System-level Early Power Estimation for Memory Subsystem in Embedded Systems

Jinsong Ji[1,2+] Chao Wang[1,2] Xuehai Zhou[1,2]

[1]Dept. of Computer Science
University of Science and Technology of China
Hefei, Anhui, 230027, China
jsji@mail.ustc.edu.cn
saintwc@mail.ustc.edu.cn

[2]Embedded System Laboratory
Suzhou Institute for Advanced Study,USTC
Suzhou, Jiangsu, 215223,China
xhzhou@ustc.edu.cn

Abstract

Early power estimation is important to guide architectural design, especially for embedded systems. Since the power consumption of memory subsystem dominates, while DRAM and NAND Flash are the two main storage mediums nowadays, we analyze the power model of DRAM and propose a power model for NAND Flash, considering its system-level behaviors. Experimental results show that the accuracy of model proposed can be up to 95% .

1. Introduction

Nowadays, power consumption is already one of the leading constraints in designing embedded systems[8]. And memory system cost is a major factor in the total power consumption[12, 7].A share of up to 80% of the total power has been attributed to the memory subsystem in the signal processing domain [2]. So system designers are paying more and more attention to memory subsystem powers.

In order to create power efficient designs, accurate power budgets for the memory subsystem are essential, whether it is for calculating battery life, planning cooling system, or determining the power supply.

Though there have been several different power estimation techniques[6, 9, 15, 16, 3, 12, 11] and different related tools, such as SimplePower[14], Wattch[1], JouleTrack[13], VPR[10] et al; most of them focus on micro-architecture level power estimation. System designers usually could not utilize these techniques or tools in early design stage, because they may not have enough micro-architecture information, or enough time to run a whole simulation for the design. Another problem is that most of the tools focus on processors or controllers, memory subsystems are ignored,to say nothing about memory hierarchies.

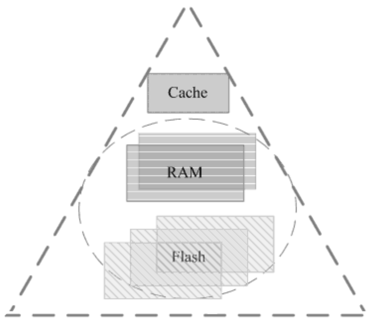


Figure 1. Typical Memory Hierarchy

In this paper, we propose a mathematical power model for typical memory subsystem in embedded systems, which can help designers gain accurate power budgets in early stage of architecture design, requiring only the knowledge of system level behaviors and related medium parameters. We first analyze the main components of modern memory hierarchy, then describe the power model of DRAM provided by Micron[5] and propose our model of NAND Flash in details; after that we show our experimental results which verified the models presented.

2 System-level Power Models

As shown in Fig.1, the typical memory hierarchy of today's embedded systems consists of three or more type of memory: Cache, RAM, Flash et al.The power consumption of Cache is usually calculated within controllers or processors.Therefore, we would like to focus on RAM and Flash in the following sections. As to RAM, though there are several kinds of RAM, such as SRAM, DRAM, VRAM et al,

978-0-7695-3348-3/08 $25.00 © 2008 IEEE
DOI 10.1109/SEC.2008.48

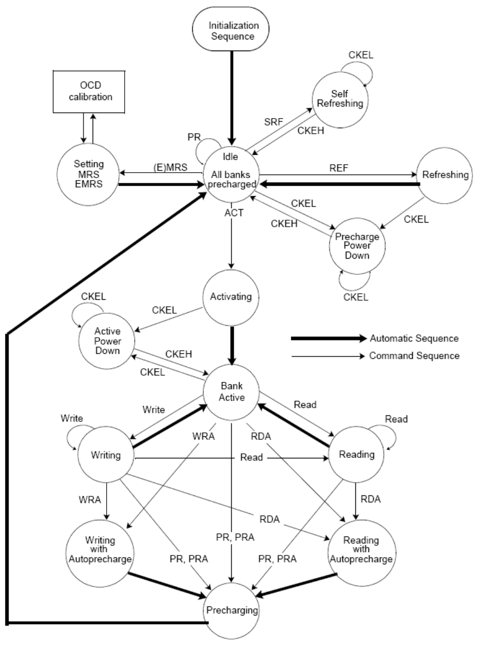


Figure 2. Typical State Machine of DRAM[4].

most of them are more or less working in the same mechanism; and the DRAMs are the most prevalent ones, so we decided to analyze the DRAM power model as a representative. As for Flash, many current designs are moving towards NAND Flash to take advantage of its higher density and lower cost, so we choose to model NAND Flash instead of NOR Flash.

