Power modeling of wide-SIMD baseband processor datapath for different clock frequencies

Martin Palkovic, Bart Vanhoof, Matthias Hartmann, Tom Vander Aa,
Antoine Dejonghe, Liesbet Van der Perre
imec, Kapeldreef 75, 3001 Heverlee, Belgium
Email: {palkovic,vanhoofb,hartma,vanderaa,dejonghe,vdperre}@imec.be

***Abstract*—In Software-Defined Radio (SDR) domain, the energy consumption is a key aspect. However, the energy consumption is usually determined only during gate-level simulation, when the processor is designed and the application is mapped. This is especially true for dynamic reconfigurable coarse-grain array (CGA) processors used in the SDR where to our knowledge no energy modeling exists above the gate-level.**

In our contribution we propose a flow how to obtain the energy numbers for opcode activation on different functional units for different clock frequencies. Then we integrate the developed model into an instruction set simulator (ISS) to evaluate energy consumption for the given application for different clock frequencies. The results allow us to obtain global picture for different frequencies before the processor is fully synthesized and evaluate full applications at the ISS level from the energy perspective already during mapping process.

Compared to our previous work, we obtain the energy per opcode by simulation of the different heterogeneous functional units (FUs) using gate-level simulations of physically synthesized FU. Also, we demonstrate our approach on the wide-SIMD (256b) heterogeneous multi-threaded instance and state-of-the-art WLAN 4x4 40MHz application we mapped onto that instance.

I. Introduction

An SDR (Software Defined Radio) system is a radio communication system in which physical layer components are implemented on a programmable or reconfigurable platform. The modulation and demodulation is performed in software and thus the radio is able to support a broad range of frequencies and functions concurrently. Nowadays, this is especially attractive because of the soaring chip development cost and respin rate in deep sub-micro era, extremely diversified market demand for different wireless standards and the fast evolution of those standards causing short time to market.

The SDR is usually composed of several processing cores. To reach maximal energy efficiency, it is more efficient to tune one (or more) cores to a given functionality (such as digital front-end, baseband, forward error correction) rather than to make a very flexible core. Various research works such as [1] have pointed towards optimally adapting the processor to make an Application Specific Instruction-set Processor (ASIP). This gives at least an order of magnitude difference in the energy efficiency compared to a general purpose DSP or RISC processor and confirms the need for a more specialized and heterogeneous ASIP solution.

In this paper we focus on energy modeling of major energy consumption core on the SDR platform, the baseband engine, which is implemented as dynamic reconfigurable array processor. Compared to our previous work [2] we model both, the static and the dynamic energy of the datapath in a more accurate way and focus on the heterogeneous functional units (FUs) of our new processor core. We synthesize and analyze those units from energy perspective for different frequencies and integrate the resulting model into our instruction-set simulator (ISS). This allows us to obtain global picture for different frequencies before the processor is fully synthesized and evaluate full applications at the ISS level from the energy perspective already during mapping process.

The rest of the paper is organized as follows. Section II overviews the state-of-the-art. Section III introduces the dynamic reconfigurable array architecture template and its compiler and highlights the differences between our new and old processor instance. Section IV explains in detail the derivation of the power model for our new instance by obtaining the power of the opcode on the given FU using physical synthesis of the FU and power estimation flow at gate level. Section V provides the results and Section VI concludes our contribution.

II. Related work

In the domain of compiler and processor exploration frameworks a lot of frameworks exists at different levels. Some of the frameworks target high level such as SUIF or Wrap-IT, other are retargetable compiler frameworks such as GCC or ACE CoSy. However, only a subset of those frameworks provides also energy modeling.

Wattch [3] and SimplePower framework, which is based on Simplescalar [4], enable architectural exploration along with energy estimation. However, their power models are not geared towards newer technologies, their parameter range is still too restricted and are geared towards high performance systems. These frameworks do not support architectural features important for embedded handheld devices such as data parallelism or SIMD (Single Instruction Multiple Data), clustered register files, software-controlled memories etc. Most of those features also occur in the dynamic reconfigurable array processors we are targeting. Furthermore, the architecture design space supported by these frameworks is oriented towards processing subsystems of high-performance computing systems (complementary to our design space) and not towards embedded systems, which is our main focus.

