

# FLPA-based power modeling and power aware code optimization for a TriMedia DSP

J. von Livonius, H. Blume, T. G. Noll  
Chair for Electrical Engineering and Computer Systems  
RWTH Aachen University,  
Schinkelstraße 2, 52062 Aachen, Germany  
Phone: +49 (0)241 80 94367, Fax: +49 (0)241 80 92282  
Email:  [{livonius, blume, tgn}@eecs.rwth-aachen.de](mailto:{livonius, blume, tgn}@eecs.rwth-aachen.de)

*Abstract*—Low power techniques are of crucial importance for the architecture design of mobile devices and it is desirable to estimate the power consumption of a system at a very early stage in the design flow. Applying programmable processors for such systems it has to be regarded that the power consumption of a programmable digital signal processor (DSP)-kernel depends on several processor-specific influencing factors. All of these factors influence more or less strongly the power consumption of a task that is executed on a DSP.

In this paper the Functional-Level Power Analysis (FLPA) approach is exemplarily applied to a TriMedia TM1300 architecture targeted for multimedia applications. The modeling of all DSP-specific features is presented and the model functions for the single blocks are discussed. Some distinctive features like the fact that the power consumption depends on the issue slot where a single instruction is performed lead to an instruction and slot sensitive extension of the "classic" FLPA-modeling technique.

Various exemplary tasks are used as benchmark applications for the FLPA model. The resulting power consumption and the estimated power consumption feature on average a maximum estimation error less than 9 %.

Furthermore, the influence of DSP-specific code optimization on the resulting power consumption is discussed.

*Keywords*— power estimation, functional level power analysis (FLPA), digital signal processor, code optimization

## I. INTRODUCTION

In the course of increasing complexity of digital signal processing applications, especially in the field of mobile applications, low power techniques are of crucial importance. Therefore, it is desirable to estimate the power

consumption of a system at a very early stage in the design flow.

Like any other architecture block the power consumption of a programmable digital signal processor (DSP)-kernel depends on several factors like the switching statistics of the input data, the clock frequency and of course the executed task itself. Besides these dependencies there are many more processor-specific influencing factors which all more or less strongly influence the power consumption of a task that is executed on a DSP. Examples include the type and rate of memory accesses, the usage of specific architecture elements like DMA controllers or dedicated co-processors, different compiler optimization settings, pipeline stalls and cache misses but also different programming styles or the choice of algorithmic alternatives.

For this reason several methodologies for power estimation that cover significant influencing factors and provide a sufficient accuracy at moderate complexity have been conceived (e.g. [2, 3, 5, 7]). One of these is the so-called Functional-Level Power Analysis (FLPA) which has been presented and successfully applied e. g. in [1, 3, 7]. The FLPA-concept is based on the distinction of the processor architecture into functional blocks like processing unit, clock network, internal memory and others. The power consumption of these blocks is described by parameterized arithmetic models. By application of a parser based automated analysis of assembler codes the input parameters of the arithmetic functions like the achieved degree of parallelism or the kind and number of memory accesses can be calculated. In [1] it could be proven that for commercial DSP-architectures a good estimation accuracy can be achieved.

In this paper the FLPA-approach is exemplarily applied to the TriMedia TM1300 architecture. This DSP features a VLIW architecture and provides several co-processors and a cache hierarchy. The target application field of this processor is multimedia signal processing. The FLPA model incl. the modeling of all DSP-specific features is presented and the model functions for the single blocks are discussed. Some distinctive features like the fact that the power consumption depends on the issue slot where a single instruction is performed are emphasized and lead to an extension of the "classic" FLPA-modeling technique.

Various exemplary tasks out of the field of video signal processing have been used as benchmark applications for the FLPA model. The resulting power consumption and the estimated power consumption feature an average estimation error less than 9 %.

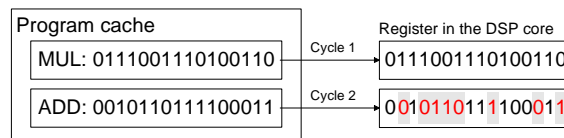
Furthermore, the influence of DSP-specific code optimization applying techniques like loop unrolling, application of restricted pointers and custom operations, etc. on the resulting power consumption is discussed. On the example of an FIR filter it is proven along several code optimization steps that the FLPA technique can also be leveraged for power-aware DSP-code optimization at an early stage of the design flow.

