# 基于ARM处理器的嵌入式软件能耗统计模型

刘啸滨[1]，郭 兵[1]，沈 艳[2]，朱 建[1]，王继禾[1]，伍元胜[1]

(1. 四川大学计算机学院 成都 610065; 2. 成都信息工程学院控制工程学院 成都 610225)

【摘要】提出了一种嵌入式软件能耗的统计模型，包括处理器、存储器和I/O控制器等硬件单元产生的能耗，然后通过分析ARM指令周期数的规律，设计了指令周期数的相应计算方法，该方法能够快速地计算软件运行时处理器产生的能耗。在高精度指令级嵌入式软件能耗模拟器HMSim中进行了模型实现。实验结果表明，该模型的能耗计算结果与实际仪器测量结果的误差在10%以内，可较准确地反映软件实现方式对系统能耗的影响程度。

**关 键 词** 嵌入式软件能耗； 嵌入式系统； 能耗统计模型； 处理器能耗

**中图分类号** TP302 **文献标识码** A doi:10.3969/j.issn.1001-0548.2012.05.024

# ARM-Based Embedded Software Statistical Energy Model

LIU Xiao-bin[1], GUO Bing[1], SHEN Yan[2], ZHU Jian[1], WANG Ji-he[1], and WU Yuan-sheng[1]

(1. School of Computer Science & Engineering, Sichuan University Chengdu 610065;

2. School of Control Engineering, Chengdu University of Information Technology Chengdu 610225)

**Abstract** The energy consumption of embedded software has become a key factor in embedded system design, and it is a fundamental work of analysis and optimization of energy consumption to measure the energy consumption of embedded software. This paper proposes a statistical model of embedded software energy consumption, including the energy consumptions of processor, memory and I/O controller, etc. Then, by analyzing the instruction cycle of ARM instruction set, a method for computing the instruction cycle number is designed so as to rapidly calculate the processor energy consumption of embedded software. This model has implemented in a high-precision instruction-level energy simulator HMSim. The experimental results show that the error rate of embedded software energy consumption estimated by this model is less than 10% compared with that measured by an electronic instrument. The achieved results also reflect how the embedded system energy consumption is influenced by different software designs.

**Key words** embedded software energy consumption; embedded system; energy statistical model; microprocessor energy consumption

在目前全球大力倡导“低碳经济”的背景下，随着嵌入式计算机系统(简称嵌入式系统)大量而广泛的使用，嵌入式系统消耗的电力能源及相关能耗优化技术是一个不容忽视的问题，已受到业内软/硬件开发商和政府的高度重视[1-3]。

嵌入式系统由嵌入式硬件和嵌入式软件组成，是典型的软件驱动执行的系统。硬件的电路活动“直接”导致了系统能耗的产生，软件的指令执行和数据存取等操作驱动了底层硬件的电路活动，“间接”导致了系统能耗的产生，这也是嵌入式软件能耗的本质含义。许多研究表明，不同的汇编指令、源程序结构、软件算法和软件体系结构造成硬件不同的工作方式，从而进一步影响系统能耗。嵌入式系统能耗可分为硬件能耗和软件能耗，软件能耗是指在软件运行期间，软件驱动直接相关硬件(包括处理器、存储器和I/O控制器等)活动产生的能耗总和，无软件运行时硬件产生的其他系统能耗一般归为硬件能耗，如处理器在空闲状态下某些硬件单元保持上电状态而产生的能耗以及I/O控制器无软件访问时产生的能耗等。因此，将系统能耗明确划分为软件能耗和硬件能耗，更便于直接观察软件实现方式(如数据结构和算法)对系统能耗的影响程度。

嵌入式软件能耗的度量是开展嵌入式软件能耗分析与优化的基础性工作。度量软件能耗最直接的

收稿日期：2010 – 11 – 17； 修回日期：2011 – 01 – 17

基金项目：国家自然科学基金(61272104, 61073045)；四川省杰出青年科技基金(2010JQ0011)；中国科学院计算技术研究所计算机体系结构国家重点实验室开放课题(ICT-ARCH201003)

作者简介：刘啸滨(1987 – )，男，博士生，主要从事嵌入式实时系统方面的研究.

