Multi-Level power consumption modelling in the AADL design flow for DSP, GPP, and FPGA

Eric SENN, Johann LAURENT, and Jean-Philippe DIGUET

Université de Bretagne Sud, Lab-STICC,
CNRS UMR3192,
F-56321 LORIENT Cedex, France

Abstract. This paper presents a method that permits to estimate the power consumption of components in the AADL component assembly model, once deployed onto components in the AADL target platform model. This estimation is performed at different levels in the AADL refinement process. Multi-level power models have been specifically developed for the different type of possible hardware targets: General Purpose Processors (GPP), Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA). Three models are presented for a complex DSP (the Texas Instrument C62), a RISC GPP (the PowerPC 405), and a FPGA from Altera (Stratix EP1S80). The accuracy of these models depends on the refinement level. The maximum error introduced ranges from 70% for the FPGA at the first refinement level (only the operating frequency is considered here) to 5% for the GPP at the third refinement level (where the component's actual source code is considered).

1 Introduction

Originally coming from the avionic domain, AADL (*Architecture Analysis & Design Language*) is now commonly used as an input modelling language for real-time embedded systems [1, 2]. It allows the early analysis of the specification, the verification of functional and non functional properties of the system, and even code generation for the targeted hardware platform [3–5]. In the context of the European project SPICES (*Support for Predictable Integration of mission Critical Embedded Systems*) [6], our aim is to enrich the AADL component based design flow to permit energy and power consumption estimations at different levels in the refinement process. However, such early verifications are only possible if power estimations are completed in a reasonable delay. Only at this condition a fast and fruitful exploration of the design space is permitted.

Significant research efforts have been devoted to develop tools for power consumption estimation at different abstraction levels in embedded system design. A lot of those tools however work at the Register Transfer Level (RTL) (this is the case for tools like SPICE, Diesel [7] and Petrol [8]), at the Cycle Accurate Bit Accurate (CABA) level ([9, 10]), and a few tools at the architectural level (Wattch [11] and Simplepower [12]). Such approaches cannot be used at high

levels because simulation times at such low abstraction levels become enormous for complete and complex systems, like multiprocessor heterogeneous platforms.

In [13] and [14], the authors present a characterization methodology for generating power models within TLM for peripheral components. The pertinent activities are identified at several levels and granularities. The characterization phase of the activities is performed at the gate level and is used to deduce the power of coarse-grained activities at higher level. Again, applying such approaches for complex processors or complete systems is not doable. Instruction level or functional level approaches have been proposed [15–17]. They however only work at the assembly level, and need to be improved to take into account pipelined architectures, large VLIW instruction sets, and internal data and instruction caches.

We introduced the *Functional Level Power Analysis* (FLPA) methodology which we have applied to the building of high-level power models for different hardware components, from simple RISC processors to complex superscalar VLIW DSP [18, 19], and for different FPGA circuits [20]. In this paper we show how this modelling approach fits into the AADL design flow and how our power models, being interoperable, are used at different refinement levels. Section 2 presents the AADL component based design flow and the deployment of the Platform Independent Models (PIM) to obtain the Platform Specific Model (PSM) of the target. Section 3 presents the methodology for power estimations and the global power analysis of a complete system described with AADL. Section 4 presents the building of power models and define the three refinement levels where they can be used. The power models of the DSP TI C62, the GPP PowerPC 405, and the FPGA Altera Stratix EP1S80 are presented as examples. The accuracy of our power estimations is finally evaluated.

2 AADL design flow

Figure 1 presents the component based AADL design flow. The *AADL component assembly* model contains all the components and connections instances of the application, and references the implementation models of the components instances from the *AADL models library*. The *AADL target platform* model describes the hardware of the physical target platform. This platform is composed of at least one processor, one memory, and one bus entity to home processes and threads execution. The *AADL deployment plan* model describes the AADL-PSM composition process. It defines all the binding properties that are necessary to deploy the processes and services model of the component-based application on the target platform. All those models are combined to obtain the AADL-PSM model of the complete component-based system. The final implementation of the system is obtained afterward through model transformations and code generation.

