AccuPower: An Accurate Power Estimation Tool for Superscalar Microprocessors

Abstract:

This paper describes the AccuPower toolset – a set of simulation tools accurately estimating the power dissipation within a superscalar microprocessor. AccuPower uses a true hardware level and cycle level microarchitectural simulator and energy dissipation coefficients gleaned from SPICE measurements of actual CMOS layouts of critical datapath components. Transition counts can be obtained at the level of bits within data and instruction streams, at the level of registers, or at the level of larger building blocks (such as caches, dispatch buffer, reorder buffer, function units). This allows for an accurate estimation of switching activity at any desired level of resolution. The toolsuite implements several variants of superscalar datapath designs in use today and permits the exploration of design choices at the microarchitecture level as well as the circuit level, including the use of voltage and frequency scaling. In particular, the AccuPower toolsuite includes detailed implementations of currently used and proposed techniques for energy/power conservations including techniques for data encoding and compression, alternative circuit approaches, dynamic resource allocation and datapath reconfiguration. The microarchitectural simulation components of AccuPower can be used for accurate evaluation of datapath designs in a manner well beyond the scope of the widely–used Simplescalar tools.

1. Introduction

It has been recognized by the processor design community that power dissipation is a first–class architectural design constraint not only for portable computers and mobile communication devices, but also for high–performance superscalar microprocessors [Mud 01]. A fair amount of research efforts has been directed towards reduction of power dissipation in this high–end systems. Proposed solutions include both microarchitectural and circuit–level techniques.

Several power estimation tools for processors have been designed, including Wattch [BTM 00], Simplepower [VKI 00] and TEM^{2}P^{2}EST [DLCD 00] to name a few. Simplepower is used only for simple 5–stage scalar pipelines and only models the execution of integer instructions; it is not applicable to superscalar processors. The major drawback of other tools is their reliance on the Simplescalar simulator [BA 97], which lumps many critical datapath artifacts like the dispatch buffer (DB, also known as an "issue queue"), the reorder buffer (ROB) and physical register files (PRFs) into a unified structure called Register Update Unit (RUU), quite unlike the real implementations, where the number of entries and the number of ports in all these structures are quite disparate. Consequently, power dissipation can not be estimated accurately on a per–component basis. Considering that in many cases, these components collectively contribute to more than half of the overall power dissipation of the chip, it is necessary to have facilities that allow the design space of these components and their interactions be modeled as accurately as possible. This is exactly one of the areas where AccuPower provides a more accurate simulation/power estimation framework than existing tools. Specifically, the main features of the AccuPower toolset are as follows:

- Detailed cycle–level simulation of all major datapath components and interconnections that mimic the actual hardware implementation, including separate and realistic implementations of the DB, register files, ROB, load–store queues and forwarding mechanisms.
- Detailed and accurate simulations of the on–chip cache hierarchy (including multiple levels of on–chip caches, interconnections, arbitration and chip–level I/O traffic).
- Built–in models for three major variants of superscalar datapaths in wide use.
- Well–instrumented facilities for collecting datapath statistics of relevance to both power and performance at the level of bits, bytes (for data and instruction flows) within logic blocks and subsystem–level components and the entire processor.
- Implementations of cutting–edge techniques for power/energy reduction at the microarchitectural level, logic level and circuit level, as well as techniques based on clock gating, voltage and frequency scaling to facilitate the exploration of the design space.
- Use of energy/power dissipation coefficients for energy dissipating events within datapath components derived from SPICE measurements of actual layouts of these components. These coefficients are used in conjunction with transition counts obtained from the microarchitectural simulation component of AccuPower to accurately estimate the power/energy dissipations. Coefficients for leakage dissipations are also provided.

We believe that short of an actual implementation, AccuPower's power estimation strategy is as accurate as it gets. As part of an ongoing effort, we are also in the process of validating the energy dissipation coefficients of critical datapath components from actual subscale implementations through MOSIS. A release of the AccuPower toolset, including the VLSI layouts and SPICE–measured data is planned for in the near future. The planned release will also incorporate models for multi–clustered datapaths; this feature is not implemented in AccuPower at this point.

2. Variants of modern superscalar datapaths.

Three variations of superscalar datapaths are predominantly in use in modern implementations. They mainly differ in two respects: how the physical registers are implemented and when the readout of the source physical registers occurs.