方法是使用电子仪器(如功率计)进行测量，由于需要将软件运行在硬件平台上进行测量，所以在硬件开发完成之前无法测量软件能耗。另一种方法是采用能耗模拟器测量软件能耗，通过计算机软件模拟某一特定嵌入式硬件系统，并采用能耗统计模型，在软件运行的同时统计软件能耗，可灵活、方便地获得嵌入式软件的能耗值。

国际上对于嵌入式软件能耗统计模型的研究历史并不长，文献[4-5]首先提出了对软件能耗进行分析的基本概念，建立了基本的指令级能耗模型，并提出了指令电流的测试方法；文献[6]测量了基于StrongARM SA-1100处理器的ARM指令集电流值，提出了软件能耗的统计方法；文献[7]在测得的指令电流值基础上，构建了基于StrongARM SA-1100处理器的指令级能耗模拟器EMSIM(embedded strongARM energy simulator)，并采用EMSIM对嵌入式软件能耗进行了分析工作；文献[8]提出了对操作系统采用服务例程级功耗建模，并具体分析了Embedded Linux操作系统共50种服务例程的功耗IPC关联模型；文献[9]利用微体系结构能耗模型估算单时钟周期指令能耗，提出了基于软件功能结构的操作系统内核能耗估算模型。在上述能耗统计模型中，主要统计了软件执行时在处理器上产生的单条指令能耗，未计算指令对、存储器、I/O控制器等产生的能耗。因此，本文针对ARM7TDMI处理器，综合考虑软件执行时在嵌入式系统各主要硬件单元产生的能耗，设计了一种基于指令周期数的处理器能耗计算方法。

# 1　嵌入式软件能耗的统计模型

嵌入式系统能耗的一般模型可表示为：

$$E=\int_{t_1}^{t_2}P(t)\mathrm{d}t=PT=IVT \tag{1}$$

式中，$E$表示能耗；$P(t)$表示瞬态功率；$P$表示平均功率(或功耗)；$I$为平均电流；$V$为工作电压；$T$为运行时间，$T=N/f$，$N$为时钟周期数，$f$为时钟频率。因此，能耗又可表示为：

$$E=IVN/f \tag{2}$$

嵌入式软件一般采用汇编语言或高级语言(如C、C++、Java等)编写，经过编译器编译成目标机可执行的二进制指令序列，这些指令在执行时将对不同的硬件单元产生影响。考虑嵌入式系统的硬件组成，嵌入式软件能耗可分为处理器(MPU)能耗、存储器能耗、I/O控制器能耗(如总线控制器、UART控制器、LCD控制器、网卡控制器等)和其他硬件单元产生的能耗，即：

$$E_{\mathrm{software}}=E_{\mathrm{mpu}}+E_{\mathrm{mem}}+E_{\mathrm{io}}+E_{\mathrm{other}} \tag{3}$$

式中，$E_{\mathrm{software}}$表示嵌入式软件能耗；$E_{\mathrm{mpu}}$表示处理器能耗；$E_{\mathrm{mem}}$表示存储器能耗；$E_{\mathrm{io}}$表示I/O控制器能耗；$E_{\mathrm{other}}$表示嵌入式系统主板上其他硬件单元的能耗，如Ethernet控制器等部件功耗，此部分能耗在软件执行时能耗值一般较小，可忽略不计。