The *Open Source AADL Environment Tool* (OSATE) [21] permits the specification of a complete system using AADL. It also permits to check some of its functional and non-functional properties. Those verifications rely on the use

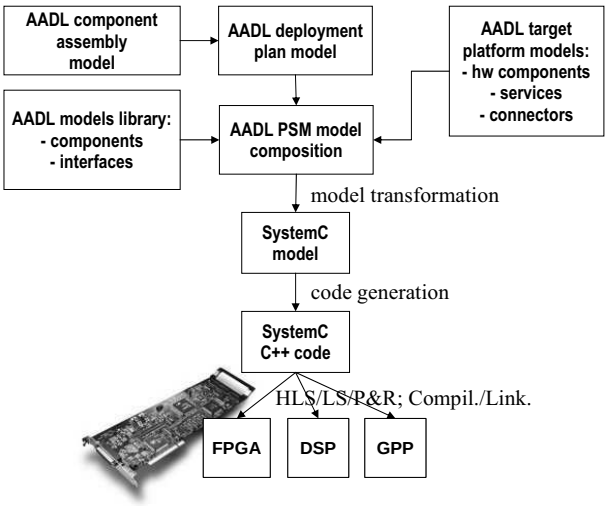

Fig. 1. AADL component based design flow

of different plug-ins included in the tool set. During the deployment, software components in the component assembly model are bound to hardware components in the target platform model [22]. According to the deployment plan model, OSATE scheduling analysis plug-in uses information embedded in the software components description to propose a binding for the system [23]. Figure 2 shows the typical binding of an application on a multiprocessor architecture. In this example, process `main_process` and its data block `data_b` are bound to the memory `sysRAM`. Threads `control_thread`, `ethernet_driver_thread` and `post_thread` are bound to the first general purpose processor `GPP1`. Thread `pre_thread` is bound to `GPP2`. Thread `hw_thread1` is, like `hw_thread2` a hardware thread. It will be implemented in the reconfigurable FPGA circuit `FPGA1`. One connection between `pre_thread` and `post_thread` has been declared using in and out `data ports` in the threads. This connection is bound to bus `sys_bus` since it connects two threads bound to two different components in the platform. Intra-component connections, like between threads `control_thread` and `ethernet_driver_thread`, do not need to be bound to specific buses. They will however consume hardware resources while being used.

In addition to communication buses, dedicated supply buses can also been declared. A *power analysis* command in the OSATE resources analysis plug-in permits to check if the power capacity of a supply bus is not exceeded. To do that, a power capacity property (`SEI::PowerCapacity`) is declared for a bus, and a power budget is declared for every component that requires an access to this bus (property `SEI::PowerBudget`). The plug-in adds all the power budgets for a bus and compares the result with its power capacity. This mechanism, even if it is interesting, is extremely limited: power budgets for every component are only a guess from the user, and are only used to compute the power consumption

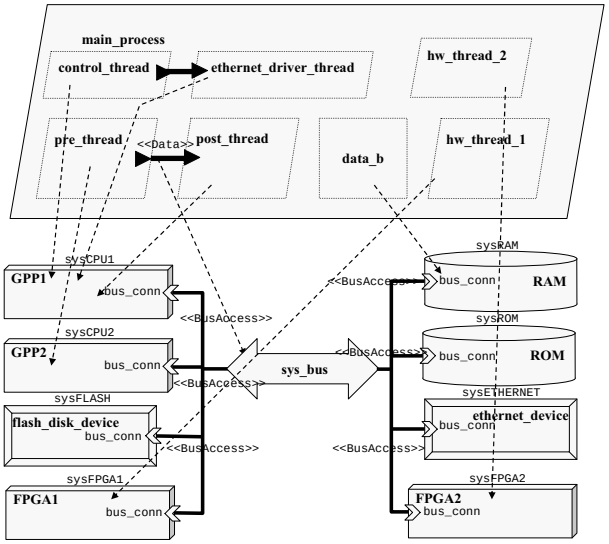

Fig. 2. Binding components to the target platform

of buses in a very simplistic way. In this paper, we propose a method to greatly enhance power analysis in the AADL flow, and to do it in an efficient way not only for buses, but for every consuming component in the system. Moreover, we propose to base power analysis on realistic power estimates, by using an accurate power estimation tool and precise power consumption models for every component in the targeted hardware platform.

3 High-level power consumption estimations

In order to complete power consumption analysis for the whole system, we need first to compute the power budget for every software component in the AADL component assembly model. This is the *power estimation* phase (1) represented with plain edges on figure 3 in the case of the binding of a thread to a processor.

