A Fast and Accurate Methodology for Power Estimation and Reduction of Programmable Architectures

Erwan Piriou, Raphaël David
CEA, LIST, Embedded Computing Lab, F-91171 Gif sur Yvette, FRANCE.
erwan.piriou@cea.fr – raphael.david@cea.fr

Fahim Rahim, Solaiman Rahim
Atrenta - Minatec Office
7 Parvis Louis Neel, 38040 Grenoble France

Abstract - We present a power optimization methodology that provides a fast and accurate power model for programmable architectures. The approach is based on a new tool that estimates power consumption from a register transfer level (RTL) module description, activity files and technology library. It efficiently provides an instruction-level accurate power model and allows design space exploration for the register file. We demonstrate a 19% improvement for a standard RISC processor.

I. Introduction

An early power estimation and exploration tool is a strong asset for optimizing an architecture. Taking into account the emergence of specialized and/or general purpose microprocessor resources in recent system-on-chip (SoC) designs, the need for fast power estimation becomes crucial. In particular, considering embedded processors, the opportunity to characterize an instruction set architecture (ISA) allows the feedback of important power figures to the compiler designer. Additionally, from a hardware point of view, the capability to highlight the cost/impact of architectural choices (e.g., the register file architecture) enables analysis of the architecture in terms of power consumption.

Tiwari et al. [1] have proposed a framework (Wattch) to analyze power consumption and optimize the design at the architectural level. This framework enables the association/correlation of a gate-level power model with a more abstract model for the design of the CPU. The Simplescalar tool [2] enables a dynamic power analysis taking into account real execution on a processor. The Xeemu simulator[3] uses the same approach and introduces more detailed microarchitectural parameters. The estimator achieves an accuracy of better than 5%. In fact, this accuracy is the result of a precise cycle accurate simulator. Such methodologies are based on gate-level power characterization, i.e., the power estimations are performed on the netlist file. Despite the accuracy of the estimation at this level, the large data set handled (and the processing time required) prevents a fast approach and an estimation on a complete application otherwise than by an average power characterization. It also prevents the designer from exploring the design space and considering different technology corner-cases.

The proposed approach considers the instruction set architecture and the RTL description as entry points. It allows the acquisition of a power scorecard for the programmable core at the instruction level while considering the architecture description at the RT level. In fact, several power scorecards could be generated depending on the chosen technology and frequency. This approach proposes to drastically reduce the computing time to rapidly obtain a power model and architectural hints for the programmable core design while remaining accurate.

978-3-9815370-0-0/DATE13/©2013 EDAA

II. Power estimation flow

The proposed power characterization starts from:

- The complete instruction set list (plus their format) and compilation toolchain
- Verilog, VHDL or SystemVerilog RTL description of the considered architecture
- Technology library

Figure 1 illustrates a four-step power estimation flow.

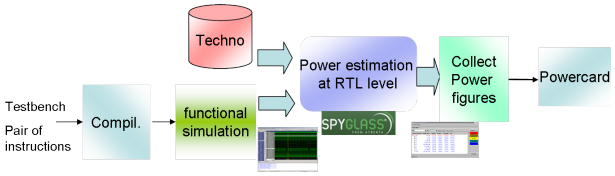

Fig. 1. Power estimation flow for programmable architectures

First, testbenches for all possible legal instruction pairs are generated in C code using asm volatile pragma features. Then, using the compilation toolchain, the program is compiled and the executable memory footprints are generated for RTL simulation. Next, simulations are performed for the hardware architecture using an RTL simulator. The activity files (.vcd, .fsdb or .saif format) are generated to feed the Atrenta® SpyGlass Power Estimate tool. Finally, all RTL and activity files are processed by the Atrenta SpyGlass RTL Power Estimation and Reduction tools. Concurrently, the technology library is linked and a functional frequency is initialized. The tool outputs the power figures for all testbenches. They are gathered to build a power scorecard. From a power analysis, the microarchitecture power budget is analyzed. The register file constitutes the greediest module.

III. Register file design space exploration

The register file (RF) is a good candidate because it consumes from 35 to 60% of the total power budget. In fact, this module optimization represents the best opportunity to save power. Our interest mainly focuses on the design space exploration of the register file while considering program execution. Nalluri et al. [4] have presented a way to

customize the register file depending on software access to the processor RF. It claims to reduce up to 55% of energy consumption by splitting a monolithic banked entity into a multi-banked configuration. Another study [5] has proposed a power reduction strategy by limiting accesses to the RF. A reuse of registers in the RF is enabled by comparing source and destinations operands of instructions. The RF accesses are reduced by 39% compared to a conventional implementation. Beyond a dedicated power improvement on the register file, we propose a structural approach by modifying both the RTL architecture of the RF module and the compiling options, especially the number of available registers. A reduction in the number of RF registers is achieved concurrently through the use of compiler options and the hardware description. Thus, different RF implementations are considered regarding effects on power consumption.