2.1 DRAM Power Model

According to the working mechanisms of DRAM, its power consists of three key components: background power, activate power, read/write power.

2.1.1 Background Power

As shown in Fig.2, CKE signal is the master on-off switch of DRAM. When CKE is LOW(CKEL), the DRAM goes into PowerDown state; and when it is HIGH(CKEH), the DRAM goes into Active state. The current changes during these transitions even without read/write access. So we can calculated these background powers easily.

$$\begin{aligned}
P_{pre-pdn} &= I_{pre-pdn} \times V_{DD} \\
P_{act-pdn} &= I_{act-pdn} \times V_{DD} \\
P_{pre-stby} &= I_{pre-stby} \times V_{DD} \\
P_{act-stby} &= I_{act-stby} \times V_{DD}
\end{aligned}$$

Meanwhile, DRAM have to refresh in the background. We assume that the device is in precharge power-down state most time except when the actual REFRESH commands are executed. Thus, the average extra power for refresh command is:

$$P_{REF} = (I_{REF} - I_{pre-pdn}) \times V_{DD}$$

2.1.2 Activate Power

All DRAM banks have to be activated (ACT) first before read/write access. After the ACT command, a large amount of current is used to decode the command/address and transfer data from the DRAM array to sense amplifiers.Once this is complete, the DRAM is maintained in an active state and draws $I_{act-pre}$ until a PRE command is issued. The PRE command restores the data from the sense amplifiers into the memory array and resets the bank for the next ACT command. Once this is complete, the device is returned to precharge state. This cycle is then repeated at tRC intervals between ACT commands. Notice that CKE is always held HIGH during this period, we have to subtract the background current it draws. Therefore, the activate power is:

$$P'_{ACT} = (I_{act-pre} - I_{act-stby}) \times V_{DD}$$

But as DRAM may work in lower clock frequency or bank interleave mode, the interval between two ACT commands is not always tRC. Fortunately, it is easy to scale the ACT current for other modes of operation.

$$P_{ACT} = (I_{act-pre} - I_{act-stby}) \times \frac{n_{READ}}{n_{ACT}} \times V_{DD}$$

2.1.3 Read/Write Power

During read/write access, assume that the currents are I_{read} and I_{write} respectively. The the read/write powers can be calculated as follows:

$$\begin{aligned}
P_{READ} &= (I_{READ} - I_{act-stby}) \times \frac{n_{READ}}{n_{ACT}} \times V_{DD} \\
P_{WRITE} &= (I_{WRITE} - I_{act-stby}) \times \frac{n_{WRITE}}{n_{ACT}} \times V_{DD}
\end{aligned}$$

But the equations above are not the complete answer. To drive the outputs, additional DQ currents are required.

$$\begin{aligned}
P_{perDQ} &= V_{OUT} \times I_{OUT} \\
P_{DQ} &= P_{perDQ} \times (n_{DQ} + n_{DQS}) \times \frac{n_{READ}}{n_{ACT}}
\end{aligned}$$

2.1.4 Total Power of DRAM

After showing the basic components above, it is now in place to calculate the total power. Considering different usage conditions, the parameters as in Table 1 are needed for

Table 1. Parameters for DRAM Model

Symbol	Comments	Typical Values
p_{pre}	Percentage of time all anks are precharged	40%
p_{ckpre}	Percentage of the PRE time when CKE is LOW	50%
p_{ckact}	Percentage of the ACT time when CKE is LOW	0%
t_{ACT}	The average time between ACT commands	120 ns
p_{READ}	Percentage of CK cycles that output read data	1%
p_{WRITE}	Percentage of CK cycles that input write data	24%

input. The total power of DRAM is:

$$P_{total} = P_{bg} + P_{ACT} + P_{R/W}$$

where:

$$\begin{aligned} P_{bg} &= P_{pre-pdn} \times p_{pre} \times p_{ckpre} \\ &+ P_{pre-stby} \times p_{pre} \times (1 - p_{ckpre}) \\ &+ P_{act-pdn} \times (1 - p_{pre}) \times p_{ckact} \\ &+ P_{act-stby} \times (1 - p_{pre}) \times (1 - p_{ckact}) \\ &+ P_{REF} \\ P_{R/W} &= (P_{READ} + P_{DQ}) \times p_{READ} \\ &+ P_{WRITE} \times p_{WRITE} \end{aligned}$$

Most of the parameters in the equations can be found in DRAM's datasheet, others such as read/write percentage can be easily estimated in early design stage.

