A Survey on Low Power Design from Architectural Perspective

Shayan Taheri
University of Central Florida, shayan.taheri@knights.ucf.edu

I. Introduction

Due to the recent advances in microelectronic technology, more functionality and performance can be provided on an integrated circuit (IC), which creates the possibility of designing smaller devices. This progress caused a shift in portable applications from low performance products such as wristwatches and calculators to high throughput and speed, and computation intensive products such as notebook computers and cellular phones. The dark side of this trend is the urgent demand of these applications for low power consumption and reasonable battery life, size, and weight. Otherwise, the usability and applicability of these applications are questionable.

Therefore, this trend makes power consumption as important as performance and area. Figure 1 shows the reason for this requirement graphically. Another driving force for diving into low power design is its important role in silicon integration (i.e. more number of transistors on a single chip or on a multi-chip module), which otherwise cooling, packaging, and reliability can be serious issues for electronic products. Even, it is a priority for large and/or parallel computing machines. Nowadays, low power design has postulate in different technologies, such as telecommunications, computers, consumer electronics, and biomedical, which shows the importance of this research area.

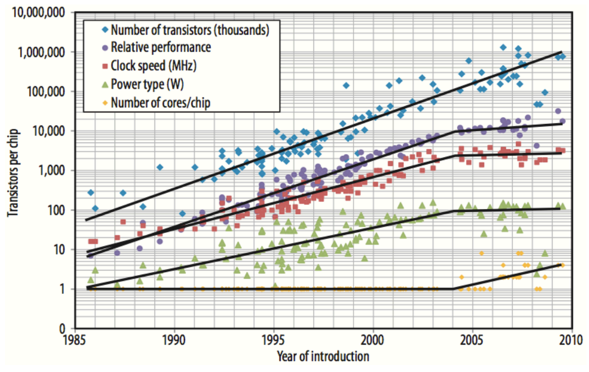


Figure 1. Requirement to low power design

Due to the ever improvement of architectural mechanisms and inclusion of more complex computational units in modern high-performance processors, power consumption and heat dissipation have become significant challenges in their design. Beside that, low power design is highly advantageous for general-purpose processors due to having idle time in their user-interactive mode. Power consumption of processors can be reduced at different levels, including fabrication technology, circuit design, micro-architecture, instruction set, run-time or operating system (O.S.), compiler, source code, algorithm, and application.

The rest of this survey is organized as follows. In section 2, origins of power dissipation in digital systems are presented. Also, general architectural solutions for power reduction are mentioned. The basic causes for power consumption from architectural perspective and the related overcomer techniques are discussed in Section 3. The survey is concluded in Section 4.

II. Origins of Power Dissipation and Solutions

CMOS circuits have both static and dynamic power dissipation. Static power is originated from to two types of leakage current: (a) sub-threshold leakage current that is generated when transistors don't turn off completely; and (b) P-N junction (diode) leakage current that is generated from existing diodes in transistor structure. While static power has its own importance, but it is insignificant in compare to dynamic power that is the dominant component of power dissipation in digital systems. This power is due to two noticeable currents: (a) switching current, which comes from the charging and discharging mechanism of load capacitance; and (b) short-circuit current, which is produced when both n-type and p-type transistors turn on during a transition. Load capacitance of a logical stage is constructed by gate output diffusion capacitance of current stage, gate input capacitance of next stage, and interconnecting metal layer capacitance.

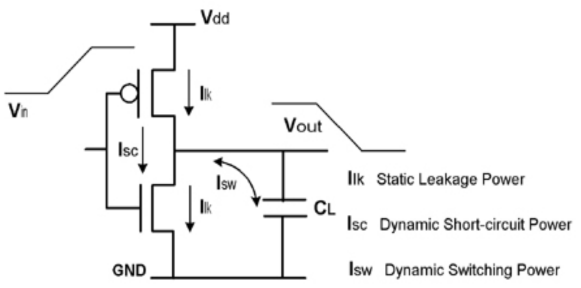


Figure 2. Static and dynamic power

Figure 2 is a graphical example for the aforementioned currents in digital systems. In this figure I_{lk} represents the leakage currents, I_{sw} is switching current, and I_{sc} is short-circuit current. Equations 1 and 2 can be used in order to obtain static and dynamic power for a single transistor. In these equations, V_{DD} is supply voltage, K_{Design} is design parameter, $I_{Leakage}$ is total leakage current, C is load capacitance, A is activity factor, and f is frequency (clock rate).

