面向低功耗优化设计的系统级功耗模型研究

李 曦,王志刚,周学海,王煦法
(中国科学技术大学计算机系,安徽合肥 230026)

摘 要: 随着嵌入式系统应用的普及,低功耗设计成为系统设计中的关键问题之一. 本文提出了一个两层构架的系统级功耗模型,包括微体系结构层模型和体系结构层模型. 微体系结构模型支持系统级硬件结构设计优化,体系结构模型则针对编译器的软件设计优化. 微结构模型以部件的结构信息特征为依据,指令级模型以微结构模型为基础. 试验证明,该模型可以满足嵌入式系统的高层设计要求.

关键词: 嵌入式系统; 低功耗优化; 系统级功耗模型
中图分类号: TP302. 7 **文献标识码**: A **文章编号**: 0372-2112 (2004) 02-0205-04

Study on System-Level Power Model for Low Power Optimization

LI Xi, WANG Zhi-gang, ZHOU Xue-hai, WANG Xu-fa
(*Dept. of Computer Science, University of Science and Technology of China, Hefei, Anhui 230026, China*)

Abstract: With the popularization of embedded systems, low power design has become one of the most challenging tasks in the embedded system development. This paper presents a novel two-level power estimation model operating at the system-level, which including a microarchitecture level model to support hardware optimization and an instruction level model to support software compiling optimization. The microarchitecture level model based upon the information of components structure. The instruction level model based upon the microarchitecture model. Thus the proposed methodology provides an accurate and rapid model to evaluate embedded system high level design.

Key words: embedded system; low power optimization; system-level power model

1 引言

低能耗设计问题正在成为嵌入式系统软硬件设计中所关注的重点. 嵌入式系统的空间、能量和重量往往都受到限制. 在电池供电的系统中,减少设备的能量消耗以延长电池使用时间,或者在保证相同的使用时间的前提下降低电池的容量和重量都是非常有意义的. 常规技术主要以微电子工艺为出发点,辅以可变电压、可变频率等手段,为系统进行降耗控制提供必要的接口. 然而,软件优化降耗会有数量级的贡献. 针对同一任务,所选择的算法不同或采用不同的实现方式,不仅性能有差别,能耗也大不一样. 因此在进行系统优化设计时,除了代码的规模和执行性能之外,功耗也是一个需要认真考虑的问题. 低功耗优化技术需要一个详细的描述系统能耗特征的功耗模型作为系统功耗评估和指导优化变换的依据. 由于嵌入式系统设计方法的特殊性,要求在前期设计中,在系统级(体系结构层、微结构层)就进行系统的低功耗优化,以便降低设计成本,缩短产品上市时间.

本文将在分析系统级功耗特征的基础上,提出一种新的系统级功耗模型,以支持系统硬件结构优化和软件优化.

2 系统级功耗模型的建立

嵌入式系统的硬件平台通常包括处理器、片内 Cache、片上存储器,以及各种 I/O 功能部件. 综合国外研究情况,当前系统级功耗模型的建立有几种途径:对于商品处理器,可以利用实验的手段(如 Tiwari[1,2]的方法),通过测量各条指令以及指令对的执行能耗而建立其功耗模型,这种模型通常为"平均功耗模型". 使用这种方式建立的模型无法区分系统内部的重点功耗源,也就无法指导系统体系结构层的设计. 在采用自底向上方法进行系统设计的设计流程中,可以基于电路级/门级/RTL 级仿真器进行模拟以获得各个组件的功耗(从版图反标而得). 这类模型可以为周期精确模型,即其所描述的功耗为系统各个节拍的功耗. 但是由于电路级参数与实现工艺相关,且电路层仿真速度很慢,这种模型不适合自顶向下的设计流程中的高层分析应用.