## 1.1　指令及指令对能耗计算

软件执行时单条指令在MPU上产生的能耗，称为指令能耗(instruction energy consumption)，记作$E_{\mathrm{instr}}$；软件执行时相邻指令间相互影响在MPU上产生的额外能耗，称为指令对能耗(或指令间能耗，inter-instruction energy consumption)，记作$E_{\mathrm{inter}}$。指令对能耗产生的主要原因包括：1) 上一条指令的执行结果作为下一条指令的操作数；2) 不同指令执行导致电路状态的频繁翻转。因此，MPU能耗包括指令能耗和指令对能耗，即：

$$E_{\mathrm{mpu}}=E_{\mathrm{instr}}+E_{\mathrm{inter}} \tag{4}$$

根据式(2)，软件总的指令能耗为：

$$E_{\mathrm{instr}}=\sum_{i=1}^{i=n}\frac{I_iN_iV_{\mathrm{mpu}}}{f_{\mathrm{mpu}}} \tag{5}$$

式中，$I_i$表示指令$i$执行时MPU平均电流；$N_i$为执行指令$i$所需的时钟周期数；$V_{\mathrm{mpu}}$为MPU的工作电压；$f_{\mathrm{mpu}}$为MPU的时钟频率；$n$为软件的指令总条数。

根据式(2)，软件总的指令对能耗为：

$$E_{\mathrm{inter}}=\sum_{i=2}^{i=n}\frac{\Delta I_{i,i-1}N_iV_{\mathrm{mpu}}}{f_{\mathrm{mpu}}} \tag{6}$$

式中，$\Delta I_{i,i-1}$表示指令$i$和前一条指令$(i-1)$形成的平均指令对电流。

$N_i$的计算方法将在第2节介绍。$I_i$的测量方法是循环执行一条指令，当电流表的值恒定时，即为执行该指令时MPU的平均电流值[4]。

指令对电流$\Delta I_{i,i-1}$的计算方法是循环执行一对指令，并将循环执行该对指令测得的电流减去该对指令电流的分权平均值，即为该对指令的平均指令对电流。一对指令的电流分权平均值计算公式为：

$$I_{\mathrm{ever}}=(I_iN_i+I_{i-1}N_{i-1})/(N_i+N_{i-1}) \tag{7}$$

式中，$I_{\mathrm{ever}}$表示一对指令的电流分权平均值；$I_i$和$I_{i-1}$分别是指令$i$和$i-1$的电流值；$N_i$和$N_{i-1}$分别是指令$i$和$(i-1)$运行所需的MPU时钟周期数。如果循环执行一对指令时测得的MPU平均电流值记为$I_{\mathrm{meas}}$，则指令对电流$\Delta I$可表示为：

$$\Delta I = I_{\text{meas}} - I_{\text{ever}} \tag{8}$$

### 1.2 存储器和I/O控制器能耗计算

根据式(2)，存储器能耗可表示为：

$$E_{\text{mem}} = \frac{I_{\text{mem}} V_{\text{mem}} N_{\text{mem}}}{f_{\text{mem}}} \tag{9}$$

式中，$I_{\text{mem}}$表示存储器在有数据读/写访问时的平均电流值；$V_{\text{mem}}$表示存储器的工作电压；$f_{\text{mem}}$表示存储器的时钟频率；$N_{\text{mem}}$表示软件运行时MPU访问存储器总的时钟周期数。

I/O控制器的能耗可分为总线控制器能耗、UART控制器能耗、LCD控制器能耗、网卡控制器能耗等，即：

$$E_{\text{io}} = E_{\text{bus}} + E_{\text{uart}} + E_{\text{lcd}} + E_{\text{net}} \tag{10}$$

下面以LCD控制器为例，说明I/O控制器能耗的统计方法。

LCD控制器能耗主要包括LCD控制器设置寄存器的能耗、LCD控制器访问显示内存的能耗、LCD控制器控制和输出图像数据的能耗。

LCD控制器设置寄存器的能耗可表示为：

$$E_{\text{lcd\_register}} = \frac{I_{\text{lcd}} V_{\text{lcd}} N_{\text{lcd\_register}}}{f_{\text{bclk}}} \tag{11}$$

式中，$E_{\text{lcd_register}}$表示LCD控制器设置寄存器的能耗；$I_{\text{lcd}}$表示软件操作LCD时LCD控制器的平均电流；$V_{\text{lcd}}$表示LCD控制器的电压；$f_{\text{bclk}}$表示数据总线时钟频率，$N_{\text{lcd_register}}$表示软件运行时LCD控制器设置寄存器总的时钟周期总数。