$$P_{Static} = V_{DD} \,.\, K_{Design} \,.\, I_{Leakage} \tag{1}$$

$$P_{Dynamic} = \frac{1}{2} \,.\, C \,.\, A \,.\, f \,.\, V_{DD}^2 \tag{2}$$

Shortage of attention to attenuate these currents properly leads to more current draw and consequently more power consumption, which causes having less effective energy capacity for battery, heating processor and affecting speed and reliability, and ground bounce (i.e. resistive and inductive supply voltage drop). Meanwhile, it should be mention that reducing power consumption may be provided at the cost of performance degradation. So, it is required to achieve a trade-off between delay and power. Equation 3 can be used in order to calculate the delay of a device. In this equation, C is load capacitance, $I_{out\,(ave)}$ is the average of flowed currents into load capacitance, V_{DD} is supply voltage, K_v technology constant, W is width, V_{th} is threshold voltage, and V_{DSAT} is drain voltage at saturation.

$$Delay = \frac{C}{I_{out\,(ave)}} \,.\, \frac{V_{DD}}{2} = \frac{C \,.\, V_{DD}}{K_v \,.\, W \,.\, (V_{DD} - V_{th} - V_{DSAT})} \tag{3}$$

As it was mentioned before, there are different design techniques in processor abstraction layers that can be utilized in order to reduce and manage power consumption. In this survey, the concentration is on power reduction and management techniques at architecture-level. Development of these techniques is mostly based upon utilization and exploitation of application characteristics and access/activation of computing or storing resources only when it is necessary. Figure 3 shows distribution of power consumption in different processor elements.

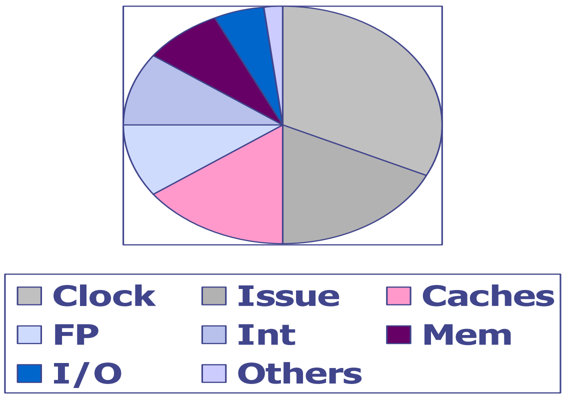


Figure 3. Processor power distribution

General architectural low power design techniques are classified into two categories: (a) processor-based, and (b) memory-based. The design techniques based on processor structure are: (1) voltage/frequency scaling (supply voltage and/or frequency reduction in idle time of processor); (2) clock gating (disabling clock to inactive dynamic components); (3) power gating (disabling supply voltage to inactive components), (4) pipeline gating (reducing mis-speculated instruction execution by fetch stalling when confidence is low); (5) pipeline balancing (adjusting effective pipeline ways for available instruction-per-cycle); and (6) efficient logical circuit for instruction-issue pipeline stage. The memory-based design techniques can be mentioned as: (1) banked or hierarchical register file; (2) sub-banked cache; (3) sequential access or way prediction caches; (4) dynamically adjusting cache size; (5) decay cache for reducing static power; and (6) low power DRAM cells with deep sleeping modes.

III. Low Power Design at Architecture-Level

According to [7], there are three basic causes (or roots) for power dissipation at architecture-level that are: (1) Program waste of energy, which is due to execution of instructions that are unnecessary for correct program execution; (2) Speculation waste of energy, which is due to speculative execution of instructions that do not commit their results; and (3) Architectural waste of energy, which is due to oversizing of processor structures. Program waste occurs due to fetch and execution of unnecessary instructions. An instruction is labeled as "Unnecessary" when it produces either dead or redundant results. An instruction that generates dedicated data for an unnecessary instruction can also be considered as unnecessary. This type of instructions doesn't have any effect on processor state.

The unnecessary instructions can be classified in this way: (a) Dead instruction (`dead`), whereby dead results are produced. A result is labeled as "Dead" if it is written in register(s) or memory, but is not read before being overwritten by another instruction result; (b) Silent Store instruction (`stores`), whereby the old and new written data to a given address of memory are the same; (c) Silent Load instruction (`loads`), whereby the old and new read data from a given address of memory and written to a destination register are the same; (d) Silent Register instruction (`sir` and `sfp`), is defined as an integer- or floating point-based register operation that produces the same result as the existing one in the destination register; (e) Silent Conditional Move instruction (`cmov`), which is an unsatisfied and not performed conditional move operation (i.e. behaving like a null operation).

