HIGH-PERFORMANCE ENERGY-EFFICIENT MICROPROCESSOR DESIGN

SERIES ON INTEGRATED CIRCUITS AND SYSTEMS

Anantha Chandrakasan, Editor
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Published books in the series:

A Practical Guide for SystemVerilog Assertions
Srikanth Vijayaraghavan and Meyyappan Ramanathan
2005, ISBN 0-387-26049-8

Statistical Analysis and Optimization for VLSI: Timing and Power
Ashish Srivastava, Dennis Sylvester and David Blaauw
2005, ISBN 0-387-25738-1

Leakage in Nanometer CMOS Technologies
Siva G. Narendra and Anantha Chandrakasan
2005, ISBN 0-387-25737-3

Thermal and Power Management of Integrated Circuits
Arman Vassighi and Manoj Sachdev
2005, ISBN 0-398-25762-4

High-Performance Energy-Efficient Microprocessor Design

Edited by

VOJIN G. OKLOBDZIJA
Integration Corp., Berkeley, California and University of California

and

RAM K. KRISHNAMURTHY
Microprocessor Research Laboratory, Intel Corp., Hillsboro, Oregon

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN-10 0-397-28594-6 (HB)
ISBN-10 0-387-34047-5 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

Printed in the Netherlands.

TABLE OF CONTENTS

INTRODUCTION

Microprocessor design is a discipline and an art. Since the introduction of the first microprocessor in 1971 containing 2108 transistors embodied in Intel's 4004, the complexity of the design has increased several orders of magnitude with contemporary multi-core processors containing over two billion transistors. Yet microprocessor design is still driven by the inspiration, passion and vision of individuals involved.

This book deals with energy efficiency in microprocessors. As the complexity increased and the number of transistors integrated on the chip skyrocketed, power became the single most important issue limiting the otherwise unlimited progress. Not only does power determine the maximal speed at which we can allow a microprocessor to run, but it also determines the form factor, packaging and price. This is particularly important as computers migrated into consumer electronics characterized by mobility, portability and battery operation. Microprocessor design is a centerpiece of contemporary electronics, computer engineering and almost every complex electronic endeavor. Today microprocessors are found in almost every product, from the personal computer to iPod, in personal digital assistants, cell phones, games consoles, digital electronic cameras and television sets. They became an indispensable part of our everyday life which is increasingly dependent on them and their sustained and reliable functioning. In almost all of these applications energy efficiency is a must.

The importance of energy-efficient processor design is also recognized in courses taught at universities, industrial courses or seminars. Several conferences have been dedicated to energy-efficient design and several workshops have augmented important conferences in attempts to highlight this particular aspect. This book describes and teaches important topics of energy-efficient processor design starting from circuits to architecture, test and design for testability. It is targeted toward engineers, practitioners and researchers, as well as graduate students who want to learn about energy-efficient microprocessor design. It is suitable for advanced undergraduate and graduate courses in electrical engineering where the subject of low-power design is taught. The book is divided into chapters that can be covered one per week, thus being suitable for universities adhering to the quarter system courses. In the next edition we

plan to introduce exercises and problems at the end of each chapter to make it more suitable for teaching. The chapters are written by the world's top experts in the field highlighting a particular aspect of their expertise.

The book starts with a chapter "Ultra-Low Power Processor Design", describing the ways of designing for energy efficiency in low- and ultra-low-power processor design. It contrasts the ways of achieving low power with the flexibility of design which is often of more importance. The techniques widely used for the power reduction of microcontrollers and DSP processors are reviewed in this chapter. They include basic CPI (clocks per instruction) reduction, gated-clock mechanisms, optimal pipeline length, hardware accelerators, reconfigurable units and techniques to reduce leakage power. This is augmented by several examples such as RISC 8-bit and 32-bit microcontrollers and DSP cores. They describe the necessary tradeoffs between flexibility and energy efficiency.

Energy-efficient design of digital circuits is discussed in the second chapter of this book. In particular this chapter describes how transistor sizing affects the energy and delay of digital circuits. It examines design methodology based on the logical effort, and shows its limitations when designing for energy efficiency. The chapter presents a new methodology for the design and analysis of digital circuits in the energy-delay space which allows for energy reduction without performance penalty.

Clocking, which is one of the most critical aspects of processor design, is described in the third chapter of this book. Clocking strategy determines processor performance and largely impacts its power consumption. Conventional clocking strategies and circuit techniques descriptions, augmented with an overview of the state-of-the art clocked storage elements used in modern microprocessors, are contained in this chapter. Emerging methods aimed at handling incoming challenges in microprocessor design are also described.

The fourth chapter is dedicated to static memory design and issues related to the design of memory peripherals. This chapter presents design techniques to reduce SRAM dynamic and static power. It explores the design of static memory structures starting with a description of the single-port six-transistor SRAM cell operation and subsequently addressing voltage-mode and current-mode differential reads, single-ended reads, and control logic operation. The chapter covers issues pertaining to SRAM reliability, testing, and yield. Subsequently, implementation of efficient multi-port register file storage cells and peripherals is described.

The fifth chapter addresses the problem of placing design blocks of widely varying sizes and shapes and interconnecting them. Stability of placement methods is now a key concern. To achieve timing closure it is essential that gate sizing, buffer insertion, and routing can be completed without large disruptions to the overall physical structure of a circuit. This chapter surveys modern

techniques for circuit placement, with an emphasis on how placement interacts with logic synthesis and routing.

The choice of the right algorithm and a corresponding adder topology is discussed in the sixth chapter. This depends on many factors closely related to the technology of implementation. With the transition to deep-submicron CMOS technologies further complexity has been introduced. Thus, it has become even more difficult to make the right selection of appropriate design topology when power consumption is included. In this chapter this complex relationship is addressed and the important factors that influence the right selection of algorithm, circuit topology, operating conditions and power consumption are explained.

Fast 32-bit and 64-bit ALU with single-cycle latency and throughput are described in Chapter 7. They are one of the most performance-limiting units within the integer and floating-point execution clusters. ALU also contribute to one of the highest power-density locations on the processor, resulting in thermal hotspots and sharp temperature gradients within the execution core. This strongly motivates energy-efficient ALU designs that satisfy the high-performance requirements, while reducing peak and average power dissipation. The chapter, describes a single-cycle 64-bit integer execution ALU fabricated in 90 nm dual-Vt CMOS technology.

Chapter 8 deals with algorithms and implementation details used in today's floating-point units. It shows the implementation of the different parts of the fused multiply-add dataflow including the counter tree, suppression of sign extension encoding, leading zero anticipation, and end around carry adder design which has a huge performance advantage over a separate multiplier and adder. With one compound operation, effectively two dependent operations per cycle can be achieved. This type of design has a huge performance advantage over a separate multiplier and adder.