LCD控制器访问显示内存的能耗为：

$$E_{\text{lcd\_framebuffer}} = \frac{I_{\text{lcd}} V_{\text{lcd}} N_{\text{lcd\_framebuffer}}}{f_{\text{mlcd}}} \tag{12}$$

式中，$E_{\text{lcd_framebuffer}}$表示LCD控制器访问显示内存的能耗；$f_{\text{mlcd}}$表示存储器时钟频率；$N_{\text{lcd_framebuffer}}$表示软件运行时LCD控制器访问显存总的时钟周期总数。

LCD控制器控制和输出图像数据的能耗为：

$$E_{\text{lcd\_data}} = \frac{I_{\text{lcd}} V_{\text{lcd}} N_{\text{lcd\_data}}}{f_{\text{pclk}}} \tag{13}$$

式中，$E_{\text{lcd_data}}$表示LCD控制器控制和输出图像数据的能耗；$f_{\text{plcd}}$表示像素时钟频率；$N_{\text{lcd_data}}$表示软件运行时LCD控制器控制和输出图像数据总的时钟周期总数。

式(9)～(13)中，$I_{\text{mem}}$和$I_{\text{lcd}}$的值可采用电流表直接测试获得。

## 2 ARM处理器指令周期数计算

由式(5)和式(6)可知指令周期数是计算指令能耗和指令对能耗的必要数据，本文以ARM系列处理器中ARM7TDMI为例，设计了指令周期数的计算方法，并在高精度指令级嵌入式软件能耗模拟器HMSim中进行了实现，可快速获得准确的指令周期数，如图1所示。图中方框代表具体的方法步骤，箭头代表上层功能模块调用下层功能模块。

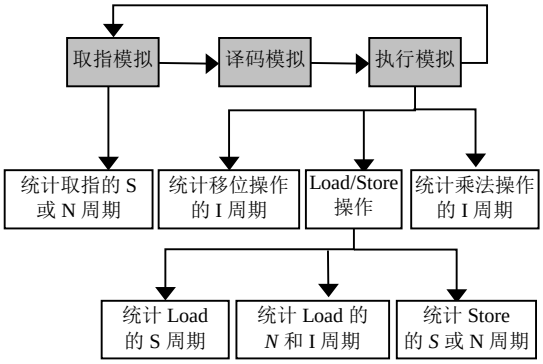


图1 指令周期数计算方法

ARM7TDMI处理器的指令周期包括下列4种类型：N周期(nonsequential cycle非连续周期)、S周期(sequential cycle连续周期)、I周期(internal cycle内部周期)和C周期(coprocessor register transfer cycle协处理器传输周期)。通常情况下，一个N周期、S周期、I周期或C周期均为MPU的一个时钟周期。ARM7TDMI指令在时序上是这4种周期的组合，部分指令的指令周期数如表1所示，其中，$n$表示传输字的数目。

**表1 ARM7TDMI的指令周期数**

| 指令类型 | 周期计数值 | 限制条件 |
|---|---|---|
| 数据处理指令 | $S$ | 单周期 |
| 数据处理指令 | $S + I$ | 寄存器相关移位 |
| 数据处理指令 | $2S + N$ | R15为目标 |
| 数据处理指令 | $2S + N + I$ | R15为目标，寄存器相关移位 |
| B, BL | $2S + N$ | |
| LDR | $S + N + I$ | 非R15目标 |
| LDR | $2S + 2N + I$ | R15为目标 |
| LDM | $(n)S + N + I$ | 非R15目标 |
| LDM | $(n+1)S + 2N + I$ | R15为目标 |

依据表1所示，指令周期数的计算方法如下：

1) 考虑ARMT7TDMI的三级流水线(取指、译码和执行)和指令周期的关系，可发现：① 大部分指令都包含了至少一个S周期，可认为是指令取指时消耗的一个时钟周期；② 观察B、BL和可修改PC值的