Datapath A (Figure 1)**:** Here, input registers that contain valid data are read out while the instruction is moved into the

DB. As the register values required as an input by instructions waiting in the DB (and in the dispatch stage) are produced, they are forwarded through forwarding buses that run across the length of the DB [PJS 96]. The dispatch buffer

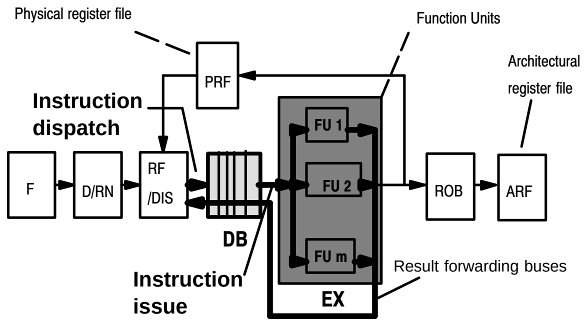

Figure 1. Datapath A

entry for an instruction has one data field for each input operand, as well as an associated tag field that holds the address of the register whose value is required to fill the data field. When a function unit completes, it puts out the result produced along with the address of the destination register for this result on a forwarding bus. Comparators associated with each DB entry then match the tag values stored in the fields (for waited–on register values) against the destination register address floated on the forwarding bus [PJS 96]. On a tag match, the result floated on the bus is latched into the associated input field. Since multiple function units complete in a cycle, multiple forwarding buses are used; each input operand field within a DB entry thus uses a comparator for each forwarding bus. Examples of processors using this datapath style are the IBM Power PC 604, 620 and the HAL SPARC 64 [Mi 9X].

Datapath B (Figure 2): Here, even if input registers for an instruction contain valid data, these registers are not read out at the time of dispatch. Instead, when all the input operands of an instruction waiting in the DB are valid and a function unit of the required type is available, all of the input operands are read out from the register file (or as they are yet to be written to the register file, using bypassing logic to forward data from latter pipeline stages) and the instruction is issued. In this case, the DB entry for an instruction is considerably narrower compared to the DB entries for Datapath A, since entries do not have to hold input register values. The dispatch/issue logic can be implemented using a global scoreboard that keeps track of instructions and register/FU availability. Alternatively, an associative logic similar to that of Datapath A can be used to update the status of input registers for instructions waiting within the DB (as shown in Figure 1(b)). Examples of processors using this datapath style are the MIPS 10000, 12000, the IBM Power 3, the HP PA 8000, 8500, and the DEC 21264 [Mi 9X, Bh 96].

Datapath C (Figure 3): Here, the ROB entry set up for an instruction at the time of dispatch contains a field to hold the result produced by the instruction – this serves as the analog of a physical register. We assume that each ROB entry may hold only 32–bit long result, thus requiring the allocation of two ROB entries for an instruction producing a

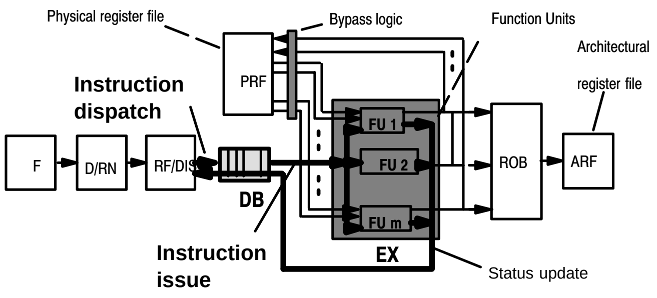

Figure 2. Datapath B

double–precision value. A dispatched instruction attempts to read operand values either from the Architectural Register File (ARF) directly (if the operand value was committed) or associatively from the ROB (from the most recently established entry for an architectural register, in case the operand value was generated but not committed. Source registers that contain valid data are read out into the DB entry for the instruction. If a source operand is not available at the time of dispatch in the ARF or the ROB, the address of the physical register (i.e., ROB slot) is saved in the tag field associated with the source register in the DB entry for the instruction. Forwarding to the waiting DB slots is performed similar to Datapath A. Examples of processors using this datapath style are the Intel Pentium II and Pentium III. [Mi 9X].

