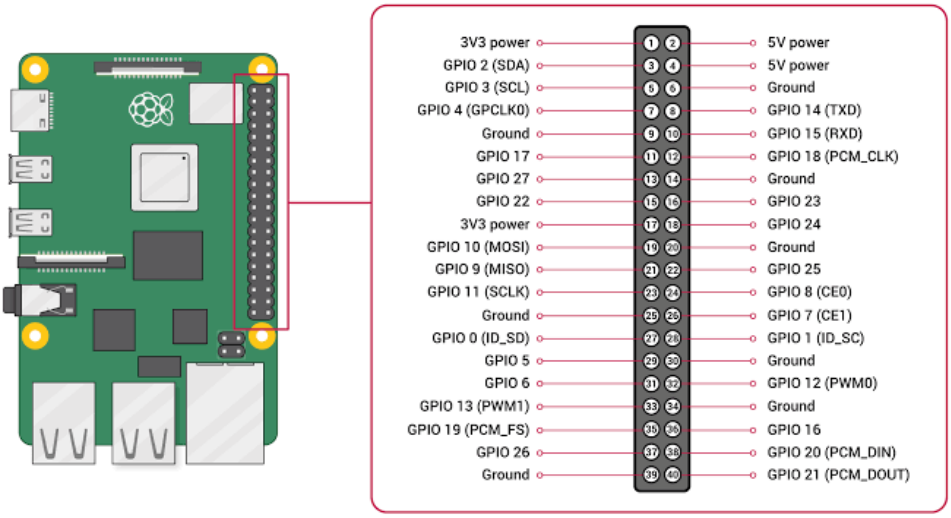


Figure 3: GPIO Raspberry Pi Documentation

Some of hardware measurement platforms could be embedded directly on the target device, giving this latter the possibility of measuring in-situ its own energy consumption.

This knowledge allows the device to perform energy-aware functionalities. **ICount** adds energy metering for free (no hardware overhead) to systems that have a built-in DC-DC boost converter in their power supply and a dedicated hardware counter (Eg. timer) by just counting the switching cycles of the regulator. **Quanto** takes advantage of the on-board energy meter **ICount** and instruments the underlying device OS to track the energy consumption of different device activities. Despite the obvious advantage of these on-boards energy metering modules, their low accuracy makes them unsuitable for fine-grained energy consumption profiling.

2.2.Software based techniques

In contrast with hardware measurements methods, software-based techniques do not require any additional materials to estimate the energy drain of a device. **Powertrace** presented in tracks system power states by measuring the time during which the device components (CPU, Flash, etc.) are in each power state (CPU Active/Sleep, TX/RX, etc.).

Device drivers are instrumented to record a **timestamp** when a component enters a new state. When the component leaves its state, the time difference is computed, then added to the

globalcorresponding component time. Like on-board measurement hardware, **Powertrace** lacks of precision when it is used to profile a precise, fine-grained and local device activity.

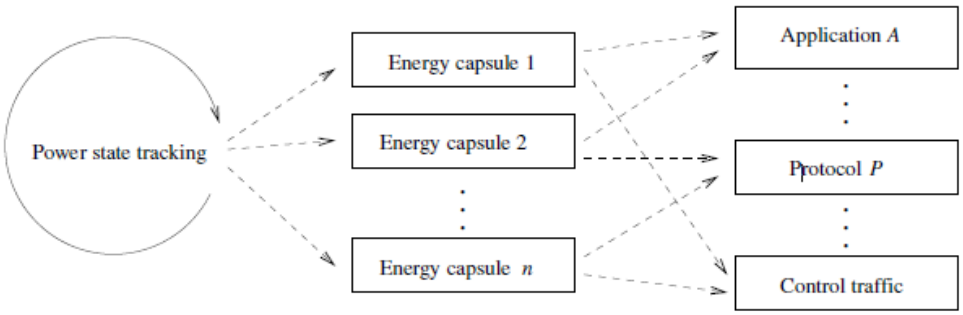


Figure 4: Powertrace power state tracking to estimate the power consumption of the system

2.3.Estimation by simulation

A last category of measurement methods takes the choice of estimating device energy consumption by simulation. We generally call them, **power simulator**. **PowerTOSSIM** is a power simulator that extends the initial **TOSSIM**, an event driven simulation environment for TinyOS applications. It instruments each peripheral components in each state, and employs code-transformation technique to estimate the number of CPU cycles executed by each node, avoiding the need to expensive instruction-based simulation. In contrast, **E-Simu** chooses to model device energy consumption by taking into account energy per instruction and per instruction switching **E-Simu** also adds a characterization of several peripherals. This allows to model the energy profile of an entire platform.

Power simulators represent good tools for quick energy consumption estimation, as they could easily be integrated in the development processes .However, power simulators fail to provide fine-grained estimations of devices in real world execution conditions, due to the unpredictable hardware energy activities and radio interferences to name only a few. Finally, simulators are based on platform architecture models, which is for most of the time, very difficult to construct, develop and validate.

3. Estimation Level & modeling

In general, different abstraction level can be used to describe a processor and its behaviors from different perspective. Similarly, energy estimation could be performed at different abstraction levels. In this section, the brief introduction of energy estimation at different abstraction level is provided, highlighting their advantages and disadvantages.

Accurate models of energy/power consumption estimation have been a crucial research field of many schemes to improve the energy efficiency. These models focus on the initial design of individual component or the whole system to provide the possibility to identify the key influences of the energy consumption.

An energy profiling is the representation of the system's energy consumption. Because of the general usage, a profiling may highlight some key factors of energy consumption of the system while abstract away some others. Energy profiling can be based on the real measurements or the estimation models. Modeling methods of energy estimation involve two important issues during the model constructing: complexity and accuracy, both of which are determined by the abstraction layer on which the energy models are set up. These two issues result in two main categories of the models: low-level and high-level energy estimation models. Low-level models which estimate the energy and power from the detailed design information include circuit level, gate level, register transfer level (RT level) and architectural level. High-level models deal with the instructions, functional units or components to profile the system energy from software point in order to avoid the hardware details.

3.1. Low-Level Power Estimation Models

The level of detail in the modeling performed by the power simulator influences both the accuracy of estimation as well as the speed of the simulator. In this section we survey the models frequently used at low level. as these power consumption estimation techniques cover a range of abstractions such as the circuit/transistor level, logic gate level, RT level

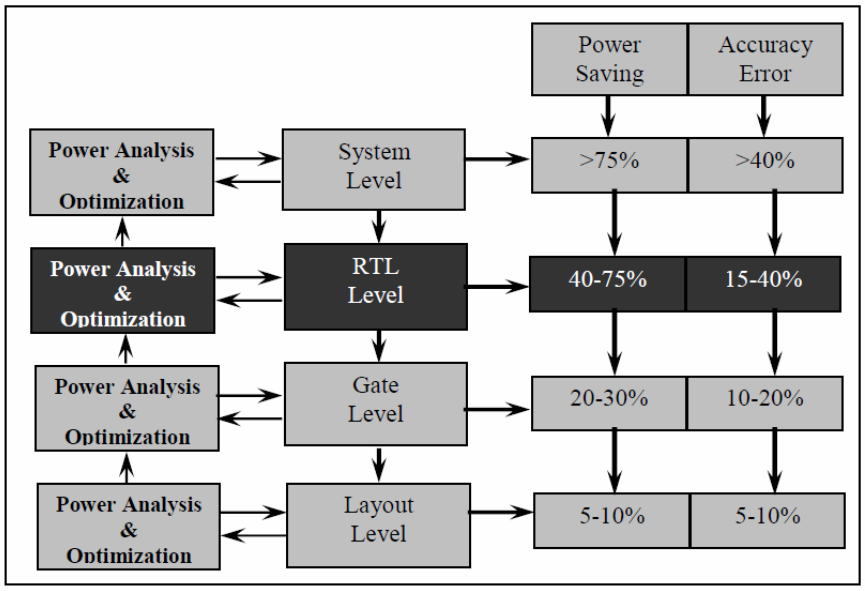


Figure 5: Power Estimation Levels

3.1.1. Circuit/Transistor-Level Estimation Models

A simple and straight method of average power estimation is to simulate the circuit behaviors to obtain the power supply voltage and current waveforms, from which the average power can be computed. Several circuit simulation-based approaches have been proposed in work. In fact, the model in this level is most used for VLSI (Very Large-Scale Integrated Circuits) design and the technology choice. The basic scheme of this approach is shown in Figure 6. A parallel RC sub-circuit is inserted into a VLSI circuit without any interference of the original circuit. The sub-circuit measures the current drawn from the voltage source and computes the average power as equation below:

$$P(t) = u \cdot i(t) = u \cdot \frac{dq}{dt} = u \cdot C_x \frac{du}{dt} = \frac{C_x V_x(t)}{t} \qquad (1)$$

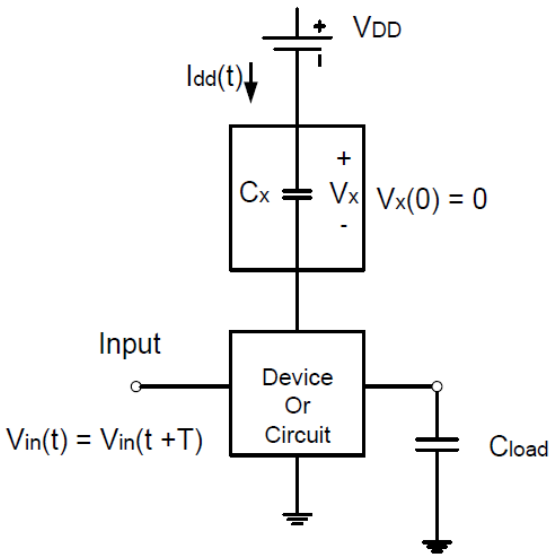


Figure 6 : A Schemes to Measure the Average Current Drawn

However, this method and other transistor-based methods are not suitable for complex circuit design with higher integration density, smaller device geometry, larger chip size and faster clock frequency caused by the rapid development of the CMOS technology.