IV. A Case Study

The 16/32bit ISA of the RISC processor [6] is considered as a case-study. For a core handling at least 74 instructions, 74 benches are required for intrinsic power characterization and 2701 benches for pair of instructions estimations. Therefore, it is not realistic to quickly provide a complete gate-level power model for the processor. In fact, an aggregate testbench would have a length of 1 millisecond for the activity file. A single analysis at the netlist level of an activity file of such a length cannot be planned.

In this case study, the developed power estimation flow at the RT level allows a speedup by a factor 15 compared to gate level methods for the RISC processor running at 400 MHz on a TSMC 45nm library. The powercard contains all average power consumption obtained for the execution of the same instructions and the execution of all possible pairs of instructions. They are classified by functionality (load/store, arithmetic, logical, branch/jmp/call and coprocessor operations), by format (i.e., 16/32 bit encoding style) and by data access (registers or immediate). These figures have been used to back annotate the standalone instruction set simulator of the chosen processor. An average accuracy of nearly 80% was obtained for the power model on a representative set of benchmarks executing motion estimation, discrete wavelet transform, and a sort algorithm compared to gate-level characterization.

Experiments have also been performed for unified/split register file architectures and different numbers of registers (physical/compiling option). Here, a 16x32 bit register unified architecture (compilation for 16 registers available) is considered as the reference baseline. The second configuration is the same architecture with only 8 registers available. Next, a 8x32 bit register unified architecture with 8 registers option is the last unified configuration. Finally, three split configurations are implemented: an 8+2 architecture (with 10 registers available), an 2x8 architecture (with 16 registers available) and a last 2x8 architecture with a clock-gated cluster.

Figure 2 depicts the power consumption of the processor core and the register file, in mW, for the unified 16x32 register file architecture. Considering the unified version, when the number of used registers is shifted from 16 to 8, the total design and the register file power consumption remain the same. However, the execution time increases by 5.6%. The

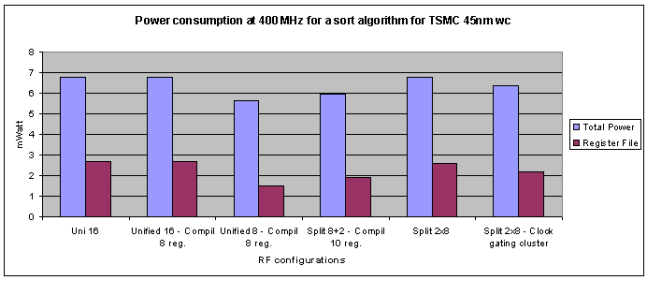

Fig. 2. Power consumption of a RISC processor and its RF

third case gives the power figure for an 8x32 unified RF implementation. The total power consumption is reduced by 17% whereas the register file component decreases by 44%. Additionally, the register file is modified to implement an 8+2 split configuration containing r0-r7 and r14-r15 (mapped to r8-r9) to avoid performance loss. Thus, the total power consumption is reduced by 12% whereas the register file component decreases by 30%. A strict splitting of the RF results in a slight gain only for the RF. Moreover, when clock gating is applied on the second part of the RF, the global power consumption decreases by 7.5% (19% for the RF).

V. Conclusions

This paper shows that a processor characterization can be done in an efficient manner using a power tool at the RT-Level rather than a strict gate-level power approach. Applying the SpyGlass Power Estimate tool resulted in a 15 fold speedup to obtain an accurate power model of a RISC processor. Moreover, a classification of instructions can be done with respect to their execution on the hardware description, allowing the back annotation of the instruction set simulator. This first step enables us to explore the impact of architectural choice. We focused on the register file design. Opportunities such as customization through compiling options have been studied. The splitting and the clock gating of one part of the RF can save 19% of the power consumption.

This opportunity for early design tools to consider programmable architectures power consumption is a promising solution to make the design process easier. It is of particular importance for compiler and hardware designers when pursuing power saving improvements. In the future, the approach would be applied on the other part of the microarchitecture.

References

[1] D.Brooks, V.Tiwari, and al. Wattch: A framework for architectural-level power analysis and optimizations. In *Intl Symp. on Computer Arch.*, 2000.

[2] D.Austin et al. Simplescalar: An infrastructure for computer system modeling. In *IEEE Computer 35*, 2002.

[3] Z. Herczeg, D. Schmidt, Á. Kiss, N. Wehn, and T. Gyimóthy. Energy simulation of embedded xscale systems with xeemu. *J. Embedded Comput.*, 3(3):209–219, August 2009.

[4] R.Nalluri et al. Customization of register file banking architecture for low power. In *Conference Proceedings:VLSI Design*, 2007.

[5] H.Takamura et al. Reducing access count to register-files through operand reuse. In *Advances in Computer Systems Architecture, Lecture Notes in Computer Science*, 2003.

[6] Charly Bechara et al. A small footprint interleaved multithreaded processor for embedded systems. In *ICECS*, pages 685–690, 2011.