Chapter 9 addresses microarchitectural techniques for avoiding defects. Error detection and correction microarchitecture can significantly reduce the probability of failure and enhance the yield and the reliable operation of a microprocessor. The concept and methods of error detection and correction are discussed, followed by a description of microarchitecture and logic design error detection and recovery techniques. The failure mechanisms of nanometer-class semiconductor VLSI circuits and commercial microprocessors using error detection and recovery techniques are presented.

Chapter 10 deals with the issue of microprocessor bandwidth. It discusses its effects on the performance of an individual processor as well its impact on the overall system. The current trends in system evolution are examined, and the implication of those trends on bandwidth is discussed. Finally, the chapter explores the technologies that are likely to emerge and satisfy those trends.

Chapter 11 explores common methods and circuit architectures used to transmit and receive data through off-chip links. It discusses the most prevalent of these techniques, focusing on the chip-to-chip communication topologies common for microprocessors, namely access to memory, processor-to-processor communication for parallel computing, and processor-to-chipset communication.

The final chapter summarizes the approaches presented in this book through an example of a system-on-chip (SOC) design where low power is imperative. The chapter is based on the processor core implementing SuperH^{TM} architecture in a 130-nm CMOS process. The processor is suited for a wide range of usage in consumer, low-power digital appliances such as cellular phones, digital still/video cameras, and car navigation systems.

Finally we would like to thank the people who helped with this project, Mark de Jongh of Springer in particular, for his diligence in executing this project and patience and understanding when things did not go as expected. The students Milena Vratonjic, Mandeep Singh and Christophe Giacomotto provided invaluable help in reviewing the chapters.

Berkeley, California
Hillsboro, Oregon
March 30, 2006

Chapter 1

ULTRA-LOW-POWER PROCESSOR DESIGN

Christian Piguet
CSEM & EPFL

Abstract: Processor energy efficiency is a major issue in a majority of products; however, it is difficult to achieve it, as it is in contradiction with the main characteristic of processors, i.e. flexibility provided by embedded software. A given function implemented in random logic could consume 100–1000 times less energy than the same function implemented in a processor and corresponding embedded software. However, flexibility is more and more required, and ultra-low-power processors are mandatory. Only a few techniques have been widely used for the power consumption reduction of microcontrollers and DSP processors. This chapter will review these techniques, which are basically CPI (clocks per instruction) reduction, gated-clock mechanisms, optimal pipeline length, hardware accelerators, reconfigurable units and techniques to reduce leakage power. Several examples will be described in more details, such as some RISC 8-bit and 32-bit microcontrollers as well as some DSP cores. The latter are a good example of the necessary tradeoffs between flexibility and energy efficiency, as many random logic-based accelerators are very often used.

Key words: Ultra-low power; CPI; gated-clock; pipeline; accelerators; reconfigurable; leakage; microcontrollers; DSP cores.

1. Introduction

For any application, several embedded processors such as microcontrollers and DSP cores, as well as a large number of embedded memories and specialized peripheral circuits, are today integrated on the same die called systems on chip (SoC). The two main constraints are computation power and low-power design. In any portable application for the consumer market, low power is generally the most dramatic issue. Furthermore, the use of deep

Vojin G. Oklobdzija and Ram K. Krishnamurthy (eds.),
High-Performance Energy-Efficient Microprocessor Design, 1–30.

submicron technologies implies solving new problems, such as very low supply voltages, leakage, wire delays, networks on chip, signal input slopes, noise and crosstalk effects.

Memories generally consume most of the power. However, the principles to reduce power consumption of memories are basically well known: the memory has to be cut in small pieces and only one piece is addressed to fetch or to store data (cache, hierarchical, divided wordline and divided bitline). However, this chapter is focused on low-power processors even if memories are larger consumers. The principles to reduce power consumption of processors are more difficult: they have to deliver more and more computation power, they have to be totally flexible, and of very low power. These requirements are fundamentally contradictory; so basically, designers have to trade off these requirements. They have to choose the right processors with more or less flexibility to achieve the required goals, which are also application-dependent.

2. Energy Efficiency Versus Flexibility

Figure 1 shows that the flexibility [1], i.e. to use a general-purpose processor or a specialized DSP processor, has a large impact on the energy required to perform a given task compared to the execution of the same given task on dedicated hardware.

2.1. The Basic Choices

At the system level there are many important choices, as shown in Figure 1. Specialized architectures in random logic are obviously better in performances

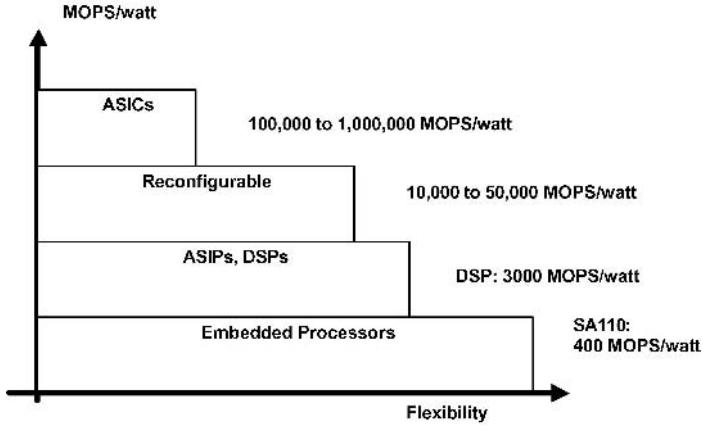

Figure 1. MOPS/watt versus flexibility.

(speed, power consumption) but are fixed without any flexibility. That is why today microprocessor-based architectures are generally selected; however, with reduced performances but much better flexibility.

It should be noted that a μP-based implementation results in very high sequencing. This is due to the μP structure that is based on reuse of the same operators and registers (ALU, accumulators). For instance, only one clock is necessary to increment a hardware counter. For its software counterpart the number of clock cycles is much higher, while executing several instructions with many clocks each. This simple example shows that the number of clock cycles executed for the same task can be very different depending on the architecture. For an electronic wristwatch 2000 instructions were necessary to update the time versus one clock for a random logic circuit.

Assuming applications presenting control tasks as well as DSP tasks, one can choose different architectures:

- a single microprocessor for control and DSP tasks
- a microcontroller with co-processors
- a microcontroller (8 or 32-bit) with a DSP processor
- configurable processors.

For the microprocessor-based approach the processor type is an important issue. As shown in Figure 1, there is a trade-off between performance and flexibility. This choice is obviously dependent on the application and required performances. Reuse is mandatory, and the designer has to choose IP cores for the microcontroller, DSP or co-processors. Configurable processors are quite interesting, as specialized instructions and execution units can be configured to the specified application.

