Functional Level Power Analysis: An Efficient Approach for Modeling the Power Consumption of Complex Processors

Johann Laurent, Eric Senn, Nathalie Julien, Eric Martin

Functional Level Power Analysis: An Efficient Approach for Modeling the Power Consumption of Complex Processors

Abstract

A high-level consumption estimation methodology and its associated tool, SoftExplorer, are presented. The estimation methodology uses a functional modeling of the processor combined with a parametric model to allow the designer to estimate the power consumption when the embedded software is executed on the target. SoftExplorer uses as input the assembly code generated by the compiler; its efficiency is compared to SimplePower's approach. Results for different processors (TI C62, C67, C55 and ARM7) and for several DSP applications provide an average error less than 5%. The accuracy and the rapidness of the estimation allow using SoftExplorer for efficiently guiding the designer in choosing the more appropriate processor for his application.

1. Introduction

The Systems-On-Chip must respect critical constraints in time, area and consumption; therefore the designer need to validate his choices early in the flow, and has to characterize the system improvements through accurate estimates. As the software part is growing in the applications, its impact on the consumption is also becoming more important. Although many researches have developed methodologies to estimate the software consumption, few of these are associated with an estimation tool. Some tools use a fine-grain representation of the processor to obtain a cycle-accurate estimation: SimplePower [1] and Wattch [2] need a RTL representation of the architecture to allow the power characterization. This low-level modeling often implies that the time required to characterize a processor and to estimate an application becomes too important to give an efficient feedback in system design. Another method, the Instruction Level Power Analysis (ILPA) consists in estimating the cost of each instruction of the assembly code [3]; the estimation time is strongly reduced but elaborating the processor model can be very time consuming for complex processors [4]. JouleTrack is the only known tool based on this approach [5]. To characterize VLIW processors, an ILPA extension, based on a fine knowledge of the processor architecture, decomposes the instruction into functionalities [6]; the pipeline stalls are hardly modeled and no tool is currently available. Furthermore, the necessary time is still important since 108 days are necessary to model a VLIW processor (Lx). We propose here to increase the abstraction level, by combining a functional level model of the processor requiring only a coarse-grain knowledge on its architecture and a parametrical model of the code; the aim is to obtain a good tradeoff between the estimation accuracy and the model complexity.

Section 2 introduces the processor modeling methodology illustrated by examples for different processors. Then, in section 3, the estimation methodology is proposed with several results for classical digital signal applications. Section 4 presents a brief comparison between SoftExplorer's and SimplePower's efficiency. Finally, section 5 illustrates the interest of this high-level approach for the Co-Design before concluding.

2. Power Model of the Processor

2.1 Functional Level Power Analysis

To estimate the power/energy consumption of an application executed on a processor, we first need to realize the power characterization of the target. The Functional Level Power Analysis (FLPA) methodology can be started from only a simple block diagram of the architecture. As sketched on Fig. 1, the first step consists in dividing the processor architecture into different functional blocks and sub-blocks, to cluster the components that are concurrently activated when a code is running. Then, the relevant consumption parameters are selected as the significant links between these blocks. There are two types of parameter: algorithmic parameter values depend on the executed algorithm (typically the cache miss rate) and architectural parameter values depend on the processor configuration settled by the designer (typically the clock frequency). Table 1 presents the global parameter set of our generic power model, which is the common base for each specific processor set.

The second step is the characterization of the processor power consumption when the parameters vary. These variations are obtained by using some elementary assembly programs (called scenario) elaborated to stimulate each block or sub-block separately. Characterization can be performed either by measurements or by simulation; for our

part, the supply current is measured on evaluation boards. Finally, a curve fitting of the graphical representation allows determining the consumption laws by regression.