The representation of modern processor in terms of transistors and nets is extremely complex and requires to undergo all the steps of the design flow and the layout, routing and parameter extraction inclusive. Furthermore, a transistor-level view of the system uses components models based on linearized differential equations and works in the continuous time domain. This implies that a simulation of more than one million transistors, even for few clock cycles, requires times that are usually not affordable and anyway not practical for the high-level power characterization.

Subsequent researchers proposed probabilistic approaches instead of directly simulating a circuit. Probabilistic approaches compute and propagate the probability for a node to change its logic state. There probability methods usually include two kinds of definition:

- Signal Probability: The average fraction of clock cycles in which the steady state value of node x is logic high;
- Transition Probability: The average fraction of clock cycles in which the value of node x at the end of the cycle is different from its initial value.

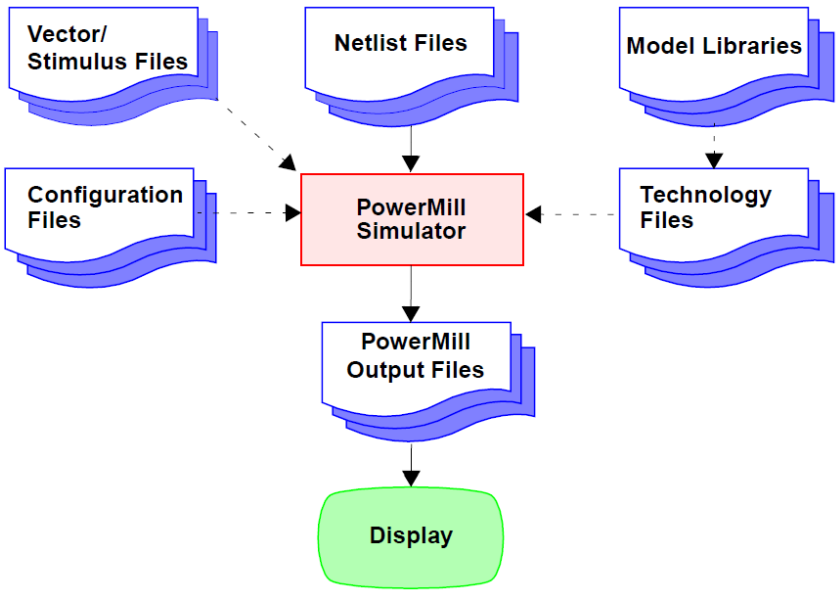


Figure 7 : The PowerMill Environment

Followed with the probabilistic logic, simulator like PowerMil, was proposed to build a transistor-level power consumption model by simulating the current and power behavior in modern deep-submicron VLSI circuits. PowerMill (Figure 7) provides piecewise linear transistor model to capture transistor characteristics from a table to greatly shorten the evaluation overhead. However, it is extremely complex to represent a circuit/transistor-level

models due to all details of the design flow, layout, routing and parameter extraction. Probabilistic-based models can achieve very good accuracy while take unbearable long time to simulate more than one million transistors, thus, they are not suitable for the real-time demand because of both the high complexity and expense. In the other side, they also require the user to specify complete information about the input patterns.

Software tools for circuit level estimation:

- SPICE
- Synopsys PowerMill™
- Cadence VoltageStorm™

3.1.2. Gate-Level Estimation Models

Gate-level estimation methods aim to describe the different gate circuit behaviors during the system runs. The advantage of such methods is that the simulations are driven by events and take place in the discrete time domain. This means that the model is enable to provide an estimation of the switching activities of the basic logic blocks without actually simulating the circuit with a large number of test patterns. Compared with the transistor-level models, gate-level estimation method is faster, can handle larger circuits, and, the most important difference is to be applied before all the circuits details are available.

A gate-level is the representation of the design in terms logic gates or technology specific components, the circuit is described in terms of gates (e.g., and, nand, most commonly because any gate can be done using it, and its transistor circuits is easy to fabricate), as shown in the example of Figure 8.

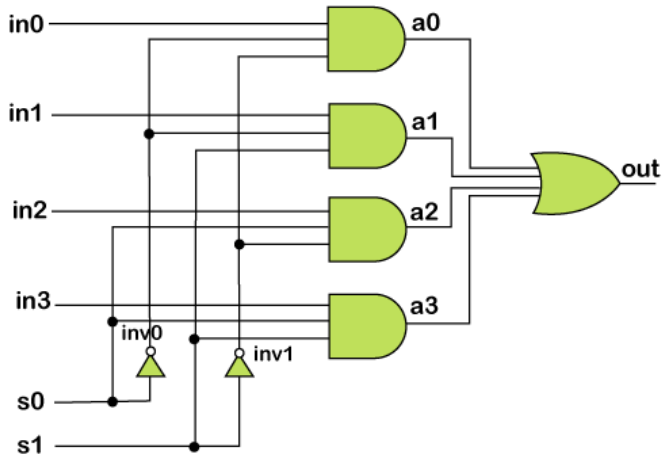


Figure 8 : A Gate-level Example

Gate-level energy estimation needs the gate switching activities (Figure 9); thus, pattern independent approaches are well-suited for this kind of power estimation. Pattern independent methods provide an estimate of the average switching activity without actually simulating the circuit with a large number of test patterns.

Gate-level energy estimation reduce the computational complexity compare with the circuit-level models, meanwhile, it does not loss too much accuracy. The accuracy of the power estimation in this level depends on the high-quality library and a good simulation model for interconnect, glitching, and gate delays. Also, high accuracy requires a large set of input vectors and consequently a long simulation time. The power estimation results in this level can be used as reference for estimations at higher levels of abstraction .

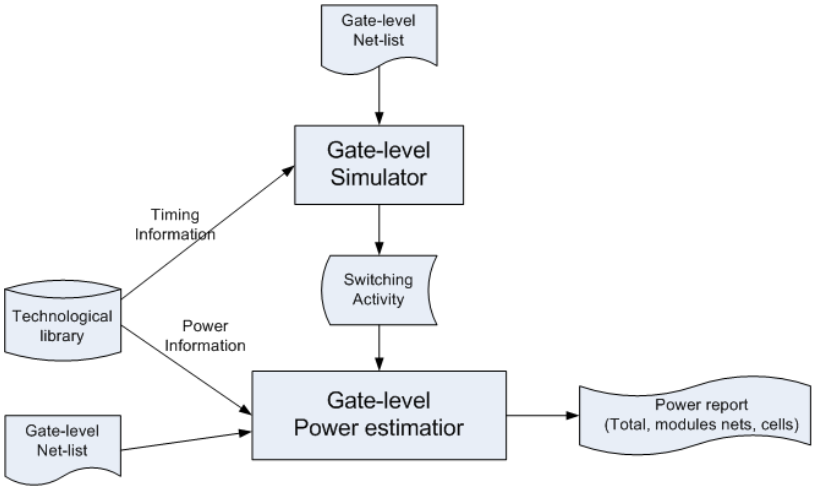


Figure 9 : Gate-level power estimation flow

There are two major types of approaches, dynamic and static, used in gate-level power estimation. Dynamic approaches simulate the circuit based on the input sequence with the system representativeness. Their main shortcomings are their very slow estimation speed and highly dependent results on the simulated sequence. Usually, the required number of simulated sequences is high to produce a valid power estimate. To address this problem, Monte Carlo simulation techniques are proposed. These techniques use an input model based on a Markov process to generate the input stream for simulation. The main difficulty is that it is not clear how the input stream can be efficiently generated when the circuit inputs exhibit complex correlations. The static techniques are implemented based on the statistical information abstract from the input sequences to estimate the internal switching activity of the circuit.

Software tools for gate level estimation:

- Synopsys Prime Power™
- Synopsys Power Compiler™

3.1.3. RT-Level Estimation Models

In digital circuit design, register-transfer level (RTL) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals (RTL example in Figure 10). A register transfer level description captures the application specific integrated circuit (ASIC) behaviors at the physical levels. The simulation approach in RT-level try to functionally estimate and collect the input sequence including blocks or network on interconnections such as adders, registers, multiplexers and the netlists. The power properties of a block could be traced by the application under the controlled operating conditions of an individual block through its input statistics.

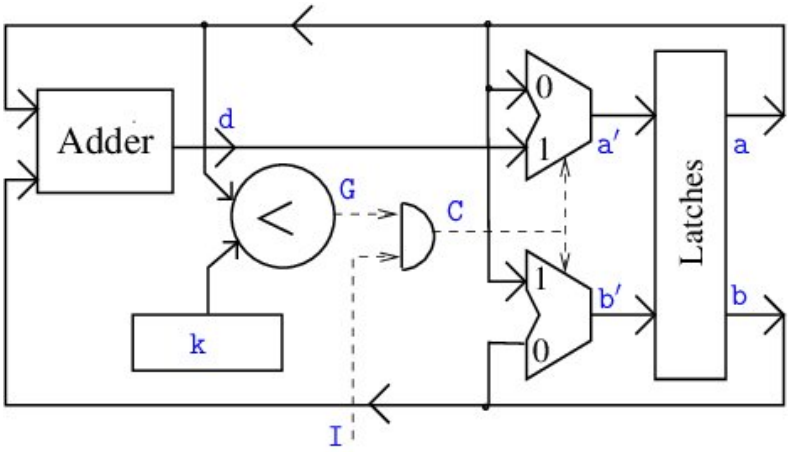


Figure 10 : RTL representation of a simple sequential circuit

Most RT-level power estimation approaches use the capacitance models with activity profiles of data or control signals, which are signal probability or switching activity. For a node to switch state and consume dynamic power, its current state must differ from its previous one, which meant that if the previous state was zero and the node was now directly set to one. Thus, signal probability (SP) is the fraction of time a signal is logic high. This probability of this occurring was referred to the switching activity (SA), thus a simple dynamic power dissipation model of a gate could be calculated by multiplying the SA, the capacitive load (C), the clock frequency (f) and the square of the supply voltage for each node as the formula 2:

$$P_{\text{dynamic}} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f_{\text{clk}} = \alpha \cdot C_L \cdot \Delta V \cdot V_{dd} \cdot f_{\text{clk}} \qquad (2)$$

Here, C_L is the load capacitance, f_{clk} is the clock frequency, V_{dd} is the supply voltage, ΔV is the swing voltage of the node, and α is the node '0→1' transition activity factor which is defined between 0 and 1.