## II. POWER ESTIMATION APPROACHES FOR PROGRAMMABLE PROCESSORS

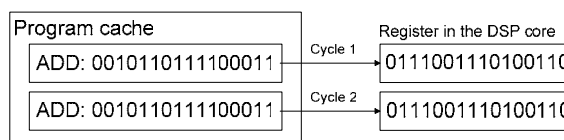
One possible straight power estimation approach for DSPs is the so-called Physical-Level Power Analysis methodology. This approach is based on the analysis of the switching activity of all circuit nodes of the DSP architecture. The requirement of this methodology is the availability of a detailed description of the processor architecture on transistor level, which is rarely given for modern DSPs and even more severe, it results in an extremely high computational effort. Architectural-Level approaches like [2] reduce this computational effort by abstracted modeling of typical architecture elements like registers, functional units or load/store queues. Therefore, these methodologies can be mainly found in the course of the development of high volume products like e.g. microprocessors. Due to their extremely high computational effort they are not suited to evaluate the power consumption of digital signal processing tasks on a DSP in practical use with acceptable computation time.

Another possibility for power estimation for DSPs is the so-called Instruction-Level Power Analysis [8]. By means of low level simulations or physical measurements the energy consumption of each instruction out of the instruction set of a given processor is determined. By

analysis of the assembler code of a software task it is then possible to estimate the specific power consumption of this program performed on a certain processor. The advantage of this approach is the ability to cover a specific part of power consumption of DSPs: the so-called inter-instruction effects. In general, the energy consumption of a DSP instruction depends on the previously executed instructions, what can be explained by means of Figure 1 and Figure 2.



**Figure 1: Sequential execution of two different DSP instructions**



**Figure 2: Sequential execution of two identical DSP instructions**

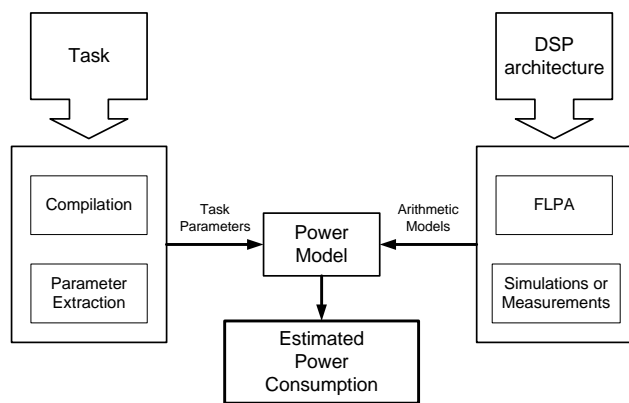
At a certain stage of a processors pipeline, instruction words are transferred from the program cache into a register in the DSP core for further processing. Figure 1 shows the situation that an ADD (addition) instruction word replaces a MUL (multiplication) instruction word in cycle 2. The numbers shaded by gray boxes show the bits in the register that switch their state in this case. In this example a Hamming distance (number of different bits of these two instruction words) of eight ( $H_d=8$ ) is resulting. Of course the sequence of two identical instructions causes no switching activity ( $H_d=0$ ), (Figure 2). Effects like this occur in many stages of a processors pipeline and as a result of these effects the energy consumption of a DSP instruction obviously depends on the previously executed instruction [4]. The Instruction-Level Power Analysis methodology allows covering such inter-instruction effects by measuring the energy consumption of groups of DSP instructions, but the huge number of possible combinations makes this approach very complex. The effort will even grow, if Very-Long-Instruction-Word (VLIW) architectures shall be modeled due to their increasing word length and their ability to issue several instructions in parallel.

An alternative approach for power estimation is the Functional-Level Power Analysis (FLPA) methodology. This methodology has been introduced in [5] and was first applied in [7] to a digital signal processor. Here, a refined extension of this methodology is presented in

order to model the TriMedia architecture sufficiently accurate. This architecture features several specific effects such as a strong dependency of the instruction related power consumption on the so-called issue slot where an instruction is performed on. This requires an extension of classical (pure) FLPA modeling techniques. In the next section the basics of classical FLPA modeling are shortly reviewed.

### III. FUNCTIONAL-LEVEL POWER ANALYSIS (FLPA)

The basic principle of the FLPA methodology is depicted in Figure 3. In a first step the DSP architecture is divided into functional blocks like fetch unit, processing unit, internal memory and others like the clocking system. By means of simulations or measurements it is possible to find an arithmetic model for each block that determines its power consumption in dependency of certain parameters. These parameters are for example the degree of parallelism, the rate with which the internal memory is accessed or the clock frequency. Most of these parameters can be automatically determined by a parser which analyzes the assembler file of a program code. The total power consumption is then given as the sum of the power consumption of each functional block.



