Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2011, Article ID 480805, 14 pages
doi:10.1155/2011/480805

Research Article

A Precise High-Level Power Consumption Model for Embedded Systems Software

Mostafa E. A. Ibrahim,[1,2] Markus Rupp (EURASIP Member),[2] and Hossam A. H. Fahmy[3]

[1] *Electrical Engineering Department, High Institute of Technology, Benha University, 13512 Benha, Egypt*
[2] *Institute of Communications and RF Engineering, Vienna University of Technology, 1040 Vienna, Austria*
[3] *Electronics and Communication Department, Faculty of Engineering, Cairo University, 12613 Cairo, Egypt*

Correspondence should be addressed to Mostafa E. A. Ibrahim, mostafa.halas@gmail.com

Received 26 February 2010; Revised 17 June 2010; Accepted 11 August 2010

Academic Editor: Xiaorui Wang

The increasing demand for portable computing has elevated power consumption to be one of the most critical embedded systems design parameters. In this paper, we present a precise high-level power estimation methodology for the software loaded on a VLIW processor that is based on a functional level power model. The targeted processor of our approach is the TMS320C6416T DSP from Texas Instrument. We consider several important issues in our model such as the pipeline stall, inter-instructions effect and cache misses. The contributions are the following. First, a precise model to estimate the power consumption of the targeted DSP, while running a software algorithm is proposed. Second, we prove the validation and precision of our model on many typical algorithms applied in signal and image processing. Third, we further validate the precision of our model on a real application applied in the video processing field. The power consumption estimated by our model is compared to the physically measured power consumption, achieving a very low average absolute estimation error of 1.65% and a maximum absolute estimation error of only 3.3%.

1. Introduction

Many applications in special areas such as hand-held computation, tiny robots, and guidance systems in automated vehicles are powered by batteries of low rating. In order to avoid frequent recharging or replacement of the batteries, there is a significant interest in low-power system design. Very Long Instruction Word (VLIW) Digital Signal Processors (DSP) are the most worthy choice for such an application domain because of their optimal performance at low power. The importance of the power constraints during the design of embedded systems has continuously increased in the past years, due to technological trends toward high-level integration and increasing operating frequencies, combined with the growing demand of portable systems. This has led to a significant research effort in power estimation and low power design. Power simulators (profilers) allow the software programmers to specify the hot spot, highly power consuming, segments of their software code as a first step towards optimizing these code segments from power perspective. Developers of power simulators have to embed a precise power consumption model in their simulators. Existing processors power simulators are available only for the lower levels of the design, at the circuit level and to a limited extent at the logic level. These tools are very slow and impractical to use to evaluate the power consumption of embedded software since the application power consumption would only be known at the very last stage of the design process. In this paper, an approach for estimating the power consumption of a VLIW DSP while running a software application is presented. The contribution of this work aims to precisely estimate the power consumption of the core processor while running a software algorithm at an early stage in the design process. The targeted DSP is the TMS320C6416T (for the rest of the paper it is referred to as C6416T for brevity) from Texas Instrument. This processor features the highest performance among the fixed-point DSPs of the C6000 DSP platforms.

The rest of the paper is organized as follows. Section 2 presents an overview of the existing power consumption modeling abstraction levels for general purpose processors. Section 3 provides a general overview of the target architecture. Section 4 describes the methodology along with the experimental setup employed in our experiments. Section 5 describes in detail the functional level analysis and the resulting mathematical formulas constituting the model for the targeted architecture functional units. Section 6 demonstrates the validation of the power estimation methodology utilizing the developed power consumption model. Finally, Section 7 summarizes the main contributions of this paper.

2. Related Work

This section summarizes the most recent contributions to the problem of power modeling and estimation. Recent approaches to model the power consumption of the software running on a processor can be separated into two main categories *low-level models* and *high-level models*. Low-level models calculate power and energy from detailed electrical descriptions, comprising circuit level, gate level, register transfer (RT) level, or system level. while, high-level models deal only with instructions and functional units from the software point of view and without electrical knowledge of the underlying architecture [1].

2.1. Low-Level Estimation Techniques. 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, and architectural level.

2.1.1. Transistor-Level Estimations. The representation of a microprocessor in terms of transistors and nets is extremely complex and requires undergoing all the steps of the design flow and the layout, routing, and parameter extraction inclusive. Furthermore, a transistor level view of the system employs component 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 a few clock cycles, requires times that are usually not affordable and anyway not practical for the high-level power characterization [2]. Thus, while providing very good accuracy; transistor-level power estimation methodology is slow and impractical for analyzing the power consumption at an early design stage. Moreover, this methodology requires the availability of lower level circuit details of the targeted processor, which is not available for most of commercial off-the-shelf processors.

The PowerMil [3] is an early attempt to build a low-level power consumption simulator. PowerMil is a transistor level simulator for simulating the current and power behavior in VLSI circuits. It is capable of simulating detailed current behavior in modern deep submicron CMOS circuits, including sophisticated circuitries such as sense-amplifiers, with speed and capacity approaching conventional gate level simulators. For more details about power estimation techniques in VLSI circuits refer to [4, 5].

2.1.2. Gate-Level Estimations. Methods to estimate the power consumption based on gate level descriptions of microprocessors or micro controller cores have been proposed in literature. The main advantage of such methods with respect to transistor-level simulation approaches is that the simulation is event-driven and takes place in a discrete time domain, leading to a considerable reduction of the computational complexity, without a significant loss of accuracy [2].

An example for the gate level power estimators is the model presented by Chou [6]. present an accurate estimation of signal activity at the internal nodes of sequential logic circuits. The power consumption estimation in Chou and Roy is a Monte Carlo based approach that take spatial and temporal correlations of logic signals into consideration.

2.1.3. RT-Level Estimations. A design described at RT-level can be regarded as a collection of blocks and a network of interconnections. The blocks are sometimes referred to as macros, adders, registers, multiplexers, and so on, while the interconnections are simply nets or group of nets. An assumption underlying the great majority of the approaches presented in the literature is that the power properties of a block can be derived from an analysis of the block isolated from a design, under controlled operating conditions. The main factor influencing the power consumption model of a macro is the input statistics [2].

Most of the research in RT level power estimation is based on empirical methods that measure the power consumption of existing implementations and produce models from those measurements. This is in contrast to the approaches that rely on information-theoretic measures of activity to estimate power [7, 8]. Measurement-based approaches for estimating the power consumption of datapath functional units can be divided into two subcategories, namely; transition sensitive and activity sensitive. The first technique, introduced by Powel and Chau [9], is a fixed-activity micromodeling strategy called the Power Factor Approximation (PFA) method. The power models are parameterized in terms of complexity parameters and a PFA proportional constant. Similar schemes were also proposed by Kumar et al. [10] and Liu and Svensson [11]. This approach assumes that the inputs do not affect the switching activity of a hardware block. To remedy this problem, activity-sensitive empirical power models were developed. These schemes are based on predictable input signal statistics; an example is the method proposed by Landman and Rabaey [12]. The overall accuracy of such models may be hampered due to incorrect input statistics or the inability to correctly model the interaction.

The second empirical technique, transition-sensitive power models, is based on input transitions rather than input statistics. The method, proposed by Mehta et al. [13], assumes that a power model is provided for each functional

unit—a table containing the power consumed for each input transition. Closely related input transitions and power patterns can be concentrated in clusters, thereby reducing the size of the table. Other researchers have also proposed similar macro-model-based power estimation approaches [14, 15].

2.1.4. Architectural-Level Estimations. Recently, various architectural power simulators have been designed that employ a combination of lower level of abstraction power consumption models. These simulators derive power estimates from the analysis of circuit activity induced by the application programmes during each cycle and from detailed capacitive models for the components activated. A key distinction between these different simulators is the estimation accuracy and estimation speed. For example, the *SimplePower* power simulator [16] employs a transition-sensitive power model for the datapath functional unit. The *SimplePower* core accesses a table containing the switch capacitance for each input transition of the functional unit exercised.