The switched capacitance and switching probability of each functional module are modeled by formulas that are a function of the module's inputs probabilities. These formulas are computed beforehand for each model using the polynomial simulation scheme, and stored in the model library. The switched capacitance for each instance of a module in the circuit can then be efficiently evaluated for its specific input probabilities. The switching probabilities at the outputs of each model can be computed in a similar manner, thus providing a means of propagating the switching probabilities through the circuit described at the RT level.

In [24], Ahuja et al. present a system-level power estimation methodology, which is based on a high-level synthesis framework and supports sufficiently accurate power estimation of hardware designs at the system level.

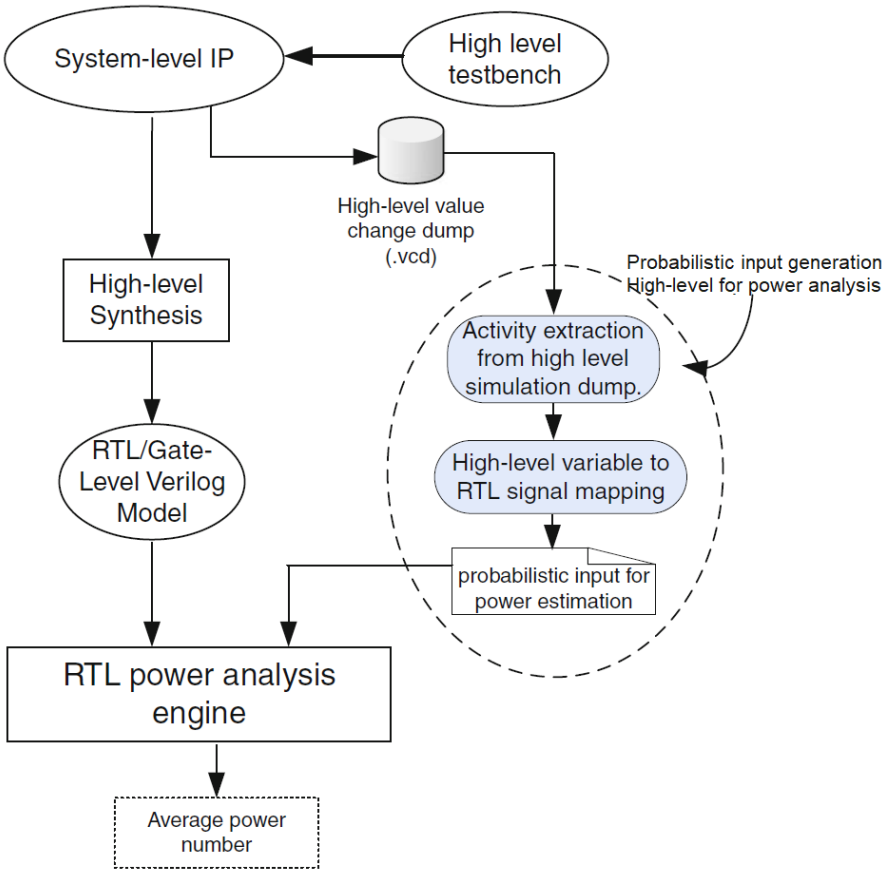


Figure 11 : Example of RTL power estimation flow

In literature we can find several methods that are based on RTL estimation, in the methodology is divided into different steps (as shown in Figure 11): first, authors convert the system-level model of the design to an equivalent cycle-accurate RTL model using specialized software in order to synthesize the high-level models to RTL implementations. Then, they simulate the high-level model and generate its VCD (Value change dump) file. After that, they apply an algorithm to the VCD file generated in order to extract the activities associated with each signal of the RTL design, such as the number of simulation ticks for which the variable value remains unchanged, from the system level simulation dump. Finally, they perform the mapping of system-level variables to RTL signals and use the algorithm outputs to find the activity information of the remaining signals, to generate the power models and to analyze the power consumption using RTL power estimator.

RT-level estimation methods cannot totally avoid the pattern-dependence problem in circuit-level or gate-level since some of the inputs provided by the users are typical behaviors. These inputs are usually based on the probability, which is defined as the average fraction of time that a signal stay in high status and the density, the average number of transitions during each second. Comparatively speaking, this information is much more easily obtained by the designers than specific input patterns are. For example, designers can estimate the average input switching frequencies through the test streams or assume the frequency by the known clock frequency. For the real implementation, the statistical approach can ne construct by the existed simulation tools and libraries, which mainly differs with the dynamic one. In the other word, dynamic modeling approach, which based on the probabilistic technique, requires the specific simulation models.

Software tools for gate level estimation:

- Cadence Genus™
- Cadence Joules™
- Synopsys Power Compiler™

3.2. High-Level Power Estimation Models

As the discussion before, low-level estimation models or simulators are suitable for the hardware designers to make a decision on combinational circuit technology choice. Based on the deliberate choices, the greatest benefits are derived by trying to assess early in the design process the merits of the potential implementation. However, low-level estimation methods are not suitable for the software or OS designers for whom it is difficult to obtain the details of the hardware. Thus, the high-level estimation models are proposed.

3.2.1. Instruction Level Esitmation Models

An instruction level power analysis (ILPA) for individual processors was first proposed in. By measuring the current drawn by the processor as it repeatedly executes distinct instructions or distinct instruction sequences, it is possible to obtain most of the information that is required to evaluate the power consumption of a program for the processor under test. In this method, a power consumption model of the Intel DX486 processor is presented. Power is modeled as a base cost for each instruction plus the inter-instruction overheads that depend on neighboring instructions. The base cost of an instruction can be considered as the cost associated with the basic processing needed to execute the instruction. However, when sequences of instructions are considered, certain inter-instruction effects come into play, which are not reflected in the cost computed solely from base cost. These effects can be summarized as:

- Circuit state: switching activity depends on the current inputs and previous circuit state. In other words, the difference between the bit pattern of two successive instructions.
- Resource Constraints: Resource constraints in the CPU can lead to stalls e.g., pipeline stalls and write buffer stalls.
- Cache Misses: Another inter-instruction effect is the effect of cache misses. The instruction timings listed in manuals give the cycle count assuming a cache hit. For a cache miss, a certain cycle penalty has to be added to the instruction execution time.

An experimental method is proposed to empirically determine the base and the inter-instructions overhead cost. In this experimental method, several programs containing an infinite loop consisting of several instances of the given instruction or instruction sequences

are used. The average current drawn by the processor core during the execution of this loop is measured by a standard off-the-shelf, dual-slope integrating digital multi-meter as shown in Figure 12. This method uses a table to record the average power of each instruction. An accurate model not only needs the pre-estimated power of each instruction but also the effects of inter-instructions. Unfortunately, such a table requires a huge space which is quite high spatial and temporal expense

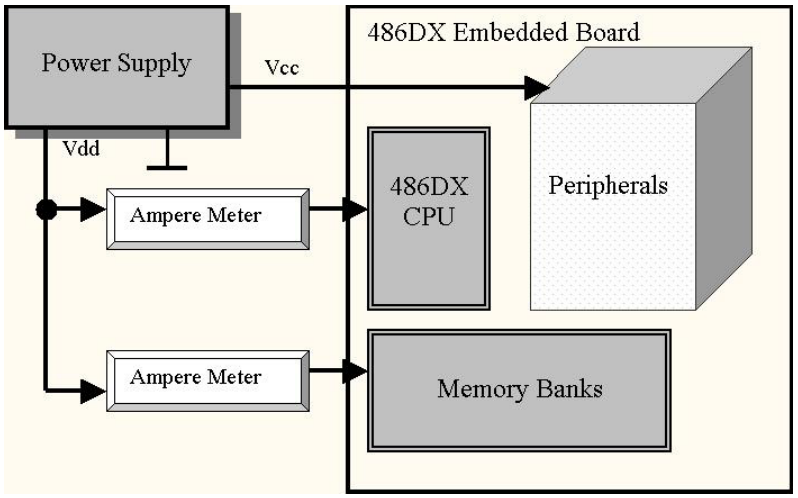


Figure 12: Experimental Setup used for current measurements

Instruction-level modeling approach faces three problems. The first is the numbers of current measurements, which has a direct relationship with the size of the instruction set architecture (ISA). The second is the number of parallel instructions in the very long instruction word (VLIW) processor. The last one is the difficulty to draw the whole picture of the full-system power consumption since this approach cannot provide any insight on the isolate other components. The first two problems cause the model is not general for utility, and the previous work discussed before made some trade-off of the estimation accuracy and the complexity of the individual instruction and the inter-instruction effects. The last one makes this model cannot distinguish the instructions executed on the processor at the specific time related which part on the system, thus the user cannot know which part consume most percentage of the energy.

3.2.2. Function-Level Estimation Models

Function-level power analysis (FLPA) is applicable to all types of processor architectures without the details of the system circuits. Instead of the classical energy characterization abstracted from the instructions, the basic idea of FLPA is to obtain the distinction energy consumption from system activities of different processor functional blocks. Thus, FLPA model first divides the target into several functional units. Then it relates the processor operations to the power activations.