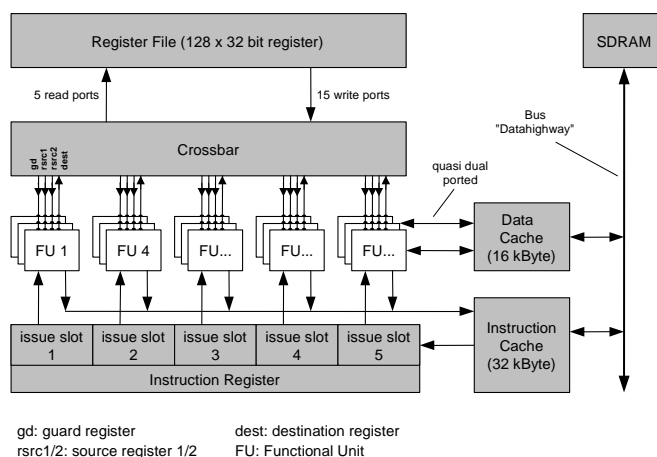
**Figure 3: The basic FLPA principle**

The left side of Figure 3 depicts the process of extracting parameters from a program which implements a task. After compilation it is possible to extract the task parameters from the assembler code. Further parameters can be derived from a single execution of the program (e.g. the number of required clock cycles). These parameters are the input values for the previously determined arithmetic models. Thus, an estimation of the power consumption for a given task can be computed. This approach is applicable to all kinds of processor architectures. Furthermore, FLPA-modeling can be applied to a processor with moderate effort and no detailed knowledge of the processors architecture is necessary.

### IV. ARCHITECTURE OF THE TRIMEDIA DSP

The digital signal processor TriMedia (available in several versions, e.g. TM1000, TM1300, etc.) [6, 9] is targeted for multimedia applications. The processor is based on a Very-Long-Instruction-Word (VLIW) -architecture. Such a VLIW-processor consists of parallel functional units, which are controlled by a long instruction word. In contrast to a Single-Instruction-Multiple-Data (SIMD) -architecture, where a single instruction word controls multiple identical parallel units, a VLIW-processor features several parallel units which are not identical and one section of the instruction word is assigned to each unit. By this, each instruction word controls several functional units.

A detailed architecture block diagram of the TriMedia is depicted in Figure 4.



**Figure 4: Principle of the VLIW-architecture of the TriMedia DSP**

Data as well as instructions can access the CPU via the bus and they are stored separately in data and instruction caches. After the decoding of the instructions the instruction word is stored in the instruction register.

According to the TriMedia architecture this is separated in five sections, so-called issue slots. Each slot contains an own instruction which in each clock cycle calls the belonging functional unit.

The TriMedia provides 28 functional units. For each instruction only a limited number of functional units and only a restricted number of possible mappings to the issue slots are possible. For example, DSP-multiplications can only be performed using the slots two and three. If multiple multiplications have to be performed in a series, these can be parallelized only according to the number of available functional units. An overview of the available number of functional units and their possible mappings to the issue slots is given in Table 1.

All instructions of the TriMedia apart from the load/store-instructions are register-register-instructions. They get their source data from one or two source registers and write the results back to a target register.

Therefore, the instruction set can be classified into load/store and arithmetic instructions. The arithmetic instructions can be further classified into several classes.

The instruction set contains simple RISC instructions as well as complex instructions which can be composed by up to eleven simple RISC instructions. These are adapted to the architecture and the targeted multimedia applications and therefore provide maximum performance.

The TriMedia supports the conditional execution of instructions which is also called guarding. Here, the execution of a command is controlled by a guard register. Hence, the generation of branches within the assembler code is avoided.

Functional Unit	Nr.	Latency	Recovery Clocks	Slot Assignment				
				1	2	3	4	5
Constant	5	1	1	•	•	•	•	•
Integer ALU	5	1	1	•	•	•	•	•
Load/Store	2	3	1				•	•
Cache	1	3	1					•
DSP ALU	2	2, 3	1	•		•		
DSP MUL	2	3	1		•	•		
Shifter	2	1	1	•	•			
Branch	3	3	1		•	•	•	
Int/Float MUL	2	3	1		•	•		
Float ALU	2	3	1	•			•	
Float Compare	1	1	1			•		
Float sqrt/div	1	17	16		•			
<b>Quantity</b>	28	-	-	5	7	7	5	4