Epic Explorer [5] is (Very Long Instruction Word) VLIW exploration framework, based on Trimaran [6] and supports energy estimation of the processing system. However, the architecture design space is limited to general purpose VLIWs. Industrial tools like Tensilica's XPRES [7], and Coware's Processor Designer [8] provide architectural and compiler retargetability, but the supported design space is limited to a restricted template. Some of the industrial players such as Target's Chess/Checkers [9] integrated the energy models at the instruction set level to their frameworks recently. However, their are targeting customized (Ultra-Low Power) ULP ASIPs and not embedded Coarse-Grain Array (CGA) architectures.

In the analytical power estimation domain which we are also targeting in this paper, there exist various research works. However, [10] is limited to the Lx processor, [11] is limited to the TI C6x processor, [12] is limited to i486 and Sparc, even though later it was extended also to other type of processors [13]. [14], [15] are accurate but limited to ARM only. The COFFEE framework [16] complements the Trimaran framework with an energy estimation engine to provide complete, consistent, flexible, fast and accurate power modeling over a large architectural space with a large amount of state of the art features. An energy estimator was used in [17] for exploring different CGA interconnect architectures which is complementary to our work. In [18] a hybrid functional and instruction level power model has been proposed. However it is mainly targeting the embedded general purpose RISC processors such as ARM. Compared to this work, our approach is more tightly coupled to a virtual machine simulation and thus should be more accurate. Moreover it is targeting a very data-parallel style namely a coarse grain array (CGA) type of embedded processor with heterogeneous FUs and different clock frequencies.

III. ADRES ARCHITECTURE AND BOADRES INSTANCE

In this section we explain the basics of our processor architecture template. Then we compare our previous and current instance.

A. ADRES architecture template

The ADRES architecture template [19], consists of an (coarse grain) array (CGA) of basic components, including functional units (FUs), register files (RFs) and an interconnect network. The array contains three types of basic components: FUs, storage resources such as RFs and read-only data memories, and interconnects that include wires, muxes and busses. The ADRES architecture is a flexible template that can be freely specified by an XML-based architecture specification language as an arbitrary combination of those elements. The architecture provides also support for multi-threading, heterogeneous CGA FUs and multiple memories. The computation-intensive kernels, typically dataflow loops, are mapped onto the reconfigurable array by the compiler using a modulo scheduling technique [20] to implement software pipelining and to exploit the highest possible parallelism, whereas the remaining code is mapped onto the VLIW processor. Except of exploiting instruction-level parallelism with modulo scheduling, also data-level parallelism is exploited by using wide-SIMD FUs and task-level parallelism by implementing the multithreading.

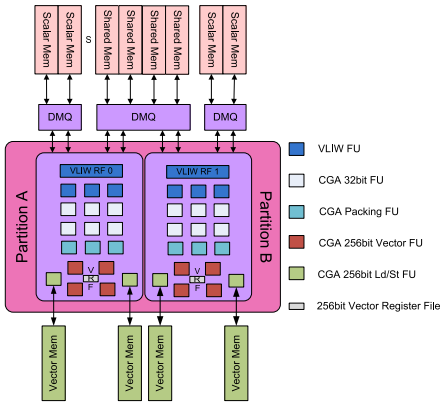
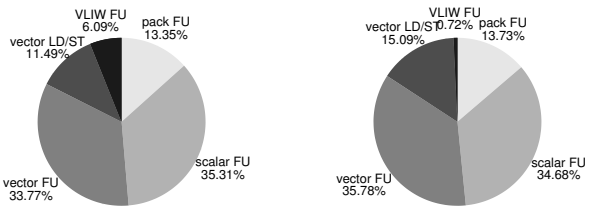


Fig. 1. Architecture diagram of our baseband processor.