2.2 NAND Flash Power Model

Compared to DRAM model, NAND Flash are simpler. In standby state, the Flash could not be accessed, the current $I^*_{standby}$ is usually small enough to be ignored. In operation state, as shown in Fig.3, a typical flash operation includes at least three stages: command, address, data. Notice that the command and address stages are almost the same for all operations (READ/PROGRAM/ERASE), except for different command codes, we denote them as cmd+addr stage.

2.2.1 Read Power

The read operation consists of only two major stages, so the average power can be calculated according to their percentages:

$$\begin{aligned} P^*_{READ} &= P^*_{READ_{cmd+addr}} \times p^*_{READ_{cmd+addr}} \\ &+ P^*_{READ_{data}} \times p^*_{READ_{data}} \end{aligned}$$

[1]We consider only the typical interface defined by ONFI(Open NAND Flash Interface).

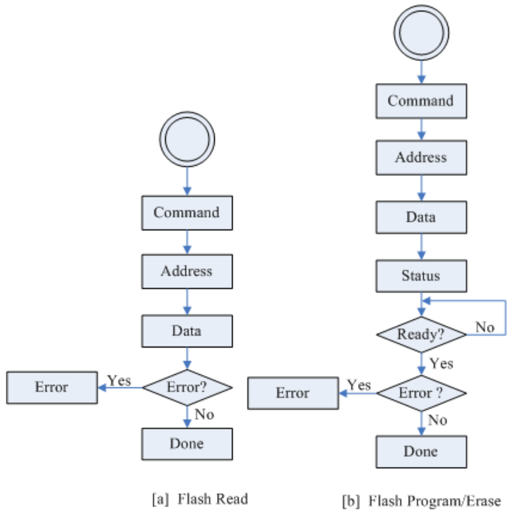


Figure 3. Flash Operation Diagram

where

$$\begin{aligned} P^*_{READ_{cmd+addr}} &= I^*_{WRITE} \times V^*_{DD} \\ P^*_{READ_{data}} &= I^*_{READ} \times V^*_{DD} \\ &+ P^*_{perDQ} \times (n^*_{DQ} + n^*_{DQS}) \\ P^*_{perDQ} &= V^*_{OUT} \times I^*_{OUT} \end{aligned}$$

Here, V^*_{DD} is the maximum voltage which can be found in the datasheet. I^*_{WRITE} , I^*_{READ} , V^*_{OUT} , I^*_{OUT} , are also provided by vendor. $p^*_{READ_{cmd+addr}}$ and $p^*_{READ_{data}}$ are the percentage of time consumed in each stage. They could be calculated easily from the timing diagrams, and are almost constant. n^*_{DQ} and n^*_{DQS} are the pin number of outputs and their strobes.

2.2.2 Program Power

For PROGRAM and ERASE operations, we have to include the stage of status, when some specific bits of status registers are read to confirm the success of operation.

$$\begin{aligned} P^*_{PROG} &= P^*_{PROG_{cmd+addr}} \times p^*_{PROG_{cmd+addr}} \\ &+ P^*_{PROG_{data}} \times p^*_{PROG_{data}} \\ &+ P^*_{PROG_{status}} \times p^*_{PROG_{status}} \end{aligned}$$

where

$$\begin{aligned} P^*_{PROG_{cmd+addr}} &= I^*_{WRITE} \times V^*_{DD} \\ P^*_{PROG_{data}} &= I^*_{WRITE} \times V^*_{DD} \\ &+ P^*_{perDQ} \times (n^*_{DQ} + n^*_{DQS}) \\ P^*_{PROG_{status}} &= I^*_{READ} \times V^*_{DD} \\ &+ P^*_{READ} \times p^*_{polling} \end{aligned}$$

Table 2. Parameters for NAND Flash Model

Symbol	Comments	Typical Values
p^*_{READ}	Percentage of READ operation	1%
p^*_{PROG}	Percentage of PROGRAM operation	24%
p^*_{ERASE}	Percentage of ERASE operation	24%