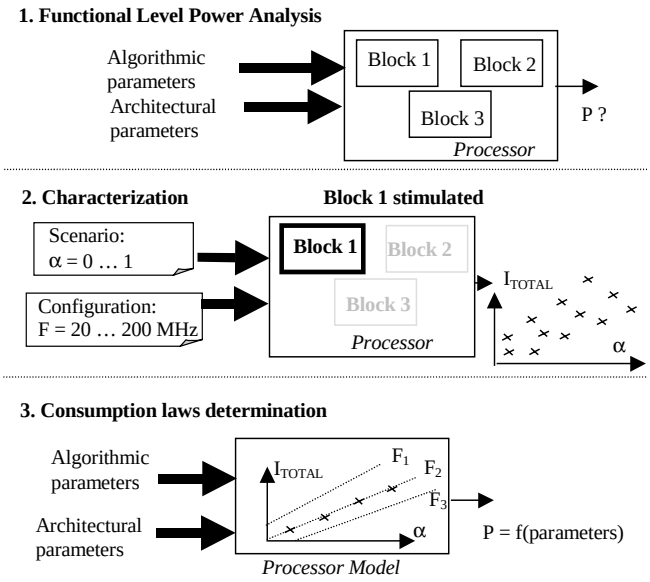

Figure 1. Processor Modeling Methodology

Table 1. Generic power model parameters

SOFTWARE PARAMETERS		
Algorithmic	Name	Description
	α	Parallelism rate
	β	Processing unit rate
	γ	Cache miss rate
	τ	External memory access rate
	ε	DMA access rate
Architectural	*W*	Data width transferred by the DMA
	F	Clock frequency in MHz
	MM	Memory mode
	DM	Data placement in memory
	PM	Power management (units in sleep mode)

2.2 FLPA Examples

The FLPA has been applied to different processors in order to demonstrate that our methodology allows characterizing VLIW processors (TI C62 and C67), low power processors (TI C55), and general processors (ARM7TDMI) as well.

2.2.1 TI C6x power model

The TI C62 and C67 processors are complex Digital Signal Processors, containing a deep pipeline and allowing to execute up to 8 instructions in parallel. The internal program memory can be used in four different memory modes (Mapped, Cache, Freeze and Bypass). A DMA (Direct Memory Access) and/or an EMIF (External Memory InterFace) realize the external memory accesses. The difference between these two processors is that the C62 is a fixed-point architecture and the C67 is a floating point one. As these processors have a similar architecture, the cutting out in functional blocks (in Fig. 2) and the selected parameter set (Table 2) are shared; however, the consumption laws will differ. Indeed, the processor used onto our development boards have neither the same technological process (0.25µm for the C62 and 0.18µm for the C67) nor the same supply voltage (2.5V for the C62 and 1.8V for the C67). Therefore these two processors will not have the same dynamic and static power consumption.

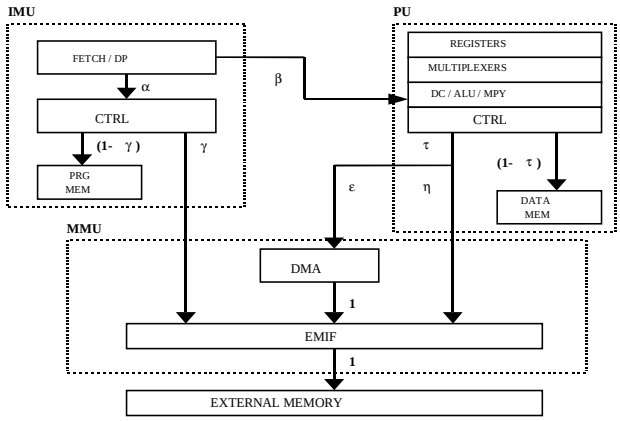

Figure 2. TI C6x Functional Level Power Analysis

The algorithmic parameters are then defined for the C6x:

- α represents the parallelism rate i.e. the number of instructions fetched at each clock cycle. When 8 instructions are fetched simultaneously then $\alpha = 1$; if only one instruction is fetched then $\alpha = 1/8$.
- β represents the processing unit rate. When 8 parallel instructions are executed then $\beta = 1$. As the No Operation instruction does not involve any processing unit, if 2 NOP are used then $\beta = 6/8 = 0.75$.
- γ represents the cache miss rate.
- ε represents the external data rate accessed by the DMA.
- τ represents the external data access rate. For the TI C6x processors, each external access leads to pipeline stalls; therefore, we can include this parameter into a new parameter, the Pipeline Stall Rate (PSR) together

with the internal memory bank conflicts, obtained from the architectural parameter DM (cf. Table 2).

As illustration, Table 3 (at the end of the paper) presents the complete power model for the Texas Instruments TMS320C6201 that has already been extensively reported [7].

2.2.2 TI C55 power model

The TI C55 is a low power processor with a fixed-point architecture that can only execute two parallel instructions; however, it only fetches one instruction at each clock cycle, and pipeline stalls never occur. Another characteristic of this processor is the possibility to automatically idle some parts of its architecture if unused. Its internal program memory can be used in the same 4 modes as the TI C6x and it also contains a DMA and an EMIF. As the C55 architecture is less complex than the C6x architecture, its power model will have less parameters (cf. Table 2).