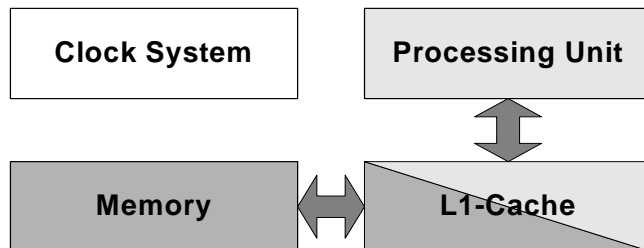
**Table 1: Functional units and possible slot mappings of the TriMedia architecture (here: model TM1300)**

## V. FLPA-MODELING OF THE TRIMEDIA DSP

The TriMedia architecture has been separated into four different modules (clock system, processing unit, external memory system and L1-cache, see Figure 5), being modeled by three FLPA functions. The L1-cache is modeled together with the processing unit in case of cache hits and together with the memory system in case of cache misses.

According to the FLPA modeling technique, arithmetic models describing the power consumption of a functional block can be found by means of simulations and measurements. Therefore, it is necessary to stimulate each block separately. This can be achieved by executing different parts of assembler code, which will be called scenarios. For the TriMedia architecture a script

based framework has been implemented running a variety of test scenarios with automatically performed measurements of the power consumption (respectively the voltage measurement on a shunt resistor on the processor board). The captured results are transferred self-controlled to a database where the results for all measurements are collected.



**Figure 5: Separation of the TriMedia TM1300 architecture into functional blocks**

Since the processing unit features five different issue slots it has been inspected which differences arise if an instruction is performed on different issue slots. The results show that there are significant differences up to 20%. Table 2 lists some examples. Here, it has to be regarded that only constant and integer ALU instructions can be performed on all issue slots (e.g. the instruction `dspuadd` can only be performed on issue slot 1 and 3, see also Table 1).

Exemplary Instructions	Issue slot dependent energy [nJ]				
	1	2	3	4	5
<code>iadd</code>	4.63	4.77	4.84	5.18	5.26
<code>iimm</code>	4.86	5.21	5.18	5.86	5.53
<code>dspuadd</code>	4.57	-	4.95	-	-
<code>funshift1</code>	4.55	4.75	-	-	-
<code>ufir8uu</code>	-	5.13	5.08	-	-

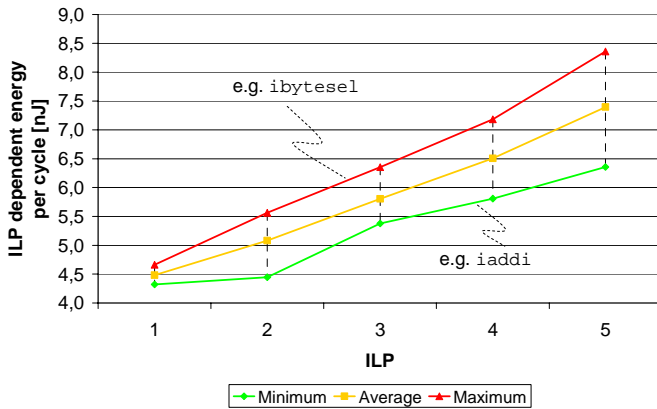
**Table 2: Issue slot dependent energy per instruction for a single instruction per cycle (ILP = 1)**

The second investigation concerns the ILP (instruction level parallelism) dependency of each instruction. The influence on the total energy per clock cycle has been analyzed for an ILP being increased stepwise from one (only one issue slot is utilized) to five (all slots are utilized). In Table 3 the ILP dependent energy per clock cycle is listed for several instructions. Again it has to be regarded, that some instructions can only be performed in one clock cycle on specific issue slots (e.g. `dspuadd` on issue slot number 1 and 3). In order to create simple modeling functions covering all possible ILP values the measured results have been extrapolated also for those ILP values which cannot be achieved in reality. These values are shaded in gray in Table 3.

Exemplary Instructions	ILP dependent energy [nJ]				
	1	2	3	4	5
iadd	4.63	4.93	6.25	6.80	7.57
iimm	4.86	5.80	6.68	8.05	9.06
dspuadd	4.57	5.47	6,61	6,67	9,78
funshift1	4.55	5.49	7,16	7,22	10,70
ufir8uu	5.13	6.57	8,02	8,52	12,34