FLPA was first introduced in. Figure 13 illustrates the process of estimating the power consumption with aid of the FLPA technique. The basic idea behind the FLPA is the distinction of the processor architecture into functional blocks like Processing Unit (PU), Instruction Management Unit (IMU), internal memory and others. First, a functional analysis of these blocks is performed to specify and then discard the non-consuming blocks. The second step is to figure out the parameters that affect the power consumption of each of the power consuming blocks. For instance, the IMU is affected by the instructions dispatching rate which in turn is related to the degree of parallelism. In addition to these parameters, there are some parameters that affect the power consumption of all functional blocks in the same manner such as operating frequency and word length of input data. The functional level power modeling approach is applicable to all types of processor architectures. Further-more, FLPA-modeling can be applied to a processor with moderate effort and no detailed knowledge of the processor's architecture is necessary [Ibrahim thesis].

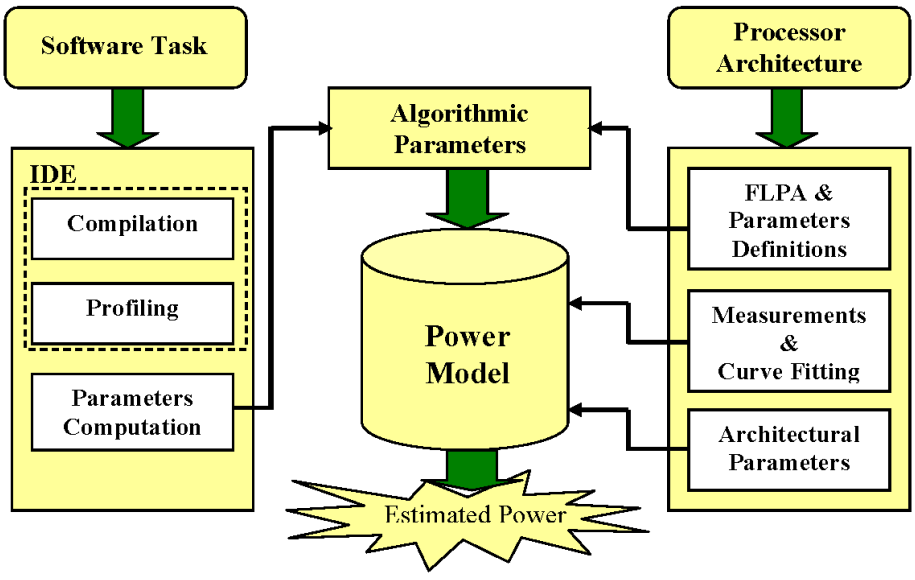


Figure 13 : Functional level power estimation general methodology.

Based on this methodology, the SoftExplore, a tool automatically performs power and energy consumption estimations, are widely used by lots of research groups. In this case, the model only requires coarse-grain knowledge on the processor architecture and it achieves a good tradeoff between the estimation accuracy and the model complexity. However, its main disadvantage is the complexity of the component's determination, the coverage of all significant influencing parameters and dependency of the corresponding power consumption on the performed instruction. Furthermore, during the determination period of consumption laws, the issue of temperature of the system which has an obvious impact on the static power consumption is not taken into account. The coefficients of the model could be different as the temperature increased.

4. Conclusion

In this chapter we review the evolution and state-of-the-art in processor's power consumption modeling and estimation methodologies that rely on the running software. In general, two main abstraction levels are surveyed in this chapter. The low-level power modeling and estimation techniques cover the circuit-level, gate-level, Register Transfer (RT)-level and the micro-architecture level.

The high-level techniques can be divided into two categories, the Instruction Level Power Analysis (ILPA) and the Functional Level Power Analysis (FLPA). This survey leads us to the appropriate power estimation technique for VLIW processors.

1. Introduction

In this chapter, we make a review on techniques used for power optimization, power saving and power management. This review involves the essential of those technics including carachteristics and benefits.

2. Power Saving Techniques: Overview

Low power design is a complex endeavor requiring a broad range of strategies from floor planning on silicon substrate to the design of application software. In Table 1, we enlisted several strategies for achieving power efficiency in an power-conscious system design. In this section, we review some of these strategies.

Power Saving Technique	Abstract Level
Manufacturing level power saving	Low level
Processor level power saving	Intermediate level
Dynamic voltage and frequency scaling (DVFS)	Intermediate level
Battery aware power saving	High level
Compiler level power saving	High level

Table 1 : Power saving techniques for embedded systems.

2.1.Manufacturing Level Power Saving

There are three major sources for the power dissipation in digital CMOS circuits, these sources are summarized in (2.1) [58].

$$P = KC_L V_{dd}^2 F + I_{SC} V_{dd} + I_{Leakage} V_{dd} \qquad (1)$$

Where C_L is the output capacitance load, V_{dd} is the supply voltage, K is the transition activity factor which is defined as the average number of times the circuit makes a power consuming transition in a single clock cycle (this term is defined as the probability that a power consuming transition occurs), and F is the operating clock frequency. The short circuit current pulse is expressed by the term I_{SC} which is generated when both n-CMOS and p-CMOS transistors are briefly turned on during the output switching, and $I_{Leakage}$ is the leakage current.

It is expected that employing low-power electronics can achieve significant power saving. Half of this power reduction will come from architecture changes and management of

switching activity. The other half of this power reduction will come from the utilization of advanced materials technology to allow reduction of V_{dd} to 1V or even below, while also reducing C_L.

2.2. Processor Level Power Saving

One software technique offered by many embedded processors to conserve power is different operating modes. These modes allow the software to scale processor power consumption to match the moment-by-moment needs of the application, for example:

- Run-mode: the processor core runs at its normal frequency; this is the normal or default operating mode.
- Idle-mode: the processor core is not clocked, but the other peripheral components operate as normal.
- Sleep-mode: this is the lowest power state for the processor.

The embedded system designer needs to determine when the system is not doing anything and how to wake up the processor when it needs to operate, and the designer need to know what events will wake up the system. For example, in an embedded system that sends some data across a network every few minutes, it makes sense to be able to shut down the device to conserve power until it is time to send the data. The device must still be able to wake up in case an error condition arises. Therefore, the designer must understand how a peripheral circuit wakes up the processor when the processor needs to operate (including how long it takes the circuit to wake up and whether any re-initialization needs to be done) .

2.3. Battery Aware Power Saving

Chiasserini and Rao have shown how battery behavior can be exploited to prolong battery life. In particular, they identify the phenomenon of charge recovery that takes place under pulsed discharge conditions as a mechanism that can be exploited to enhance the capacity of an energy cell. The bursty nature of many data traffic sources suggests that there might be a natural fit between the two. P. Rong and M. Pedram in address the problem of maximizing the utilization of the battery capacity of the power source for a portable electronic system under a given performance constraint. They propose a new stochastic model of a power-managed battery-powered electronic system, which is based on Continuous-Time Markovian Decision Processes (CTMDP).

In this model, two important characteristics of today's rechargeable battery cells, i.e., the current rate-capacity characteristic and the relaxation-induced recovery are considered.

2.3.1. Battery-Aware Power Management Based on Markovian

With the rapid progress in semiconductor technology, chip density and operation frequency have increased, making the power consumption in battery-operated portable devices a major concern. High power consumption reduces the battery service life. The goal of low-power design for battery-powered devices is thus to extend the battery service life while meeting performance requirements. Dynamic power management (DPM) - which refers to a selective, shut-off or slow-down of system components that are idle or undemtilized - has proven to be a particularly effective technique for reducing power dissipation in such systems. Early DPM works described predictive shutdown approaches b a d on "time-out" policy. A power management approach based on discrete-time Markovian decision processes was proposed in The discrete-time model requires policy evaluation at periodic time intervals and m y thus consume a large amount of power dissipation even when no change in the system state has occurred. To overcome this shortcoming, a model based on continuous-time Markovian decision processes (CTMDP) was proposed in The policy change under this model is asynchronous and thus more suitable for implementation as part of a real-time operating system environment. Reference [SI also improved on the modeling technique of by using time-indexed semi-Markovian decision processes. Although the abovementioned DPM techniques may successfully reduce the system power consumption, they are not able to obtain the optimal policy far a battery-powered system. This is because the characteristics of battery power source are not properly modeled or exploited in these techniques. As demonstrated by research results in the total energy capacity that a battery can deliver during its lifetime is strongly related to the discharge current rate. More precisely, as the discharge current increases, the deliverable capacity of the battery decreases. This phenomenon is called the (current) rate-capacity characteristic. Another important property of batteries, which was analyzed and modeled in is named the relaxation phenomenon (or recovery effect). It is caused by the concentration gradient of active materials in the electrode and electrolyte formed in the discharge process. Driven by the concentration gradient, the

active material at the electrolyte-electrode interface, which is consumed by the electrochemical reactions during discharge, is replenished with new active materials through diffusion. Thus, the battery capacity is somewhat recovered during a no-use state. Due to these non-linear characteristics, a minimum power consumption policy does not always

necessarily result in the longest battery service life because the energy capacity of its power sources may be not fully exploited when the cut-off voltage of the battery is reached.

A number of battery models have been proposed. These can be divided into *0x0* categories: electrochemical model and stochastic model. The electrochemical models are based on diffusion equations and provide an accurate description of the underlying electrochemical process. A low-level model for lithium batteries and a high-level model for the time-varying load were proposed in [SI y d respectively. The electrochemical models require a'

predetermined workload profile. However, in most real situations, the workload is unknown a priori and often evolves as a random process. In these cases, stochastic models are needed. Stochastic models describe the battery behavior as a stochastic process whose

parameters are extracted from the electrochemical characteristics of the simulated battery. Some stochastic models have been reported in the literature, e.g., a discrete-time VHDL model and a discrete time Markovian chain model. The stochastic model is a Markovian chain of the battery's states of charge with forward and backward transitions corresponding to the normal discharge and recovery effect, respectively. The load is expressed as a stochastic demand on the charge units.