The use of a transition-sensitive approach has both design challenges as well as performance concerns during simulation. The first concern is that the construction of these tables is time-consuming. Unfortunately, the size of this table grows exponentially with the size of the inputs. The table construction problem can be addressed by partitioning and clustering mechanisms. The second concern is the performance cost of the table lookup for each component access in a cycle. In order to overcome this cost, simulators such as *SoftWatt* [17] and *Wattch* [18] utilize a simple fixed-activity model for the functional unit. These simulators only track the number of accesses to a specific component and utilize an average capacity value to estimate the power consumed. Even the same simulator can employ different types of power models for different components. For example, *SimplePower* estimates the power consumed in the memories utilizing analytical models [19]. In contrast to the datapath components that utilize a transition-sensitive approach, these models estimate the power consumed per access and do not accommodate the power differences found in sequences of accesses. One of the most widely used microarchitectural power simulators is *Wattch* [18]. *Wattch* is a power simulator for superscalar, out-of-order, processors. It has been developed with aid of the infrastructure offered by *SimpleScaler* [20]. The power estimation engine of *Wattch* is based on the *SimpleScaler* architecture, but in addition, it supports detailed cycle-accurate information for all models, including datapath elements, memory, control logic, and clock distribution network [21].

While providing good accuracy, low-level power estimation methodologies are slow and impractical for analyzing the power consumption at an early design stage. Moreover, these methodologies require the availability of lower level circuit details or a complete Hardware Description Language (HDL) design of the targeted processor, which is not available for most of the commercial off-the-shelf processors.

2.2. High-Level Estimation Techniques. Recently, the demand has increased for high level power estimation simulators that allow an early design space exploration from the power consumption perspective. The existing high-level power estimation models can be classified into two main categories, Instruction Level Power Analysis (ILPA) and Functional Level Power Analysis (FLPA).

2.2.1. Instruction Level Power Analysis. An instruction level power model for individual processors was first proposed by Tiwari et al. [22]. 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. Tiwari et al. modeled the power consumption of the Intel DX486 processor. Power is modeled as a base cost for each instruction plus the interinstruction 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 interinstruction effects come into play, which are not reflected in the cost computed solely from base cost. These effects can be summarized as the following.

(i) 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.

(ii) Resource constraints: resource constraints in the CPU can lead to stalls, for example, pipeline stalls and write buffer stalls.

(iii) Cache misses: another interinstruction effect is the effect of cache misses. The instruction timings listed in manuals provide 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 by Tiwari et al. 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 multimeter. Much more accurate measuring environments have been proposed to precisely monitor the instantaneous current drawn by the processor instead of the average current. One of these approaches has employed a high-performance current mirror based on bipolar junction transistors as current sensing circuit. The power profiler in the work of Nikolaidis et al. [23] receives as input the trace file of executed assembly instructions, generated by an appropriate processor simulator, and estimates the base and interinstruction energy cost of the executed program taking into account the energy sensitive factors as well as the effect of pipeline stalls and flushes. The main disadvantage of this approach is the current measuring complexity [24].

Another approach, to reduce the spatial complexity of instruction-level power models is presented in [25]. Therein, interinstruction effects have been measured by considering only the additional energy consumption observed when a generic instruction is executed after a no-operation (NOP) instruction.

An attempt to modify the original ILPA to create an instruction level power model with a gate level simulator is carried out by Sama et al. [26]. In this approach, the power cost values were obtained through a power simulator rather than actual measurement; thus modeling is possible at design time and can be part of microarchitecture and/or instruction set architecture exploration. More researchers attempted to enhance the original Tiwari ILPA power consumption modeling technique as in [27–29].

The ILPA-based methods have some drawbacks, one of these drawbacks is that the number of current measurements is directly related to the number of instructions in the Instruction Set Architecture (ISA) and also the number of parallel instructions composing the very long instruction in the VLIW processor. The problem of instruction level power characterization of K-issue VLIW processor is $O(N^{2K})$ where N is the number of instructions in the ISA and K is number of parallel instructions composing the VLIW [30]. Also they do not provide any insight on the instantaneous causes of power consumption within the processor core, which is seen as a black-box model. Moreover, the effect of varying data (as well as address) is ignored in the ILPA models, though this effect can be accounted by an additive factor [31].

2.2.2. Function Level Power Analysis. FLPA was first introduced by Laurent et al. in [32]. The functional level power modeling approach is applicable to all types of processor architectures. Furthermore, FLPA modeling can be applied to a processor with moderate effort, and no detailed knowledge of the processors circuitry is needed. 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 [32]. First, a functional analysis of these blocks is performed to specify and then discard the nonconsuming blocks (those with negligible impact on the power-consumption). 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.

Laurent et al. [32, 33] presented a functional level power consumption model for the TI C62x DSP series. The C62x series has four internal memory modes which are handled by Laurent et al. model. The targeted architecture in our proposed model is the TI C64x. There are significant differences between the C64x and the C62x architectures. The internal program and data memories of the C62x have been replaced by two level-1 caches in the C64x. Moreover, the C64x includes a level-2 SRAM that is utilized both for the data and program with the ability to be partially configured as level-2 cache memory. Moreover, the C64x has the ability to utilize SIMD instructions. The number of registers is also doubled (2×32 in place of 2×16).

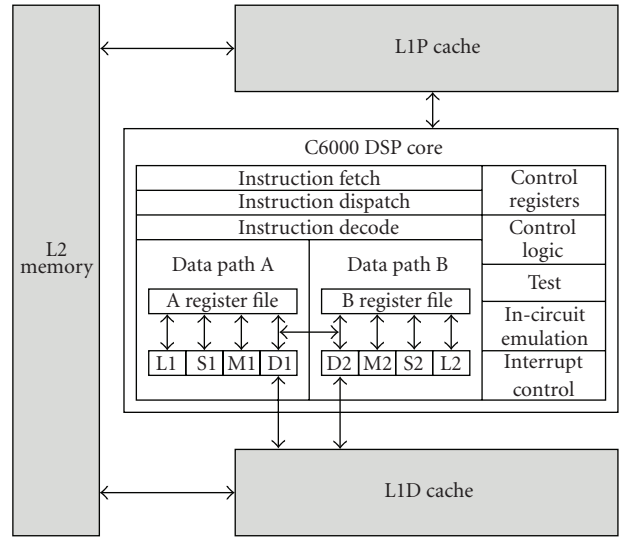

Figure 1: C6416T block diagram.

Unlike the model presented in [32, 33] that considers the load and store instructions as a part of the processing unit submodel, we had to consider the internal memory as a separate functional block. Hence, we believe that our proposed model is substantially different from the model presented in [32, 33] but required for accurately modeling the C64 family.

The work presented in [34] briefly points to the C64x but is lacking details. Furthermore, as of now (submission time of the paper) the model for the C64x is not included in the library of the latest SoftExplorer tool [35]. Unlike the model introduced in [36] that employs the parallelism factor as the affecting parameter for the processing units (PUs) block, the fact that the NOP does not require any PU for its execution convinced us that another parameter yields a better description of the PUs. Moreover, as we will explain later, the level-1 data cache memory submodel is different as well.

3. Target Architecture

In this section we briefly consider the target processor architectural features and the experimental setup used in our work.

A block diagram of the C6416T CPU is shown in Figure 1. It is one of the highest performance fixed-point DSPs that features deep pipeline (11 stages), eight 32-bit instructions/cycle, twenty-eight operations/cycle, and up to 8000 MIPS. The VLIW TMS320C64x+ DSP core consists of six ALUs (32-/40-bit), each supports single 32-bit, dual 16-bit, or quad 8-bit arithmetic per clock cycle, two multipliers support four 16×16-bit multiplications (32-bit results) per clock cycle or eight 8×8-bit multiplications (16-bit results)

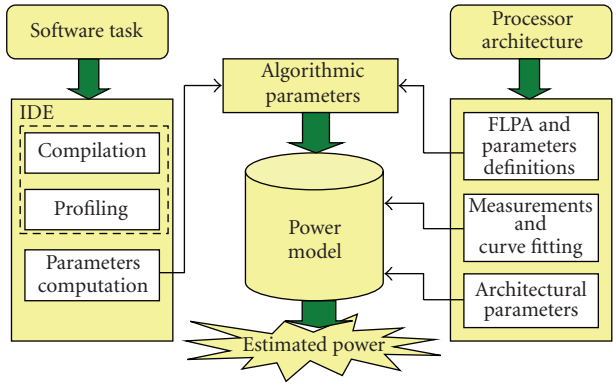

Figure 2: Functional level power estimation general methodology.