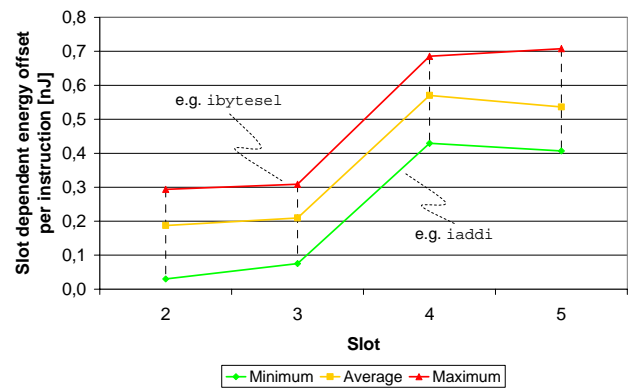
**Table 3: ILP dependent energy per cycle (values shaded with gray are extrapolated, since these instructions are only available on two issue slots)**

In Figure 6 the ILP dependent behavior of the energy conversion per clock cycle for integer ALU instructions is visualized. This depiction shows that on the one hand the processor features a linear dependency of the energy from the ILP (averaged over all ALU instructions). On the other hand the processor shows for some specific but frequently used instructions (e.g. iaddi) a non-linear dependency of the energy from the ILP. The other functional units feature comparable results.



**Figure 6: ILP dependent energy per cycle for integer ALU instructions**

In Figure 7 the so-called slot dependent energy offset is depicted over the different slots for integer ALU instructions. The results are stored in form of offset values which have to be added on top of the base ILP energy values. Only those curves for the maximum (corresponding to instructions like ibytesel), the minimum (corresponding to instructions like iaddi) and the average energy conversion are depicted. Concrete energy offset values for some exemplary instructions are given in Table 4.



**Figure 7: Slot dependent energy offset per instruction for integer ALU instructions**

Exemplary Instructions	Issue slot dependent energy offset [nJ]			
	2	3	4	5
iadd	0.14	0.21	0.55	0.63
iimm	0.35	0.32	0.99	0.66
dspuadd	-	0.38	-	-
funshift1	0.20	-	-	-
ufir8uu	0.00	-0.05	-	-

**Table 4: Issue slot dependent energy offset per instruction**

Depending on the instruction which is performed a difference of up to 55 % for the energy occurs (fdiv = 3.64 nJ to umul = 5.66 nJ). Regarding at the same time the issue slot dependency, this results in a strong dependency of the complete power respectively energy model on the specific instruction and issue slot distribution of each application. This has to be regarded in a sufficiently accurate TriMedia power modeling. It is one of the main differences compared to "classic" FLPA-models (e.g. in [3, 7]) and specific for the TriMedia DSP architecture. In [1] it has been shown that modeling the TMS320C6416 architecture it is sufficiently accurate to model it on a higher level assuming always typical instruction mixes and regarding mainly task specific parameters like the achieved ILP. This is not sufficient for the TriMedia. One reason for this different behavior is that the C6416 architecture can be divided into more fine grained functional blocks, because of more detailed simulation and run time reports. Another reason is that the C6416 features at maximum eight different instructions per cycle whereas the TriMedia architecture features only five issue slots providing a non-symmetric power behavior. Concluding from this, it is important to model the TriMedia architecture differently than using the "classic" FLPA approach.

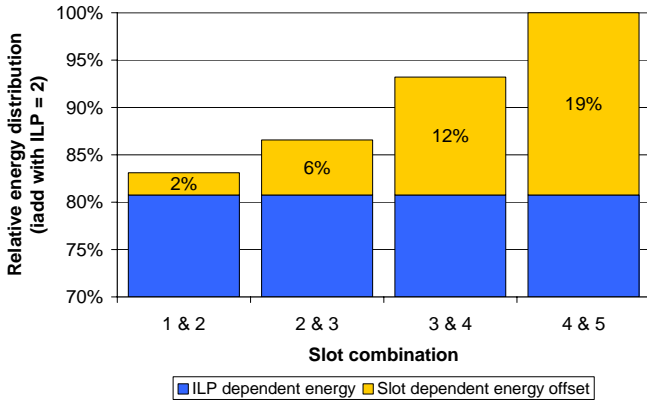
A simple example demonstrates the influence of the slot dependent energy offset on the total energy conversion. In Figure 8 the results for a clock cycle in which two iadd-instructions are performed in parallel are de-

pictured. This corresponds to an ILP value of two. Several different mappings of this instruction pair on the available issue slots are possible. For example, this instruction pair can be mapped on issue slot number 1 & 2 resulting in the lowest possible energy conversion. The maximum energy conversion is achieved if these two instructions are performed on issue slot 4 & 5. This example demonstrates that in the presented model the resulting energy conversion is composed of two parts:

- the ILP dependent base energy value for each instruction  $W_{ILP}$ ,
- the slot dependent energy offset which has to be added on top of the base value  $W_{Slot}$ .