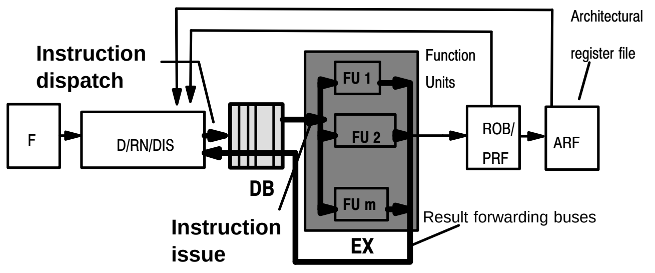

Figure 3. Datapath C

3. Implementation

The AccuPower tool consists of the three components: a microarchitectural simulator, which is a greatly modified version of the Simplescalar; the VLSI layouts for major datapath components and caches in a 0.5micron process; and power coefficients obtained from the SPICE simulations. (The public release planned will use 0.18 and 0.25 micron layouts as well.) Figure 4 summarizes the overall power/energy estimation methodology used in AccuPower.

3.1 Microarchitectural simulators

To support the three superscalar datapath configurations discussed in section 2, we significantly modified the Simplescalar simulator [BA 97] and implemented separate structures for the DB, the ROB, the rename table, the physical register files and the architectural register files. Three versions of the simulator were designed – one for each datapath configuration discussed above. We also accurately modeled the pipeline structures and various interconnections within the datapath, namely the dispatch buses, issue buses, result buses and commit buses. In a typical superscalar processor, multiple sets of such buses are needed to sustain the dispatch/issue/commit rate. Traffic on each such bus and read/write activity within the register files implementing the datapath storage components are separately monitored and analyzed as discussed later.

The extent of our modifications to the Simplescalar can be gauged by the fact that barely 10% of the original code was retained. The *sim–outorder.c* file was completely rewritten to support a true cycle–by–cycle out–of–order instruction execution. This is in contrast to the original Simplescalar code, where instructions are actually executed in–order at the dispatch stage and the effects of out–of–order execution are achieved through the convoluted manipulations with the RUU. Significant modifications have also been incorporated into the cache simulator, as discussed below. Specifically, the changes that were made to the Simplescalar to simulate the realistic superscalar pipelines are as follows:

(i) We split the monolithic cache access stage as used in Simplescalar into two stages to mimic the real–world situation where cache accesses – even L1 cache accesses – are typically performed in multiple cycles and provided a support for pipelined cache.

(ii) We modeled the contention for the bus between L1 and L2 caches. This is important because the L1 caches are typically split with separate caches for instructions and data. L2 cache, however, is typically unified. Situation when both I1 miss and D1 miss occur in the same cycle are certainly possible and they require proper modelling to arbitrate for the access to the L2 cache.

(iii) Along similar lines, we modelled the contention for the off–chip interconnection from the L2 cache to the DRAM modules.

(iv) The decode stage of the Simplescalar datapath, where instruction dispatching and register renaming is performed, was split into two stages, as it is unrealistic to perform the fairly complicated operations of dispatching, register renaming and source register readout in a single cycle. The only exception to this is Datapath C, due to the use of associative addressing for performing a source physical register lookup instead of an indirect addressing through a rename table.

(v) We also assumed realistic delays on the interconnections, noting that it takes a full cycle to distribute the result produced by one of the FUs to the waiting operands in the DB.

(vi) For Datapaths A and B, the branch misprediction handling was implemented by waiting till all instructions prior to the mispredicted branch commit, after which the contents of the ROB, the DB, the physical register files and the rename table were flushed and the execution was restarted from the correct PC. Of course, the branch misprediction penalty for Datapaths A and B was higher than in Simplescalar implementation. For Datapath C, we tagged each instruction with the id of the preceding branch and selectively flushed the ROB and the DB on a misprediction by using the associative search for the appropriate tags. This is doable for the Datapath C, because the ROB incorporates rename table and physical register files. For Datapaths A and B, such selective flushing results in the inconsistent values of the valid bits stored within physical register files.

To summarize, we attempted to design a simulator that would closely mimic the actual microarchitecture and hardware implementations of real CPUs on a cycle–by–cycle basis. The focus of the AccuPower tool is to facilitate the exploration of the design space of superscalar processors and gauge the impact of well–used and cutting–edge techniques for saving power and/or energy. The tool also supports the exploration of circuit–level techniques and the more standard power reduction techniques like voltage and frequency scaling.

3.2 VLSI layouts

For the estimating the energy/power for the key datapath components using AccuPower, the transition counts and event sequences gleaned from the microarchitectural simulator were used, along with the energy dissipations for each type of event, as measured from the actual VLSI layouts using SPICE. CMOS layouts for the on–chip caches, DB, PRF, ARF and DB in a 0.5 micron 4 metal layer CMOS process (HPCMOS–14TB) were used for the key datapath components to get an accurate idea of the energy dissipations for each type of transition. We are in the process of migrating these layouts to a 0.18 micron process and making SPICE measurements, but the results for the 0.5 micron layouts are, however, quite representative, although greatly scaled down compared to what one would see with 0.18 micron implementations running at a faster clock rate! The exception to this claim are the wire dissipations *outside* these components. (Dissipations on such wires at small feature sizes become relatively dominant.)