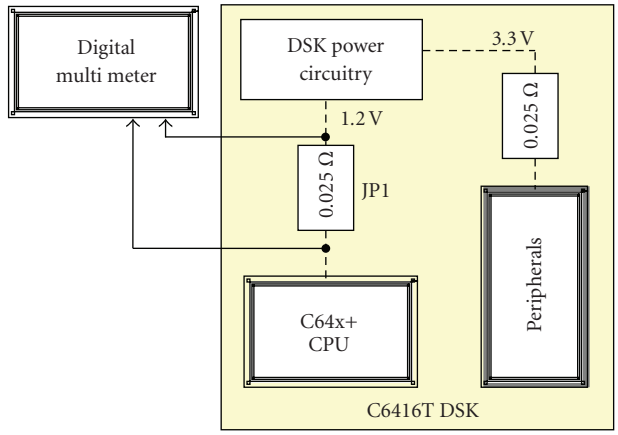

Figure 3: Current measurement setup.

per clock cycle, 64-/32-bit general purpose registers, and nonaligned load-store architecture. Instruction set features byte-addressable (8-/16-/32-/64-bit data). L1/L2 memory architecture composed of 16 kB two-way set associative L1D cache, with a 64-byte line size and 128 sets, 16 k-byte direct-mapped L1P cache, with a 32-byte line size and 512 sets, and 1024 k-byte L2 unified mapped RAM/cache, with flexible allocation configurations [37].

4. Methodology

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 [32]. Figure 2 illustrates the process of estimating the power consumption with aid of the FLPA technique. At first, a functional analysis of these blocks is performed to specify and then discard the nonconsuming blocks (those with negligible impact on the power consumption). The second step is to figure out the parameters that affect the power consumption of each of the power consuming blocks. This kind of parameters is called algorithmic parameters as they are computed from the software algorithm. For instance, the IMU is affected by the instructions dispatching rate which in turn is related to the parallelism degree. 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 which are called architectural parameters [33].

By means of simulations or measurements it is possible to find an arithmetic function for each block that determines its power consumption depending on a set of parameters. Hence, to determine the arithmetic function for each functional block, the average supply current of the processor core is measured in relation with the variation of the affecting parameter. These variations are achieved by a set of small programs, called scenarios. Such scenarios are short programs written in assembly language. Each program consists of an unbounded loop with a body of several hundreds of certain instructions that individually invoke each block. The power consumption rules are finally obtained by curve-fitting the measurement values [33].

All the current measurements are carried out on the TMS320C6416T DSP Starter Kit (DSK) manufactured by Spectrum Digital Inc. There are three power test points on this DSK for DSP I/O current, DSP core current, and system current. The operating frequency ranges from 600 MHz to 1200 MHz and the DSP core voltage is 1.2 V. The C/C++ compiler embedded in the Code Composer Studio (CCS3.1) from Texas Instruments is used for getting the binaries to be loaded to the DSP. The current drawn by the DSP core while running an algorithm (IDD) is captured by the Agilent 34410A 6.5 digit digital multimeter (DMM). This DMM features very high DC basic accuracy, actually 0.003% of the reading plus 0.003% of the range [38]. As shown in Figure 3 the current is captured in terms of the differential voltage drop across a 0.025 Ω sense resistor inserted, by the DSK manufacturer, in the current path of the DSP core. The input differential voltage drop is divided by 0.025 Ω to obtain the current drawn. Several assembly language scenarios have been developed to separately stimulate each of the functional blocks. Each scenario consists of an unbounded loop with a body of more than 1000 instructions, to avoid the effect of branching instructions on the measured current.

The parameters that affect the power consumption for each functional block can be extracted from the assembly code generated by the CCS3.1. Some parameters cannot be extracted directly from the assembly code, such as the execution time and the data cache miss rate. Therefore, the code should be run at least once to obtain these parameters with the aid of the code profiler.

5. C6416T Power Consumption Model

After applying the FLPA, the C6416T architecture is subdivided into six distinct functional blocks (clock tree, instruction management unit, processing unit, internal memory, L1 data cache and L1 program cache) as shown in Figure 4. The L2 unified memory accesses are considered via handling the L1 data and program cache misses. The L2 cache misses are not handled in our model as the L2 cache misses require the data or program to be accessed from an external memory that is beyond the scope of our proposed model. Moreover,

Table 1: Methodology of computing algorithmic parameters.

Parameter	Computation methodology
α	No. of fetch packets/No. of execution packets
β	(No. of executed instructions—NOP instructions)/Total code cycles
ε	(No. of L1D read hits/Total code cycles) · 100
λ	(No. of L1D write hits/Total code cycles) · 100
γ	((No. of L1D read misses + No. of L1D write misses)/No. of L1D references) · 100
δ	(No. of L1P misses/No. of L1P references) · 100
PSR	No. of CPU stall cycles/Total code cycles

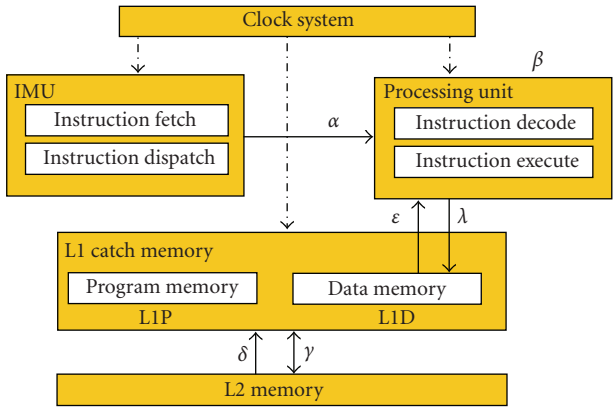

Figure 4: Functional level power analysis for C6416T.

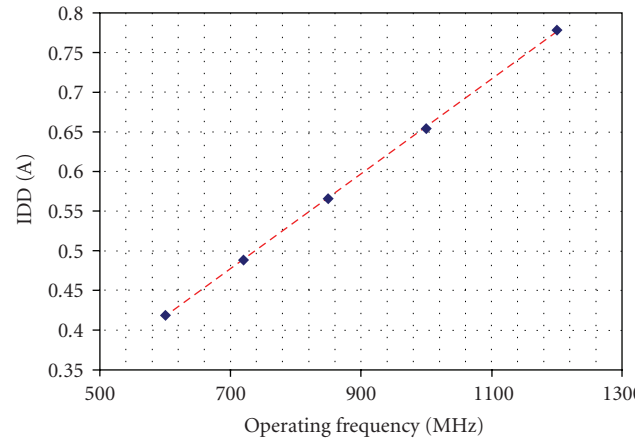

Figure 5: Model function of the C6416T clock tree.

there are extreme differences between individual realizations of external memories in terms of their access time and power consumption. The utilization of the FLPA also results in the algorithmic parameters that affect the power consumption for the prementioned functional blocks.

The dispatching rate α represents the average number of execution packets per fetch packet. The processing rate β stands for the average number of active processing units per cycle. The internal memory read/write access rates ε/λ, respectively, express the number of memory accesses divided by the number of required clock cycles for executing the code segment under investigation. The data cache miss rate γ corresponds to the number of data cache misses divided by the total memory accesses. Finally, the program cache miss rate δ corresponds to the number of program cache misses divided by the total program cache references. Table 1 shows how these algorithmic parameters of the proposed model are computed with the aid of the C6416T profiler. The C6416T profiler, which is embedded in the CCS3.1, offers many statistics regarding the program under investigation that are utilized in the process of computing our proposed model such as number of execution packets, number of NOP instruction cycles, and number of L1D cache misses.

5.1. Static and Clock Distribution Power Consumption Submodel. The static power consumption of any processor includes the power consumed due to leakage current and the clock distribution network. It is not possible at the functional level analysis to differentiate between those types of power consumption. Hence, both static and clock distribution power consumption are considered in a single submodel as the static and clock distribution power consumption model. From now on when we talk about the operating frequency effect on the power consumption, actually the effect of static and clock distribution is meant. It is clear that the parameter that affects the power consumption of the clock tree functional block is the operating frequency. Thus, we modeled the effect of the operating frequency on the power consumption. The operating frequency linearly affects the current drawn by the DSP core and hence, also linearly affects the power consumption of the processor. Figure 5 shows the relation between the operating frequency and the current drawn by the DSP core.