2.4. Compiler Level Power Saving

Compiler design techniques contribute to energy saving in several ways. Kolson et al. address the problem of allocating memory to variables in embedded DSP (digital signal processing) software. The goal is to maximize simultaneous data transfers from different memory banks to registers. In several DSP applications mentioned in two registers are loaded with the required data and an arithmetic operation is performed. Loading two registers with a single double transfer instruction draws a little more current than a move instruction. Both instructions take one clock cycle each. However, energy is saved by employing the double transfer, because the double transfer instruction loads the two registers in one clock cycle, whereas it takes two clock cycles to sequentially load the registers. Instructions with memory operands have much higher energy cost than instructions with register operands This suggests that energy can be saved by suitably assigning the live variables of a program to registers. But, a processor has only a small number of registers. When the number of simultaneous live variables is larger than the number of available registers, some of the variables must be spilled to memory. Register assignment for loop variables is important because loops are typically

executed many times. Algorithms for optimal register assignment to loop variables are presented in These algorithms can be included in the code generation part of a compiler.

2.4.1. Influence of compiler optimizations on power and energy usage

Recently some attempts to understand the scope of compiler optimizations, from the perspective of power dissipation and energy consumption of programmable processors have been introduced. Tiwari et al presented an instruction level power model, following the same methodology published in for a Fujistu 3:3v, 40MHz DSP. Moreover, the effect of two architectural features (dual-memory accesses, and packing of instructions into pairs) on the energy consumption has been exposed.

With the help of a cycle-accurate energy simulator (SimplePower), a source-to-source code translator, and a number of benchmark codes, Kandemir et al studied the influence of five high-level compiler optimizations, such as loop unrolling and loop fusion, on energy consumption. Valluri et al provided an evaluation of some general and specific optimizations in terms of the power/energy consumption of the Alpha processor while running some SpecInt95 and SpecFp95 benchmarks. The processor in their work was simulated by means of Wattch (A frame work for analyzing processor power consumption at architectural level) Chakrapani et al also presented a study into the effect of compiler optimization on the energy usage of an embedded processor. Their work targets an ARM embedded core and they use an RTL level model along with Synopsys Power Compiler to estimate power. Seng et al revised the effect of the Intel compiler general and specific optimizations, for energy and power consumption, for a Pentium 4 processor running some benchmarks extracted from Spec2000. Azzemi examined the effect of loop unrolling factor, grafting depth and blocking factor on the energy and performance for the Philips Nexperia media processor ”PNX1302”. But, they interchangeably use the term energy and power for the same meaning. Hence the improvement in energy is directly related to the performance enhancement. Finally, Casas et al studied the effect of various compiler optimizations on the energy and power usage of the low power C55 DSP from Texas Instruments. Their work was based on a physical measurement platform for measuring the current drawn by the DSP core. The work does not consider the effect of the compiler optimizations on many performance measures that significantly affect the power and energy usage, such as the memory access and the instructions per cycle.

2.4.2. Compiler Optimizations

In this section, we introduce the optimizations evaluated (i.e., linear loop transformations, tiling, unrolling, fusion, fission, and scalar expansion) and discuss their impact on energy consumption. Linear loop transformations attempt to improve cache performance, instruction scheduling, and iteration-level parallelism by modifying the traversal order of the iteration space of the loop nest. The simplest form of loop transformation, called loop interchange, can improve data locality (cache utilization) by changing the order of the loops. From the power consumption point of view, by applying this transformation we can expect a reduction in the total memory power due to better utilization of the cache For the power consumed in other parts of the system, we do not anticipate a major variation for this small example after the transformation. This may not be true in general, though, as some loop transformations can result in complex loop bounds and array subscript expressions that can potentially increase the power consumed in the core datapath. Another important technique used to improve cache performance is blocking or tiling. When it is used for cache locality, arrays that are too big to fit in the cache are broken up into smaller pieces (to fit in the cache). When we consider power, potential benefits from tiling depend on the changes in power dissipation induced by the optimization on different system components. We can expect a decrease in power consumed in memory, due to better data reuse On the other hand, in the tiled code we traverse the same iteration space of the original code using twice as many loops (in the most general case); this entails extra branch control operations and macro calls. These extra computations might increase the power dissipation in the core. Loop unrolling reduces the trip count of a given loop by putting more work inside the nest with the aim of reducing the number of memory accesses and promoting the register reuse. From the power point of view, fewer accesses to the memory means less power dissipation. In addition, we can also expect a reduction in the power consumed in the register file and data buses.

2.4.3. Compiler optimizations for low power systems

Most current compiler optimizations focus on improving execution time. With the increasingly widespread use of embedded systems, however, power/energy consumption is also becoming an important issue. This is particularly true for battery-operated devices where power consumption has first class status along with performance and form factor.This paper makes the following contributions. First, we present two low-level (back-end) compiler optimizations for energy reduction. An important conclusion drawn from evaluating the impact of these optimizations is that compiling for power/energy is different from compiling

for execution cycles. Second, we evaluate widely used state-of-the-art high-level compiler optimizations from a power consumption perspective. Further, we compare the relative impact of these high-level optimizations on both energy and performance metrics in order to identify any differences in optimizing for energy and performance. Finally, we cover a set of optimizations that are designed specifically for exploiting low power features supported by the hardware. In particular, we show how loop and data transformation can be used to exploit low-power mode control mechanisms available in some memory architectures.

3. Power management

Power management is, any electronic device feature that allows users to manage the quantity of power consumed by an underlying device, with minimal impact on performance. Power management allows the switching of devices in various power modes, each with different power usage characteristics related to device performance. Power management has become a major issue in the design of Embedded systems. There are many adverse effects like unstable thermal properties of the die, effect the system performance and also result in increasing power consumption. This makes power consumption issue sometimes more important than speed of the system. The power management in Embedded systems design can be achieved at different levels like, at chip level, architectural level, application level and system level. Among all these, system-level power management techniques have importance, because, to reduce the power consumption, it's to better to follow an intelligent power-aware technique than low-power design.

Power management techniques

Based on their main power saving approach, power management techniques can be classified as follows.

a) DVFS (Dynamic voltage and frequency scaling)
b) Dynamic Power Management (DPM)

3.1. Dynamic Voltage & Frequency Scaling (DVFS)

DVFS is a method of changing the voltage and frequency of an electronic system based on performance and power requirements. This method basically refers to the change in voltage levels supplied to various system components during runtime so as to reduce the

overall system power consumption while maintaining the throughput requirement. In a system primarily the power consumption is monitored by the following equation:

$$P = CV^2f$$

Where P is the power consumed, C is the switching capacitance, V is the supply voltage and f is the frequency of operation. It is possible to control the amount of consumed power by simply adjusting voltage-frequency pairs. Several commercial Embedded processors support DVFS technology for saving the power. The basic idea is to tune voltage and frequency pairs within a predefined level, to reach the required power and performance level.

The system level controller guides the global on-chip controller with the suitable power budget. The global controller checks for voltage, frequency and power usage of each core. Based on these parameters, the global controller actuates voltage and/or frequency as and when required. DVFS methodology is widely applied to almost all modern processors in Embedded Systems, rather than only in general purpose applications Hua discussed the importance of DVFS in saving the power in Embedded computing systems using DVFS and addressed DVFS can also be used in real-time applications. In discussed the energy minimization factor when running real-time applications that have strict reliability and deadline necessities. Checkpointing, Dynamic Voltage Frequency Scaling (DVFS) and backward fault recovery techniques are used to confirm the satisfaction of the running application's reliability and deadline necessities.

Quan proposed DVFS based methods for optimizing energy in real-time Embedded systems. Studied the problem of determining the optimal voltage schedule, for a real-time system with fixed-priority jobs, running on a variable voltage processor. Also shown that in designing many real-time Embedded systems, proper usage of variable voltage processors can dramatically reduce system energy consumption.

Edward's khan et presented an investigative algorithm, based on mathematical optimization techniques to identify energy-aware processor frequencies for soft real-time Embedded systems. Using the mathematical models, it was possible to identify the desired frequency for a task assuming availability of constant range of frequencies. The proper selection was based on choosing, the closest frequencies which are lesser and bigger than the desired frequency and some of the time each of them should be used to reach the deadline from real-time tasks. This approach provided performance improvements for the issue of multiple target deadlines for each task.

Wonyoung Kim discussed DVFS methods for each core, in chip-multiprocessors (CMP) systems for optimizing energy. They experimented the DVFS methods by considering practical overheads, voltage transition time and advantages, by using on-chip voltage regulators in a 4-core multiprocessor.

3.1.1. What is DVFS technique

Dynamic voltage and frequency scaling (DVFS) is a technique that aims at reducing the dynamic power consumption by dynamically adjusting voltage and frequency of a CPU.