Figure 4 shows the amount of wasted energy by each class of unnecessary instructions relative to the total amount of processor energy usage (in percentage), for an out-of-order Alpha processor. The configuration of this processor is presented in Table 1. The processor was simulated for 300 million committed instructions using the conjunction of SMTSIM and Wattch simulators. Meanwhile, the employed benchmarks were chosen from SPEC2000 bench set. These results don't encompass the wasted energy by data generators (i.e. parent instructions) of these instructions. The `all` category includes all classes of unnecessary instructions. A number of solutions have been presented for this root of energy waste that work based on dynamically identifying, predicting, and eliminating unnecessary instructions.

Parameter	Value
Fetch bandwidth	8 instructions per cycle
Functional Units Instruction Queues	3 FP, 6 Int (4 load/store) 64-entry FP, 64-entry Int
Inst Cache	64KB, 2-way, 64-byte lines
Data Cache	64KB, 2-way, 64-byte lines
L2 Cache (on-chip)	1 MB, 4-way, 64-byte lines
Latency (to CPU)	L2 18 cycles, Memory 300 cycles
Pipeline depth	8 stages
Min branch penalty	6 cycles
Branch predictor	21264 predictor
Instruction Latency	Based on Alpha 21164

Table 1. The examined processor configuration

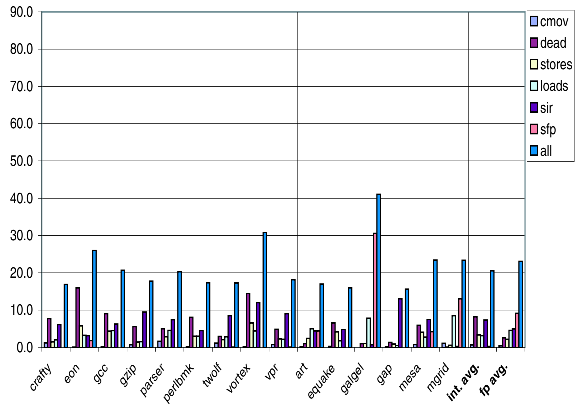


Figure 4. Wasted energy by unnecessary instructions

Speculation waste arises from wrong speculative fetch and execution of instructions due to mis-predicted branches. These instructions are entered into pipeline and consume resources, without being committed and changing processor state. Figure 5 shows the percentage of wasted energy by mis-speculated instructions. Possible solutions for this problem are: (a) improving the accuracy of branch predictors; (b) pipeline gating (i.e. keeping instructions from continuing down a processor pipeline once a low-confidence branch has been fetched); and (c) multithreading execution of instructions.

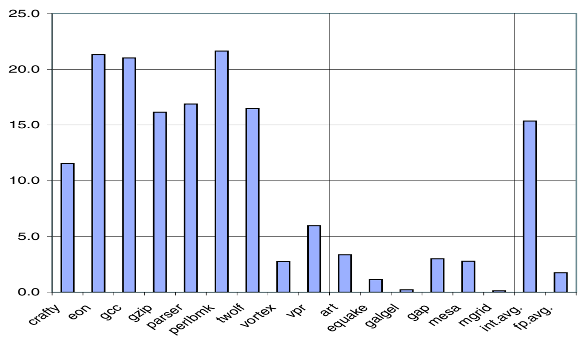
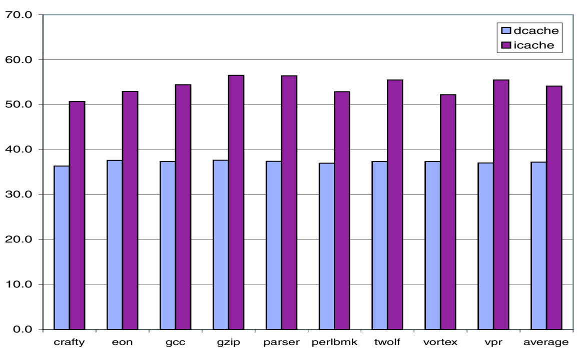
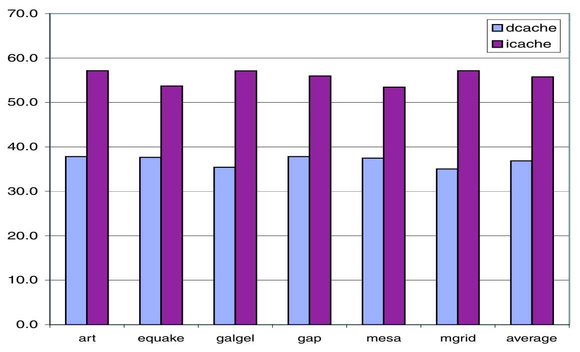


Figure 5. Wasted energy by mis-speculated instructions

Architectural waste is originated from oversizing of architectural structures, such as caches, instruction queues, branch predictors, and etc. Regularly, architectural structures are designed with large capacity for performance improvement. However, their capacities might be much larger than the enough amounts for exploitation of a given program. This causes extreme power dissipation due to retention of large amount of data, which might not be used in the next phases of program. Possible solutions for this problem can be mentioned as: (a) reconfigurable cache structures; (b) power downing a cache parts that are unused; (c) dynamically reconfiguration of the window size of issue queue; and (d) hierarchical design of cache, issue queue, and register file. Figure 6 shows the percentage of saved energy by an adaptive reconfigurable (AR) cache in compare to a fixed cache.