为了兼顾目标代码优化与体系结构设计优化的需求,我们采用两层结构模型建立系统级功耗模型的方案:底层为微结构层,基于部件的功耗模型,上层为指令级功耗模型,基于体系结构层特征. 部件功耗用于知道系统体系结构设计优化,

收稿日期:2003-02-17;修回日期:2003-05-30
基金项目:国家自然科学基金(No. 60273042);安徽省自然科学基金(No. 03042101)

指令级功耗模型用于指导编译器进行低功耗指令选择和调度.

3 部件级功耗模型的特征

部件级功耗模型用于描述系统各个模块(如存储器,cache,ALU)的功耗,是在系统级对软硬件协同设计结果进行评估的基础,并且能够指导系统体系结构的设计,同时,它也是建立指令级功耗模型的基础.

对于 CMOS 电路,其主要功耗由各节点电容的充放电造成,其基本功耗模型为[3]:

$$P = 0.5\, C_L V_{dd}^2 A f$$

其中 C 为负载电容,A 为电路开关活动因子,f 为电路工作频率,$C \times A$ 又称为开关电容. 该模型说明对于给定的电路其功耗是由电路的开关活动决定的,而开关活动又是由电路的前一个输入和当前输入决定的.

一种直观而简单的建模方法是 Huzefa 等人在文[4]中提出的前输入相关模型. 这种方法建立了一张以前一输入和当前输入为索引的开关电容表,在仿真时通过查表得出每次输入所引起的开关电容. 这种功耗模型能够提供相当精确的评估,但是它非常复杂以至于很难实现,如果一个部件有 n 个输入,采用这种模型需要建立和查找一张 2^{2n} 大小的表,这个工作量是相当惊人的.

我们在建立功耗属性库时采用了文[5]中提出的基于部件结构信息的功耗建模方法,简称为"部件信息模型". 这种方法对结构信息明确的功能部件的建模比较准确,而且也极大的缩小了查找表的长度,如表 1 所示. 下面以一个 1 位的 2 选 1 的多路选择器(图 1)为例分析部件信息模型的建模原理.

表 1 部件信息模型实验数据表

功能部件	查找表长度		平均功耗比较(μW)		
	前输入相关模型	部件信息模型	SPICE	部件信息模型	误 差
Mux21	64	16	264.22	257.68	- 2.47 %
Mux41	4096	256	355.81	348.12	- 2.16 %
Mux81	4194304	16384	1033	980.21	- 5.11 %
1-bitALU	65536	484	1157.9	1084.1	- 6.37 %
2-bitALU	1048576	15376	2715.8	2644.37	- 3.45 %
4-bit Fast Adder	262144	26244	3060.4	2997.26	- 2.06 %

观察图 1 可以发现各个输入并非直接作用于输出,而是经过中间节点的翻转后,间接作用于输出的. 例如:D0,SD 的变化先影响到部件 G1 和 G3 间的信号 a,再影响到最终的结果. 通过以上分析,可以通过在真值表中加入中间信号为该多路选择器建立一张扩展的真值表,如表 2. 对比真值表中深色的两行,不难发现这两行的输入虽然不同但是输出和中间信号都是相同的,换言之当输入由 000 切换至 010 时内部信号和输出不发生任何翻转,因此,可以把这两个输入归为一组. 根据同样的分析,可以把 001 和 101、011 和 111、100 和 110 分别归为一组,这样就可以将查找表的长度由原来的 64 变为 16 了. 该实例分析说明,这种部件信息模型既缩短了查找表长度,又克服了 Mehta 等人在文[6]中提出的基于簇的功耗分析方法的一些缺陷,是建立部件级功耗模型的一种可行的选择.