5.2. IMU Power Consumption Submodel. The IMU unit of the C6416T processor consists of two main subunits which are the instructions fetching unit and the dispatching unit. The IMU fetches eight instructions per cycle as one fetch packet. The dispatch unit then subdivides this fetch packet into execution packets. Since the C6416T has eight functional units, it is capable of simultaneously executing up to eight instructions. Consequently, the dispatch unit can divide the fetch packet into ne (maximum parallelism) to eight (sequential) execution packets. Thus, the dispatching rate is strongly affecting the instruction parallelism. Therefore, it is

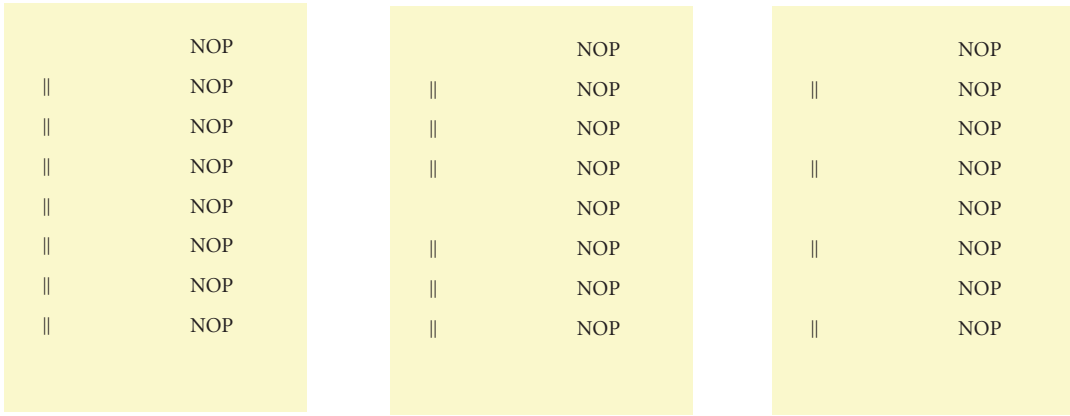

Figure 6: Screen shots of the scenarios for varying α.

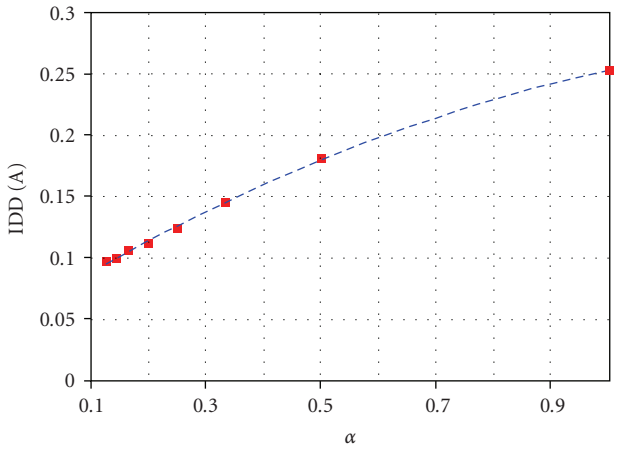

Figure 7: Model function of the C6416T IMU at $F = 1000$ MHz.

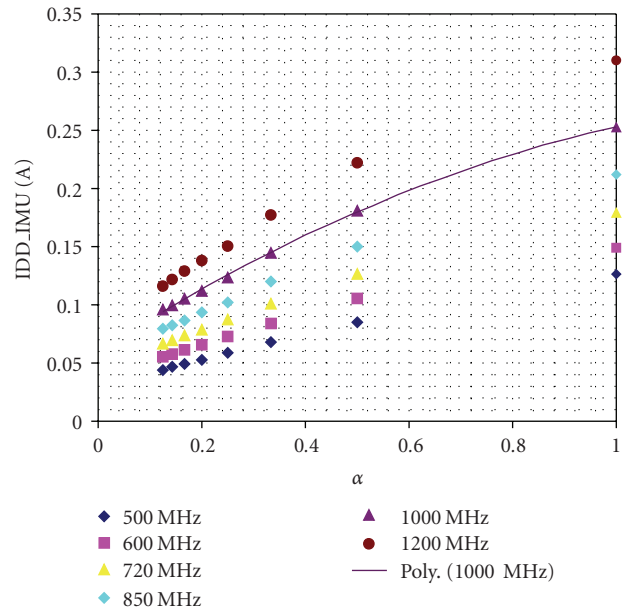

Figure 8: Model function of the C6416T IMU at different frequencies.

obvious that the dispatch rate is the parameter that affects the power consumption of the IMU.

Since the NOP instruction does not require any processing unit for its execution, each of the proposed scenarios to invoke the IMU is composed of an unbounded loop with more than 1000 no operations (NOPs). These scenarios vary the dispatch rate (number of fetch packets divided by the number of execution packets) from 0.125 to 1.0. Figure 6 shows screen shots of the scenarios to vary the dispatch rate, where the pipe symbols (||) indicate parallel instructions, that is, the first instruction without || together with the successor instructions with || is executed in the same clock cycle. Figure 7 indicates the characteristics of the current drawn by the core processor with a varying dispatch rate when the operating frequency is adjusted to 1000 MHz. Figure 8 indicates that varying α is independent of varying the operating frequency.

By curve fitting the measurement values in Figure 7 the arithmetic function is obtained:

$$\mathrm{IDD}_{\mathrm{IMU}} = -0.0918\alpha^2 + 0.284\alpha + 0.0603. \quad (1)$$

The quality of the fitting process is measured by the value R-squared (R^2): a number from 0 to 1, which is the normalized square of the residuals of the data after the fit. This value expresses what fraction of the variance of the data is explained by the fitted trend line. It reveals how closely the estimated values for the trend line correspond to the actual data. A trend line is most reliable when its R^2 value is at or close to 1.0 [39]. Since the R^2 value for the arithmetic function in (1) equals 0.9994 then (1) is an excellent fit for the curve values in Figure 7.

The arithmetic function in (1) does not consider the effect of pipeline stalls. Many reasons cause the pipeline to stall. For instance, one data cache miss stalls the pipeline for at least six cycles. Hence, the arithmetic function in (2) is

```
        ADD A0, 5, A1
||      SUB A2, 4, A10
||      AND A4, A0, A3
||      MPY A6, A7, A15
||      ADD B0, B4, B2
||      SUB B4, 15, B3
||      AND B5, A0, B6
||      MPY B5, B4, B7
```

$\alpha = 1$ $\beta = 8/8$ (a)

```
        ADD A0, 5, A1
||      NOP
||      AND A4, A0, A3
||      NOP
||      NOP
||      SUB B4, 15, B3
||      NOP
||      MPY B5, B4, B7
```

$\alpha = 1$ $\beta = 4/8$ (b)

```
        ADD A0, 5, A1
||      NOP
||      NOP
||      NOP
||      NOP
||      NOP
||      NOP
||      MPY B5, B4, B7
```

$\alpha = 1$ $\beta = 2/8$ (c)

Figure 9: Difference between β and α.

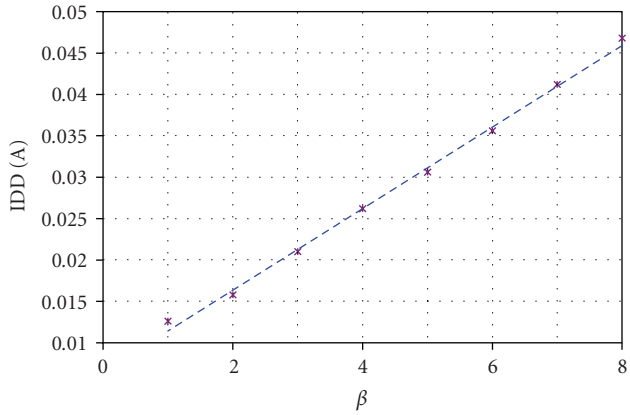

Figure 10: Model function of the C6416T processing units at $\alpha = 1$ and $F = 1000$ MHz.

presented to account for the pipeline stall effect"

$$\mathrm{IDD_{IMU}} = (-0.0918\alpha^2 + 0.284\alpha + 0.0603)(1 - \mathrm{PSR}), \quad (2)$$

where PSR stands for pipeline stall rate which can be expressed as the number of pipeline stall cycles divided by the total cycles required for executing the code segment under investigation.