2.2.3 ARM7TDMI power model

The ARM7TDMI is the simplest processor that we have modeled yet. It has a scalar architecture and its internal program memory can be used in 3 modes (Mapped, Cache and Bypass). Previous works on the StrongArm have established that the power consumption essentially depends on the clock frequency and the supply voltage [5]. Our own consumption measurements on the ARM7TDMI have fully validated this trend: the power consumption variations corresponding to various programs are under 8% of the global consumption. No algorithmic parameter is then required to model the ARM7, as represented in Table 2.

Table 2. Specific parameter set for various processors

SOFTWARE PARAMETERS		C62	C67	C55	ARM7
Algorithmic	α	X	X		
	β	X	X	X	
	γ	X	X	X	
	τ	*in PSR*	*in PSR*		
	ε	X	X	X	
	PSR	X	X		
Architectural	*W*	X	X	X	
	F	X	X	X	X
	MM	X	X	X	X
	DM	*in PSR*	*in PSR*		
	PM			X	

The complete modeling time was about 30 days for the TI C6x and 15 days for the ARM7. With an ILPA approach, Bona and al. have characterized a VLIW processor (Lx) in 108 days [6].

3. Power/Energy Estimation Methodology

As the architectural parameters are settled by the designer, once the processor model is known, only the algorithmic parameters have to be calculated for each software execution in order to compute the estimated power consumption. Our automatic tool, SoftExplorer, has been developed to estimate the power/energy consumption of an application in few seconds. Through a graphic interface, the user must provide some architectural parameters: the clock frequency F, the memory mode MM and eventually the data width W if the DMA is used. He also has to quantify some algorithmic parameters like the pipeline stall rate and the cache miss rate; such data are usually available through a dynamic profiling of the application in the processor environment. From the assembly code, SoftExplorer computes the other algorithmic parameter values, the parallelism rate α and the processing unit rate β, as follows:

$$\alpha = \frac{NFP}{NEP} \times (1 - PSR) \quad \beta = \frac{1}{NPU_{\max}} \frac{NPU}{NEP} \times (1 - PSR) \quad (1)$$

NFP and *NEP* respectively stand for the number of Fetch and Execute Packets. *NPU* is the number of processing units and *NPUmax* is the maximum number of processing units, which can be used in parallel. In the example for the TI C6x proposed on Fig. 3, the 8 instructions are fetched together then NFP=1. An Execute Packet clusters the parallel instructions so NEP=3 and as the load and store instructions also involve a processing unit, NPU=7. So, if PSR = 0, α and β are respectively equal to 0.33 and 0.29.

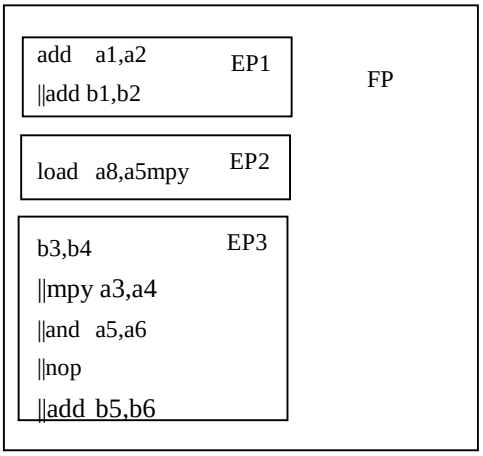

Figure 3. Algorithmic parameter computation for the C6x

To validate the estimation accuracy, several signal and image processing applications have been analysed: a Finite Impulse Response filter (FIR), a Discrete Wavelet Transform (DWT), an Enhanced Full Rate Vocoder for

GSM (EFR), and a MPEG-1 decoder (MPEG). The last two benchmarks are extracted from MediaBench. Table 4 presents the SoftExplorer power estimation results in Watt and the associated error between the physical measures and the estimates.

The average error between the estimates and the measurements, for all targets, is around 2.5%. The error for the ARM processor is slightly more important, due to a very simple power model. However, such results achieve the accuracy/complexity tradeoff required for high level design.

Furthermore, SoftExplorer computes the consumption for each loop and/or function of the program and provides a graphical representation in order to spot the power and/or energy critical parts of the application. For more details, this tool will be in demonstration at the DATE'04 University Booth.