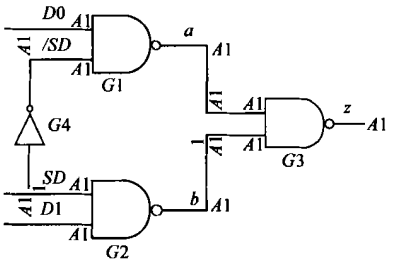


图 1 2 选 1 的多路选择器

表 2 1 位的 2 选 1 的多路选择器扩展真值表

D0	D1	SD	A	b	/ SD	Z
0	0	0	1	1	1	0
0	0	1	1	1	0	0
0	1	0	1	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	1
1	0	1	1	1	0	0
1	1	0	0	1	1	1
1	1	1	1	0	0	1

有了 1 位的 2 选 1 的多路选择器的功耗数据后,就能够对不同的 2 选 1 的多路选择器进行分析,因为更多位的 2 选 1 的多路选择器都是有这种多路选择器扩展而得. 这种方法为周期精确的微结构仿真器 simplepower[7] 所使用,其平均误差为 13 %.

4 指令级功耗模型的特征

指令级功耗模型是编译器对应用程序代码进行低功耗优化的基础. 指令级功耗模型以系统的 ISA 为切入点,描述了指令序列执行时的功耗特征. 对指令级而言,每条指令的执行涉及到数据通路上的一系列相关部件,对应于其电路的开关活动. 不同的操作码和寻址方式以及地址和数据的编码方式,确定了每条指令的基本功耗. 指令的基本功耗为其执行时相关数据通路上各个功能部件的功耗之和. 对于一个程序,其功耗由静态(基本)功耗和动态功耗两部分构成. 静态功耗为程序中所有指令的基本功耗之和,动态功耗由程序执行时指令间的相互影响造成,是与电路状态变化有关的附加运行开销,它包括各种流水线停顿、Cache 状态以及访存操作的影响.

为了说明的指令级功耗分析方法,我们对 simplepower 所仿真的体系结构进行建模. SimplePower 基于一个五级流水线的体系结构,这五级流水线分别是:取指(IF),译码(ID),执行(EXE),访存(MEM)和写回(WB),其指令集是 SimpleScalar 指令集的一个子集(整数操作集). 限于篇幅,在此仅就 IF 阶段(其结构如图 2 所示)的功耗(用开关电容表示)进行分析,其他几个阶段的分析方法与此类似,只是涉及更多的部件.

根据体系结构的定义,IF 阶段将完成的功能包括:根据

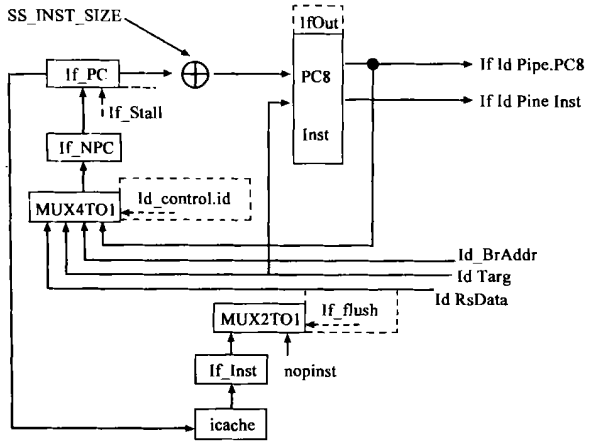


图 2　流水线 IF 阶段的结构

PC 值从存储器中取出指令;根据流水线控制信号 If_FLUSH 选择将取出的指令或者 NOP 指令放入流水线存储器文件 IfOut 中的 inst 寄存器;将 PC 值增加指令长度后放入 IfOut 中的 PC8 寄存器,供 ID 阶段计算分支地址时使用;根据控制信号从 IfOut 中的 PC8、分支地址 IdBrAddr 等候选地址中选择正确的地址放入 NPC 中;根据流水线 STALL 与否决定是否将 NPC 写入 PC 中.因此,根据 IF 阶段的结构可得该阶段的功耗,应该包括以下几个部分:

(1) 访存取指功耗.从存储系统中取指所产生的开关电容 IF_Fetch_$C(w_n)$.由于不同的指令其编码不同,取指时引起的总线上的切换会有所不同,所以它的功耗与具体的指令相关,对于该部件采用分类统计平均值的方法计算:

$$\text{IFFetch\_}C(w_n) = \text{IF\_Fetch\_SUM}(w_n) / \text{Ins\_}N(w_n)$$

其中 IF_Fetch_SUM(w_n)为该部件在执行 w_n 时的总功耗,Inst_N(W_n)为该部件在执行 w_n 指令的总条数.