2.2. Processor Types

There are several points to fulfill in order to save power. The first point is to adapt the data width of the processor to the required data. This results in a relatively increased sequencing to manage, for instance, 16-bit data on an 8-bit microcontroller. As shown in Table 1, for a 16-bit multiply, 30 instructions are required (add-shift algorithm) on a 16-bit processor, while 127 instructions are required on a 8-bit machine (double precision). However, a better architecture would be to have a hardware $16 * 16$ bit parallel-parallel multiplier with only one instruction to execute a multiplication.

Another point is to use the right processor for the right task. For control tasks do not use DSP processors; they are largely inefficient. Conversely, do not use 8-bit microcontrollers for DSP tasks! For instance, to perform a JPEG compression on an 8-bit microcontroller requires about 10 million executed instructions for a $256 * 256$ image (CoolRISC, 10 MHz, 10 MIPS, 1 second per

Table 1. Number of executed instructions in 8-bit microcontrollers

	No. of instructions		No. of instructions	
	CoolRISC 88: in the code	CoolRISC 88: executed	PIC 16C5x: in the code	PIC 16C5x: executed
8-bit multiply linear	30	30	35	37
8-bit multiply looped	14	56	16	71
16-bit multiply linear	127	127	240	233
16-bit multiply looped	31	170	33	333

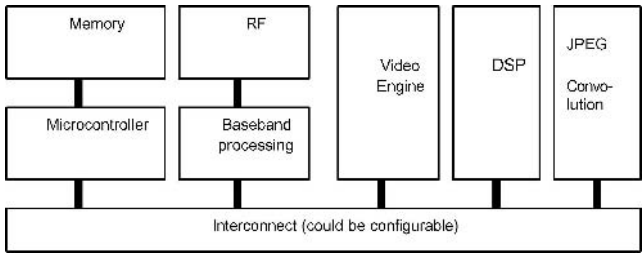

Figure 2. Heterogeneous architectures with many different processors.

image). It is very inefficient! Factor 100 in energy reduction can be achieved with JPEG dedicated hardware.

Figure 2 shows an interesting architecture to save power. For any applications there is some control that is performed by a microcontroller (the best machine to perform control). But in most applications there is also a main task to execute, that could be a DSP task, convolution, JPEG or another task. The best architecture is to design a specific machine (co-processor) to execute this task. So this task is executed by the smallest and most energy-efficient machine. Most of the time both microcontroller and co-processors are not running in parallel.

3. Power Reduction Techniques

This section will review power reduction techniques, which are basically CPI (clocks per instruction) reduction, gated-clock mechanisms, optimal pipeline length, hardware accelerators, reconfigurable units and techniques to reduce leakage power. At the architecture level most of the improvements in single processor performance are likely to come from lowering CPI.

3.1. Overview of Power Reduction Techniques

Future SoC will contain several different processor cores on a single chip. This results in parallel architectures, which are known to be less dynamically power-hungry than fully sequential architectures based on a single processor [2]. The design of such architectures has to start with very high-level models in languages such as System C, SDL or MATLAB. The very difficult task is then to translate such very high-level models in application software in C and in RTL languages (VHDL, Verilog) to be able to implement the system on several processors. One could think that many tasks running on many processors require a multitask but centralized operating system (OS), but regarding low power, it would be better to have tiny OS (2K or 4K instructions) for each processor [3], assuming that each processor executes several tasks. Obviously, the latter solution is easier to implement, even if performances could be reduced due to the inactivity of a processor that has nothing to do at a given time frame.

One has to note that most of the dynamic power can be saved at the highest levels. At the system level partition, activity, number of steps, simplicity, data representation and locality (cache or distributed memory instead of a centralized memory) have to be chosen (Figure 3). These choices at high level are, however, strongly application-dependent.

At the architecture level many low-power techniques aiming at dynamic power reduction have been proposed (Figure 3). The list could be: gated clocks,

	Dynamic Power			Static Power
High-level	Reduction of the number of executed tasks, steps and instructions. Processor types. Processor versus random logic. Reconfigurability			Remove units that do nothing or nearly nothing
Architecture	Asynchronous Encoding	Parallel Pipeline	Simplicity	Architectures with less inactive gates
Circuit Layout	Gated Clock	Sub 1V. DVS Low V_T	Low-power Library and Basic cells	Gated Vdd, MTCMOS, VTCMOS, DTMOS stacked transistors
	Activity reduction	Vdd reduction	Capacitance reduction	

Figure 3. Power reduction techniques.

pipelining, parallelization, very low Vdd, several Vdd, variable Vdd (DVS or dynamic voltage scaling) and V_T, activity estimation and optimization, low-power libraries, reduced swing, asynchronous and adiabatic. Some are used in industry, but some are not, such as adiabatic and asynchronous techniques. At the lowest levels, for instance a low-power library, only a moderate factor (about 2) in dynamic power reduction can be reached. At the logic and layout level the choice of a mapping method to provide a netlist and the choice of a low-power library are crucial. At the physical level layout optimization and technology have to be chosen.

However, leakage or static power becomes more and more important in very deep submicron technologies, and low-power design methodologies have to take into account the total power, i.e. dynamic and static power [4].

3.2. Leakage Reduction at Architecture Level

Many techniques have been proposed to reduce leakage power, such as gated Vdd, multi-V_T technologies, DTMOS, VTCMOS and static or dynamic SATS [4]. These techniques are effective at the cost of more or less complex circuits or technologies. Another technique already proposed is weak inversion logic for which transistors work in a weak inversion regime [5]. Despite the use of these techniques, reducing static power in very deep submicron technologies has to be addressed at all design levels, including system and architecture levels. This section addresses this problem at architecture level.

3.2.1. Activity, logic depth and number of gates

By analyzing the ratio of dynamic over static power it is observable that microprocessors presenting a low or very low activity will present a too large static power compared to dynamic power. Performing a logic function with a very small activity can be considered as far from the optimum, as the microprocessor is idle most of the time. In other words, if a transistor or a logic gate is not switching for a very long period it is not very efficient, as the ratio between the switching time (related to dynamic power) and the idle time (for which the transistor or the logic gate is leaky) is very small. This is obviously the case when the microprocessor is in sleep mode, as no dynamic power is present. Hence the only consumed power is the static one. However, in that case nothing can be done at the architecture level and only circuit techniques [4] can be used to reduce leakage.

The presented design methodology aims at searching an optimum in which one has better use of switching transistors or gates in the reference period of time, i.e. an increased activity in such a way that dynamic power is not too small

compared to static power. Obviously, this relative increase of activity has to be understood as useful, and not, for instance, by suppressing gated clocks or increasing glitches.