Here all parameters have the same meanings with READ operation, except for $p^*_{polling}$. $p^*_{polling}$ is the ratio of data polling during the whole programming progress, it depends on the software implementation. If user uses R/B pins to determine the end of operation, then there would be only one polling operation; otherwise user has to read the status register time and time again to check whether the programming is still in progress. The maximum number of polling is $k = \lfloor t^*_{status}/t^*_{polling} \rfloor$, where t_{status} is the total time of status checking and $t_{polling}$ is the time of one single data polling. So,

$$p^*_{polling} = \frac{t^*_{polling} \times k}{t^*_{status}} \text{ where } k \in [1, \lfloor t^*_{status}/t^*_{polling} \rfloor].$$

2.2.3 Erase Power

ERASE operation is mostly the same as PROGRAM, except that ERASE doesn't have data stage. So,

$$\begin{aligned} P^*_{ERASE} &= P^*_{ERASE_{cmd+addr}} \times p^*_{ERASE_{cmd+addr}} \\ &+ P^*_{ERASE_{status}} \times p^*_{ERASE_{status}} \end{aligned}$$

where

$$\begin{aligned} P^*_{ERASE_{cmd+addr}} &= I^*_{WRITE} \times V^*_{DD} \\ P^*_{ERASE_{status}} &= I^*_{READ} \times V^*_{DD} \\ &+ P^*_{READ} \times p^*_{polling} \end{aligned}$$

2.2.4 Overall NAND Flash Power

Considering the parameters in Table 2 for input, we have:

$$\begin{aligned} P^*_{total} &= P^*_{READ} \times p_{READ} + P^*_{PROG} \times p_{PROG} \\ &+ P^*_{ERASE} \times p_{ERASE} \end{aligned}$$

Now we can estimate the total power of NAND Flash with only the knowledge of operation percentages.
What is more, with these two key component models, we could estimate the power of whole memory subsystem, requiring only the hierarchy and system-level information.

3 Experimental Results

For evaluation, we use Xilinx's Vertex-II Pro Development Board for DRAM model and Altera's Cyclone Development Board for NAND Flash model.

Table 3. Measured Power data of DDR DRAM

Operations	Assembly Code	Binary Code	Current (A)	Power (W)
Only Bootloop			0.87	4.35
No Memroy Access	BRI 0	B8000000	0.9	4.5
Write all "0"	ADDIK r3,r0,0	30600000	1.21	6.05
	IMM 10240	B0002800		
	SWI r3,r0,0	F8600000		
	BRI -8	B800FFF8		
Write all "1"	ADDIK r3,r0,-1	3060FFFF	1.16	5.8
	IMM 10240	B0002800		
	SWI r3,r0,0	F8600000		
	BRI -8	B800FFF8		
Read	IMM 10240	B0002800	1.35	6.75
	LWI r3,r0,0	E8600000		
	BRI -8	B800FFF8		

[a]The Working Voltage of Development Board is 5 V

3.1 DRAM Model

We developed benchmarks in assembly within Xilinx EDK, debugged them with XMD and measured the current. First, we downloaded the bootloop program into its embedded CPU, measured the current of board without DRAM module to gain the basic power for all other modules on board. Then we installed the DRAM module and downloaded a benchmark with one single loop, without accessing DRAM, to measure the background power of DRAM. After that, we downloaded benchmarks reading and writing DRAM to gain the related powers. Finally, we got the results in Table 3 and related powers calculated below:

$$\begin{aligned} P_{bg} &= (4.5 - 4.35)/8\,W = 18.75mW \\ P_{READ} &= (6.75 - 4.5)/8\,W = 281.25mW \\ P_{WRITE} &= (6.05 + 5.8)/2 - 4.5)/8\,W = 178.125mW \end{aligned}$$

Meanwhile, we got 12.9mW, 229 mW and 187.6 mW respectively from the DRAM model. Fig. 4 shows the comparison of two data sets, indicating that the accuracy of DRAM model could reach 95%.