The DRESC compiler tool chain, supporting the architecture template, consists of the IMPACT C compiler frontend [21] and of the DRESC compiler backend. IMPACT, a VLIW compiler framework, profiles and parses the C source code to an intermediate representation (Lcode), and applies several optimizations. These include extensive inlining and hyperblock formation by means of predication to eliminate control flow from inner loops. Those loops are then mapped onto the ADRES array mode with a modulo-scheduling algorithm exploiting the high parallelism of the loop kernels. This array mapping is fully retargetable, as the target ADRES instance is described in an XML file that is fed to the compiler together with the C code of the application. Traditional ILP scheduling techniques are applied to achieve high performance in the non-kernel parts of the application by executing them in the VLIW mode. The DRESC compiler backend generates scheduled code for both the CGA and the VLIW.

B. BOADRES processor instance

In this paper we focus on power estimation of our new BOADRES architecture instance (see Figure 1). Similar to the previous FLAI-ADRES architecture instance [2], there is also clear separation between VLIW mode (when VLIW FUs are active) and CGA mode (when CGA FUs are active). Compared to the FLAI-ADRES where all CGA FUs were identical, BOADRES has four different types of CGA FUs: 256b vector FUs, 32b scalar FUs, 256b packing FUs, and 256b load/store FUs. Together with VLIW FUs this results in five different types of FUs (see Figure 1). Similar to FLAI-ADRES also BOADRES supports multithreading. Compared to FLAI-ADRES, there are also different types of memories. Besides the local memory for each thread and global memory for cross-thread communication that were present also in the multi-threaded FLAI-ADRES, there are also vector memories attached to the vector load/store FUs.



(a) WLAN payload profiling (b) LTE profiling

Fig. 2. Profiling of the opcode occurrence on different types of FUs in the BOADRES processor for different wireless standards.

The heterogeneous FUs w.r.t. bit-width in BOADRES mean that energy of each opcode depends on the type of the unit the opcode was issued. E.g. *mov* operation has different energy on 32bit scalar FU and on 256bit vector FU. Thus the energy does not depend on the opcode only, but also on the FU type where the code was issued. The FUs of the same type can have different instructions supported. E.g. some of the vector FUs might support complex multiplication *cmul16*, others not. The energy thus depends on the concrete unit the opcode is executed on, but for the purpose of this paper we assume that all the FUs of the same type have the same characteristics and support the same instruction set. As we will see in the Section V this is still giving relevant results.

IV. Power model derivation

In this section we analyze the wireless applications w.r.t. different opcodes used, then we synthesize the type of FUs which are mostly active in the wireless applications and perform power estimation on those units for different opcodes and arguments and finally we integrate those results in our ISS.

A. Opcode profiling

Before doing power profiling of different opcodes, we have to know which opcodes are most used and what is the distribution across the different types of FU. Thus before starting with creation of the energy model, we profiled two wireless application, LTE 2x2 20MHz and WLAN 4x4 40MHz payload using our Adres Virtual Machine (AVM) Instruction Set Simulator (ISS). The distribution of the operations across the different types of FUs is depicted in Figure 2.

As we see from the Figure 2, more than 2/3 of the opcodes are issued in 256bit vector FUs and 32bit scalar FUs, so those will be also the FUs that we will profile (especially because 256bit vector FUs will be power hungry). Less than 1/3 of the opcodes is issued on pack FUs and 256bit vector load/store FUs. We will not synthesize those FUs for the energy profiling purpose, but we will estimate those values based on the synthesized 256bit vector FUs. Only few percent of the opcodes are executed on the VLIW. For WLAN, the amount is larger due to much tighter timing constraints ($4\mu s$ compared to $500\mu s$ in LTE), smaller CGA kernels and the multi-threading applied. For VLIW FUs we estimate the power consumption based on synthesized scalar FUs and their opcodes.