(2) 指令选择功耗.为了对流水线的相关进行处理,IF 阶段有一个 FLUSH 选择器,它用于选定下一条指令.如果有 If_FLUSH 信号,它将选择 NOP 指令放入流水线寄存器中,否则它选择根据 PC 取出的指令放入流水线寄存器中.它的功耗与具体的指令编码相关,对于该部件采用分类统计平均值的方法计算,如下式:

$$\text{IF\_FLUSH\_Mux\_}C(w_n) = \text{IF\_FLUSH\_Mux\_}C\text{\_SUM}(w_n) / \text{Inst\_}N(w_n)$$

其中 IF_FLUSH_Mux_C_SUM(w_n)为该部件在执行 w_n 时的总功耗,Inst_$N(w_n)$为该部件在执行 w_n 指令的总条数.

(3) 计算 PC8 的操作功耗.对 PC 加上指令长度时加法器所消耗的开关电容 IF_PCAdder_$C(w_n)$.该部分能耗与具体指令无关,用平均值代替.

$$\text{IF\_PCAdder\_}C(w_n) = \text{IF\_PCAdder\_}C\text{\_SUM} / \text{Inst\_}N$$

其中 IF_PCAdder_C_SUM 为运行某测试程序的过程中该部件消耗的总的开关电容,Inst_N 为执行指令的总数.

(4) 选定 NPC 的多路选择器所产生的开关电容 IF_NextPC_Mux_$C(w_n)$.该部分能耗与具体指令的关联关系很难把握,用平均值代替.

$$\text{IF\_NextPC\_Mux\_}C(w_n) = \text{IF\_NextPC\_Mux\_}C\text{\_SUM} / \text{Inst\_}N$$

其中 IF_NextPC_Mux_C_SUM 为运行某测试程序的过程中该部件消耗的总的开关电容,Inst_N 为执行指令的总数.

(5) 寄存器 NPC 的值写入 PC 时所产生的开关电容 IF_Pipereg_$C(w_n)$.该部分能耗与具体指令无关,用平均值代替.但是 STALL 信号直接影响到该部件的功耗,因此其功耗表达如下:

$$\text{IF\_Pipereg\_}C(w_n) = A_{\text{IF\_Pipereg\_}C} + m_{\text{IF\_Pipereg\_}C} \times p_{\text{IF\_Pipereg\_}C} \times S_{\text{IF\_Pipereg\_}C}$$

其中 $A_{\text{IF_Pipereg_}C}$为正常运行某测试程序的过程中该部件消耗的平均开关电容,$m_{\text{IF_Pipereg_}C}$为发生 STALL 时的平均停顿周期,$p_{\text{IF_Pipereg_}C}$为平均发生 STALL 的概率,可以通过统计执行的总周期数和 STALL 发生的周期计算出 $p_{\text{IF\_Pipereg\_}C}$,$S_{\text{IF_Pipereg_}C}$为发生 STALL 时的平均功耗.