The conventional definition of activity is the factor a in the dynamic power formula: $P = a * f * C * Vdd^2$. This means that a is the ratio between the number of switching gates in a clock period over the total number of gates. Combinational circuits generally present activities around 1–5%. One has also to consider idle gates when they are connected in series. If there are 20 gates in series for a given pipeline stage or logic block (logic depth LD = 20), these 20 gates have to switch in series in a clock period. So only 1/20 of the clock period is used for switching; the rest of the time the considered gate is only a leaky gate. In running modes, with more and more leaky transistors, it seems that it could be better to avoid designing very inactive gates and therefore to search for an optimum ratio of dynamic power over static power. Leakage energy could be considered roughly proportional to the number of gates and to the duration of the clock period. It is not the case of the dynamic energy that is only proportional to the number of switching gates. To have a better balance of dynamic versus static energy, it could be mandatory to decrease significantly the total number of gates, generally resulting in increase of global activity.

3.2.2. Optimal total power

This optimum of total power (dynamic + static) is roughly obtained with similar amount of static and dynamic power [6, 7]. To reduce the total power consumption an attractive goal could be to have fewer, but more active, transistors or gates to perform the same logic function. If a given logic function requires 10,000 gates for its implementation and presents an activity factor of 1%, this means that on average 100 gates are switching in a clock period. If the same logic function could be implemented with only 1000 gates, keeping the same number of switching gates (100), the activity will be 10%, with the same dynamic power but with a leakage reduced by a factor of 10 due to the reduction of the total number of gates [7].

Figure 4 shows at the left such a theoretical situation with the same dynamic power and a significant reduced leakage power for the architecture containing fewer gates. This is however a naive situation, as nothing is said about the speed of the two architectures. The architecture with the smaller number of gates could be slower and would therefore require a larger Vdd to achieve the same speed, impacting the dynamic power.

Figure 4 (at the right) shows a practical situation comparing a 16×16 multiplier automatically synthesized (Synopsys) with two different architectures. Architecture A is realized with four RCA multipliers working in parallel (3250 gates) and the second architecture B is implemented with two Wallace

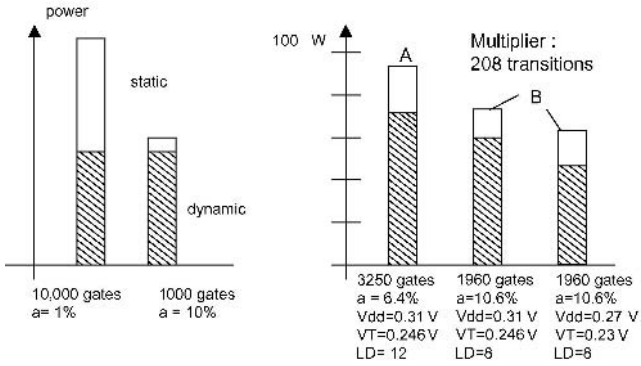

Figure 4. Various architectures with the same number of transitions for a given function (white rectangle: static; filled rectangle: dynamic).

multipliers working in parallel (1960 gates). Both present 208 transitions for executing a 16×16 multiply ($3250 \text{ gates} * 6.4\% = 1960 \text{ gates} * 10.6\% = 208$ transitions), showing that various architectures with a quite different number of gates can achieve their function with the same number of transitions. Architecture B (second bar, at the same Vdd and V_T) with fewer gates than A presents a larger activity, a smaller leakage but also a smaller dynamic power (smaller transistors, due to improved delay slack). However, at the optimum of total power consumption this architecture B can be supplied with smaller Vdd and V_T (third bar) to meet the same speed performances as A, with better results in total power consumption. This shows, however, that the speed performances are a very important parameter in such comparisons.

The goal of designing architectures with more global activity is not to reduce only leakage power with same dynamic power (Figure 4 at left), but to find an optimum of the total power, as shown at the right of Figure 4. This optimum has to be searched for the same speed constraints, and looking at first and last bars at the right of Figure 4, architecture B with less gates and more activity is better in terms of optimal total power; however, with a similar ratio of dynamic over static power of respectively 3.2 for A and 2.5 for B. This example shows very clearly that the reduction of the total number of gates and the increase of the activity could result in the same or even smaller dynamic power and optimal total power. It is the paradigm shift, not increasing the number of switching gates, but reducing the total number of gates. In this respect the resulting activity, although increased, can be considered as a useful activity. However, as already pointed out, the reduction in the total number of gates and the increase of activity cannot be a goal *per se*, as some architectures for a given logic function with significantly less gates could be extremely slow. So one has to dramatically increase the supply voltage and decrease the V_T

for satisfying the speed constraints, resulting in a much larger total power. It could be the case, for instance, for a sequential multiplier using an add-shift mechanism compared to a parallel multiplier.

3.2.3. Design methodologies based on optimal total power

In refs 7 and 10, it is shown that at optimal total power, the ratio Ion/Ioff of a single device in a given technology is proportional to K1*(LD/a), i.e. proportional to architectural parameters such as logical depth (LD) and activity (a). K1 is not fixed as shown in refs 8 and 9, but goes from 3 to 6 depending on the architecture. Dynamic over static power is also equal to K1, i.e. dynamic power is 3 to 6 times larger than static power at the optimum of the total power. It turns out that Ion/Ioff is quite low in very deep submicron technologies, showing that leakage tolerant architectures do have to present small LD and large activities. So pipelined architectures, reducing LD proportionally to the number of stages with the same activity, have to be preferred to parallel architectures that reduce LD and activity. In other words, parallel architectures do reduce LD, but the transistor count is also largely increased, having too much impact on the number of inactive gates and therefore the leakage.

Obviously, what is searched for is design methodologies [10, 11] starting from a given logic function architecture and generating a new architecture in which the number of cells is reduced, the number of transitions for this given logic function is constant, the activity is increased, but the logic depth is also constant (or even smaller). If the logic depth is increased, as is generally the case by increasing sequencing to reduce the gate count, the speed constraints have to be satisfied by increasing Vdd and going down with V_T, resulting in a larger optimal total power.

4. Low-power Microcontrollers

4.1. Eight-bit Embedded Microcontrollers

The most well-known 8-bit microcontrollers are Intel 8051-based (many versions), Motorola 68K, Microchip PIC, Zilog eZ80Acclaim! and many Japanese microcontrollers. Basically, all these 8-bit microcontroller architectures are based on old CISC (complex instruction set computer) architectures, with instruction formats containing several bytes. This means that several memory accesses are required to execute a single instruction, and it is impossible to execute an instruction in a single clock cycle. So this results in a CPI (clock per instruction) of 4–20. To provide good MIPS performances (million of instructions executed per second), according to the following

formula: MIPS $= f/\text{CPI}$, they have to be clocked at relatively high f frequencies. As the dynamic power consumption is proportional to f, they present a figure MIPS/watt that is not reduced as it could be.