3.2 NAND Flash Model

In order to verify the NAND Flash power model, we also developed benchmarks in NIOS II IDE. We used the C APIs provided by Altera's flash driver to access the flash, then collected the current data in Table 4, and calculated its powers below:

$$P^*_{READ} = (3.78 - 3.69)\,W = 90mW$$

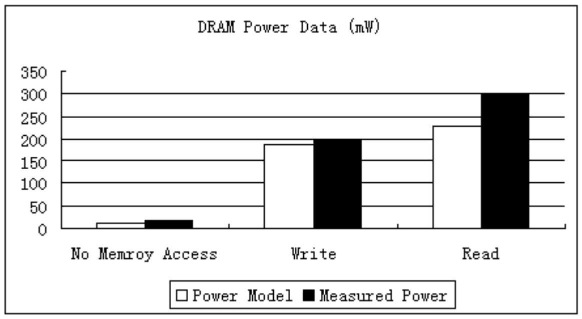


Figure 4. Comparison of DRAM Power

$$P^*_{WRITE} = (3.96 - 3.69)\,W = 270mW$$
$$P^*_{ERASE} = (3.96 - 3.69)\,W = 270mW$$

Meanwhile, we got 142.3mW, 217.3mW and 217.3mW from the NAND Flash model. The measured data is not so good as DRAM model, but the accuracy is still above 80%. After examining the data carefully, we found out that the accuracy is somehow affected by the precision of our measuring instruments and the working voltage of Development Board: the working voltage is 9 V, while the precision of current measuring instruments is 0.01 A under 9 V , which indicates that we can't get power data between 180 mW and 270 mW. Another reason may be that the software driver in NIOS II IDE is too complicated, it may cause extra powers during execution.

4 Summary and Conclusions

This paper analyzes a mathematical power model of DRAM and presents another one for NAND Flash. These models estimate the total power of memory subsystem, requiring only the percentage of different operations and the parameters provided by memory medium datasheets . Experimental results show that the accuracy of them are 95% and 80%. The accuracy of Flash model was affected by the precision of our measure instruments. Compared against other power estimation techniques and tools, the models are simpler but accurate enough. With these two models, system designers can make an early power estimation easily and accurately without too much effort. Refining the model of NAND Flash by considering more about the microoperations is one of our future work.

Acknowledgements

This article was supported by 863 Research and Development Program. And we would like to thank Xilinx Inc. for providing the Development Board and Mr. Wang Hengcai for providing measure devices.

Table 4. Measured Power data of NAND Flash

Operations	Current(A)	Power(W)
Only Bootloop	0.37	3.33
No Memroy Access	0.41	3.69
Read	0.42	3.78
Erase	0.44	3.96
Program	0.44	3.96

[a]The Working Voltage of Development Board is 9 V

References

[1] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA*, pages 83–94, 2000.

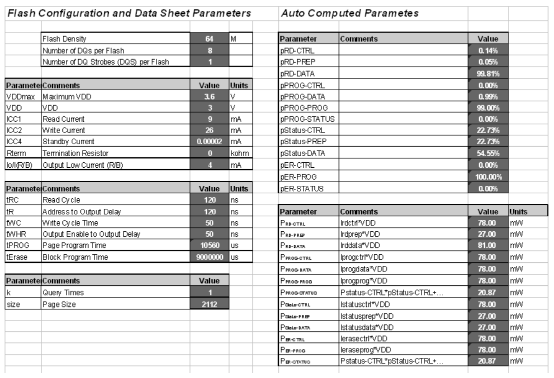
[2] F. Catthoor, F. Franssen, S. Wuytack, L. Nachtergaele, and H. D. Man. Global communication and memory optimizing transformations for low-power signal processing systems. In *VLSI Signal Processing Workshop*, pages 178–187, Oct. 1994.

[3] J. Coburn, S. Ravi, and A. Raghunathan. Power emulation: a new paradigm for power estimation. page 700, 2005.

[4] H. Corp. DDR2 SDRAM device operation and timing diagram. *DateSheet*, 2005.

[5] J. Janzen. Calculating memory system power for DDR SDRAM. *Designline*, 10(2), 2001.

[6] K. Kang, J. Kim, H. Shim, and C.-M. Kyung. Software power estimation using ipi(inter-prefetch interval) power model for advanced off-the-shelf processor. In *GLSVLSI '07: Proceedings of the 17th great lakes symposium on Great lakes symposium on VLSI*, pages 594–599, New York, NY, USA, 2007. ACM Press.