3.1.2. Analysis of DVFS technique

Dynamic voltage frequency scaling (DVFS) is accepted as a technique to reduce power and energy consumption of microprocessors Lowering only the operating frequency fclk can reduce the power consumption but the energy consumption remains the same because the computation needs more time to finish. Lowering the supply voltage Vdd can reduce a significant amount of energy because of the quadratic relation between power and Vdd as given in Equation 2. Lowering the supply voltage and operating frequency reduces the power and energy consumption further. Figure 2 shows the power saving achievable by using variable Vdd

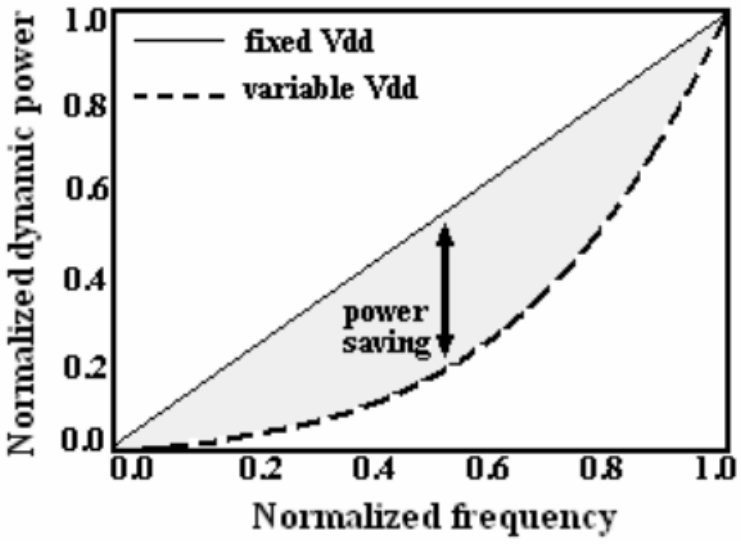


Figure 2. Power saving achievable by Using variable Vdd

When the clock frequency f_{clk} is reduced by half, this lowers the processor's power consumption and still allows task to complete by deadline, the energy consumption remains the same. Reducing the voltage level Vdd by half reduces the power level further without any

corresponding increase in execution time. As a result, the energy consumption is reduced significantly, but the appropriate performance is remained. This is shown in Figure 3:

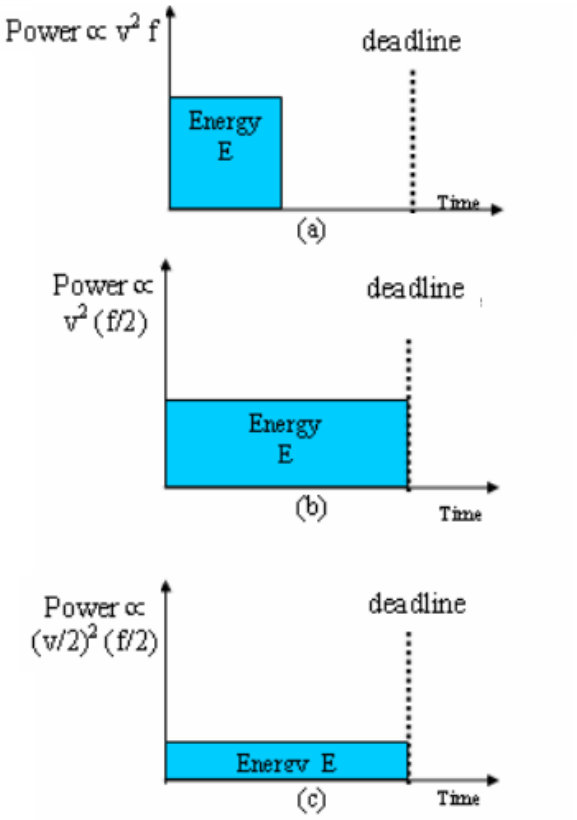


Figure 3. Energy consumption vs. power consumption for a task, which is ready at 0 and complete at T, with maximum clock frequency f_{clk}

There are three key components for implementing DVFS technique in processors :

1. An operating system which intelligently vary the processor speed.
2. A control loop which generates the voltage required for the desired speed.
3. A microprocessor which operates over a range of voltages.

The relationship between these three components is shown in Figure 4

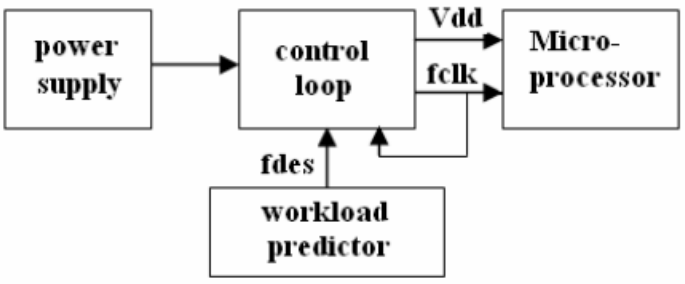


Figure 4. DVFS required components

3.2. Dynamic Voltage Scaling (DVS)

In contrast to clock gating, which can only be applied to idle components, DVS targets components that are in active state, but serving a light workload. It has been proposed as a mean for a processor to deliver high performance when required, while significantly reducing power consumption during low workload periods. The advantage of DVS can be observed from the power consumption characteristics of digital static CMOS circuits (2) and the clock frequency equation:

$$delay \propto \frac{V}{(V - V_K)^{\alpha}} \qquad \text{and} \qquad f_{CLK} \propto \frac{(V - V_K)^{\alpha}}{V} \qquad (3)$$

Where V is the supply voltage, and f_{CLK} is the clock frequency. α ranges from 1 to 2, and V_k depends on threshold voltage at which velocity saturation1 occurs.

Decreasing the power supply voltage would reduce power consumption quadratically as shown in Equation (2). However, this would create a higher propagation delay and at the same time force a reduction in clock frequency as shown in Equation (3). While it is generally desirable to have the frequency set as high as possible for faster instruction execution, the clock frequency and supply voltage can be reduced for some tasks where maximum execution speed is not required. Since processor activity is variable, there are idle periods when no useful work is being performed and DVS can be used to eliminate these power-wasting idle times by lowering the processor's voltage and frequency.

In order to clearly show the advantage of DVS techniques, Figure 2 compares DVS with the simple On/Off scheme, where the processor simply shuts down when it is idle (during time 2~4, 5~7 and 8.5~11 in the figure). DVS reduces the voltage and frequency, spreading the workload to a longer period, but more than quadratically reducing energy consumption. A quick calculation from Figure 2 shows about 82% reduction in power based on Equation (2) because $E_{DVS} / E_{On/Off} = (4\times(0.5)^3 + 3\times(0.33)^3 + 4\times(0.375)^3) / (2\times1^3 + 1\times1^3 + (1.5)\times1^3) = 0.82/4.5 = 0.18$. Note that each task workload, which is represented by the area inside the rectangle in Figure 2, remains the same for both the simple On/Off and DVS mechanisms.

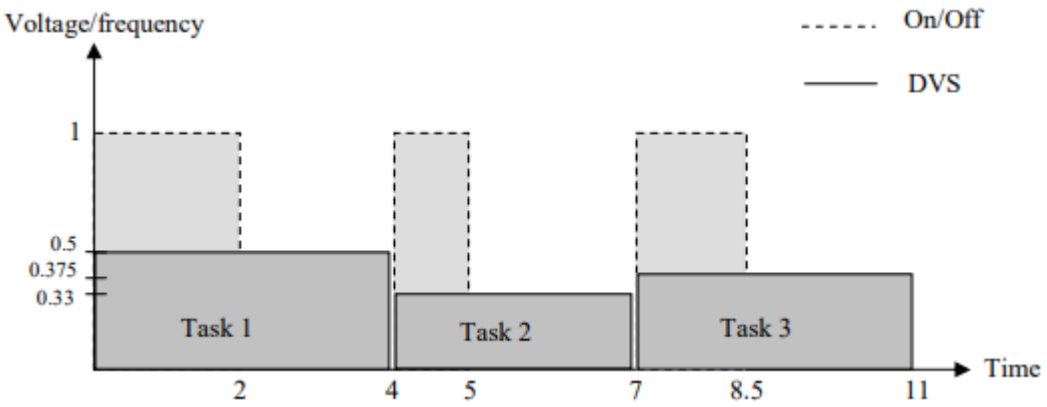


Figure 2: Voltage scheduling graph with On/Off and DVS mechanisms.

Current custom and commercial CMOS chips are capable of operating reliably over a range of supply voltages and there are a number of commercially available processors that support DVS mechanisms. Table 6 shows the Mobile Intel Pentium III processor with 11 frequency levels and 6 voltage levels with two performance modes: Maximum performance mode and battery optimized performance mode. The maximum performance mode is designed to provide the best performance while the battery optimized performance mode provides the balance between performance and battery lifetime. Crusoe processor from Transmeta, Inc. also has variable voltage and frequency as presented in Table 7.

3.3. Dynamic Power Management (DPM)

DPM techniques, which are based on timeout or event prediction, assume multiple components with multiple modes. Subsystem shutdown decision can be based on fixed idle Time, adaptive timeout, or profile and runtime history The simplest power management policy is time-out with a fixed or predicted amount of time before the system's shutdown or power-up Stochastic models are used to address the uncertainty in system behaviors. Simunic et al combined stochastic-modeled power management with dynamic voltage scaling to achieve significant power reduction in portable systems.

DPM techniques can be effective for minimizing energy and time penalties on average, but they have several limitations. First, most treat either power or timing as an objective or penalty, rather than a constraint. In real systems, the max power is a real, hard constraint, whose violation can lead to malfunction. Max power was not of central concern previously, but as we consider additional power sources such as solar whose maximum output can vary, they must be strictly satisfied This becomes especially important as we increase the dynamic range of power by increasing parallelism. Second, they have not considered intercomponent

dependency in a system, with the exception of Qiu, Qu and Pedram in Their Generalized Stochastic PetriNet (GSPN) can capture some dependencies in clientserver applications. However, only one server is modeled to process an incoming request.

Figure 1 summarizes the features of the techniques surveyed here. The last column shows our new approach, mode selection, which combines the advantages of existing approaches. It is constraint-driven, enabling us tomake power/performance trade-offs with out hardwiring specific goals or policies in the design.

	DPM		DVS		
	[35], [9]	[3], [25]	[7], [29], [30], [27]	[18], [19]	MS
Timing as Constraint	N	N	Y	Y	Y
Power as Constraint	N	N	N	N	Y
Timing overhead	Y	Y	N	N	Y
Power overhead	Y	Y	N	N	Y
Multiple resources	N	Y	N	Y	Y

Figure 1: Comparison of dynamic power management (DPM), dynamic voltage scaling (DVS), and mode selection (MS).