Next, the power budgets of software components are combined to get the power budgets for hardware components. This is the *power analysis* phase (2) represented with thick dotted edges on the figure. Using timing information, the *energy analysis* will be performed afterwards (thin dotted edges).

The challenge for our power estimation tool is to provide a realistic power budget for every component in the application. This tool gathers several information in the system's AADL specification at different places, here from the process and thread descriptions, and from the processor specification. It also uses binding information that may come from a preliminary scheduling analysis. The tool uses the power model of the targeted hardware component (here a processor) to compute the power budget for the (software) component. In fact, it

determines the input parameters of the model from the set of information it has gathered. This process is repeated, not only for threads bound to processors, but also for any possible binding of software components onto hardware components, and that means: (i-) threads onto processors or FPGA, (ii-) processes and data onto memories, and (iii-) inter-components connections onto buses.

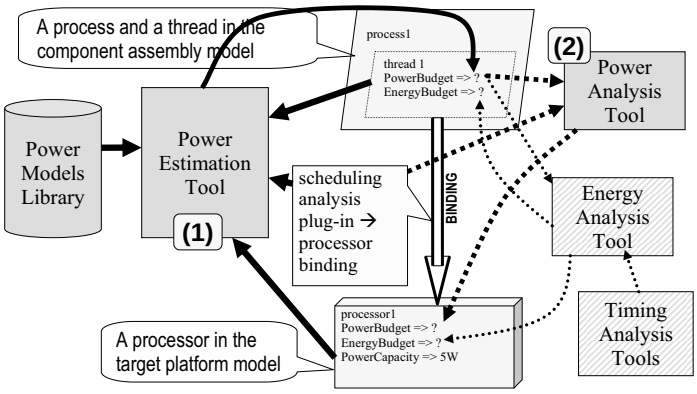

Fig. 3. Power and Energy consumption estimation in the AADL design flow

Once the power budgets have been computed for every component in the application, the power analysis is performed. The power analysis tool retrieves all the component power budgets, together with additional information from the specification, and computes the power budget for every hardware component in the system. Then it computes the power estimation for the whole system. The result of the scheduling analysis (which gives the load of processors) is also taken into account at this level. Indeed, whenever a processor is idle, its power consumption is at the minimum level. Scheduling analysis is performed using basic information on the threads properties defined as properties for each thread implementation in the AADL component assembly model: dispatch_protocol (periodic, aperiodic, sporadic, background), period, deadline, execution_time ...

This paper will concentrate on the power estimation phase, no further details will be given on the power analysis. Energy analysis will be finally performed using information from the timing analysis tools currently being developed by some of our partners in the SPICES project.

4 Multi-level power models

Our Power Estimation Tool, *PET*, is an evolution of the former *SoftExplorer*, initially dedicated to power and energy consumption estimation for processors (from simple RISC General Purpose Processors to very complex VLIW Digital Signal Processors) [24]. This tool comes with a library of power models for every hardware component on the platform.

Our objective is to allow power estimation at different levels in the flow. This involves the use of multi-level power models, which are models that can be used with more or less information, depending on the refinement level. In fact, while the specification is being refined, more information is available and power estimations get more precise.

Let's consider a case study platform including one GPP (the PowerPC 405), one DSP (the Texas Instrument C62), and one FPGA circuit (the Xillinx Virtex 400E). The description of those components' power models can be found respectively in [25], [18], and [20]. Power models are built following our Functional Level Power Analysis methodology [19]. The component's architecture is firstly analysed and relevant parameters regarding its power consumption are identified. Then physical measurements are performed to assess the evolution of the power consumption with the models' input parameters (using little benchmarking programs called "scenario"), and finally power consumption laws are established.

4.1 A complex Digital Signal Processor

The TI C62 processor has a complex architecture. It has a VLIW instructions set, a deep pipeline (up to 15 stages), fixed point operators, and parallelism capabilities (up to 8 operations in parallel). Its internal program memory can be used like a cache in several modes, and an *External Memory Interface* (*EMIF*) is used to load and store data and program from the external memory [26]. In the case of the C62, the 6 following parameters are considered. The *clock frequency* (F) and the *memory mode* (MM) are what we call *architectural parameters*. They are directly related to the target platform and the hardware component, and can be changed according to the users will. The influence of F is obvious. The C62 maximum frequency is 200MHz (it is for our version of the chip); the designer can tweak this parameter to adjust consumption and performances.