Flash Configuration and Data Sheet Parameters

Flash Density	64	M
Number of DQs per Flash	8	
Number of DQ Strobes (DQS) per Flash	1	

Paramete	Comments	Value	Units
VDDmax	Maximum VDD	3.6	V
VDD	VDD	3	V
ICC1	Read Current	9	mA
ICC2	Write Current	26	mA
ICC4	Standby Current	0.00002	mA
Rterm	Termination Resistor	0	kohm
Io/I(R/B)	Output Low Current (R/B)	4	mA

Paramete	Comments	Value	Units
tRC	Read Cycle	120	ns
tR	Address to Output Delay	120	ns
tWC	Write Cycle Time	50	ns
tWHR	Output Enable to Output Delay	50	ns
tPROG	Page Program Time	10560	us
tErase	Block Program Time	9000000	us

Paramete	Comments	Value
k	Query Times	1
size	Page Size	2112

Auto Computed Parametes

Parameter	Comments	Value
pRD-CTRL		0.14%
pRD-PREP		0.05%
pRD-DATA		99.81%
pPROG-CTRL		0.00%
pPROG-DATA		0.99%
pPROG-PROG		99.00%
pPROG-STATUS		0.00%
pStatus-CTRL		22.73%
pStatus-PREP		22.73%
pStatus-DATA		54.55%
pER-CTRL		0.00%
pER-PROG		100.00%
pER-STATUS		0.00%

Parameter	Comments	Value	Units
$P_{RD-CTRL}$	Irdctrl*VDD	78.00	mW
$P_{RD-PREP}$	Irdprep*VDD	27.00	mW
$P_{RD-DATA}$	Irddata*VDD	81.00	mW
$P_{PROG-CTRL}$	Iprogctrl*VDD	78.00	mW
$P_{PROG-DATA}$	Iprogdata*VDD	78.00	mW
$P_{PROG-PROG}$	Iprogprog*VDD	78.00	mW
$P_{PROG-STATUS}$	Pstatus-CTRL*pStatus-CTRL+...	20.87	mW
$P_{Status-CTRL}$	Istatusctrl*VDD	78.00	mW
$P_{Status-PREP}$	Istatusprep*VDD	27.00	mW
$P_{Status-DATA}$	Istatusdata*VDD	27.00	mW
$P_{ER-CTRL}$	Ierasectrl*VDD	78.00	mW
$P_{ER-PROG}$	Ieraseprog*VDD	78.00	mW
$P_{ER-STATUS}$	Pstatus-CTRL*pStatus-CTRL+...	20.87	mW

Figure 5. Model Screenshot

[7] L. Kruse, E. Schmidt, G. Jochens, A. Stammermann, A. Schulz, E. Macii, and W. Nebel. Estimation of lower and upper bounds on the power consumption from scheduled data flow graphs. *IEEE Trans. Very Large Scale Integr. Syst.*, 9(1):3–15, 2001.

[8] T. N. Mudge. Power: A first class design constraint for future architecture and automation. In *HiPC*, pages 215–224, 2000.

[9] M. Onouchi, T. Yamada, K. Morikawa, I. Mochizuki, and H. Sekine. A system-level power-estimation methodology based on ip-level modeling, power-level adjustment, and power accumulation. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 547–550, New York, NY, USA, 2006. ACM Press.

[10] K. K. W. Poon, S. J. E. Wilton, and A. Yan. A detailed power model for field-programmable gate arrays. *ACM Trans. Des. Autom. Electron. Syst.*, 10(2):279–302, 2005.

[11] E. Schmidt, G. Jochens, L. Kruse, F. Theeuwen, and W. Nebel. Automatic nonlinear memory power modelling. In *DATE '01: Proceedings of the conference on Design, automation and test in Europe*, page 808, Piscataway, NJ, USA, 2001. IEEE Press.

[12] E. Schmidt, G. von Cölln, L. Kruse, F. Theeuwen, and W. Nebel. Memory power models for multilevel power estimation and optimization. *IEEE Trans. Very Large Scale Integr. Syst.*, 10(2):106–108, 2002.

[13] A. Sinha and A. Chandrakasan. Jouletrack - a web based tool for software energy profiling. In *Design Automation Conference*, pages 220–225, 2001.

[14] N. Vijaykrishnan, M. T. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. In *ISCA*, pages 95–106, 2000.

[15] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 554–557, New York, NY, USA, 2006. ACM Press.

[16] L. Zhong, S. Ravi, A. Raghunathan, and N. K. Jha. Rtl-aware cycle-accurate functional power estimation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(10):2103, 2006. 0278-0070.