3.3.3. CPU-level DPM

The intuition behind power saving at the CPU-level comes from the basic energy consumption characteristics of digital static CMOS circuits, which is given by

$$E \propto C_{eff} V^2 f_{clk} \qquad (4)$$

Where C_{eff} is the effective switching capacitance of the operation, V is the supply voltage, and f_{clk} is the clock frequency The DPM techniques presented in this section reduce the power consumption by targeting one or more of these parameters? Subsection 3.1.3.1.1 discusses techniques to reduce the switching activity of the processor, mainly at the Datapath and buses. In Subsection 3.1.3.1.2 *clock gating* techniques are discussed, which reduce power consumption by turning off the idle component's clock, *i.e.* $f_{clk} = 0$.

3.4.3.1 Reducing switching activity

Reducing switching activity plays a major role in reducing power consumption. A number of such optimization techniques have been proposed to reduce switching activity of internal buses and functional units of a processor. In case of buses, energy is consumed when wires change states (between 0 and 1). Different techniques are used to reduce the switching

activity on buses by reducing the number of wire transitions. Stan and Burleson proposed bus-invert coding where the bus value is inverted when more than half the wires are changing state In other words, when the new value to be transmitted on the bus differs by more than half of its bits from the previous value, then all the bits are inverted before transmission. This reduces the number of state changes on the wire, and thus, save energy.

Henkel and Lekatsas proposed a more complicated approach where cache tables are used on the sending and receiving sides of the channel to further reduce transitions That is, when a value "hit" is observed at the input of the channel, the system will only send the index of the cache entry instead of the whole data value, which will reduce the number of transitions. Finally,

Wen et al. used bus transcoding to reduce bus traffic and thus power based on data compression on bus wires. As an enhancement to this technique, transition coding was also proposed where the encoding of data represents the wire changes rather than the absolute value, which simplifies the energy optimization problem.

On the other hand, the processor's switching activity can also be reduced by using power aware compiler techniques. Although applied at compile time, these are considered as DPM techniques because their effect is closely tied to the system's run-time behavior. For example, in instruction scheduling technique instructions are reordered to reduce the switching activity between successive instructions. More specifically, it minimizes the switching activity of a data bus between the on-chip cache and main memory when instruction cache misses occur

Cold scheduling prioritizes the selection of the next instruction to execute based on the energy cost of placing that instruction into the schedule. Another compiler-based technique called register assignment focuses on reducing the switching activity on the bus by re-labeling the register fields of the compiler-generated instructions. A simulator, such as *SimplePower* is used to parameterize the compiler with sample traces. In other words, it records the transition frequencies between register labels in the instructions executed in consecutive cycles and this information is then used to obtain a better encoding for the registers such that the switching activity and consequently the energy consumption on the bus is reduced.

3.4.3.2 Clock gating

Clock gating involves freezing the clock of an idle component. Energy is saved because no signal or data will propagate in these frozen units. Clock gating is widely used because it is conceptually simple; the clock can be restarted by simply de-asserting the clock-freezing signal. Therefore, only a small overhead in terms of additional circuitry is needed, and the component can transit from an idle to an active state in only a few cycles. This technique has been implemented in several commercial processors such as Alpha 21264 and PowerPC 603 The Alpha 21264 uses a hierarchical clocking architecture with gated clocks. Depending on the instruction to be executed, each CPU unit (*e.g.*, floating point unit) is able to freeze the clock to its subcomponents (*e.g.*, adder, divider and multiplier in floating point unit).

The PowerPC 603 processor supports several sleep modes based on clock gating. For this purpose, it has two types of clock controllers: global and local. Clocks to some components are globally controlled while others are locally controlled. For example, consider PLL (Phase Locked Loop) that acts mainly as a frequency stabilizer and does not depend on global clock.

Even though clocks to all units are globally disabled and the processor is in sleep state, the PLL can continue to function which makes a quick wake-up (within ten clock cycles) possible. On the other hand, if the PLL is also turned off, maximum power saving would be achieved but the wake-up time could be as long, to allow the PLL to relock to the external clock.

3.3.4. Complete system-level DPM

As discussed before, the CPU does not dominate the power consumption of the entire system. Other system components, such as disk drives and displays, have a much larger contribution. Therefore, it is necessary to consider all of the critical components of the system to effectively optimize power. A well-known system-level power management technique is shutting down hard drives and displays when they are idle. A similar idea can also be applied to other I/O devices to save energy. However, changing power states of hardware components incurs not only time delay but also energy overhead. Consequently, a device should be put to sleep only if the energy saved justifies the overhead. Thus, the main challenge in successfully applying this technique is to know when to shut down the devices and to wake them up.

A straightforward method is to have individual devices make such decisions by monitoring their own utilization. One clear advantage of this device-based scheme is transparency, *i.e.*, energy saving is achieved without involving or changing application or system software. On the contrary, this scheme may perform poorly because it is unaware of the tasks requesting the service of the device. Software-based DPM techniques have been proposed to alleviate this problem. Application or system software takes full responsibility on power-related decisions assuming that devices can operate in several low power modes using control interfaces such as Advanced Configuration and Power Interface (ACPI)

3.4.3.3 Hardware device-based DPM policies

Hardware device-based policies observe hardware activities and workloads of the target device and change power states accordingly. They are usually implemented in hardware or device drivers without direct interaction with application or system software as illustrated in Figure 3. Based on prediction mechanisms for future device usage, these methods can be classified into three categories: *Time-out*, *Predictive*, and *Stochastic* policies

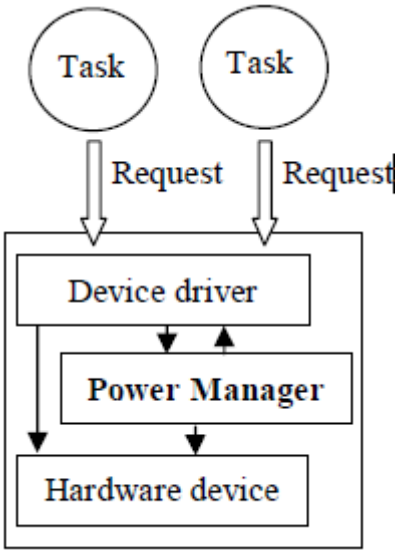


Figure 2: Hardware device-based DPM

3.4.3.4 Software-based DPM policies

While hardware device-based power management policies can optimize energy-performance of individual devices, they do not consider system-wide energy consumption due to the absence of global information. Therefore, software-based DPM policies have been proposed to handle system- level power management. Advanced Configuration and Power Interface (*ACPI*) proposed as an industrial standard by Intel, Microsoft and Toshiba, provides

a software-hardware interface allowing power managers to control the power of various system components. Application and operating system-based DPM techniques, which will be discussed later in this section, utilize such interface to conserve power. Although application-based schemes can be the most effective because future workloads are best known to applications, OS-based schemes have a benefit that existing applications do not need to be re-written for energy savings.

- Advanced Configuration and Power Interface (ACPI)

ACPI evolved from the older Advanced Power Management (*APM*) standard targeting desktop PCs Implemented at the BIOS (Basic I/O Services)-level, APM policies are rather simple and deterministic. Application and OS make normal BIOS calls to access a device and the APMaware BIOS serve the I/O requests while conserving energy. One advantage of APM is that the whole process is transparent to application and OS software. ACPI is a substitute for APM at the system software level. Unlike APM, ACPI does not directly deal with power management. Instead, it exposes the power management interfaces for various hardware devices to the OS and the responsibility of power management is left to application or operating system software. Figure 4 overviews the interactions among system components in ACPI . The front-end of the ACPI is the ACPI-compliant device driver. It maps kernel requests to ACPI commands and maps ACPI responses to I/O interrupts or kernel signals. Note that the kernel may also interact with non-ACPI-compliant hardware through other device drivers.

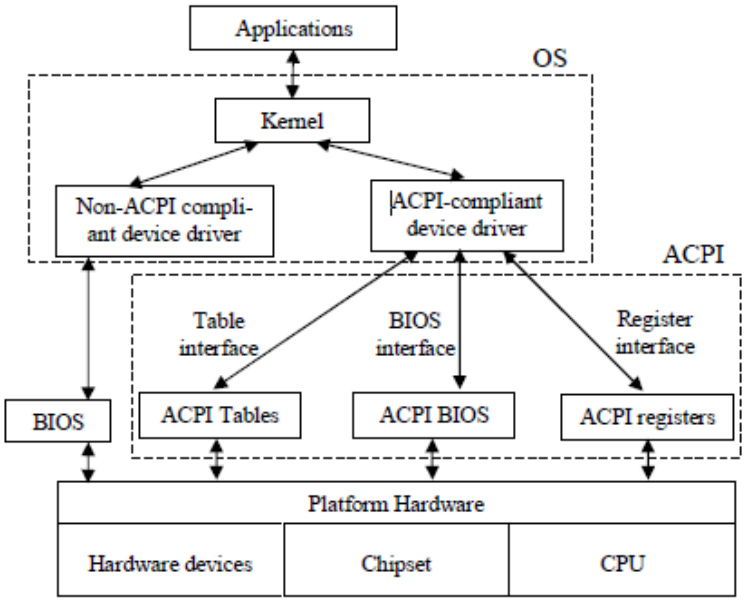
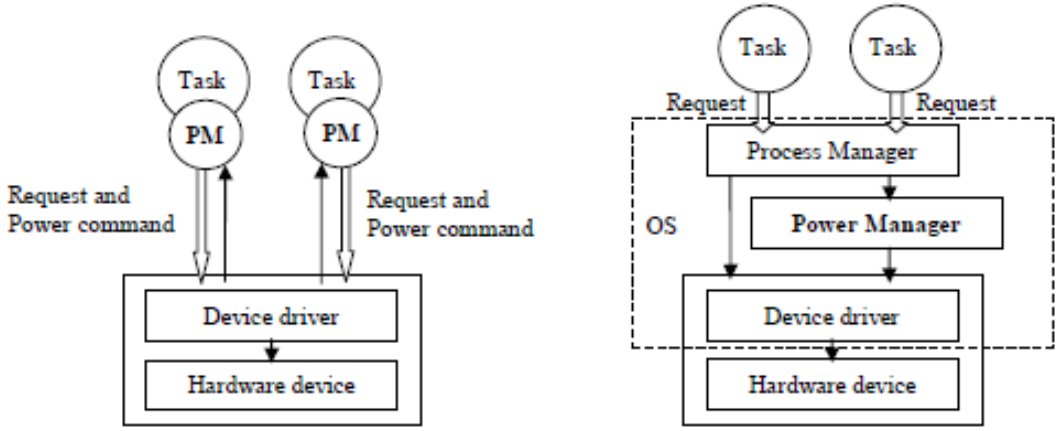


Figure 3: Interaction among system components with ACPI

3.3.5. Application-based DPM

Application-based DPM policies were made possible by the emergence of the ACPI standard state above. These policies move the power manager from the device or hardware level to the application level. The application, which is the source of most requests to the target device, is now in charge of commanding the power states of that device. These policies allow application programs to put a device in the fully working state, send it to sleep mode, wake it up, or receive notice about the device power-state changes. For example, Microsoft's *OnNow* and *ACPI4Linux* support power management for ACPI-compliant devices. Figure 8(a) illustrates the application-based DPM scheme.