The remaining parameters are called *algorithmic parameters*; they directly depend on the application code itself. The *parallelism rate* α assesses the flow between the processor's instruction fetch stages and its internal program memory controller inside its IMU (Instruction Management Unit). The activity of the processing units is represented by the *processing rate* β. This parameter links the the IMU and the PU (Processing Unit). The activity rate between the IMU and the MMU (Memory Management Unit) is expressed by the *program cache miss rate* γ. The *pipeline stall rate* (PSR) counts the number of pipeline stalls during execution. It depends on the mapping of data in memory and on the memory mode.

The memory mode MM illustrates the way the internal program memory is used. Four modes are available. All the instructions are in the internal memory in the *mapped mode* (MM_M). They are in the external memory in the *bypass mode* (MM_B). In the *cache mode*, the internal memory is used like a direct mapped cache (MM_C), as well as in the *freeze mode* where no writing in the cache is allowed (MM_F). Internal logic components used to fetch instructions

(for instance tag comparison in cache mode) actually depends on the memory mode, and so the power consumption.

A precise description of the C62 power model and its building may be found in [18]. The variation of the power consumption with the input parameters, more precisely the fact that the estimation is not equally sensitive to every parameter, allows to use the model in three different situations.

In the first situation, only the operating frequency is known. The tool returns the average value of the power consumption, which comes from the minimum and maximum values obtained when all the others parameters are being made to vary. The designer can also ask for the maximum value if a higher bound is needed for the power consumption.

In the second situation, we suppose that the architectural parameters (here F and MM) are known. We also assume that the code is not known and that the designer is able to give some realistic values for every algorithmic parameter. If not, default values are proposed, from the values that we have observed running different representative applications on this DSP (see table 1).

In the third situation, the source code is known. It is then parsed by our power estimation tools: the value of every algorithmic parameter is computed and the power consumption is estimated, using the power model and the values enter by the user for the frequency and memory mode.

Table 1. Default algorithmic parameters for the C62

	α	β	PSR
LMSBV_1024	1	0,625	0,385
MPEG_1	0,687	0,435	0,206
MPEG_2_ENC	0,847	0,507	0,28
FFT_1024	0,5	0,39	0,529
DCT	0,503	0,475	0,438
FIR_1024	1	0,875	0,666
EFR_Vocoder_GSM	0,578	0,344	0,045
HISTO (image equalisation by histogram)	0,506	0,346	0,499
SPECTRAL (signal spectral power density estimation)	0,541	0,413	0,288
TREILLIS (Soft Dcision Sequential Decoding)	0,55	0,351	0,038
LPC (Linear Predictive Coding)	0,684	0,468	0,171
ADPCM (Adaptive Differential Pulse Code Modulation)	0,96	0,489	0,194
DCT_2 (imag 128x128)	0,991	0,709	0,435
EDGE DETECTION	0,976	0,838	0,173
G721 (Marcus Lee)	1	0,682	0,032
AVERAGE VALUES	0,7549	0,5298	0,2919

The error introduced by our tool obviously differs in these three situations. To calculate the maximum error, estimations are performed with given values for the parameters known in the situation, and with all the possible values of the remaining unknown parameters. The maximum error comes then from the difference between the average and the maximum estimations. This is repeated

for every valid set of known input parameters. The final maximum error is the maximum of the maximum errors. Table 2 gives the maximum error in the three situations above, which correspond to three levels of the specification refinement. Note that the maximal errors computed at level 2 are really pessimistic since we assume here that the designer is completely (100%) wrong on his evaluation of all the input parameters. If his evaluation of those parameters is only 50%, or 25% wrong, then the error introduced by our tool is reduced as well.

Table 2. Maximum errors for the C62 power model (Power in mW)

Known parameters	Memory Mode	Max Power	Min Power	Average Power	Max Error
Level 1					
Frequency	X	3037	848	2076	59%
Level 2					
Frequency, MM, α, β, γ, PSR	Mapped	2954	848	1809	53%
	Cache	2955	756	1778	57%
	Freeze	3018	882	1801	51%
	Bypass	3037	2014	2397	21%
Level 3					
F, MM, and the source code is provided	Max Error = 8%, Average Error = 4%				

4.2 A more simple General Purpose Processor

The PowerPC 405 is a light version of the IBM PowerPC, embedded in the Xilinx VirtexII Pro FPGA. It includes one prefetch instruction unit that allows to reduce the number of pipeline stalls due to instruction misses, two caches (16KB each) (one for the data and the other for instructions). These two caches can be involved separately and use LRU policy. They are two ways associative with 32 bytes line size. And three TLB translating addresses from logical to physical (2 shadow TLB - one for the instructions and one for data - are used and coupled with an unified one).