指令，其共同特点是可改变MPU的流水线。这些指令在改变流水线时，将比不改变流水线时多出一个S周期和一个N周期。可认为当这类指令执行时，流水线上已预取的指令和已译码的指令将无效，需要重排流水线，此时MPU将多消耗一个S周期和一个N周期。基于以上特点，在指令周期数计算方法中，若流水线顺序执行，则在MPU取指时统计一个S周期；若流水线重排，在MPU第一次取指时统计一个N周期，后续取指则统计S周期。

2) 在数据处理指令中，如果第二个操作数为寄存器移位寻址方式，则该指令的执行将多一个I周期，在指令周期数计算方法中统计一个I周期。

3) 乘法指令周期中都包含了一个S周期和多个I周期，由于在指令周期数计算方法中取指时统计了一个S周期，因此，在乘法指令执行时只统计I周期。这些I周期是有规律的，在指令周期数计算方法中可根据这些规律设计函数统计I周期数。

4) 观察Load/Store指令的指令周期数，Load/Store指令包括单寄存器操作指令LDR/STR和多寄存器操作指令LDM/STM。在指令周期数上，单寄存器操作指令可视为多寄存器操作指令的一种特殊情况，即多寄存器操作指令传送一个字的数据时，其指令周期数等同于单寄存器操作指令。如LDR和LDM指令，当LDM指令传送一个字的数据时，其指令周期数等同于LDR指令；当LDM指令传送$n$个字的数据时，比LDR指令多$(n-1)$个S周期。在LDR指令执行时或LDM指令传送第一个字的数据时，在指令周期数计算方法中统计一个N周期和一个I周期；在LDM指令传送后续$(n-1)$个字的数据时，在指令周期数计算方法中统计$(n-1)$个S周期。Store指令特点与此类似。

上述分析的指令包含了ARM7TDMI指令集中所有的存储器访问指令、数据处理指令和跳转指令，是ARM7TDMI使用最频繁的指令。实验表明，依据以上分析统计的指令周期数与实际的指令周期数完全一致。而其他一些指令的指令周期数只在指令取指时统计一个S周期，由于这些指令出现的频率很低，因此对指令周期数的计算影响可忽略不计。

在完成了指令周期数的计算后，本文构造了一个完整的嵌入式软件能耗统计模型，综合考虑软件执行时在嵌入式系统各硬件单元产生的能耗，包括处理器、存储器和I/O控制器等单元。

## 3　仿真实验与分析

实验环境包括能耗模拟器平台HMSim、目标平台Winbond W90P71开发板、能耗测量仪器日置HIOKI3334功率计及交叉编译器采用的ARM ADS v1.2开发工具。

HMSim是一种高精度指令级嵌入式软件能耗模拟器，包括指令集模拟器(instruction set simulator，ISS)、I/O接口控制器模拟器和能耗统计模型，能够模拟ARM7TDMI处理器的指令执行、并进行相应能耗统计。首先，采用ARM交叉编译器将目标程序编译成ELF格式的文件，然后HMSim将文件中的指令和数据读入HMSim模拟的存储器中，逐条取出指令、模拟执行，在指令执行的同时根据能耗统计模型计算指令在执行过程中的能耗，最后以函数为单位输出能耗统计结果，包括函数名、函数的执行次数、函数运行消耗的指令周期数、指令能耗、指令对能耗以及UART控制器能耗和LCD控制器能耗。

为了验证本文提出的嵌入式软件能耗统计模型的有效性，首先在HMSim中实现了该模型，然后设计了5个典型C语言测试程序，进行能耗统计精度验证和能耗优化结果对比两个实验。