(a) Application-level power management (b) OS-level power management

Figure 4: Software-based DPM policies

Alternatively, Lu *et al.* proposed a software architecture that exports power management mechanisms to the application level through a template This scheme makes power state decisions based on information about the system parameters such as power at each state, transition energy, delay, and future workload. This software architecture differs from the ACPI-based techniques in that the power management is centralized to one application, which makes it safer and more efficient in a single application system.

3.3.6. Operating system-based DPM

Application-based power management has several drawbacks. First, they require modifications to existing applications, and thus implementing these policies will place additional burden on the programmers. Second, advances in technology constantly change hardware parameters, which make a policy optimized for a certain environment inefficient after a device is replaced. Finally, different programs may set the same device to different

power states causing the system become unstable. OS-based DPM techniques use the operating system's knowledge of all the running processes and their interaction with the hardware to optimize energy performance. A number of OS-based DPM schemes have been proposed in the literature Figure 5(b) illustrates the implementation of OS-based power management. Lu *et al.* proposed *task-based* power management, which uses process IDs at the OSlevel to differentiate tasks that make I/O requests This has a major advantage over device- based policies in that it offers a better understanding of device utilization as well its future usage pattern. For example, an OS-based DPM scheme, called *power-oriented process scheduling*, schedules tasks by clustering idle periods to reduce the number of power-state transitions and state-transition overhead Finally, Gniady *et al.* proposed to use program counters to predict I/O activities in the operating system Their *program-counter access predictor* dynamically learns the access patterns of an application using path-based correlation to match a particular se38 quence of program counters leading to each idle period. This information is then used to predict future occurrences of this idle period and thus optimize power.

3.3.7. DPM ARCHITECTURE

A number of strategies for saving energy using DVFS techniques have been developed We believe that sincethe most effective way to manage energy consumption is highly dependent on the particulars of the embedded system, a generic power management architecture are best served by leaving the workings of the DVFS system completely transparent to most tasks, and even to the core of the OS itself. These considerations led to the development of an architecture for policy-guided dynamic power management called DPM. It is important to note at the outset that DPM is not a DVFS algorithm, nor a power-aware operating system such as described in nor an all-encompassing power management control mechanism such as ACPI. Instead, DPM is an independent module of the operating system concerned with active power management. DPM policy managers and applications interact with this module using a simple API, either from the application level or the OS kernel level. Although not as broad as ACPI, the DPM architecture does extend to devices and device drivers in a way that is appropriate for highly integrated SOC processors (but not discussed here). A key difference with ACPI is the extensible nature of the number of power manageable states possible with DPM. While DPM is proposed as a generic feature for a general-purpose operating system, our practical focus has been the implementation of DPM for Linux. DPM implementations are expected to be included in embedded Linux

distributions needs to be flexible enough to support multiple platforms with differing requirements. Part of this flexibility is the requirement to support "pluggable" power management strategies that allow system designers to easily tailor power management for each application. Although excellent results have been obtained with kernel-level approaches for DVFS, we believe that the requirements for simplicity and flexibility

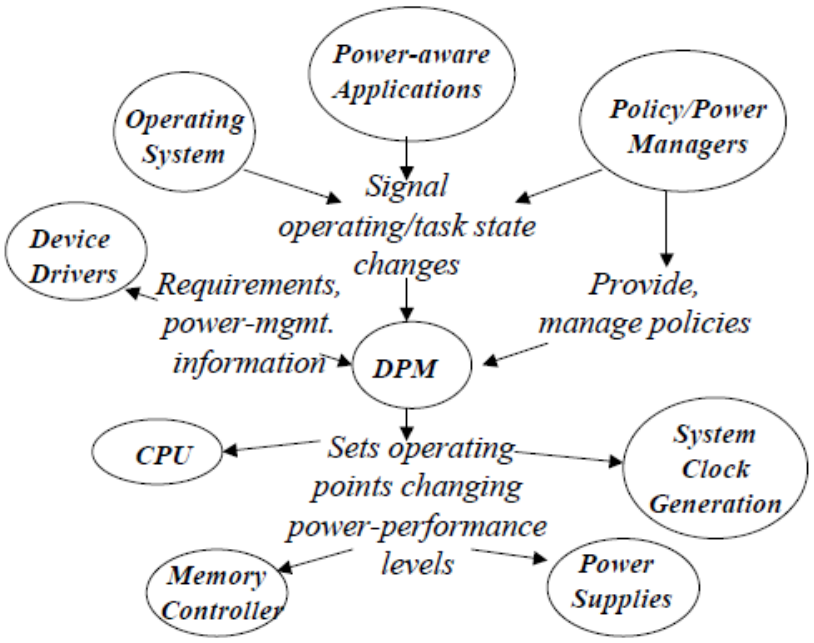


Figure 5: DPM Architecture for the 405LP and other processors. The following is a brief overview of the DPM architecture.

4. Conclusion

In this chapter, we presented different techniques for power optimization from the design to the system operation. In the other hand, we reviewed the most employed power management techniques to optimize system energy consumption, these techniques allow embedded systems to control and adjust operating frequency or voltage supply to a current functioning mode.

General Conclusion

In the context of this thesis, the main objective is to study techniques for optimizing and managing energy in embedded systems, we introduced various energy technologies for consumption monitoring, energy saving and optimization. Usually several of these techniques are used for electronic and computing system.

This present work is accomplished in the graduation final project, it allowed us to study the following:

We did an overview in chapter 1 of the integrated systems and of how energy consumption and sources are improved, and in chapter 2 we provided an audit of the analysis of energy and its technologies, which provide us with real or estimated results and consumption models as well as techniques for managing, improving and maintaining energy consumption.

In the last chapter we presented the principals techniques for power optimization and power management, these techniques reduce energy consumption and enhance lifetime system. In embedded system, it is important to improve perpetually device performance while reducing power need.

Finally, we can say that for embedded systems there is a lot of work to be done, to carry out multiple constraint improvement, above all performance and power consumption, without complicating the design stages.

Reference

[1] Google.omni.sci/technical.glossary/embedded systems

[2] Programming embedded systems.with C and GNU development tools

[3] Ramesh U. B. K.SentillesS.Crnkovic I.

[4] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2013.

[5] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso, "Experimental evaluation of lifetime bounds for wireless sensor networks," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, 2005.

[6] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler, "Energy metering for free: Augmenting switching regulators for real-time monitoring," in *Information Processing in Sensor Networks (IPSN), 2008.*, April 2008.

[7] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking energy in networked embedded systems," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2008.

[8] Cherifi, N., Grimaud, G., Vantroys, T., &Boe, A. (2015, August). Energy consumption of networked embedded systems. In *2015 3rd International Conference on Future Internet of Things and Cloud* (pp. 639-644). IEEE.

[9] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Networklevel power profiling for low-powerwireless networks," SICS, Tech. Rep., 2011.

[10] Cherifi, N., Grimaud, G., Vantroys, T., &Boe, A. (2015, August). Energy consumption of networked embedded systems. In *2015 3rd International Conference on Future Internet of Things and Cloud* (pp. 639-644). IEEE.

[11] V. Shnayder, M. Hempstead, B.-R. Chen, and M. Welsh, "Power-TOSSIM: Efficient Power Simulation for TinyOS Applications," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[12] N. Fournel, A. Fraboulet, and P. Feautrier, "Embedded software energy characterization: Using non-intrusive measures for application source code annotation,"*Journal of Embedded Computing*, Jul. 2009.

[13] Cherifi, N., Grimaud, G., Vantroys, T., &Boe, A. (2015, August). Energy consumption of networked embedded systems. In *2015 3rd International Conference on Future Internet of Things and Cloud* (pp. 639-644). IEEE.

[14] Ibrahim, M., Rupp, M., &Fahmy, H. (2011). A precise high-level power consumption model for embedded systems software. *EURASIP Journal on Embedded Systems*, *2011*, 1-14.

[15] Najm, F. N. (1994). A survey of power estimation techniques in VLSI circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *2*(4), 446-455.

[16] Ibrahim, M., Rupp, M., &Fahmy, H. (2011). A precise high-level power consumption model for embedded systems software. *EURASIP Journal on Embedded Systems*, *2011*, 1-14.

[17] Huang, C. X., Zhang, B., Deng, A. C., &Swirski, B. (1995, April). The design and implementation of PowerMill. In *Proceedings of the 1995 international symposium on Low power design* (pp. 105-110).

[18] Ibrahim, M. E. A. (2009). Power/energy estimation and optimization for software-oriented embedded systems. *PhD Disseration*.

[19] Ibrahim, M. E. A. (2009). Power/energy estimation and optimization for software-oriented embedded systems. *PhD Disseration*.

[20] Ibrahim, M. E. A. (2009). Power/energy estimation and optimization for software-oriented embedded systems. *PhD Disseration*.