However, for power consumption or for energy consumed per instruction, RISC microcontrollers are much better. As the energy per clock cycle is roughly the same in RISC and CISC microcontrollers (about the same number of transistors), RISC-like microcontrollers with a single clock cycle per instruction provide a significant advantage regarding energy per instruction. Only two RISC 8-bit microcontrollers are commercially available:

- Xemics/Semtech-CSEM CoolRISC (www.coolrisc.com)
- Atmel AVR (www.atmel.com)

4.2. CoolRISC 8-bit Microcontroller

The CoolRISC is a three-stage pipelined core [12, 13]. The branch instruction is executed in only one clock. In that way no load or branch delay can occur in the CoolRISC core, resulting in a strictly CPI $= 1$ (Figure 5). For each instruction the first half clock is used to precharge the program memory. The instruction is read and decoded in the second half of the first clock. As shown in Figure 5, a branch instruction is also executed during the second half of this first clock, which is long enough to perform all the necessary transfers. For a load/store instruction, only the first half of the second clock is used to store data in the RAM memory. For an arithmetic instruction the first half of the second clock is used to read an operand in the RAM memory or in the register set, the second half of this second clock to perform the arithmetic operation and the first half of the third clock to store the result in the register set.

Furthermore, to reduce the energy per clock cycle, the gated clock technique has been extensively used in the design of the CoolRISC core (Figure 6). The ALU, for instance, has been designed with input and control registers that are loaded only when an ALU operation has to be executed. During the execution of another instruction (branch, load/store), these registers are not clocked, thus

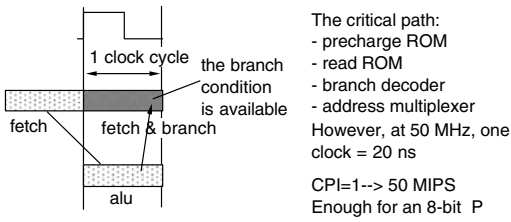

Figure 5. No branch delay.

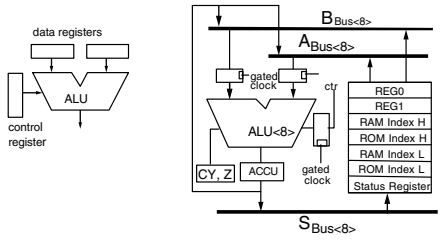

Figure 6. Gated-clock ALU.

no transition occurs in the ALU (Figure 6). A similar mechanism is used for the instruction register; thus, in a branch, which is executed only in the first pipeline stage, no transition occurs in the second and third stages of the pipeline. It is interesting to see that gated clocks can be advantageously combined with the pipeline architecture; the input and control registers implemented to obtain a gated clocked ALU are naturally used as pipelined registers. Automatic gated clocks insertion is proposed today in many papers, such as ref. 14 and CAD tools [15].

A main issue in the design of processor cores [16] is the design of the clock tree with the smallest possible clock skew and avoiding any timing violations. In deep submicron technologies this design becomes more and more difficult as wire delays are larger and larger compared to gates delays. Generally, clock trees do have very large buffers, resulting in a significant increase in power consumption. Today, most processor cores are based on a single-phase clock and are based on D-flip-flops. As clock input slopes are a key factor in D-flip-flop reliability, as clock input capacitances of D-flip-flops in standard cell libraries are significant, as clock skew is an issue, the design of clock trees becomes more and more difficult in deep submicron technologies.

Another approach than the conventional single-phase clock with D-flip-flops (DFF) has been shown to be more effective. This is based on using only latches with two non-overlapping clocks. This clocking scheme has been used for the 8-bit CoolRISC microcontroller [17] as well as for other DSP cores such as ref. 18. Figure 7 shows this latch-based clocking scheme that has been chosen to be more robust to clock skew, flip-flop failures and timing problems at very low voltage [17]. The clock skew of Ø1 (respectively Ø2) has to be shorter than half a period of CK, i.e. Ø1 can be delayed for half a period of CK before Ø1 and Ø2 become active at the same time. However, this clocking scheme requires two clock cycles of the master clock CK to execute a single instruction. This means 100 MHz is necessary to generate 50 MIPS (CoolRISC with CPI = 1), but the two Øi clocks and the two clock trees are at 50 MHz. Only a very small logic block is clocked at 100 MHz to generate two 50 MHz clocks.

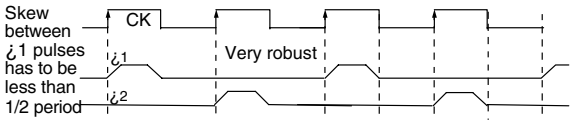

Figure 7. Latch-based clocking schemes.

The design methodology using latches and two non-overlapping clocks has many advantages over the use of D-flip-flop methodology. Due to the non-overlapping of the clocks, and the additional time barrier caused by having two latches in a loop instead of one D-flip-flop, latch-based designs support greater clock skew before failing than a similar D-flip-flop design (each targeting the same MIPS). This allows the synthesizer and router to use smaller clock buffers and to simplify the two clock trees generation, which will reduce the power consumption of the two clock trees. Despite the fact that this clocking scheme requires two different clock trees, each of them sees a much smaller clock capacitance, and due to clock skew relaxed constraints it is likely that the total power consumption is smaller that the power of a single clock tree using D-flip-flops.

Furthermore, if the chip has clock skew problems at the targeted frequency after integration, it is possible with a latch-based design to reduce clock frequency. The result is that the clock skew problem will disappear, allowing the designer to test the chip functionality and eventually to detect other bugs or to validate the design functionality. Another advantage with a latch design is time borrowing [17]. The latch design has additional time barriers, which stop the transitions and avoid unneeded propagation of signal and thus reduce power consumption (Figure 8). Using latches can also reduce the number of MOS in a design. For example, a microcontroller has $16 * 32$-bits registers, i.e. 512 DFF or 13,312 MOS (using DFF with 26 MOS). With latches the master part of the registers can be common for all the registers, which gives 544 latches or 6528 MOS (using latches with 12 MOS). In this example the register area is reduced by a factor of 2.

Using latches for pipeline registers significantly reduces power consumption when using such a scheme in conjunction with clock gating. The clock gating of each stage (latch register) of the pipeline with individual enable signals, reduces the number of transitions in the design compared to the equivalent DFF design, where each DFF is equal to two latches clocked and gated together.