(a) Simulation by integer-based benchmarks



(b) Simulation by floating point-based benchmarks

Figure 6. Saved energy by AR cache

IV. Conclusion

The limiting factor for the functionality of current and future electronic portable and wearable devices is power dissipation. Due to the ever increase in computational complexities of these devices and more demand for boosting performance, low power design can be challenging and effective techniques are required to achieve a trade-off between these two design parameters. In this survey, low power design is studied from architectural perspective. There are three roots of power dissipation at architecture-level: program waste, speculation waste, and architectural waste.

These roots of energy waste are originated from unnecessary instructions, mis-speculated instructions, and oversized architectural structures respectively. The simulation results show that program waste can cause nearly 37.7% and 47.0% energy misusage for integer- and floating point-based operations. Mis-speculated instructions can cause more than 15% energy misusage for integer-based operations, while less than 2.5% for floating point-based operations. Leveraging adaptive reconfiguration cache can help to save energy almost 55% and 37% for integer- and floating point-based operations. While it is a good demonstration for a significant opportunity in low power design, it also indicates the requirement for design and development of energy efficient architectural mechanism and structures.

V. References

[1] Ernst, Dan, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler et al. "Razor: A low-power pipeline based on circuit-level timing speculation." In Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on, pp. 7-18. IEEE, 2003.

[2] Ghose, Kanad, and Milind B. Kamble. "Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation." In Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on, pp. 70-75. IEEE, 1999.

[3] Shiue, Wen-Tsong, and Chaitali Chakrabarti. "Memory exploration for low power, embedded systems." In Proceedings of the 36th annual ACM/IEEE Design Automation Conference, pp. 140-145. ACM, 1999.

[4] Burd, Thomas D., and Robert W. Brodersen. "Energy efficient CMOS microprocessor design." In System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on, vol. 1, pp. 288-297. IEEE, 1995.

[5] Su, Ching-Long, Chi-Ying Tsui, and Alvin M. Despain. "Low power architecture design and compilation techniques for high-performance processors." In Compcon Spring'94, Digest of Papers., pp. 489-498. IEEE, 1994.

[6] Pedram, Massoud. "Power minimization in IC design: principles and applications." ACM Transactions on Design Automation of Electronic Systems (TODAES) 1, no. 1 (1996): 3-56.

[7] Seng, John S., and Dean M. Tullsen. "Architecture-Level Power Optimization-What Are the Limits." Journal of Instruction-Level Parallelism 7, no. 7 (2005): 3.

[8] Athas, William C., Lars J. Svensson, Jeffrey G. Koller, Nestoras Tzartzanis, and Eric Ying-Chin Chou. "Low-power digital systems based on adiabatic-switching principles." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 2, no. 4 (1994): 398-407

[9] Scott, Jeff, Lea Hwang Lee, John Arends, and Bill Moyer. "Designing the Low-Power M• CORE TM Architecture." In Power Driven Microarchitecture Workshop, pp. 145-150. 1998.

[10] Luo, Yan, Jia Yu, Jun Yang, and Laxmi Bhuyan. "Low power network processor design using clock gating." In Proceedings of the 42nd annual Design Automation Conference, pp. 712-715. ACM, 2005.

[11] Burd, Thomas D., Trevor Pering, Anthony J. Stratakos, and Robert W. Brodersen. "A dynamic voltage scaled microprocessor system." Solid-State Circuits, IEEE Journal of 35, no. 11 (2000): 1571-1580.

[12] Lee, Seongsoo, and Takayasu Sakurai. "Run-time voltage hopping for low-power real-time systems." In Proceedings of the 37th Annual Design Automation Conference, pp. 806-809. ACM, 2000.

[13] Mathew, Binu, Al Davis, and Mike Parker. "A low power architecture for embedded perception." In Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems, pp. 46-56. ACM, 2004.

[14] Pouwelse, Johan, Koen Langendoen, and Henk Sips. "Dynamic voltage scaling on a low-power microprocessor." In Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 251-259. ACM, 2001.

[15] Hans, Martin. "Architectural aspects of design for low static power consumption." PhD diss., Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2004.