Furthermore, the example shows that depending on the resulting issue slot distribution energy conversion differences up to 17 % and more occur.

The energy values  $W_{ILP}$  and  $W_{Slot}$  are stored in a database in order to support the instruction refined FLPA model. The total amount of data which has to be stored can be computed as follows: For each instruction out of the instruction set of the processor, the base energy values (for an ILP of one to five) and the issue slot dependent offsets (four additional offset values per instruction) have to be stored. In case of the TriMedia (here: TM1300) the database stores 169 instructions times nine values.



**Figure 8: Influence of the slot dependent energy offset on the total energy for different exemplary slot combinations**

The resulting model for the energy conversion of the TriMedia consists of three components: The energy of

- the processing unit  $W_{PU}$  (including the Level 1 cache access in case of cache hits),
- the external memory  $W_{Mem}$  (including the Level 1 cache access in case of cache misses),
- the clock system  $W_{CS}$ .

Hence, the total energy conversion for the TriMedia  $W_{TM}$  amounts to

$$W_{TM} = W_{PU} + W_{MEM} + W_{CS} . \quad (1)$$

The total energy of the processing unit can be separated into the ILP dependent energy and the Slot dependent energy offset totalized over all clock cycles  $C$

$$W_{PU} = \sum_{i=1}^C (W_{PU,ILP}(i) + W_{PU,Slot}(i)) . \quad (2)$$

An overview of the used parameters and functions is given in Table 5.

The ILP dependent energy per clock cycle results in the average energy of a specific instruction mix for each clock cycle. For each instruction issued on the  $S=5$  slots the ILP dependent energy value  $W_{ILP}$  is taken from the database, totalized over the slots and finally normalized with the corresponding ILP value of the inspected cycle:

$$W_{PU,ILP}(i) = \frac{1}{ILP(i)} \cdot \sum_{j=1}^S W_{ILP}(OP(i, j), ILP(i)) \quad (3)$$

In order to compute the total slot dependent energy offset  $W_{PU,Slot}$  the individual energy offsets  $W_{Slot}$  for each instruction and slot are also taken from the database and totalized

$$W_{PU,Slot}(i) = \sum_{j=2}^S W_{Slot}(OP(i, j), j) . \quad (4)$$

The energy conversion  $W_{Mem}$  of the external memory shows a linear behavior to the measured stall cycles and the clock frequency

$$W_{MEM} = (c \cdot \chi + d \cdot \delta) \cdot f_{Clock} . \quad (5)$$

The required task-specific parameters  $\chi$  (instruction cache stall cycles) and  $\delta$  (data cache stall cycles) can be acquired by simulating the given task once and analyzing subsequently the so-called stat-file produced by the TriMedia simulator with a parser.

The total energy conversion of the clock system can be described in dependency on the clock frequency as

$$W_{CS} = (a \cdot f_{Clock} + b) . \quad (6)$$

parameter / function	description
$W_{ILP}(op, ilp)$	energy of instruction $op$ for an instruction level parallelism of $ilp$ (stored in database)
$W_{Slot}(op, j)$	energy offset of instruction $op$ for the execution on slot $j$ (stored in database)
$ILP(i)$	instruction level parallelism in cycle $i$
$OP(i, j)$	instruction in cycle $i$ and issue slot $j$
$C$	total number of instruction cache cycles
$S$	number of execution slots ( $S = 5$ in case of the TriMedia TM1300)
$\chi, \delta$	instruction and data stall cycles
$f_{Clock}$	clock frequency
$a, b, c, d$	coefficients for polynomials

**Table 5: Parameter list for the TriMedia TM1300 model functions**

The calculation of the energy conversion of the processing unit  $W_{PU}$  for a single cycle  $i$  and an exemplary instruction mix with an ILP of four is given in the following. Such an instruction mix for a specific cycle can be extracted from the assembler code. Compare to Table 3 and 4 for further details.