B. Functional unit synthesis

As mentioned in Subsection III-B, the energy of the opcode depends on the type of the FU the opcode is issued on. Most of the opcodes are issued on the vector and scalar FUs as depicted in Figure 2. Thus we decided to focus on those types of FUs and derive the power consumption of the other types of FUs based on the synthesis of vector and scalar FU. As mentioned in Subsection III-B the energy within the same type of FU can differ based on what type of instruction set is supported on that FU. In this paper, we assume that for the same type of FU, the same (maximal) instruction set is supported. We have synthesized one scalar FU and one vector FU with the maximal instruction set for different clock frequencies (1GHz, 800MHz and 600MHz) using Cadence RTL Compiler Ultra physical synthesis [22] and TSMC 40nm standard library. After the synthesis, we have applied the different opcodes on the given type of FU and have analyzed both, the leakage (static) and switching (dynamic) power that is consumed when an opcode is used. Thus, we can have a detailed view on the leakage as well as switching power, but in this paper we work only with the total power, even when breakdown into leakage and switching power is always possible. The power does not depend only on the opcode but also on the input data. Thus we have developed two models, first where the opcode is analyzed with the random input data, the second where the opcode is analyzed with the real data from our AVM ISS. We call the first model RAND power model and the second one AVM power model. The power numbers for the AVM power models are lower compared to RAND power model. This has two reasons; First, the data toggling for the same opcode when the input data is derived from AVM is not so large, because of similarity of the data for the same opcode. Second, not all the bits are toggling for real data. E.g. for the complex multiplication *cmul16* with three source operands, the first two are the two 32bit complex numbers (in 8 SIMD slots) that will be multiplied, the third operand is the shift factor applied after the multiplication. The shift factor is using usually only a few bits and not all 256 input bits of the third source. On the other hand, in reality the opcodes will be interleaved with other opcodes, so we assume that the realistic numbers will be between the RAND and AVM power model.

C. Power estimation

In Figure 3 we see comparison of the RAND and AVM power model for ten most frequently occurring opcodes in WLAN 4x4 40MHz payload processing for the vector FU and the scalar FU for 1GHz. The power is normalized to most consuming operation power, which is complex conjugate multiplication *ccmul16* on a vector FU. We see large difference between the vector and scalar operations, e.g. 256bit complex conjugate multiplication *ccmul16* is consuming $6\times$ more power than the 32bit complex conjugate multiplication *ccmul2*. This is less than factor of 8 because of the control overhead, that both, the vector and the scalar FU have to bear.

We also see big disproportion between the RAND power model and the AVM power model. As mentioned before, this is

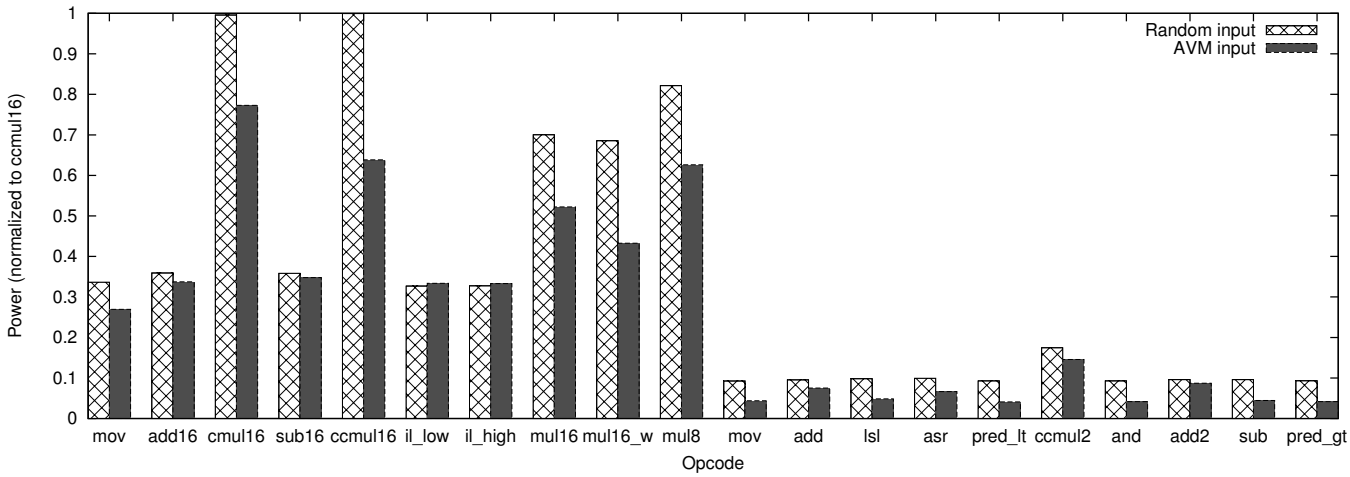


Fig. 3. Comparison of power (normalized to *ccmul16*) for different opcodes on different types of FUs (first ten opcodes are for vector FU, last ten for scalar FU) running at 1GHz when using the random input and input from our AVM ISS.