Measurements show that among the most important factors in the PowerPC 405 consumption are its frequency and the frequency of the bus to which it is connected. The processor can be clocked at 100, 150, 200 or 300 MHz, and, depending on the processor frequency, the bus (OCM or PLB) frequency can take different values between 25 to 100 MHz. Another important parameter to consider is the configuration of the memory hierarchy associated to the processor's core, and that means, which caches are used (data and / or instruction) and where is located the primary memory (internal / external). Once again, the component's power model can be used at three refinement levels.

At the first refinement level, our model gives a rough estimate of the power consumption for the software component, only from the knowledge of the processor and some basic information on its operating conditions. The only information

we need is the processor frequency and the frequency of the internal bus (OCM or PLB) to which the processor is connected inside the FPGA. They are two architectural parameters of the PowerPC 405. They will be defined as a property of the AADL processor implementation of the PowerPC 405 in the AADL specification. The maximum error we get here is 27%.

At the second refinement level, we have to add some information about the memories used. We have to indicate which caches will be used in the PowerPC 405 (data cache, instructions cache, or both), and if its primary memory is internal (using the FPGA BRAM memory bank) or external (using a SDRAM accessed through the FPGA I/O). Indeed, while building the power model for the PowerPC 405, we have observed that it draws quite different power and energy in those various situations [25]. Table 3 show the maximal errors we obtain here for every valid set of known input parameters, the others being unknown. The maximum error we obtain is 15,3% and the average error is 6,6%. The first line indicates 0% because in this configuration, there are not remaining unknown parameters that can change the power consumption of the processor.

Table 3. Maximal Errors for the PowerPC 405 at refinement level 2 (Power in mW)

	$F_{processor}/F_{bus}$ (MHz)									
	300/ 100	200/ 100	100/ 100	200/ 66	200/ 50	150/ 50	100/ 50	100/ 33	100/ 25	Max Error
2 caches BRAM	2595	2555	2515	2262	2134	2104	2084	1938	1869	0%
2 caches SDRAM_MAX	3129	3091	3053	2760	2604	2585	2566	2400	2321	
2 caches SDRAM_MIN	2719	2681	2643	2350	2194	2175	2156	1990	1911	
Error	7,0%	7,1%	7,2%	8,0%	8,5%	8,6%	8,7%	9,3%	9,7%	9,7%
BRAM_MAX	2570	2515	2460	2241	2112	2084	2057	1920	1856	
BRAM_MIN	2472	2440	2408	2177	2053	2037	2021	1890	1829	
Error	1,9%	1,5%	1,1%	1,4%	1,4%	1,1%	0,9%	0,8%	0,7%	1,9%
Icache_BRAM&BRAM_MAX	2662	2592	2522	2305	2170	2135	2100	1957	1890	
Icache_BRAM&BRAM_MIN	2497	2451	2405	2193	2071	2048	2025	1897	1836	
Error	3,2%	2,8%	2,4%	2,5%	2,3%	2,1%	1,8%	1,6%	1,4%	3,2%
Icache_SDRAM_MAX	3432	3267	3102	3155	3103	3020	2938	2882	2856	
Icache_SDRAM_MIN	2600	2478	2356	2403	2367	2306	2239	2208	2190	
Error	13,8%	13,7%	13,7%	13,5%	13,5%	13,4%	13,5%	13,2%	13,2%	13,8%
Dcache_BRAM_MAX	2595	2555	2515	2262	2124	2104	2084	1938	1869	
Dcache_BRAM_MIN	2588	2544	2500	2254	2118	2096	2074	1930	1861	
Error	0,1%	0,2%	0,3%	0,2%	0,1%	0,2%	0,2%	0,2%	0,2%	0,3%
Dcache_SDRAM_MAX	3535	3497	3459	3133	2960	2941	2922	2741	2622	
Dcache_SDRAM_MIN	2744	2706	2668	2375	2218	2199	2180	2015	1936	
Error	12,6%	12,8%	12,9%	13,8%	14,3%	14,4%	14,5%	15,3%	15,1%	15,3%

At the lowest refinement level, the actual code of the software component is parsed. In the case of the PowerPC 405, what is important is not exactly what instruction is executed, but rather the type of instruction being executed. We have indeed exhibited that the power consumption changes noticeably from memory access instructions (load or store in memory), to calculation instructions (multiplication or addition). As we have seen before, the place where the data is stored in memory is also important, so the data mapping is also parsed here. The average error we get at this level is 2%. The maximum error is 5%. Logically,

that corresponds to the max and average errors for the set of consumption laws for the component.