Slot	1	2	3	4	5
Instruction	iadd	ufir8uu	iadd	nop	iimm
$W_{ILP}(OP, 4)$	6.80	9.47	6.80	0	8.05
$W_{Slot}(OP, j)$	0	0	0.21	0	0.66

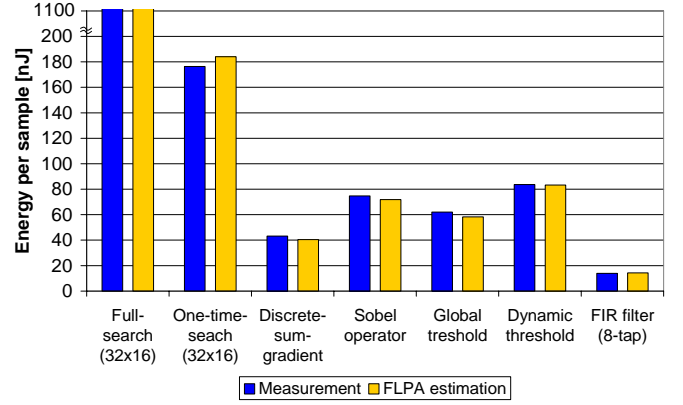
$$\begin{aligned}
 W_{PU,ILP}(i) &= \frac{1}{4} \cdot \sum_{j=1}^5 W_{ILP}(OP(i, j), ILP(i)) \\
 &= \frac{1}{4} \cdot \left( W_{ILP}(iadd, 4) + W_{ILP}(iadd, 4) + \right. \\
 &\quad \left. W_{ILP}(ufir8uu, 4) + W_{ILP}(iimm, 4) \right) \\
 &= \frac{1}{4} \cdot (6.80 + 6.80 + 9.47 + 8.05) \text{ nJ} \\
 &= 7.78 \text{ nJ}
 \end{aligned}$$

$$\begin{aligned}
 W_{PU,Slot}(i) &= \sum_{j=2}^5 W_{Slot}(OP(i, j), j) \\
 &= \left( W_{Slot}(ufir8uu, 2) + W_{Slot}(iadd, 3) + \right. \\
 &\quad \left. W_{Slot}(nop, 4) + W_{Slot}(iimm, 5) \right) \\
 &= (0 + 0.21 + 0 + 0.66) \text{ nJ} \\
 &= 0.87 \text{ nJ}
 \end{aligned}$$

$$\begin{aligned}
 W_{PU}(i) &= W_{PU,ILP}(i) + W_{PU,Slot}(i) \\
 &= (7.78 + 0.87) \text{ nJ} \\
 &= 8.65 \text{ nJ}
 \end{aligned}$$

## VI. BENCHMARKING OF THE TRIMEDIA-FLPA-MODEL

For the evaluation of the FLPA-model the estimated power consumption respectively energy conversion was compared to measured values for a variety of digital signal processing tasks. Regarding the distribution of the energy conversion for the single modeled blocks of the processor it is obvious that the part of the total energy per sample associated with the clock system is a constant offset for each task. Therefore, so-called *absolute energy values* (incl. the clock system) and so-called *differential energy values* (without the clock system) have to be differentiated. The comparison of estimated and measured values shows a maximum error of less than 6 % for the absolute energy conversion and 9 % for the differential energy conversion. The differential energy conversion for several algorithms is presented in Figure 9.



**Figure 9: FLPA estimation results and measurements for the TriMedia architecture**

The error between measurement and estimation result is much smaller than the maximum energy dynamics between different applications. From Figure 9 it can be seen that the difference between the energy per output sample (one output motion vector per pixel) for a computational intensive full-search motion estimation algorithm and the energy per output sample for a simple eight-tap FIR filter is in a range of a factor of 80. Therefore, the estimation error is much smaller than the energy dynamics of the processor.

## VII. POWER AWARE CODE OPTIMIZATION

One interesting application for power estimation approaches (especially for approaches which are based on high level estimations) is that they can be applied for power aware code optimizations during the design phase of a new application. For actual DSP architectures a variety of different code optimization techniques ranging from loop unrolling to the application of restricted pointers or the application of DSP specific custom instructions are available. It has been proven that these techniques can be applied successfully in order to increase the throughput of an application. It is desirable to give the software designer a possibility to estimate the consequences of his code optimizations on the resulting power consumption. Such a possibility is given applying approaches like the FLPA modeling presented before.

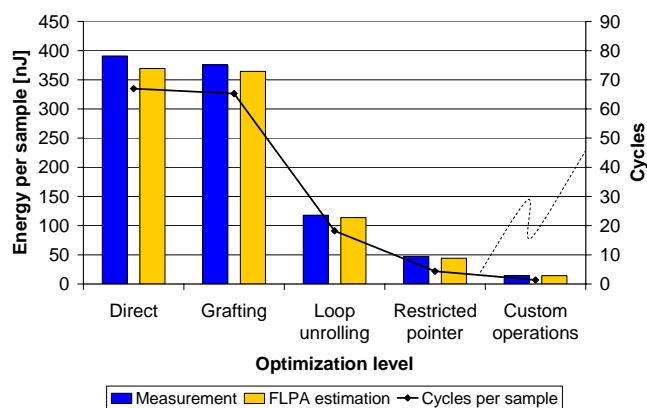
In order to evaluate the estimation accuracy while optimizing the code the following example will be discussed:

A basic FIR filter featuring eight taps has been implemented using straight forward C-code (this reference optimization example has been taken from [9]). Then, this code has been optimized in four steps. From step to step the following code optimizations have been included

1. grafting,
2. loop unrolling for integer wise memory access,
3. additional use of restricted pointer,
4. additional use of custom operations.