The latch-based design allows a very natural and safe clock gating methodology. Figure 8 shows a simple and safe way of generating enable signals for clock gating. This method gives glitch-free clock signals without the adding of memory elements, as is needed with DFF clock gating. Synopsys handles very nicely the proposed latch-based design methodology. It performs nicely

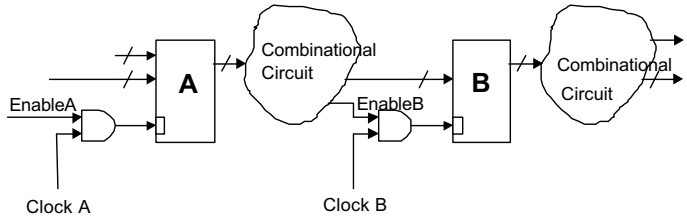

Figure 8. Latch-based clock gating.

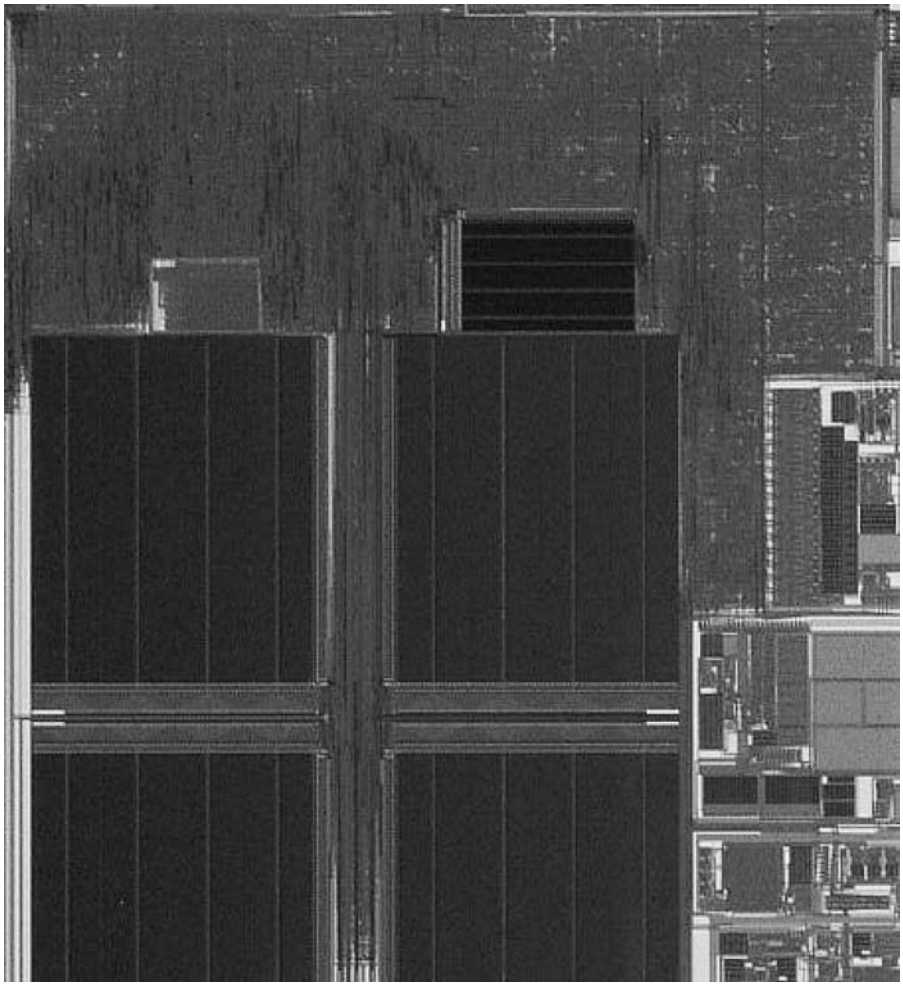

Figure 9. CoolRISC core and SRAM memories in 0.18 µm technology.

the time borrowing and correctly analyzes the clocks for speed optimization. So it is possible to use this design methodology with Synopsys, although there are a few points of discussion linked with the clock gating.

This clock gating methodology cannot be inserted automatically by Synopsys. The designer has to write the description of the clock gating in his VHDL code. This statement can be generalized to all designs using the above latch-based design methodology. Synopsys can do automatic clock gating for a pure double-latch design (in which there is no combinatorial logic between the master and slave latch), but such a design causes a loss of speed over a similar DFF design. The most critical problem is to prevent the synthesizer from optimizing the clock gating AND gate with the rest of the combinatorial logic. To ensure a glitch-free clock this AND gate has to be placed as shown in Figure 8. This can be easily done manually by the designer by placing these AND gates in a separate level of hierarchy of his design or placing a "don't touch" attribute on them.

It is interesting to note that the CoolRISC core in a quite old TSMC 0.25 μm consumes about 10 μW/MIPS, resulting in 100,000 MIPS/W at 1.05 V. This is roughly better for a factor of 2 over a similar design using D-flip-flops [17]. Figure 9 shows a CoolRISC core with SRAM memories embedded in a SoC chip integrated in 0.18 μm TSMC technology.

4.3. 32-bit Microcontrollers

Embedded 32-bit RISC microcontrollers are used in portable applications for which low power consumption is an issue [19]. These 32-bit cores are quite small, from 30,000 MOS to about 100,000 MOS. They are very power-efficient. The main characteristics are as follows:

- load/store architectures (RISC);
- instructions of 32 bits (ARM, MIPS, StrongArm) or 16 bits (SH, Thumb, TinyRISC);
- 32 registers (ARM, MIPS) or 16 registers (SH);
- small number of generic instructions;
- single pipeline, with a peak CPI = 1;
- the number of pipeline stages was generally three for the first 32-bit cores (ARM7, Thumb, MiniRISC, TinyRISC) some years ago;
- for most recent 32-bit cores the number of pipeline stages has been significantly increased to achieve larger frequencies, from five (MIPS R3000, Hitachi SH, StrongArm, X-Scale, ARM9) to six pipeline stages (ARM10) or even eight pipeline stages (MIPS32 24K).

The basic principle of a pipelined μP is to execute N-cycles instructions in an overlapped fashion in a N-stages pipeline. At each cycle an instruction

is provided to the pipeline, and an instruction is completed. This results in the execution of one instruction per cycle. However, there are some drawbacks, such as pipeline stalls. Branch hazards occur when the target instruction after a branch is determined in the second, third or xth stage of the pipeline. This can be solved by the insertion of load delay or branch delay (reducing the throughput) or by using a bypass technique: the preceding instruction is able to transfer its result directly to the next instruction if needed.

The resolution of pipeline hazards can be achieved while using several different approaches:

- Static approach, for which it is the compiler that is responsible for re-organizing the code and/or inserting NOP instructions. Generally the code size is increased [20, 22].
- Dynamic approach, for which the processor hardware is in charge to solve the pipeline bubbles at run-time. Generally load and branch delays are inserted, but code could also be reorganized dynamically, resulting in out-of-order execution [20, 22].
- Pipelined multithreaded architecture.
- Short pipeline with branch instructions executed in one clock (CoolRISC [12]).