Table 4. SoftExplorer power estimation results

		PROCESSORS			
		C55	**C62**	**C67**	**ARM7**
FIR	P (W)	0.46	2.36	0.59	0.22
	Error	*-1.5%*	*-2.8%*	*+1.4%*	*+4%*
DWT	P (W)	0.39	3.83	0.9	0.22
	Error	*+2.3%*	*+2%*	*+6%*	*+7%*
EFR	P (W)	0.43	2.62	0.95	0.22
	Error	*+2.1%*	*-0.6%*	*-1%*	*+3%*
MPEG-1	P (W)	0.4	5.59	0.93	0.22
	Error	*-1.6%*	*-4%*	*2.4%*	*+8%*
Maximum Error		*2.3%*	*-4%*	*6%*	*+8%*

4. SoftExplorer/SimplePower Comparison

We will compare the efficiency of our method with the SimplePower's approach on the SPEC-95 benchmarks (provided with SimplePower) and on other signal-processing applications.

4.1 SimplePower

SimplePower is an automatic tool developed at the Pennsylvania State University [1]. This tool computes the equivalent capacitance corresponding to the total capacitance switched during the program execution. The architecture characterization methodology is the following:

- First, for each class of functional unit (bit independent and bit dependent), the switched capacitance is computed as a function of previous and present input vectors.
- Then, one capacitance table is created for each unit.
- The last step consists in the dynamic profiling: for each clock cycle, the capacitance switched due to the executed instructions is computed. The total switched capacitance is computed as the capacitance sum for each cycle.

To model a processor with this methodology, a RTL representation of the architecture is compulsory; such an information is often unavailable with commercial processors. The architecture currently characterized is a modified MIPS IV (integer part only). However, the clock tree and the control unit are not included in the model, although the clock tree consumption is known as an important part of the total consumption in processors. Due to the RTL abstraction level, modeling another target would be very time consuming

SimplePower results always underestimate the consumption relatively to HSPICE simulations with a maximum error about 15% [1].

4.2 Comparison Results

SimplePower has been executed on a Ultra Sparc III+ (900MHz, 1Go of RAM) and SoftExplorer on a PC (Athlon 1GHz, 256Mo of RAM); a first computer benchmarking has been achieved to confirm that the workstation is always faster than the PC for every type of application.

Afterwards, power estimation has been performed on SPEC-95 benchmarks and typical signal and image processing applications (DSP) with both SimplePower and SoftExplorer. As the modeled processors are different (a modified MIPS-IV for SimplePower and TI C62, C67, C55 and ARM7 for SoftExplorer), it is not relevant to compare the estimates. Nevertheless, another indicator of the tool efficiency is the estimation time, corresponding to the total time necessary to compile, to profile and to estimate the benchmark consumption; results are proposed in Table 5.

Table 5. SimplePower/SoftExplorer estimation time

		SimplePower	SoftExplorer
SPEC-95	Bubble	35s	4s
	Hanoi	20s	3s
	Heap	9s	3s
	Matmult	1s	2s
	Perm	67s	4s
	Quick	6s	3s
	Test	<1s	2s
DSP	FFT 1024	17s	3s
	FIR 1024	4360s	3s
	LMS 1024	24728s	4s
	DWT 512x512	142267s	10s
	MPEG-1	10s	8s

SoftExplorer is generally faster than SimplePower especially for the classical signal-processing applications.

Indeed, SoftExplorer realizes a static trace of the application whereas SimplePower realizes a dynamic trace, increasing the estimation time when the execution time is high. On the other hand, this dynamic trace allows SimplePower to estimate all types of applications (control oriented or data oriented and also data dependant applications). Furthermore, it provides to the designer a more detailed information, cycle by cycle, allowing fine-grain code optimizations. Besides, as SoftExplorer uses a static trace of the application, a dynamic profiling must also be performed to determine the average execution time and then the global energy consumption.

Therefore, SoftExplorer is a power estimation tool intended to the system design assistance; it realizes the suitable tradeoff between the estimation accuracy and time in order to ensure a rapid and reliable feedback to the designer. It also provides a graphical representation of the power and energy consumption for each program loop and function. So, the designer can rapidly focus his optimization efforts onto the critical parts of the code according to the application constraints.

5. Application to the Co-Design

To be efficient, the system designer has to take into account the consumption constraints, in addition to the area and the execution time, as soon as possible in the design flow. Especially, he has to select the more appropriate target for his application. Here, we will show how our estimation tool can be helpful to realize the processor selection.