5.3. PU Power Consumption SubModel. The data path of the C6416T consists of eight functional units. These functional units can work simultaneously if the dispatch unit succeeds to compose an execution packet with eight instructions. Unlike the model in [36] that uses the parallelism degree as the affecting parameter for the processing unit submodel, the fact that the NOP does not require any PU for its execution convinced us that another parameter yields a better description of the PUs.

The new parameter is the processing unit rate which expresses the average number of active processing units per cycle. Figure 9 illustrates the difference between the dispatch rate and the processing unit rate. Another important parameter that affects the processing unit power consumption is the word length of the data operands. In the C6416T the word length varies from 8 bits to 32 bits. Thus, in our model 16-bit word length has been chosen to be the typical word length.

More than 1000 different instructions compose the scenarios that vary the processing unit rate, that is to account for the inter-instructions effect. The current measured from the DSK is the sum of the clock tree, IMU, and the PU currents. To attain only the current drawn by the PU, the IMU and clock tree currents are subtracted from the measured current.

Figure 10 depicts the effect of varying the number of active PU per cycle on the current drawn by the core processor.

$$\mathrm{IDD_{PU}} = (-0.0049\beta + 0.0065)(1 - \mathrm{PSR}). \quad (3)$$

The arithmetic function in (3) results in an excellent fit for the curve values in Figure 10 with an R^2 value of 0.9982. Compared to other functional units such as clock tree or the IMU, it is clear that the PU does not significantly contribute to the total power consumption of the core processor. It is important to mention that the scenario for invoking the PU does not include any memory instructions. The internal memory operations are handled in a separate scenario.

5.4. Internal Memory Power Consumption SubModel. As mentioned in Section 5.3 the internal memory operations are separately handled. That is because of its distinct execution characteristics. Two categories of memory operations are included in the instruction set of the C6416T DSP load and store. The load instructions represent the read of data from the data cache (if the operand exist in the data cache) to a specific register from the processor's register file. The store instructions represent the write of data into the memory, according to the data cache write policy.

The C64x+ architecture is capable of performing two memory operations per cycle. The affecting parameters for the internal memory submodel are the memory read access rate ε and the memory write access rate λ. The memory access rate is defined as the number of memory references (read and write) divided by the algorithm execution time.

Figure 11 illustrates snapshots of different scenarios to vary the memory read access rate ε from 20% to 180% (as two memory operations can be simultaneously executed). All of those scenarios conducted with the same $\alpha = 1/4$; where D1 and D2 are the processing functional units that

	LDHU	.D1	*A25++,A26		LDHU	.D1	*A25++,A26		LDHU	.D1	*A25++,A26
||	LDHU	.D2	*B25++,B26	||	LDHU	.D2	*B25++,B26	||	LDHU	.D2	*B25++,B26
	NOP				LDHU	.D1	*A25++,A27		LDHU	.D1	*A25++,A27
||	NOP			||	LDHU	.D2	*B25++,B27	||	LDHU	.D2	*B25++,B27
	NOP				LDHU	.D1	*A25++,A28		LDHU	.D1	*A25++,A28
||	NOP			||	LDHU	.D2	*B25++,B28	||	LDHU	.D2	*B25++,B28
	NOP				NOP				LDHU	.D1	*A25++,A29
||	NOP			||	NOP			||	LDHU	.D2	*B25++,B29
	NOP				NOP				LDHU	.D1	*A25++,A30
||	NOP			||	NOP			||	LDHU	.D2	*B25++,B30
	NOP				NOP				LDHU	.D1	*A25++,A31
||	NOP			||	NOP			||	LDHU	.D2	*B25++,B31
	NOP				NOP				LDHU	.D1	*A25++,A26
||	NOP			||	NOP			||	LDHU	.D2	*B25++,B26
	NOP				NOP				LDHU	.D1	*A25++,A27
||	NOP			||	NOP			||	LDHU	.D2	*B25++,B27
	NOP				NOP				LDHU	.D1	*A25++,A28
||	NOP			||	NOP			||	LDHU	.D2	*B25++,B28
	NOP				NOP				NOP		
||	NOP			||	NOP			||	NOP		
	$\alpha = 1/4$	$\varepsilon = 20\%$			$\alpha = 1/4$	$\varepsilon = 60\%$			$\alpha = 1/4$	$\varepsilon = 180\%$	

Figure 11: Snapshots of different scenarios for varying ε.

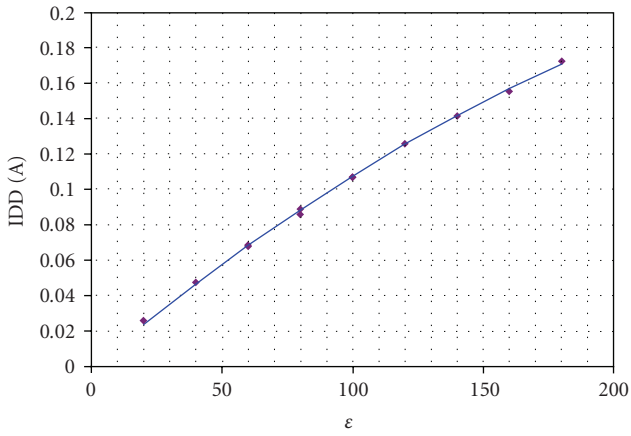

Figure 12: Model function of the C6416T internal memory read at $\alpha = 1$ and $F = 1000$ MHz.

are responsible for load and store operations [40]. Figure 12 shows the measured current values for different ε values

$$\mathrm{IDD}_{\mathrm{Int\_Mem\_Read}} = \left(-2 \cdot 10^{-6}\varepsilon^2 + 0.0012\varepsilon\right)(1 - \mathrm{PSR}). \quad (4)$$

The arithmetic function in (4) results in an excellent fit for the curve values in Figure 7 with an R^2 value of 0.9995. When we tried to fit the values of the curve in Figure 12 with a linear arithmetic function, we hit upon that the resultant R^2 value equals 0.98, equivalent to an error of 2%. As the final model will be the summation of the different functional block submodels the final model estimation error will be an accumulation of the submodels errors. Therefore, we decided to minimize the curve fitting error as much as possible. Hence, we choose the arithmetic function in (4) to represent the internal memory read submodel.

In the same manner (5) represents the current drawn from the CPU while running different scenarios that vary the memory write access rate λ. This equation results in an R^2 value of 0.9978

$$\mathrm{IDD}_{\mathrm{Int\_Mem\_Write}} = \left(-10^{-5}\lambda^2 + 0.0049\lambda\right)(1 - \mathrm{PSR}). \quad (5)$$

Hence, (6) represents the total internal memory model:

$$\mathrm{IDD}_{\mathrm{Int\_Mem}} = \mathrm{IDD}_{\mathrm{Int\_Mem\_Read}} + \mathrm{IDD}_{\mathrm{Int\_Mem\_Write}}. \quad (6)$$

5.5. L1 Data Cache Power Consumption SubModel. The L1 data cache functional block represents the flow of data from the L1 data cache to L2 memory and vice versa. Different scenarios are prepared to stimulate the effect of the data cache miss.

The data cache miss rate is used as the affecting parameter for the L1 data cache functional block. Taking into account the fact that the L1D cache is a two-way associative cache, different scenarios that vary the number of data cache misses per fixed number of memory accesses have been developed. In this scenario, a deterministic way for forcing the data cache misses is followed. First, arbitrary data are preloaded into both blocks of set 0. Second, data from L2 memory with addresses that must be mapped into set 0 blocks are loaded to L1D cache. The new data, from L2 memory, addresses are different from those already preloaded to set 0. Hence, a data cache miss occurs as illustrated in Figure 13.