By application of these optimization techniques the number of required clock cycles for filtering a data block of 400 samples can be reduced from 26788 cycles for the straight forward implementation to 547 cycles for the fully optimized implementation using custom operations.

Figure 10 depicts the corresponding energy conversion per output sample (per filtering result) over the single optimization steps. In parallel the estimated energy conversion values are depicted. It can be seen that the maximum error is less than 6 %. The total dynamics of the energy conversion due to the complete code optimization amounts to about 2800 %. Therefore, this estimation provides a good accuracy which proves that such a power/energy estimation technique is a helpful means for performing power aware code optimization even in an early stage of the design of an application.



**Figure 10: Influence of DSP suited code optimization on the energy conversion, comparison to estimation results**

### VIII. SUMMARY

In this paper the concept of so-called Functional-Level Power Analysis (FLPA) has been extended and refined in order to model the power consumption of a TriMedia VLIW DSP-architecture. The appropriate separation of the processor architecture into functional blocks has been demonstrated. The power consumption of these blocks has been described in terms of parameterized arithmetic models. It has been shown that a sufficiently accurate FLPA model for this architecture has to include the strong issue slot and instruction dependency of the power consumption. In order to perform efficient power estimation a code parser has been implemented, which allows automatic analysis of the assembler codes and

profiling files of applications to be performed on the TriMedia. This parser yields the input parameters of the arithmetic models like e.g. the achieved degree of parallelism, the type and number of memory accesses or the distribution of instructions. An evaluation of this estimation technique has been performed applying a variety of basic tasks of digital signal processing. Resulting estimated energy figures were compared to physically measured values. A resulting maximum estimation error of less than 6 % for absolute energy conversion and 9 % for differential energy conversion is achieved. The application of this methodology allows to efficiently evaluate different parameter settings of a programmable processor, different coding styles, compiler settings, algorithmic alternatives etc. concerning the resulting power consumption. This has been proven also by analyzing an FIR filter example where a basic code has been optimized in several steps. For this optimization process the FLPA based estimation yields good estimation results describing the effect of each optimization step. This optimization example demonstrates that such an FLPA-based power estimation technique can be effectively leveraged within power-aware software development for DSPs.

### REFERENCES

- [1] Blume, H.; Schneider, M.; Noll, T. G.: "Power Estimation on a Functional Level for Programmable Processors", Proc. of the TI Developers Conference 2004, Houston, Texas, February 2004
- [2] Brooks, D.; Tiwari, V.; Martonosi, M.: "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", Proc. of the ISCA, 2000, pp. 83-94
- [3] Laurent, J.; Julien, N.; Senn, E.; Martin, E.: "Functional Level Power Analysis: An Efficient Approach for Modelling the Power Consumption of Complex Processors", Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04), Munich, 2004
- [4] Marwedel, P.: "Fast, predictable and low-energy memory references through memory-architecture aware compilation", Proceedings of the DSP Design Workshop 2003, Dresden, 25.11.2003
- [5] Qu, G.; Kawabe, N.; Usami, K.; Potkonjak, M.: "Function Level Power Estimation Methodology for Microprocessors", Proc. of the Design Automation Conference 2000, pp. 810-813
- [6] Rathnam, S.; Slavenburg, G.: "An Architecture overview of the programmable multimedia processor TM1000", Proc. Compeon, IEEE CS Press, 1996, pp. 319-326
- [7] Senn, E.; Julien, N.; Laurent, J.; Martin, E.: "Power Consumption Estimation of a C Program for Data-Intensive Applications", Proc. of the PATMOS Conference 2002, pp. 332-341
- [8] Tiwari, V.; Malik, S.; Wolfe, A.: "Instruction Level Power Analysis and Optimization of Software", Journal of VLSI Signal Processing 1996, pp. 1-18
- [9] TriMedia TM1300, Digital Signal Processor, Datasheet, Philips Semiconductors, <http://www.philips.com>; ("DATA BOOK TM1300: Media Processor")