Table 7. Performance results for DSP applications

Application	Target	Texe (ms)	P (W)	E (mJ)
DWT	C55	4.615	**0.39**	**1.78**
	C62	**2.32**	3.83	8.88
	C67	**2.32**	0.9	2.09
EFR	C55	3.14	**0.43**	**1.37**
	C62	**4.05**	2.62	10.63
	C67	**4.05**	0.95	3.85
MPEG-1	C55	4.09	**0.4**	1.64
	C62	**0.0404**	5.59	0.23
	C67	**0.0404**	1.45	**0.038**

As we will concentrate on DSP applications, only three different targets are proposed: the TI C62, the C67 and the TI C55. Results in Table 7 confirm that the C55, a low power processor, is always the less power consumer. Concerning the execution time *Texe*, the C62 and the C67 are generally faster, because of the strong parallelism; one exception is for the EFR application, where many pipeline stalls are generated. As the degree of parallelism and the clock frequency are the same for the C62 and the C67, we can see that the execution time is therefore the same. On the other hand, the power consumption for these two processors differs because neither the technology nor the supply voltage are identical.

Finally, the resulting energy *E* presents a particular effect for the MPEG application. Indeed, although the C62 uses around 14 time more power than the C55, as its execution time is 100 less, finally the energy consumed by the C62 is 7 time lower than for the C55.

Actually, to compare the processor performances in the case of a real-time application like the MPEG decoder, the global energy must be considered, including idle periods. From the data throughput rate, we deduct the time constraint of the application $T_{constraint}$, representing the maximum execution time to avoid loss of information. Once the code executed, the processor is now assumed to be in idle mode up to the time constraint (Fig. 4).

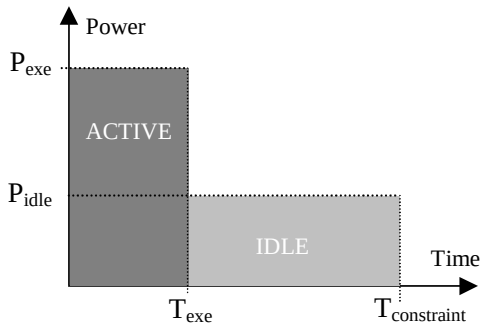

Figure 4. Real-time constrained application

Various clock frequencies have been applied; in each case, the resulting global energy E_{global} is computed through Equation 2, considering that P_{exe} is the power consumption during the execution period and P_{iddle} is the power consumption during the idle period.

$$E_{global} = P_{exe} \times T_{exe} + P_{idle} \times (T_{constra\,int} - T_{exe}) \quad (2)$$

This equation can be rewritten as equivalent to:

$$E_{global} = (KF + C) \times \frac{N}{F} + K'F \times (T_{constra\,int} - \frac{N}{F}) \quad (3)$$

In Equation 3, KF and K'F represent respectively the execution and dynamic power terms; the static part is C. N is the number of execution cycles and F the processor frequency. The results for several MPEG applications with different real-time constraints are given in Fig. 5: for a QCIF image (88x72) at 5 and 10 images/s and for a CIF image (352x288) at 25 and 30 images/s. The curve variations are fully explained by developing Equation 3 with a high frequency part proportional to F and a low frequency part proportional to 1/F.

Actually, the best clock frequency in term of energy is highly dependent on the real-time constraint and no a priori decision can be made. The energy variations always follow the same trend but, with the QCIF image, the more energy

efficient clock frequency (30 MHz for the QCIF-5 and 60 MHz for the QCIF-10) is within the C62 functional range. On the contrary, for the CIF image, the best clock frequency is 600 MHz; it implies that, for this application, the C62 maximum clock frequency must be chosen.

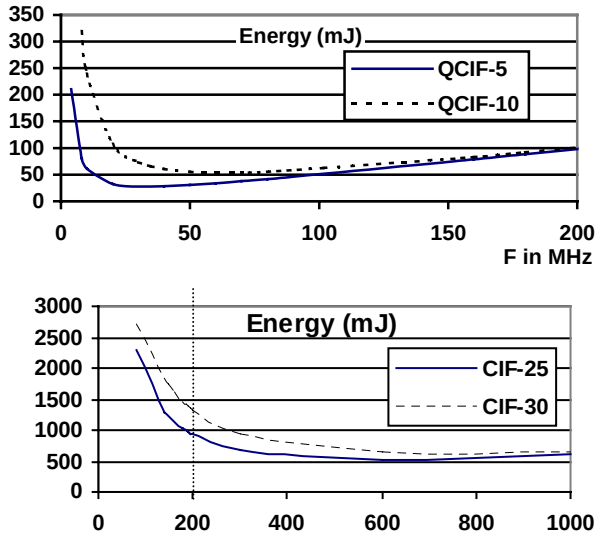

Figure 5. Energy characteristics for the MPEG decoder

As the power consumption is not a sufficient parameter for embedded applications, the energy consumption must also be considered. In these conditions, selecting the best target with the appropriate parameters is a very complex task; to guide the designer, a reliable estimation is then highly valuable.