Figure 14 shows the effect of varying the data cache miss rate on the current drawn by the core processor. The arithmetic function in (7) results in an excellent fit for the curve values in Figure 14 with an R^2 value of 0.9909. Although the quadratic term in (7) is very small compared to the linear term, it has great impact on the R^2 value. Discarding the quadratic term in (7) results in an R^2 value of 0.9272 with an error of 7.28%, thus, (7) is a very

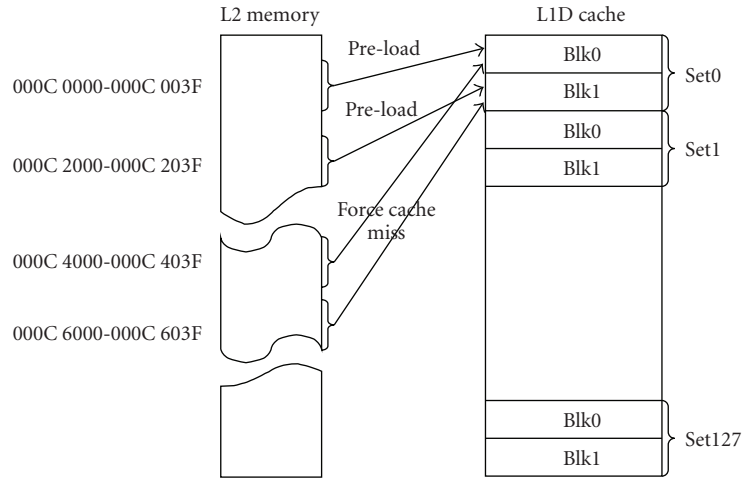

Figure 13: Scenario for forcing a data cache miss.

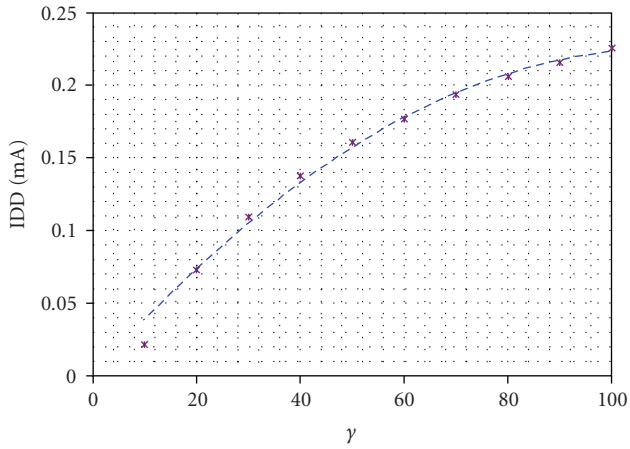

Figure 14: L1D cache miss rate versus measured CPU current.

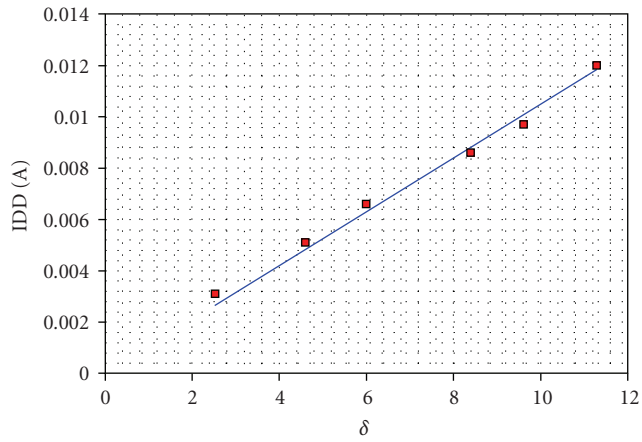

Figure 15: L1P cache miss rate versus measured CPU current.

suitable function to represent the L1D cache misses power consumption:

$$\mathrm{IDD}_{\mathrm{L1D}} = \left(-2 \cdot 10^{-5}\gamma^2 + 0.0041\gamma\right)(1 - \mathrm{PSR}). \quad (7)$$

The arithmetic function in (7) differs from the corresponding linear function that was proposed in [36] for the cache functional block. The squared-function yields a better description for the L1 data cache block due to the fact that L1 data cache pipelines the cache misses, to decrease the resulting pipeline stalls. The proposed model in [33] did not separately investigate the effect of data cache misses; instead it is included in the processing unit functional block [41].

5.6. L1 Program Cache Power Consumption SubModel. With the aid of the profiler of the C6416T device accurate cycle simulator, different scenarios are prepared that arbitrarily vary the program cache miss rate δ. Figure 15 shows the effect of varying the program cache miss rate on the current drawn by the core processor. The best fit for the measured values in Figure 15 is obtained as indicated in (8) with an R^2 value of 0.9889.

$$\mathrm{IDD}_{\mathrm{L1p}} = 0.0011\delta(1 - \mathrm{PSR}). \quad (8)$$

The complete FLPA power consumption model for the C6416T fixed-point high-performance VLIW DSP is shown in Table 2, while the complete model with exact constant values at an operating frequency of 1000 MHz is illustrated in Table 3.

6. Model Validation

Some common signal and image processing benchmarks from Texas Instruments libraries are used for demonstration purpose as described in Table 4. The input data for all used benchmarks are located in the internal data memory. All the benchmarks are executed in an infinite loop to obtain a stable reading on the DMM.

First of all, all optimization options which are included in the CCS3.1 are turned off because these optimization options

Table 2: Complete power consumption model for C6416T DSP.

Functional unit	Functional unit power consumption submodel
Clock distribution	$P_{\mathrm{F}} = (\mathrm{a1} \cdot F + \mathrm{a2}) \cdot V_{\mathrm{core}}$
IMU	$P_{\mathrm{IMU}} = (\mathrm{b1} \cdot \alpha^2 + \mathrm{b2} \cdot \alpha + \mathrm{b3})(1 - \mathrm{PSR}) \cdot F \cdot V_{\mathrm{core}}$
Processing units	$P_{PU} = (\mathrm{c1} \cdot \beta + \mathrm{c2})(1 - \mathrm{PSR}) \cdot F \cdot V_{\mathrm{core}}$
Memory read	$P_{\mathrm{MemR}} = (\mathrm{d1} \cdot \varepsilon^2 + \mathrm{d2} \cdot \varepsilon)(1 - \mathrm{PSR}) \cdot F \cdot V_{\mathrm{core}}$
Memory write	$P_{\mathrm{MemW}} = (\mathrm{e1} \cdot \lambda^2 + \mathrm{e2} \cdot \lambda)(1 - \mathrm{PSR}) \cdot F \cdot V_{\mathrm{core}}$
L1D cache	$P_{\mathrm{L1D}} = (\mathrm{g1} \cdot \gamma^2 + \mathrm{g2} \cdot \gamma)(1 - \mathrm{PSR}) \cdot F \cdot V_{\mathrm{core}}$
L1P cache	$P_{\mathrm{L1P}} = (\mathrm{h1} \cdot \delta)(1 - \mathrm{PSR}) \cdot F \cdot V_{\mathrm{core}}$
Total power	$P_{\mathrm{T}} = P_{\mathrm{F}} + P_{\mathrm{IMU}} + P_{PU} + P_{\mathrm{MemR}} + P_{\mathrm{MemW}} + P_{\mathrm{L1D}} + P_{\mathrm{L1P}}$

Table 3: Complete power consumption model for C6416T DSP at $F = 1000$ MHz.

Functional unit	Functional unit power consumption submodel
Clock distribution	$\mathrm{P}_{\mathrm{Clock\_Distribution}} = (0.0006F + 0.0574) \times \mathrm{V}_{\mathrm{core}}$
IMU	$\mathrm{P}_{\mathrm{IMU}} = (-0.0918\alpha^2 + 0.284\alpha + 0.0603)(1 - \mathrm{PSR}) \cdot \mathrm{V}_{\mathrm{core}}$
Processing units	$\mathrm{P}_{PU} = (-0.0049\beta + 0.0065)(1 - \mathrm{PSR}) \cdot \mathrm{V}_{\mathrm{core}}$
Memory read	$\mathrm{P}_{\mathrm{Mem\_Read}} = (-2 \cdot 10^{-6}\varepsilon^2 + 0.0012\varepsilon)(1 - \mathrm{PSR}) \cdot \mathrm{V}_{\mathrm{core}}$
Memory write	$\mathrm{P}_{\mathrm{Mem\_Write}} = (-10^{-5}\lambda^2 + 0.0049\lambda)(1 - \mathrm{PSR}) \cdot \mathrm{V}_{\mathrm{core}}$
L1D cache	$\mathrm{P}_{\mathrm{L1D}} = (-2 \cdot 10^{-5}\gamma^2 + 0.0041\gamma)(1 - \mathrm{PSR}) \cdot \mathrm{V}_{\mathrm{core}}$
L1P cache	$\mathrm{P}_{\mathrm{L1P}} = 0.0011\delta(1 - \mathrm{PSR}) \cdot \mathrm{V}_{\mathrm{core}}$

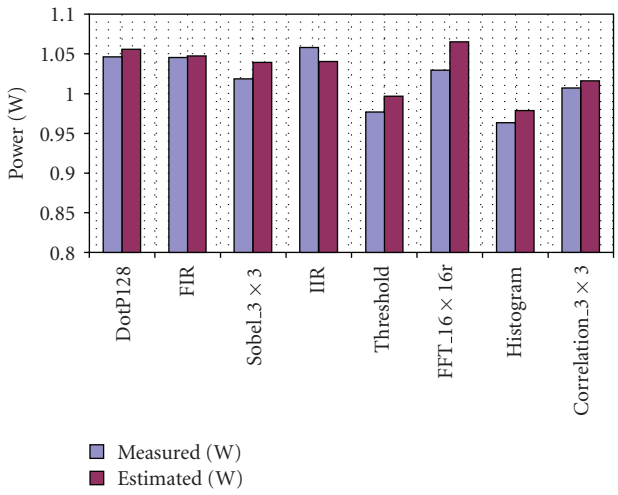

Figure 16: Estimated versus measured power consumption of the C6416T at $F = 1000$ MHz.