The register files that implement the ROB and DB were carefully laid out to optimize the dimensions and allow the use of a 300 MHz clock. A Vdd of 3.3 volts is assumed for all the measurements. (The value of the clock rate was determined by the cache cycle time, as determined by the cache layouts for a two–stage pipelined cache in the same technology.) In particular, these register files feature differential sensing, limited bit line driving and pulsed word line driving to save power. Augmentations to the register file structures for the DB (mainly in the form of comparators with each of the 3 source operand fields and the four result/tag buses) were also fully implemented; a pulldown comparator was used for associative data forwarding to entries within the DB and the device sizes were carefully optimized to strike a balance between the response speed and the energy dissipations. For each energy dissipating event, SPICE measurements were used to determine the energy dissipated. These measurements are used in conjunction with the transitions counted by the hardware–level, cycle–by–cycle simulator to estimate dynamic energy/power accurately. Actual layout data was also used for estimating the leakage power of the layouts in the smaller feature sizes.

3.3 Speeding up the execution – multithreading

The instrumentation needed to determine the bit level activities within data flow paths and data storages (both explicit and implicit) and log all major switching activities slows down the simulation drastically. To get reasonable overall simulation performance with all the instrumentation in place, we resorted to the use of multithreading.

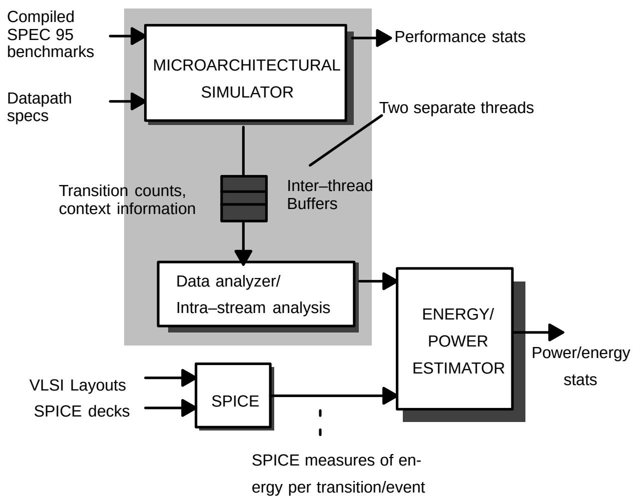
Compiled SPEC 95 benchmarks
Datapath specs
MICROARCHITECTURAL SIMULATOR
Performance stats
Two separate threads
Transition counts, context information
Inter–thread Buffers
Data analyzer/ Intra–stream analysis
ENERGY/ POWER ESTIMATOR
Power/energy stats
VLSI Layouts
SPICE decks
SPICE
SPICE measures of energy per transition/event

Figure 4. Power Estimation Methodology

Specifically, we use a separate thread for the data stream analysis, as shown in Figure 4. The two–threaded implementation is run on SMPs to get an acceptable level of simulation speed – one that approached close to that of the original Simplescalar without the heavy instrumentation. The data acquired from basic instrumentation within the main simulation thread is buffered and fed into a separate thread where it was analyzed for the lack of entropy within significant byte slices and all byte slices within a data item as well across consecutive data items within a data stream.

With a single thread implementing all of the simulation, instrumentation and analysis, the overall simulation speed was reduced by as much as 40% compared to the original Simplescalar simulation without any modification and instrumentation. With both threads in place as shown in Figure 4, and with the use of inter–thread buffers of an optimized size, the overall simulation time achieved was often significantly better on a SMP compared to the single–threaded implementation. The performance of the dual–threaded version was also acceptably close to that of the original Simplescalar simulator without any of the enhancements and the instrumentation.

Public release of AccuPower is planned in the very near future. Validations of the SPICE–measured data from MOSIS–supported implementations of some critical datapath components are also planned as part of an ongoing effort.