6. Conclusion

A high-level estimation methodology and the associated tool SoftExplorer have been presented. The processor is modeled through a functional analysis and the software model is parametric: algorithmic parameters, depending on the code execution, are determined by the estimation process. The average error of the SoftExplorer results against the physical measurements is about 2.4% (except for the ARM7 with 5%). The accuracy and the rapidity of the tool make it convenient to give metrics in the Co-Design step, to choose both the target and the suitable algorithm and to return to the designer a reliable indication about algorithm optimizations. Future works will add other processor models to our library and further comparisons will be performed with existing tools and methods.

7. References

[1] W. Ye, N. Vijaykrishnan, M. Kandemir, M.J. Irwin "The Design and Use of SimplePower: A Cycle Accurate Energy Estimation Tool," in *Proc. Design Automation Conf.*, June 2000, pp. 340-345.

[2] D. Brooks, V. Tiwari, M. Martonosi "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *Proc. Int. Symp. on Computer Architecture*, June 2000, pp. 83-94.

[3] V. Tiwari, S. Malik, A. Wolfe "Power analysis of embedded software: afirst step towards software power minimization." IEEE Trans. on VLSI Systems, vol.2, N°4, Dec 1994, pp. 591-593.

[4] B. Klass, D.E. Thomas, H. Schmit, D.F. Nagle "Modeling Inter-instruction Energy Effects in a Digital Signal Processor", proc. of Int. Symp. On Circuits And Systems ISCAS, 1998.

[5] A. Sinha, A. P. Chandrakasan "JouleTrack - A Web Based Tool for Software Energy Profiling", in *Proc. DAC*, June 2001, p220

[6] A. Bona, M. Sami, D. Sciuto, C. Silvano, V. Zaccaria, R. Zafalon, "Energy Estimation and Optimization of Embedded VLIW Processors based on Instruction Scheduling*", in Proc. DAC'02*, June 10-14, 2002, New Orleans, USA, pp.886-891..

[7] self-reference

Table 3. Power Model of the TMS320C6201

Consumption laws $I_{TOTAL} = I_{MM} + I_{DMA}$	
Configuration Parameters : MM Memory Mode = {MAPPED, BYPASS, FREEZE, CACHE}; F Clock Frequency ; W Width of the data transferred by the DMA. **Algorithmic parameters** : α parallelism rate ; β processing unit rate ; γ Cache miss rate ; ε DMA access rate ; PSR Pipeline Stall Rate.	
$I_{MAPPED} = 5.21\alpha(1\text{-}PSR)F + 4.19F + 42.40\,\alpha(1\text{-}PSR) + 7.6 + 0.64\,\beta(1\text{-}PSR)F$	
$I_{BYPASS} = 9.87F + 39 + 0.64\,\beta(1\text{-}PSR)F$	
IF $\gamma > 0$	$I_{CACHE} = (8.55F + 184)\,\alpha(1\text{-}PSR)^{[-0.1249\mathrm{Log}(\gamma)\ -0.002276\ ]} + 0.64\,\beta(1\text{-}PSR)F$ $I_{FREEZE} = (9.07F + 118)\,\alpha(1\text{-}PSR)^{[-0.14\mathrm{Log}(\gamma)-0.0011]} + 0.64(1\text{-}PSR)F$
IF $\gamma = 0$	$I_{CACHE} = 4.36(1\text{-}PSR)F + 4.09F + 187.83(1\text{-}PSR) + 53.45 + 0.64\,\beta(1\text{-}PSR)F$ $I_{FREEZE} = 4.43(1\text{-}PSR)F + 4.72F + 203.1(1\text{-}PSR) - 38.52 + 0.64\,\beta(1\text{-}PSR)F$
IF $F \le F_{memory}$ $I_{DMA} = (0.077\,W\,F + 2.12F + 2.05W + 94.72)\,\varepsilon$ ELSE $I_{DMA} = (-0.083\,W\,F + 4.9F + 24.93W - 476.16)\,\varepsilon$	