due to more toggling in the RAND power model input and also using all 256bits for all three inputs of the FU. The difference is the biggest one for the operations with shift factors such as *ccmul16*, *cmul16* or *mul16_w* as discussed in Subsection IV-B.

D. Integration of the power model into the ISS

After obtaining the power numbers for RAND and AVM power model for vector and scalar FUs synthesized for 1GHz, 800MHz and 600MHz and deriving the power numbers for packing, load/store and VLIW FUs we integrated those numbers into our AVM ISS. Compared to our previous work for FLAI-ADRES [2] we derived special energy map for each type of FU as below:

```
std::map<unsigned short, std::vector<float> >
  energy_map[NUMBER_OF_FU_TYPES];
boost::assign::insert( energy_map[FU_TYPE] )
  ( OPC , boost::assign::list_of()(POWER_LIST) );
```

where the POWER_LIST is the power list for different types of energy (static/dynamic/sum), different power models (RAND/AVM) and different clock frequencies (1GHz/800MHz/600MHz) for the given opcode on the given type of FU. After this addition, our ISS also does provide information of the power consumed in the given cycle on given FU based on opcode that is issued in that cycle on that FU.

V. Results

We have used our developed power model integrated into AVM ISS to provide the power insight on the WLAN 4x4 40MHz payload processing. The results are depicted in Figure 4 and Figure 5.

In Figure 4 we see the dynamism in the power depending on the time (clock cycle). We can clearly distinguish regions with high power corresponding to the CGA mode. First region is corresponding to CFO compensation, following four regions representing the four CGA radix loops of 128pt FFT, the two only slightly higher regions correspond to two CGA tracking loops running on the scalar FUs and equalization and

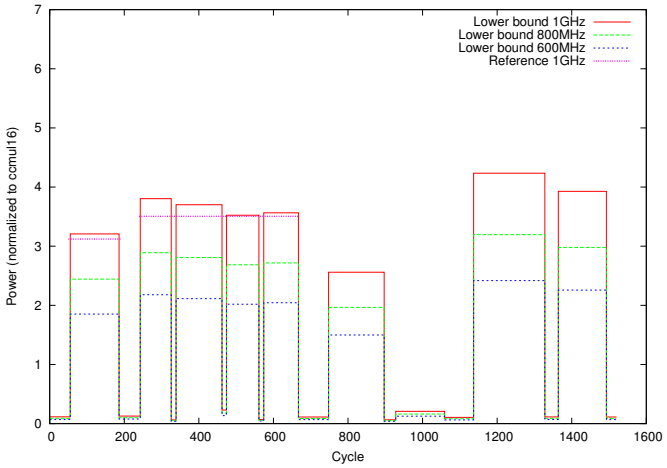


Fig. 5. Average power (normalized to *ccmul16*) for different kernels WLAN 4x4 40 MHz payload for different clock frequencies (1GHz, 800MHz and 600MHz) when using the AVM energy model and the reference of the CFO estimation and 128pt FFT utilizing the power-flow at gate level (logic synthesis).

demapping processing being the last two regions. The upper bound corresponds to the RAND power model, the lower bound corresponds to the AVM power model. For comparison, we provided also the reference that is corresponding to running two individual kernels, namely CFO compensation and 128pt FFT, on the fully synthesized BOADRES (logic synthesis) where our model was calibrated towards the logic synthesis of the fully synthesized BOADRES. The details of the calibration are out of the scope of this paper, but from comparison with 128pt FFT and CFO compensation we can see that our model is pretty accurate after calibration.