Prediction techniques can be used to guess the target instruction and to start to fetch (sometimes to execute) it before the branch is resolved. Obviously, in case of misprediction, the processor has to go back. The cost of a branch and of mispredicted branches becomes so high in terms of clock cycles that prediction techniques become very sophisticated. A given branch instruction is generally not 50% taken and 50% not taken. More than 90% of branch instructions are taken (or not taken) at 90% to 99%. Prediction techniques are based on this fact, i.e. based on the prediction that the branch will do what it did last time [21]. There are several techniques to implement a prediction mechanism. For instance, the first ARM having prediction is the the ARM10 (static prediction: backward taken, forward not taken).

The most well-known 32-bit microcontrollers are as follows: ARM, Thumb, Atmel AT91, StrongArm, X-Scale, MIPS, ColdFire, Hitachi SH SuperH, Embedded X86, Transmetta Crusoe, XAP3, NECVR4 and many others. Just to illustrate the energy efficiency of some 32-bit RISC cores, let us see some well-known cores. ARM7 was a three-stage pipeline core. ARM8 and ARM9 have been designed while extending the pipeline to five stages with no prediction (all taken) and branches executed in two clock delays. The ARM10 has six stages with a static prediction. With this prediction logic the CPI will be around 1.5. ARM11 has eight pipeline stages. In some deep submicron technologies one has the following numbers:

- In 0.18 µm and 1.8 V, ARM9 core achieves 330 MHz, 365 MIPS, 120 mW, 3000 MIPS/watt.

- In 0.13 µm ARM1026 with caches achieves 475 MHz, 500 MIPS, 250 mW, 2000 MIPS/watt.
- The ARM11 processor family delivers up to 550 MHz of performance with power as low as 0.24 mW/MHz using a 0.13 µm foundry process, so 4000 MIPS/watt.

XScale is Intel's implementation of the fifth generation of the ARM architecture. It is the successor to the Intel StrongARM microcontrollers. The number of pipeline stages was extended to seven or eight stages. The most interesting feature of the XScale architecture is what Intel calls "Dynamic Voltage Scaling". In 2001, XScale was dissipating 450 mW at 600 MHz and 1.3 V and much less at reduced supply voltage [23]. In 0.18 µm and 1.3 V, XScale with caches achieves 600 MHz, 660 MIPS, 750 mW, 880 MIPS/watt.

MIPS microcontrollers have been designed following the original MIPS and MIPS-X of Stanford [24]. These first RISC machines were based on the "delayed branch" mechanism that was handled by the compiler (called Reorganizer). In the MIPS design the instruction following a branch is always executed. The compiler is responsible for filling this delay slot with a useful instruction or, if it is not possible, by a NOP. The original pipeline provided five stages. The number of transistors was about 115,000. Performances today are as follows:

- MIPS32 4 Kp: 1 mW/MHz or 1 mW/MIPS (1000 MIPS/watt).
- MIPS32-R2 4KEc Pro: 255-300 MHz, five pipe stages, 0.12–0.37 mW/MHz (8500 MIPS/watt for 255 MHz and 0.12 mW/MHz) [25].
- MIPS32-R2 24K Pro: 400–550 MHz, eight pipe stages, 0.5 mW/MHz [25] (2000 MIPS/watt).

As the pipeline becomes deeper and deeper due to increased frequencies, branch prediction is used for MIPS, such as the MIPS32 24K with eight-stage, dynamic prediction (four clocks penalty in case of mispredicted branch).

Adding DSP instructions to a RISC core was one of the first ideas to provide flexible cores for both control and DSP tasks (see Section 5). Many companies are offering such combinations, such as ARM, ARC and Tensilica. However, while provided DSP extensions are meaningful, the resulting performances are impacted. Generally, the resulting core is larger, maximum frequency is reduced of 30% and power consumption is increased. Control as well as DSP tasks are not energy efficiently executed on these cores. If some companies like MIPS claim no performance degradation (MIPS32 24KE), it means that the DSP extensions are very limited or that MAC-like operations are executed in two clock cycles [26].

There is a clear trend for multicore and multithreaded architecture for high-performance microprocessors [27]. For embedded microcontrollers into SoC, as applications are more specific, generally the multicore approach is

implemented with different processors including DSP cores and co-processors. A variety of chip vendors have recently announced multicore SoC designs, including PMC-Sierra (MIPS), Broadcom (MIPS), RMI Raza Microelectronics Inc. (MIPS), Freescale (PPC), Cavium (MIPS), and ARM Ltd. (ARM). Multicore advantages include a better ratio of performance to power usage, less heat dissipation, and a smaller physical footprint. One prime market for multicore SoC appears to be networking equipment such as firewalls that deeply inspect packets or perform compute-intensive spam filtering.

Most microprocessors are single-threaded computers, i.e. they are executing a single flow of instructions stored in a program. This means that the dependency between sequential instructions is quite high. For multi-threaded computers or multiple-context processors, several independent tasks are executed at the same time by the processor. Instructions of different tasks are completely independent. So pipeline bubbles as well as limited parallel execution could be avoided, providing excellent throughput. A multi-threaded scheme is generally used for very-high performance microprocessors, but it has also been used for small, 8-bit microcontrollers like the low-power CSEM Punch [28]. The main problem in multi-threaded computers is the number of active tasks. The processor is fully utilized only if all the tasks are active. However, if only one task is active the throughput is drastically reduced. SUN is also designing multi-threaded microprocessor architectures called Niagara [29]. So all branch and load delays are removed, all the very sophisticated prediction mechanisms about branch instructions are removed, this "new" microprocessor seems so simple! But the idea is quite old.

5. Low-power DSP Architectures

DSP microprocessors are being used more and more in high-volume applications, such as portable phones, hard-disk drives, audio and image processing, hearing aids, wired and wireless communication devices. DSP processors are different from RISC processors: they are dedicated for executing arithmetic operations and have to be very energy-efficient in executing DSP algorithms [30].

5.1. Main Features of DSP Architectures

DSP architectures are specialized in the execution of digital signal processing (DSP) algorithms. The basic DSP operation is the multiply-accumulate (MAC), i.e. a sum of multiplications used, for instance, in digital filters, correlation and Fast Fourier transforms. The goal is to execute the MAC in one clock (CPI $= 1$) while using a pipeline. The accumulator provides extra bits

to accommodate growth of the accumulated result (for 16-bit data the accumulators are 40 bits, i.e. 32-bit multiplication result + 8 extra bits for the accumulation).