综上所述,IF 阶段在执行某条指令时的开关电容可由下式表述:

$$\text{IF\_}C(w_n) = \text{IF\_NextPC\_Mux\_}C(w_n) + \text{IF\_Pipereg\_}C(w_n) + \text{IF\_FLUSH\_Mux\_}C(w_n) + \text{IF\_PCAdder\_}C(w_n) + \text{IF\_Fetch\_}C(w_n)$$

5　实验结果

基于上文的分析,我们建立了指令级功耗模型评估器 xpPower.xpPower 可以统计出指令级功耗模型所需的有关参数,包括某部件在执行某条指令时的平均功耗,STALL 时的平均功耗,STALL 的概率等.我们以 bubble(对 100 个随机数进行冒泡查找)、heap(对 100 个数进行堆查找)、quick(对 100 个数进行快速查找)、perm(对 7 个数进行置换排序)等四个程序为测试激励,得到了程序的总功耗和各条指令的功耗(图 3、图 4).

图 3 是利用实验中抽取的指令级功耗模型对测试程序进行评估的结果.为了说明实验结果的有效性,我们将评估结果与 SimplePower 的评估结果进行了比较,误差在 10 %以内.可见,用这种功耗模型指导编译优化是可行的.

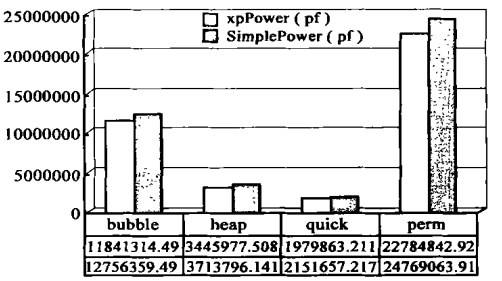


图 3　各个测试程序的总功耗评估结果

从流水线停顿与否的角度来看,可以把指令的功耗分为停顿的功耗和正常运行的功耗.正常运行的功耗在部件层功耗已知的情况下比较好统计,因为在一条指令正常运行时,其数据通路上哪些部件被激活是一定的,只要通过部件层功耗模型统计出被激活的部件功耗就可以得到指令正常运行时的功耗.不过,对于周期精确的部件层功耗模型来说,指令功耗

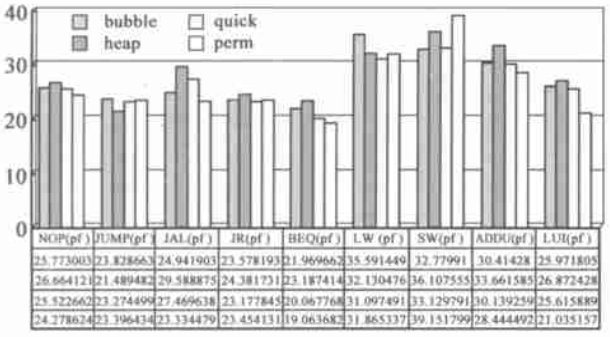
应该与前一条指令相关的,因为指令不同,部件的前一个输入也就不同,功耗也就不同.而系统运行时精确的停顿功耗难于统计,因为它不仅与当前指令有关,而且与前几条指令有关,甚至还与 CPU 的存储系统有关.直观考虑,为了得到一个准确的功耗模型,可以建立一个指令的功耗的函数,该函数与当前指令、前几条指令、存储系统、调度策略都相关.但是这样做一方面问题的复杂度会迅速提高,例如,对于一个指令集为 n 条指令的 CPU 如果仅对当前指令建模,复杂度为 n.如果建模时不仅考虑当前指令,而且考虑了前一条指令,复杂度就会变为 $n2$.另一方面,有些影响停顿功耗的因数是很难量化的,如调度策略.由于上述原因,本文没有考虑指令间的前后关系,以及调度策略等,而是以平均值表示了指令正常运行时的功耗,以概率的形式将停顿的功耗分布在各条指令功耗中:

$$E(w_n) = A(w_n) + T(w_n)$$

$$T(w_n) = m(w_n) \times p(w_n) \times S(w_n)$$

其中,p 表示停顿/延时概率,m_s 为执行 w_n 时出现停顿/延时的典型周期数,S_s 为每个停顿/延时周期的能耗.使用这种方法对指令功耗进行建模时,无论从建模难度还是从评估难度上都会大大降低问题的复杂度.但这种方法也有其不足之处,它使得指令功耗的精确度有所下降.