4. The use of AccuPower

AccuPower can be used to obtain realistic measurements in a variety of ways. These include:

1) The raw data collection. This falls into two categories:

a) The tool monitors the bit–level datapath activity on the interconnections, dedicated transfer links, and read/write activity of the register files that implement the datapath storage components. We also provide the data analyzer

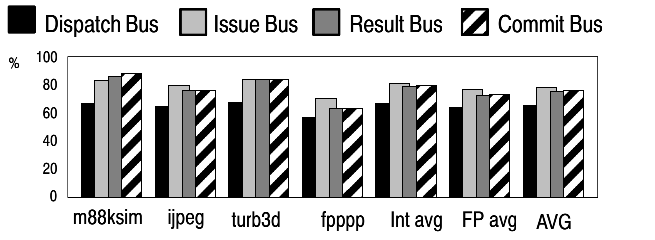

Figure 5. % of bits not driven using bit-slice invariance on top of zero byte encoding for Datapath A

(implemented as a separate thread for performance reasons, as discussed above) to examine the collected data streams on the presence of zero–bytes. Presence of zero bytes has been exploited to reduce the switching activity and hence power dissipations in caches [VZA 00] and in other datapath components [GPK+ 00]. In addition, the data analyzer estimates the percentage of bits that did not change their values since the previous value had been driven on the same link. Considerable power savings can be achieved on the

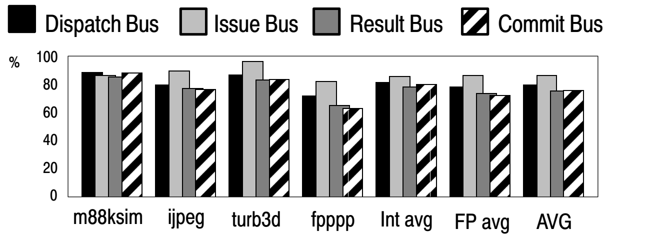

Figure 6. % of bits not driven using bit-slice invariance on top of zero byte encoding for Datapath B

datapath interconnections by not driving such bits. [GPK+ 00]. To further reduce the number of bits driven on the interconnections, such bit–slice invariance can be used on top of zero–byte encoding. Representative resulting percentages of bits that do not have to be driven on the datapath interconnections are shown for Datapaths A and B in Figures 5 and 6 respectively.

b) In AccuPower, the occupancies of individual datapath resources, as measured by the number of valid entries, are monitored and recorded. This capability is currently implemented for the DB, the ROB, and the LSQ. Support for caches will be added in the near future. Figure 7 shows the occupancies of the DB, the ROB, and the LSQ obtained from the execution of *fpppp* benchmark on AccuPower simulator. Resource occupancies have been recorded at every cycle and the averages have been taken for every 1 million of simulated cycles. This occupancy statistics can be used to drive the decision to dynamically resize the resources, in case the occupancy sampling shows that the resource is currently overcommitted. Correlations among the resource occupancies can be further exploited to reduce the overhead of the control logic needed to implement the dynamic resizing. Such dynamic resizing of multiple resources has been done in [PKG 01] with very small impact on the performance. Achieved power savings and performance degradation can be measured using AccuPower for all SPEC 95 benchmarks for a variety of resizing parameters, as discussed in [PKG 01]. Representative results for power savings within the ROB for Datapath C, obtained from AccuPower, are shown in Figure 8.

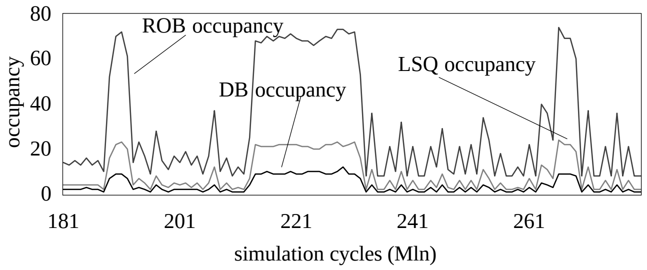

Figure 7. The occupancies of the ROB, the DB and the LSQ

2) Accurate datapath component–level power estimation. Here, we use transition counts obtained from the architectural simulation in conjunction with the capacitance coefficients obtained from SPICE simulations. Figures 8 and 9 represent some of the typical measurements.

3) Exploration of various power reduction techniques. These include the use of zero–encoding [VZA 00], dynamic resource allocation using occupancy sampling [PKG 01] and the use of partitioned datapath components [ZK 00]. Figures 5, 6 and 8 show some representative measurements.

4) Exploration of alternative circuit–level techniques. These currently include the use of fast dissipate–on–match comparators for associative lookup within the DB, the ROB and the TLB, bit–line segmentation within register files and zero–byte encoding. Figure 9 demonstrates the representative savings, as measured by AccuPower.