Another main feature of a DSP processor is to complete several memory accesses in a single clock. Simultaneously, instruction fetch from the program memory, as well as two operands fetch from two data memories and one result store, have to be performed in a single clock. DSP architectures are either load/store (RISC) architectures (input registers to the datapath) or memory-based architectures in which the operands are directly fetched from the data memories to be processed. However, load/store architectures are capable of executing in parallel an arithmetic operation and register-memory transfers (to fetch operands for the next arithmetic operation).

The third basic DSP feature is specialized addressing modes. Both data RAM are addressed through two banks of pointers with pre- or post-incrementation/decrementation as well as circular addressing (modulo). These addressing modes provide efficient management of arrays of data, to which a repetitive algorithm is applied. These operations are performed in a specialized address generation unit.

The fourth basic feature is the capability to perform efficiently loops with zero overhead. Loop or repeat instructions are able to repeat 1 to N instructions without loop counter (no need to initialize and to up-date a loop counter). These instructions are fetched from the memory and stored in a small cache memory in which they are read during the execution of the loop.

All these features have been introduced in DSP cores to increase performances, mainly to reduce the number of clock cycles to execute some tasks. Compared to microcontrollers, what is executed in one clock cycle in a DSP core could take 10–20 clock cycles in microcontrollers.

5.2. Single MAC DSP Cores

Figure 10 shows a typical DSP datapath (in this case, 24-bit Motorola 56,000) producing a single MAC per clock cycle. It contains two data buses that are connected to four 24-bit input registers. It is therefore a load/store architecture in which operands have to be loaded into these few input registers before processing. These registers contain two operands that can be multiplied and accumulated in two 56-bit accumulators with 8 guard bits. The guard bits allow the DSP to perform a series of MAC operations without arithmetic overflow. The extra bits in the accumulators are capable of accommodating the bit growth resulting from the repeated additions. Memory-register transfers can be executed in parallel with an arithmetic operation, for instance:

$$A \leftarrow X0 * Y0, X0 \leftarrow RAM(R0), RAM(R1) \leftarrow Y1;$$

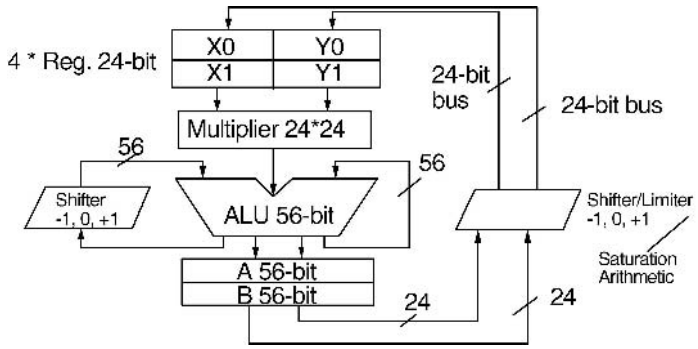

Figure 10. Typical single MAC DSP datapath.

Shifters are used to scale the operands or the results to avoid overflows. In Figure 10 there are two shifters with limited capabilities. One is used in the MAC and the other is used to scale the results that have to be stored in the data memory. Overflows are generally solved by using saturation arithmetic, i.e. an overflow value is replaced by the largest number that can be represented. It is called a limiter in Figure 10.

Many single MAC DSP cores are still offered on the market, such as Ceva-TeakLite [31] running at 200 MHz in 0.13 µm and consuming, for instance, 0.1 mA/MHz in an audio platform dedicated to MP3 stereo decoding.

5.3. RISC Cores with DSP Enhancements

As mentioned above, adding DSP instructions to a RISC core was one of the first ideas to provide flexible cores for both control and DSP tasks. Many companies have designed such cores. There are two options: adding these instructions definitively, such as ARM, MIPS and SH3-DSP, or giving the possibility to customers to add the desired instructions (customizable ARC, Hyperstone and Tensilica).

Cores having control and DSP instructions result in lower performances, as the cores are 30% bigger, maximum frequency reduced by 30% and power consumption increased. Neither the control nor the DSP tasks are really energy-efficiently executed on these cores. If some companies claim no performance degradation, it means that the DSP extensions are very limited [26]. For instance, in ARM DSP extensions there are no X/Y memories, no zero overhead looping or no special addressing modes [32].

ARC is offering synthesizable soft macros that can be configured to meet specific performances. In a four-stage pipeline the instruction set contains 32 separate instructions; the first 16 are predefined and the last 16 are available

for customers. They may be instructions from an ARC library or completely new instructions defined by customers. This is a much better approach than the previous one as the customer is free to add only the required DSP instructions for a given task. ARC 605 is a five-stage pipeline with static branch prediction. The 605 is the smallest one [33] in 0.13 μm, 0.36 mm^2, 250 MHz maximum frequency and 0.06 mW/MHz. This means 15 mW at 250 MHz, 250 MIPS, so 17,000 MIPS/W. Tensilica Xtensa proposed an instruction set containing 78 basic immutable commands. Customers are also free to add their special-purpose instructions by defining them using the provided hardware RTL language. Not only new instructions can be specified, but also I/O ports can be modified depending on the memories used, as well as the pipeline number of stages [34]. The RTL code includes gated clocks for powering down various blocks and can be synthesized with any library. The process of synthesis could be as short as 8 hours. Tensilica Xtensa LX has five to seven pipeline stages and occupies 0.2 mm^2 in 0.13 μm (19,000–25,000 gates). It consumes 0.038 mW/MHz, or 15 mW at 390 MHz [33]. This means 390 MIPS, so 26,000 MIPS/W.

5.4. More Computation Power Required

Various DSP architectures can be and have been proposed to reduce power consumption significantly while keeping the largest throughput. Furthermore, many portable applications require a significantly increased throughput, due to new, quite sophisticated DSP algorithms and to wireless communication. Beyond the single MAC DSP core of 5–10 years ago, it is well known that parallel architectures with several MAC working in parallel allow designers to reduce supply voltage and power consumption at the same throughput. That is why many VLIW or multitask DSP architectures have been proposed and used, even for hearing aids. The key parameter to benchmark these architectures is the number of simple operations executed per clock cycle, up to 50 or more. Another key point is the design of specialized execution units, such as Viterbi and Turbo code. These dedicated execution units in DSP are mandatory to speed up these algorithms while significantly reducing power consumption [35].

However, there are some drawbacks regarding very parallel architectures such as VLIW or superscalar DSP processors. The very large instruction words of VLIW of up to 256 bits significantly increase the energy per memory access. Some instructions in the set are still missing for new, better algorithms. Finally the growing core complexity and transistor count (roughly 2 million transistors for the cores in a hearing-aid circuit) become a problem because leakage is roughly proportional to transistor count.

To be significantly more energy-efficient there are basically two ways however, impacting either flexibility or the ease of programming. The first one is