In Figure 5 we depict the comparison of the power when comparing the AVM power model for different frequencies. We have averaged the power over each kernel loop and have put also the reference for CFO estimation and 128pt FFT similar to Figure 4. The derived model can be used to obtain

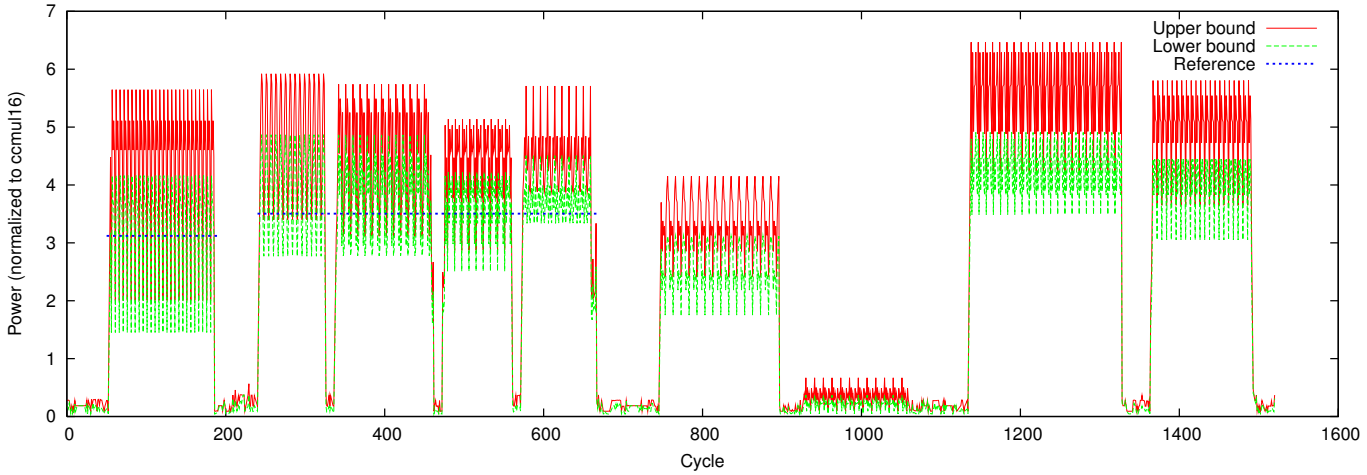


Fig. 4. Power (normalized to *ccmul16*) evolution for WLAN 4x4 40 MHz payload for 1GHz clock when using the RAND energy model, the AVM energy model and the reference of the CFO estimation and 128pt FFT utilizing the power-flow at gate level (logic synthesis).

global picture of the power consumption of the application and its dynamics as well as the idea of power consumption for different frequencies before the processor is fully synthesized. As mentioned before, it allows us also evaluate full applications at the ISS level from the energy perspective already during mapping process. To compare power consumption of BOADRES processor with another SDR baseband solutions is out of the scope of this paper. It is also rather difficult task, because the processors have to be designed with the same technology, the test applications have to be the same and the results have to be obtained from chip measurements to have fair comparison.

VI. Conclusion

In this paper we presented the derivation of the power model of our new BOADRES processor architecture and utilization of the model for obtaining better insight on the power consumption of wireless application we are mapping on the processor. The derivation of the model has its clear steps and can be repeated for any other instance. In our future work we would like to focus on the derivation of the model when each FU within the same type has different instruction set and better calibration of our model.

References

[1] W.J.Dally, J.Balfour, D.Black-Shaffer, J.Chen , R.C.Harting, V.Parikh, J.Park and D.Sheffield, "Efficient Embedded Computing," *IEEE Computer Magazine*, vol. 41, no. 7, pp. 27–32, July 2008.

[2] M.Palkovic, M.Hartmann, O.Allam, P.Raghavan, F.Catthoor, "Time-space energy consumption modeling of dynamic reconfigurable coarse-grain array processor datapath for wireless applications", *Proc. IEEE Wsh. on Signal Processing Systems (SIPS)*, Cupertino CA, IEEE Press, pp.134-139, Oct. 2010.

[3] D.Brooks, V.Tiwari, M.Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", *Proc of the 27th International Symposium on Computer Architecture (ISCA)*, pp.83–94, June 2000.