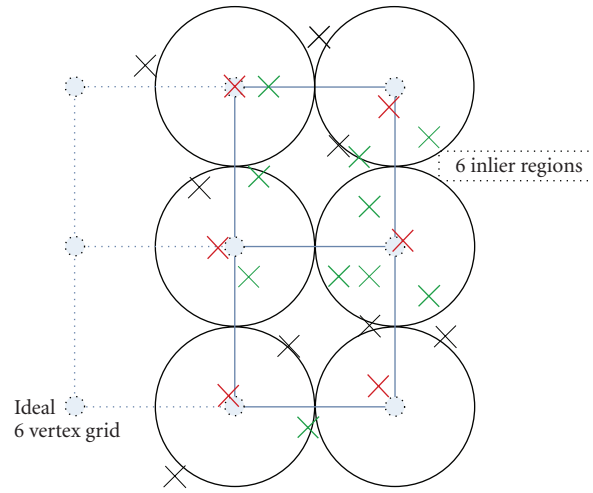

Figure 17: Illustration of the plants scatter-plot.

affect the speed or the code size only and are not dedicated to power optimization. The second step is to compile the benchmarks.

The required parameters for the model are calculated either statically from the generated assembly files or with the aid of the CCS3.1 profiler for the parameters that cannot be estimated statically such as the data cache miss rate. For instance, the processing unit rate which is defined as the average number of active processing units per cycle is calculated from the assembly code. The parameter β is the result of dividing the number of processing units (equals the number of instructions excluding the NOP) by the number of cycles per code iteration. Figure 16 presents the result of the estimated power consumption versus the measured one for the benchmarks listed in Table 4. The absolute average estimation error is 1.65% while the worst is 3.3%.

Unlike almost all the benchmarks of Figure 16, the physically measured power consumption for the IIR benchmark is higher than the estimated power. By investigating the generated assembly code of the IIR benchmark from the CCS3.1 we find that all the load and store operations inside the loops are of operand size 32 bits unlike the case in

Table 4: Benchmarks used for our experiments.

Benchmark	Description
DotP128	Dot product of a vector of 128 16-bit elements
m100	Matrix multiplication for 2 100×100 square matrices
FIR	Computes a real FIR filter, Input data and filter taps are 16-bit
Sobel 3×3	Apply Sobel filter of 3×3 window to an image of 8192 pixels
Thresholding	Performs a thresholding operation on an input image of 8192 pixels
Histogram	Takes histogram of an image of 8192, 8-bit pixels
IIR	Performs an auto-regressive moving-average (ARMA) filter with 4 auto-regressive filter coefficients and 5 moving-average filter coefficients
FFT16×16	Performs a mixed radix forwards FFT using a special sequence of coefficients
Correlation 3×3	Performs a point by point multiplication of the 3×3 mask with an input image

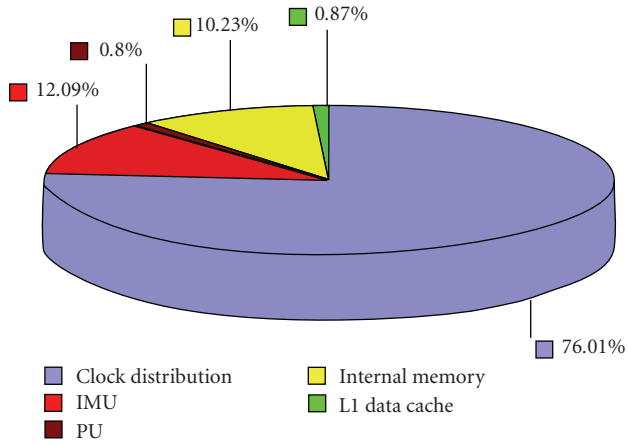

Figure 18: Average functional units contribution to the processor power consumption.

the other benchmarks; for example in the FFT 16×16 all the load and store instructions are of operand size 16 bit. Refereing to Section 4, we indicated that the word length of the instructions' operand affects the power consumption of almost all the functional blocks. Moreover, in Section 5 we stated that "in our proposed model we choose 16-bits word length to be the typical word length." Therefore the estimated power in the IIR benchmark is slightly higher than the physically measured power.

Moreover, we utilized a part of a real application, bad weeds recognition, for validating our developed power consumption model. The employed part of the application is a restricted set exhaustive search algorithm named Elastic Graph Model. Elastic Graph Model is a method for detecting nearly regular located objects in images. It proceeds as follows: any local maximum of the scatter plot is supposed to represent the left upper node of the ideal six-vertex grid. For any of the remaining five ideal grid nodes a subset of local maxima is determined from the remaining local maxima in the scatter plot. Hence, we obtain for any grid node a set of candidates, the so called inliers, which lie within a certain region around the ideal grid node. In Figure 17 these inliers are highlighted in green. Thus, the number of hypotheses for the elastic graph is any possible combination of inliers for any node. Any of these hypotheses is evaluated by two measures: the external energy, which represents the weight of the six local maxima of the respective hypothesis indicated in Figure 17 by thicker or thinner crosses and the internal energy, which is a measure that accumulates the squared error of the edge length between two hypothetical nodes (for all seven edges) and the ideal edge length (which is one in the usual case), also weighted by an edge weight (typically also one for all edges).

The power-consumption of the Elastic Graph Matching (EGM) algorithm is estimated with the aid of our proposed power consumption model described in Section 2. The estimated power consumption equals 1.0498 W while the physically measured power consumption equals 1.061 W, resulting in an estimation error of only 1%.

7. Conclusion

In this paper, we developed a precise functional level estimation technique to estimate the power consumption of the embedded software running on a programmable processor. The commercial off-the-shelf VLIW DSP C6416T from Texas Instruments is utilized as the targeted platform. The inter-instructions as well as the pipeline stall effects have been investigated in our proposed model. The validation and precision of our model have been proven by estimating the power consumption of many typical algorithms applied in signal and image processing. We further validated the precision of our developed model on a real application applied in the video processing field. The power consumption estimated by our model, is compared to the physically measured power consumption, achieving a very low absolute average estimation error of 1.65% and an absolute maximum estimation error of only 3.3%.

Although our methodology for modeling the TI C6416T VLIW processor power consumption is applicable for other VLIW processors, for different targeted processor

architecture we can reconsider the distinction of the processor architecture into functional blocks as shown in Figure 4. For example, if the new target architecture includes a floating-point unit, Figure 4 has to be modified by including a new functional block that represents the floating-point unit together with the parameters affecting its power consumption such as the access rate. Thus, as long as the power consumption of the functional blocks remains independent, our presented power estimation methodology is intended to work equally on all VLIW architectures.

Furthermore, the proposed model allows us to figure out the processor functional units that are dominantly contributing to the power consumption. Figure 18 illustrates the contribution percentages of the processor architectural function blocks to the total power consumption of the processor core. It is clear that the clock distribution is the largest contributor while the processing unit is the smallest contributor.

Finally, we have evaluated the global optimization levels of the CCS as well as two specific architectural features of the C6416T, namely, Software Pipelined Loop (SPLoop) and the Single Instruction Multi Data (SIMD) from the perspective of energy and power consumption [42, 43].