4.3 Field Programmable Gate Arrays

FPGA (Field Programmable Gate Arrays) are now very common in electronic systems. They are often used in addition to GPP (General Purpose Processors) and / or DSP (Digital Signal Processors) to tackle data intensive dedicated parts of an application. They act as hardware accelerators where and when the application is very demanding regarding the performances, that typically for signal or image processing algorithms. In this case again power estimation can be performed at different refinement levels.

At the highest levels, the code of the application is not known yet. The designer needs however to quickly evaluate the application against power, energy and / or thermal constraints. A fast estimation is necessary here, and a much larger error is acceptable. The parameters we can use from the high-level specifications are the frequency F and the occupation ratio β of the targeted FPGA implementation, that we consider as architectural parameters, and the activity rate α. The experienced designer is indeed able to provide, even at this very high-level, a realistic guess of those parameters' value. As explained before, to obtain the model, i.e. the mathematical equation linking its output to the parameters, we performed a set of different measurements on the targeted FPGA. For different values of the occupation ratio, and for different values of the frequency, we made the activity rate varying and measured the power consumption.

At our first refinement level, only the frequency is known. Our power estimation tool uses the model to estimate, at the given frequency, the power consumption with $\alpha = \beta = 0{,}1$ and with $\alpha = \beta = 0{,}9$. Then it returns the average value between those minimal and maximal values. The maximal errors we obtain for $F = 10$MHz and $F = 90$MHz (upper bound for the Altera Stratix EP1S80) are given table 4.

At the next refinement level, the two architectural parameters F and β, are known to the user. Like in the case of the former processor's models, default values are proposed for α and also β, coming from a set of representative applications. The maximal error introduced in this case ranges from 6,9% to 44,8%. To determine this error we compute the maximum and minimum estimations for the four extreme (F, β) couples, and compare them to the estimations with α default value.

At the lowest refinement level, the source code (a synthesizable hardware description of the component behaviour, may be written in VHDL or SystemC ...) is used. A High-Level Synthesis tool [27] permits to estimate the amount of resources necessary to implement the application, and given the targeted circuit, to obtain its occupation ratio (β) and its activity rate (α). Those two parameters and the frequency are finally used with the model.

Table 4. Maximum errors for the Altera Stratix EP1S80 (Power in mW)

Known parameters	Max Power	Min Power	Average Power	Max Error
Level 1				
Frequency (F=10MHz)	789	307	548	44%
Frequency (F=90MHz)	4824	835	2830	70%
Level 2				
Frequency, α, β				
F=10MHz, β=0,1	353	307	324	8,8%
F=10MHz, β=0,9	789	544	667	24,1%
F=90MHz, β=0,1	931	835	883	6,9%
F=90MHz, β=0,9	4824	2435	3630	44,8%
Level 3				
F and the source code is provided	Max Error = 4,2%, Average Error = 1,3%			

5 AADL Property Sets

Table 5 and 6 show the property sets associated respectively to the TI C62 and the PowerPC 405 for power estimation at the three refinement levels defined above. Table 7 shows the property set for the FPGA Alter Stratix EP1S80.

Table 5. Property set for the TI C62

TI C62 property set

```
Processor_Frequency : aadlreal applies to (processor);
Processor_Memory_Mode : TIC62::Processor_Memory_Mode_Type applies to (processor);
Processor_Parallelism_Rate : aadlreal applies to (processor);
Processor_Processing_Rate : aadlreal applies to (processor);
Processor_Cache_Miss_Rate : aadlreal applies to (processor);
Processor_Pipeline_Stall_Rate : aadlreal applies to (processor);
Processor_Memory_Mode_Type : type enumeration (CACHE,FREEZE,BYPASS,MAPPED);
Processor_Parallelism_Rate_Default : constant aadlreal => 0,7549;
Processor_Processing_Rate_Default : constant aadlreal => 0,5298;
Processor_Cache_Miss_Rate_Default : constant aadlreal => 0,25;
Processor_Pipeline_Stall_Rate_Default : constant aadlreal => 0,2919;
```

As described section 3, the power estimation tool, when it is invoked, extracts relevant information (a set of parameters) from the AADL specification, then computes the components' power consumption, and returns the results to fill the power budget properties for the software components. The binding makes it possible to put in relation components in the AADL component assembly model with the power models of hardware components on the targeted platform.