在图 4 中列出了部分在实验中得出的指令功耗数据.分析这些数据我们发现从不同测试程序中得到的各条指令的功耗虽然基本相同,但还是存在一定的差别.这就是由于没有考虑指令前后关系造成的,不同的程序指令的前后关系不一样,这就使得该指令停顿的概率、部件输入的切换都不同,进而造成功耗的不一样.为了使指令的功耗数据更准确,我们采用对应用程序进行分类的方法,即在对某类程序进行优化前,以典型的程序(指令的前后关系基本一致)作为测试程序,统计出其功耗模型后用于指导该类程序的优化.



NOP(pf)	JUMP(pf)	JAL(pf)	JR(pf)	BEQ(pf)	LW(pf)	SW(pf)	ADDU(pf)	LUI(pf)
25.773003	23.828663	24.941903	23.578193	21.969662	35.591449	32.77991	30.41428	25.971805
26.664121	21.489482	29.588875	24.381731	23.187414	32.130476	36.107555	33.661585	26.872428
25.922662	23.274499	27.469638	23.177845	20.067768	31.097491	33.129791	30.139259	25.615889
24.278624	23.396434	23.334479	23.454131	19.063682	31.865337	39.151799	28.444492	21.035157

图 4 不同测试程序中的各个指令功耗的对比

6 总结

嵌入式系统的应用对传统的设计方法提出了挑战,在其软硬件设计空间搜索(DSE)的过程中,设计者必须仔细研究不同的设计选择对系统功耗的可能影响,而不是仅仅使用性能作为唯一的设计约束,因此迫切需要支持系统级设计的高层辅助工具,以缩短系统的研制时间.本文提出了一个用于嵌入式系统低功耗设计优化的一种两层构架的系统级功耗模型,以支持系统的体系结构和软件低功耗优化.通过实验比较,该模型的精度可以满足系统高层设计优化的要求.同时,我们在部件级功耗仿真器 SimplePower 的基础上,实现了指令级功耗仿真器 xpPower,为进一步实现系统级低功耗设计环境奠定了基础.

参考文献:

[1] V Tiwari, S Malik, A Wolfe. Power analysis of embedded software: A first step towards software power minimization[J]. IEEE Trans. On VLSI Systems, 1994, 12: 384 - 390.

[2] V Tiwari, S Malik, A Wolfe, M Lee. Instruction level power analysis and optimization of software[J]. Journal of VLSI Signal Processing, Kluwer Academic Publishing, Boston, 1996: 1 - 18.

[3] F N Najm. Transition density: A new measure of activity in digital circuits[J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and System, 1993, 12(2): 310 - 323.

[4] Huzefa Mehta, Robert M. Owens, Mary J Irwin. Instruction-level power profiling[A]. Proc International Conference on Acoustics, Speech and Signal Processing[C]. Atlanta, Georgia, May 1996. 3326 - 3329.

[5] J Lin, W Shen, J Jou. A power modeling and characterization method for macro-cells using structure information[A]. IEEE/ACM International Conference on Computer-Aided Design[C]. San Jose, California, 1997. 502 - 506.

[6] H Mehta, R Owens, M J Irwin. Energy characterization based on clustering[A]. In Proc. DAC-33: ACM/IEEE Design Automation Conf[C]. Las Vegas, NV, June 1996. 702 - 707.

[7] Ye W, Vijaykrishnan N, Kandemir M, Irwin M J. The design and use of simple power: A cycle-accurate energy estimation tool[A]. Design Automation Conference, 2000[C]. Los Angeles, California, Proceedings 2000, 2000. 340 - 345.

作者简介:

李 曦 男,1963 年 8 月生于北京,高工,研究方向:软硬件协同设计方法.

王志刚 男,1978 年 2 月生于云南,博士生,研究方向:SOC 体系结构设计方法.