[4] T.Austin, E.Larson, D.Ernst, "SimpleScalar: an infrastructure for computer system modeling", *IEEE Computer Magazine*, Vol.35, No.2, pp.59–67, 2002.

[5] G.Ascia, V.Catania, M.Palesi, D.Patti, "EPIC-Explorer: A parameterized VLIW-based platform framework for design space exploration", *Proc of ESTIMedia*, pp.3–4, 2003.

[6] Trimaran 2.0: An Infrastructure for Research in Instruction-Level Parallelism, *http://www.trimaran.org*, 1999.

[7] R.E.Gonzalez, "Xtensa: A configurable and extensible processor", *IEEE Micro*, 20(2), 2002.

[8] CoWare Inc.: CoWare Processor Designer, *http://www.coware.com/products/processordesigner.php*, 2008.

[9] Target IP Designer, *http://www.retarget.com*, 2008.

[10] L.Benini, D.Bruni, M.Chinosi, C.Silvano, V.Zaccaria, "A Power Modeling and Estimation Framework for VLIW-Based Embedded System", *ST Journal of System Research*, Vol.3, No.1, pp.110–118, April 2002.

[11] M.Schneider, H.Blume, T.-G.Noll, "Power estimation on functional level for programmable processors", *Advances in Radio Science*, Vol.2, pp.215–219, May 2004.

[12] V.Tiwari, S.Malik, A.Wolfe, "Power analysis of embedded software: a first step towards software power minimization", *IEEE Trans. on VLSI Systems*, Vol.2, No.4, pp.437–445, Dec. 1994.

[13] V.Tiwari, S.Malik, A.Wolfe, M.Lee, "Instruction-level power analysis and optimization of software", *J. of VLSI Signal Processing*, No.13, Kluwer, Boston, pp.223–238, 1996.

[14] N.Chang, K.Kim, H.G.Lee, "Cycle-accurate energy consumption measurement and analysis: case study of ARM7TDMI", *ISLPED '00: Proceedings of the 2000 international symposium on Low power electronics and design*, pp.185–190, 2000.

[15] A.Sinha, A.P.Chandrakasan, "JouleTrack - A Web Based Tool for Software Energy Profiling", *Proc of Design Automation Conference (DAC)*, June 2001.

[16] P.Raghavan, M.Jayapala, A.Lambrechts, J.Absar, F.Catthoor, "Playing the trade-off game: Architecture exploration using Coffee", *ACM Trans. on Design Automation for Embedded Systems (TODAES)*, Vol.14, No.3, Article 36, pp.1–12, June 2009.

[17] A.Lambrechts, P.Raghavan, M.Jayapala, B.Mei, F.Catthoor, D.Verkest, "Interconnect Exploration for Energy Versus Performance Tradeoffs for Coarse Grained Reconfigurable Architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.17, no.1, pp.151–155, Jan 2009.

[18] H.Blume, D.Becker, M.Botteck, J.Brakensiek, T.G.Noll, "Hybrid Functional and Instruction Level Power Modeling for Embedded Processors," *Embedded Computer Systems: Architectures, Modeling, and Simulation: Proc. 6th Int. Workshop SAMOS 2006,* Lecture Notes in Computer Science, vol. LNCS 4017, pp. 237–247. Springer, 2006.

[19] B.Mei, S.Vernalde, D.Verkest, H.De Man, R.Lauwereins, "ADRES: An architecture with tightly coupled vliw processor and coarse-grained reconfigurable matrix," in *Proc. IEEE Conf. on Field-Programmable Logic and its Applications (FPL)*, Lisbon, Portugal, pp. 61–70, Sept. 2003.

[20] B.Mei, S.Vernalde, D.Verkest, H.De Man, R.Lauwereins, "Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling", *Proc. 6th ACM/IEEE Design and Test in Europe Conf.(DATE)*, Munich, Germany, pp.296–301, March 2003.

[21] The IMPACT Group, *http://www.crhc.uiuc.edu/Impact*.

[22] Cadence, *http://www.cadence.com/rl/Resources/datasheets/encounter_rtlcompiler.pdf*.