As we have just seen, depending on the information refinement, coarse or fine precision power estimations will be performed. Given the refinement level, information to be provided to the estimation tool depends on the selected target (which component). The information is more general if the refinement level is high, it will be more dedicated to the target if the refinement level is low. The set of properties that are used by the estimation tool actually depends on the

Table 6. Property set for the PowerPC 405

PowerPC 405 property set
Processor_Frequency : aadlreal applies to (processor); Processor_Bus_Frequency : aadlreal applies to (processor); Processor_Primary_Memory : PPC405::Primary_Memory_Type applies to (processor); Processor_Data_Cache : aadlboolean applies to (processor); Processor_Instructions_Cache : aadlboolean applies to (processor); Primary_Memory_Type : type enumeration (BRAM,SDRAM);

Table 7. Property set for the Altera Stratix EP1S80

FPGA Altera Stratix EP1S80 property set
FPGA_Frequency : aadlreal applies to (fpga); FPGA_Activity_Rate : TIC62::Processor_Memory_Mode_Type applies to (fpga); FPGA_Occupation_Ratio : aadlreal applies to (fpga); FPGA_Activity_Rate_Default : constant aadlreal => 0,4; FPGA_Occupation_Ratio_Default : constant aadlreal => 0,5;

component itself, and more precisely, on its power model. Even between two components of the same type, another set of specific properties might be necessary since another set of configuration parameters might apply. This is the case here for the two *processor* components TI C62 and PowerPC405. The property set of the processor comes finally as a part of its power model, and, as this, will remain separated from the general property set associated to the current AADL working project for the application being designed in the OSATE environment.

6 Conclusion

We have presented a method to perform power consumption estimations in the component based AADL design flow. The power consumption of components in the AADL component assembly model is estimated whatever the targeted hardware resource, in the AADL target platform model, is: a DSP (Digital Signal Processor), a GPP (General Purpose Processor), or a FPGA (Field Programmable Gate Array). A power estimation tool has been developed with a library of multi-level power models for those (hardware) components. These models can be used at different levels in the AADL specification refinement process. We have currently defined three refinement levels in the AADL flow. At the lowest level, level 3, the (software) component's actual business code is considered and an accurate estimation is performed. This code, written in C, or C++, for standard threads, can also be written in VHDL or SystemC for hardware threads. At level 2, the power consumption is only estimated from the component operating frequency, and its architectural parameters (mainly linked to its memory configuration in the case of processors). At level 1, the highest level, only the operating frequency of the component is considered.

Three power models have been presented for the TIC62 GPP, the PowerPC405 GPP, and the Altera Stratix EP1S80 FPGA. The maximum errors introduced by these models, at the three refinement levels, are given table 8.

Table 8. Maximal errors summary

Component	Max Error Level 1	Max Error Level 2	Max Error Level 3
TI C62	59%	57%	8%
PowerPC 405	27%	15,3%	5%
Altera Stratix EP1S80	70%	44,8%	4,2%

In the frame of the SPICES project, we are currently working at the integration of our *Power Estimation Tool* and *Power Analysis Tool* in the AADL OSATE tool environment.