Acknowledgments

This work has been funded by the Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms as well as the COMET K-Project Embedded Computer Vision (ECV) in conjunction with the Austrian Institute of Technology (AIT).

References

[1] C. J. Bleakley, M. Casas-Sanchez, and J. Rizo-Morente, "Software level power consumption models and power saving techniques for embedded DSP processors," *Journal of Low Power Electronics*, vol. 2, no. 2, pp. 281–290, 2006.

[2] C. Brandolese, *A codesign approach to software power estimation for embedded systems*, Ph.D. disseration, Politecnico di Milano, Institute of Electronics and Information, 2000.

[3] C. X. Huang, B. Zhang, A. Deng, and B. Swirski, "Design and implementation of PowerMill," in *Proceedings of the International Symposium on Low Power Design (ISLPED '95)*, pp. 105–109, ACM, New York, NY, USA, April 1995.

[4] F. N. Najm, "Survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994.

[5] S. Gupta and F. N. Najm, "Power macromodeling for high level power estimation," in *Proceedings of the 34th Design Automation Conference (DAC '97)*, pp. 365–370, ACM, New York, NY, USA, June 1997.

[6] T. Chou and K. Roy, "Accurate power estimation of CMOS sequential circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 4, no. 3, pp. 369–380, 1996.

[7] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures of energy consumption at register transfer level," in *Proceedings of the International Symposium on Low Power Design (ISLPED '95)*, pp. 81–86, ACM, New York, NY, USA, 1995.

[8] J. N. Rabaey and M. Pedram, *Low Power Design Methodologies*, vol. 336 of *The Springer International Series in Engineering and Computer Science*, 1996.

[9] S. Powell and E. M. Chau, "Estimating power dissipation of VLSI signal processing chips: the PFA technique," in *VLSI Signal Processing IV*, pp. 250–259, 1990.

[10] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low-power VLSI systems," *IEEE Design and Test of Computers*, vol. 12, no. 3, pp. 70–84, 1995.

[11] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, 1994.

[12] P. E. Landman and J. M. Rabaey, "Activity-sensitive architectural power analysis for the control path," in *Proceedings of the International Symposium on Low Power Design (ISLPED '95)*, pp. 93–98, ACM, New York, NY, USA, April 1995.

[13] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy characterization based on clustering," in *Proceedings of the 33rd Annual Design Automation Conference (DAC '96)*, pp. 702–707, ACM, New York, NY, USA, June 1996.

[14] Q. Wu, Q. Qiu, M. Pedram, and C.-S. Ding, "Cycle-accurate macro-models for RT-level power analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 4, pp. 520–528, 1998.

[15] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli, "Regression models for behavioral power estimation," *Integrated Computer-Aided Engineering*, vol. 5, no. 2, pp. 95–106, 1998.

[16] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: a cycle-accurate energy estimation tool," in *Proceedings of the 37th conference on Design Automation (DAC '00)*, pp. 340–345, ACM, New York, NY, USA, June 2000.

[17] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin et al., "Using complete machine simulation for software power estimation: the softWatt approach," in *Proceedings of the 8th International Symposium on High-Performance Computer Architecture (HPCA '02)*, pp. 141–151, IEEE Computer Society, Washington, DC, USA, February 2002.

[18] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *SIGARCH Computer Architecture News*, vol. 28, no. 2, pp. 83–94, 2000.

[19] M. B. Kamble and K. Ghose, "Analytical energy dissipation models for low power caches," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '97)*, pp. 143–148, ACM, New York, NY, USA, August 1997.

[20] D. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," *SIGARCH Computer Architecture News*, vol. 25, no. 3, pp. 13–25, 1997.

[21] M. Pedram, *Power Aware Design Methodologies*, edited by J. M. Rabaey, Norwell, Mass, USA, Kluwer Academic Publishers, 2002.

[22] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, 1994.

[23] S. Nikolaidis, N. Kavvadias, P. Neofotistos, K. Kosmatopoulos, T. Laopoulos, and L. Bisdounis, "Instrumentation set-up for instruction level power modeling," in *Proceedings of the 12th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS '02)*, pp. 71–80, Springer, London, UK, 2002.

[24] S. Nikolaidis, N. Kavvadias, T. Laopoulos, L. Bisdounis, and S. Blionas, "Instruction level energy modeling for pipelined processors," *Journal of Embedded Computing*, vol. 1, no. 3, pp. 317–324, 2005.

[25] B. Klass, D. E. Thomas, H. Schmit, and D. F. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in *Proceedings of the Power Driven Microarchitecture Workshop in Conjunction with International Syymposism Computer Architecture*, June 1998.

[26] A. Sama, J. F. M. Theeuwen, and M. Balakrishnan, "Speeding up power estimation of embedded software," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '00)*, pp. 191–196, ACM, New York, NY, USA, 2000.

[27] J. T. Russell and M. F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in *Proceedings of the IEEE International Conference on Computer Design (ICCD '98)*, pp. 328–333, IEEE Computer Society, Washington, DC, USA, October 1998.

[28] H. Mehta, R. M. Owens, and M. J. Irwin, "Instruction level power profiling," in *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing (ICASSP '96)*, pp. 3326–3329, IEEE Computer Society, Washington, DC, USA, 1996.

[29] V. Steven, R. Gentile, D. R. Kaeli, and G. Olivadoti, "Developing energy-aware strategies for the blackfin processor," in *Proceedings of the High Performance Embedded Computing (HPEC '04)*, Septemper 2004.

[30] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria, "An instruction-level energy model for embedded VLIW architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 9, pp. 998–1010, 2002.

[31] M. Balakrishnan, "Low Power Design," Lectures, 2008, http://embedded.cse.iitd.ernet.in/homepage/course/low_power/index.shtml.

[32] J. Laurent, E. Senn, N. Julien, and E. Martin, "High level energy estimation for DSP systems," in *Proceedings International Workshop on Power And Timing Modeling and Optimization and Simulation (PATMOS '01)*, pp. 311–316, September 2001.

[33] E. Senn, N. Julien, J. Laurent, and E. Martin, "Power consumption estimation of a C program for data-intensive applications," in *Proceedings of the 12th International Workshop on Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation (PATMOS'02)*, pp. 332–341, Springer, London, UK, 2002.

[34] E. Senn, J. Laurent, N. Julien, and E. Martin, "SoftExplorer: estimating and optimizing the power and energy consumption of a C program for DSP applications," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 16, pp. 2641–2654, 2005.

[35] "SoftExplorer: Processors Power Estimation Tool," http://portal.acm.org/citation.cfm?id=1287311.

[36] M. Schneider, H. Blume, and T. G. Noll, "Power estimation on functional level for programmable processors," *Journal of Advances in Radio Science*, vol. 2, pp. 215–219, 2005.

[37] Texas Instruments Inc., "TMS320C6416T, Fixed Point Digital Signal Processor, Datasheet," SPRS226J, November 2003, http://www.ti.com.

[38] Agilent Technologies Inc., "Agilent 34410A Digital Multimeter, Datasheet," 5989-3738EN, October 2007, http://www.home.agilent.com/agilent/product.jspx?pn=34410A.

[39] N. R. Draper and H. Smith, *Applied Regression Analysis*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, NY, USA, 2nd edition, 1981.

[40] Texas Instruments Inc., "TMS320C6416T, Technical Overview," SPRU395B, January 2001, http://www.ti.com.

[41] M. E. A. Ibrahim, M. Rupp, and H. A. H. Fahmy, "Power estimation methodology for VLIW digital signal Processor," in *Proceedings of the Conference on Signals, Systems and Computers (SSC '08)*, pp. 1840–1844, IEEE Computer Society, Asilomar, Calif, USA, October 2008.

[42] M. E. A. Ibrahim, M. Rupp, and S. E.-D. Habib, "Compiler-based optimizations impact on embedded software power consumption," in *Proceedings of the IEEE Joint North-East Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA '09)*, pp. 247–250, IEEE, Toulouse, France, June 2009.

[43] M. E. A. Ibrahim, M. Rupp, and S. E.-D. Habib, "Performance and power consumption trade-offs for a VLIW DSP," in *Proceedings of the IEEE International Symposium on Signals, Circuits and systems (ISSCS '09)*, pp. 179–200, IEEE, Iasi, Romania, July 2009.