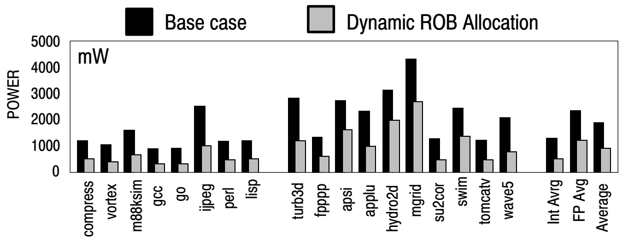

Figure 8. Total power dissipation within the ROB for a 4-way superscalar processor. (Datapath C)

5) Explorations of alternative datapath architecture. As mentioned above, three datapath configurations are currently supported. Figures 5 and 6 can be used to compare the bit–slice inactivities in Datapaths A and B.

5. Conclusions

This paper presented AccuPower – the accurate simulation tool for estimating power dissipation within various flavors of modern superscalar datapaths on a per–component basis. AccuPower consists of microarchitectural simulator in the form of significantly redesigned Simplescalar and the actual VLSI layouts for the components of a superscalar processor. We also showed how AccuPower can be used to estimate power savings achieved by several proposed microarchitectural and circuit–level techniques.

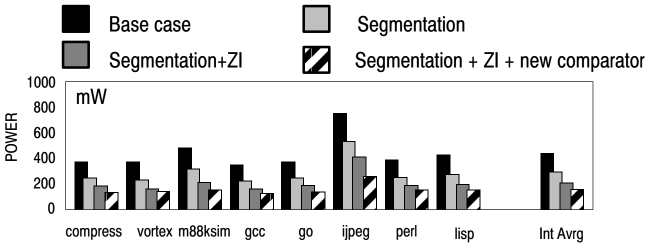

Figure 9. Total power dissipation within DB using bitline segmentation, power efficient comparators and zero-byte encoding (ZI). (Only SPEC95int results for datapath A are shown)

6. References:

[BA 97] Burger, D., and Austin, T. M., "The SimpleScalar tool set: Version 2.0", Tech. Report, Dept. of CS, Univ. of Wisconsin–Madison, June 1997 and documentation for all Simplescalar releases (through version 3.0).

[Bh 96] Bhandarkar, D., "Alpha Implementations and Architecture – Complete Reference and Guide", Digital Press, 1996.

[DLCD 00] Dhodapkar, A., Lim, C., Cai, G., Daasch R, "TEM^{2}P^{2}EST: A Thermal Enabled Multi–Model Power/Performance ESTimator", in Workshop on Power–Aware Computer Systems, held in conjunction with ASPLOS, 2000.

[GPK+ 00] Ghose, K., Ponomarev, D., Kucuk, G., Flinders, A., Kogge, P., and Toomarian N.,"Exploiting Bit–slice Inactivities for Reducing Energy Requirements of Superscalar Processors," in Proc. of Kool Chips Workshop, Micro–33, 2000.

[Mi 9X] Microprocessor Report, various issues, 1996–1999.

[Mud 01] Mudge, T., "Power: A First–Class Architectural Design Constraint", IEEE Computer, April 2001, pp. 52–58.

[PJS 96] Palacharla, S., Jouppi, N. P. and Smith, J.E., "Quantifying the complexity of superscalar processors", Technical report CS–TR–96–1308, Dept. of CS, Univ. of Wisconsin, 1996.

[BTM 00] Brooks, D., Tiwari, V., Martonosi, M., "Wattch: A Framework for Architectural–Level Power Analysis and Optimizations", Proc. of ISCA–27, 2000.

[PKG 01] Ponomarev, D., Kucuk, G., Ghose, K., "Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources", to appear in MICRO–34.

[VZA 00] Villa, L., Zhang, M. and Asanovic, K., "Dynamic Zero Compression for Cache Energy Reduction", in Proc. of Micro–33, 2000.

[VKI+ 00] Vijaykrishnan, N., Kandemir, M., Irwin, M.J. et al, "Energy–Driven Integrated Hardware–Software Optimizations Using SimplePower", in Proc. 27–th Int'l. Symposium on Computer Architecture, pp.95–106.

[WM 99] Wilcox, K., Manne, S., "Alpha processors: A History of Power Issues and a Look to the Future". In Cool–Chips Tutorial, November, 1999.

[ZK 00] Zyuban, V. and Kogge, P., "Optimization of High–Performance Superscalar Architectures for Energy Efficiency", in Proc. of Int'l Symposium on Low–Power Electronics and Design, 2000, pp. 84–89.