References

1. P. Feiler, B. Lewis, and S. Vestal, "The sae architecture analysis & design language (AADL) A standard for engineering performance critical systems," in *IEEE International Symposium on Computer-Aided Control Systems Design*, Munich, october 2006, pp. 1206–1211.
2. "SAE - Society of Automative Engineers, SAES AS5506," v1.0, Embedded Computing Systems Committee, SAE, November 2004.
3. T. Vergnaud, "Modélisation des systèmes temps-réel embarqués pour la génération automatique d applications formellement vérifiées," Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications de Paris, France, 2006.
4. A. Rugina, K. Kanoun, and M. Kaniche, "Aadl-based dependability modelling," LAAS, Tech. Rep., 2006, number = 06209.
5. J. Hugues, B. Zalila, and L. Pautet, "Rapid prototyping of distributed real-time embedded systems using the aadl and ocarina," in *Proceedings of the 18th IEEE International Workshop on Rapid System Prototyping (RSP'07)*. IEEE Computer Society Press, may 2007, pp. 106–112, porto Alegre, Brazil.
6. The SPICES ITEA Project Website. [Online]. Available: http://www.spices-itea.org/
7. P. Research, "Diesel user manual." Philips Electronic Design and Tools Group, Tech. Rep., june 2001.
8. R. Peset-Lopis and K. Goossens, "The petrol approach to high-level power estimation," in *Proceedings of the ISLPED*, Monterey, California, USA, august 1998.
9. M. Loghi, M. Poncino, and L. Benini, "Cycle-accurate power analysis for multiprocessor systems-on-a-chip," in *Proceedings of the GLSVLSI*, Boston, Massachusetts, USA, april 2004.
10. R. BenAtitallah, S.Niar, A.Greiner, S.Meftali, and J.L.Dekeyser, "Estimating energy consumption for an mpsoc architectural exploration," in *in ARCS06*, Frankfurt, Germany, 2006.
11. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. International Symposium on Computer Architecture ISCA'00*, 2000, pp. 83–94.
12. W. Ye, N. Vijaykrishnam, M. Kandemir, and M. Irwin, "The design and use of simplepower: a cycle accurate energy estimation tool," in *Proc. Design Automation Conference DAC'00*, June 2000.
13. N.Dhanwada, I. Lin, and V.Narayanan, "A power estimation methodology for systemc transaction level models," in *International conference on Hardware/software codesign and system synthesis*, 2005.

14. I. Lee, H. Kim, P. Yang, S. Yoo, E. Chung, K.Choi, J.Kong, and S.Eo, "Powervip: Soc power estimation framework at transaction level," in *Proc. ASP-DAC*, 2006.
15. V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Trans. VLSI Systems*, vol. 2, pp. 437–445, 1994.
16. S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, "An accurate and fine grain instruction-level energy model supporting software optimizations," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS'01*, 2001, pp. 3.2.1–3.2.10.
17. G. Qu, N. Kawabe, K. Usami, and M. Potkonjak, "Function-level power estimation methodology for microprocessors," in *Proc. Design Automation Conf. DAC'00*, 2000, pp. 810–813.
18. N. Julien, J. Laurent, E. Senn, and E. Martin, "Power consumption modeling of the TI C6201 and characterization of its architectural complexity," *IEEE Micro, Special Issue on Power- and Complexity-Aware Design*, Sept./Oct. 2003.
19. J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional Level Power Analysis: An efficient approach for modeling the power consumption of complex processors," in *Proc. Design Automation and Test in Europe DATE*, Paris, France, march 2004.
20. E. Senn, N. Julien, N. Abdelli, D. Elleouet, and Y. Savary, "Building and using system, algorithmic, and architectural power and energy models in the fpga design-flow," in *Intl. Conf. on Reconfigurable Communication-centric SoCs 2006*, Montpellier, France, July 2006.
21. The SAE AADL Standard Info Site. [Online]. Available: http://www.aadl.info/
22. H. Balp, E. Borde, G. Hak, and J.-F. Tilman, "Automatic composition of AADL models for the verification of critical component-based embedded systems," in *Proc. of the Thirteenth IEEE Int. Conf. on Engineering of Complex Computer Systems (ICECCS)*, march 2008, Belfast, Ireland.
23. F. Singhoff, J. Legrand, L. Nana, and L. Marc, "Scheduling and memory requirements analysis with aadl," in *Proceedings of the 2005 annual ACM SIGAda international conference on Ada*, 2005, atlanta, GA, USA.
24. 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," *the EURASIP Journal on Applied Signal Processing, Special Issue on DSP-Enabled Radio*, vol. 2005, no. 16, September 2005.
25. E. Senn, J. Laurent, E. Juin, and J. Diguet, "Refining power consumption estimations in the component based aadl design flow," in *FDL'08, ECSI Forum on specification & Design Languages*.
26. *TMS320C6x User's Guide, Texas Instruments Inc.*, 1999.
27. P. Coussy, G. Corre, P. Bomel, E. Senn, and E. Martin, "High-level synthesis under I/O timing and memory constraints," in *ISCAS'05, International Symposium on Circuits and Systems*, May 2005, kobe, Japan.