### 3.1　能耗统计精度验证

本文选取了C语言中的5个经典算法，包括冒泡排序(Bubble)、约瑟夫环(Josephus)、矩阵乘法(Matrix)、八皇后(Queen)和迷宫(Maze)算法，其中冒泡排序和约瑟夫环算法主要是验证循环结构体，矩阵乘法主要是验证乘法指令，八皇后算法验证了递归调用，迷宫算法采用了回溯法。测试结果如表2所示，第1列为程序名，第2列表示采用功率计实际测量在开发板上程序运行的能耗，第3列表示在HMSim上程序运行的能耗，其中误差的计算公式为：(E_HMSim–E_W90P71) / E_W90P71×100。

**表2　能耗统计精度验证**

| 程序名 | E_W90P71/nJ | E_HMSim/nJ | 误差/(%) |
|---|---|---|---|
| Bubble | 183 600 | 171 645.95 | –6.51 |
| Josephus | 2 925 | 2 984.01 | 2.02 |
| Matrix | 2 934 | 2 875.71 | –1.99 |
| Queen | 140 400 | 145 489.04 | 3.63 |
| Maze | 464 400 | 462 662.34 | –0.37 |

### 3.2　能耗优化结果对比

本文的实验主要是为了验证程序优化前后的能耗值能否在HMSim上有效地体现出来。同样使用上述5个C语言测试程序，分别对其进行O2级别的优化编译，在开发板和HMSim上运行，得到的能耗值优

化结果对比如图2所示。通过实验结果可以看出：

1) 使用能耗统计模型得出的能耗值与程序在目标板上运行得出的能耗值相比，误差在10%以内，说明本文提出的能耗统计模型具有较高的统计精度。

2) 能耗模拟器HMSim和目标板能耗优化结果的比例基本一致，表明采用本文提出的能耗统计模型得出的软件能耗优化比例和实际效果是一致的。因此，该能耗统计模型能够用于快速估计软件能耗，为软件能耗的分析与优化奠定重要的基础。

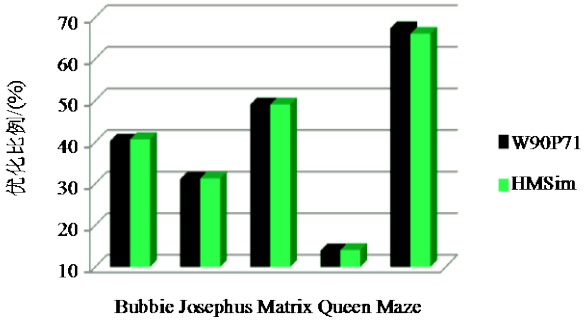


图2 两种平台能耗优化结果对比

## 4 总 结

本文提出了一种嵌入式软件能耗的统计模型，将嵌入式软件产生的能耗主要分为处理器能耗、存储器能耗和I/O控制器能耗，并通过理论分析与仿真实验验证了模型的可行性和实用性。该模型通过分析ARM指令集指令周期的规律，设计了相应指令周期数的计算方法，基于指令周期数可灵活地计算软件运行时产生的处理器能耗。同时，该模型计算的软件能耗与实际仪器测量结果的误差在10%以内，具有较高的统计精度，可用于软件开发早期阶段快速地估计软件的能耗。下一步的研究包括考虑其他I/O控制器和外设等硬件单元的嵌入式软件能耗统计模型，以进一步提高嵌入式软件能耗的统计精度和实用性。另外，本文的模型是基于ARM指令的功耗模型，而Intel Atom处理器在嵌入式系统中逐步得到应用，考虑到其未来在消费电子领域的巨大市场空间，将会对基于X86的软件能耗模型开展深一步的研究工作。

## 参 考 文 献

[1] 郭兵, 沈艳, 邵子立. 绿色计算的重定义与若干探讨[J]. 计算机学报, 2009, 32(12): 2311-2319.
GUO Bing, SHEN Yan, SHAO Zi-li. The redefinition and some discussion of green computing[J]. Chinese Journal of Computers, 2009, 32(12): 2311-2319.

[2] HE Zhi-hai, CHENG Wen-ye, CHEN Xi. Energy minimization of portable video communication devices based on power-rate-distortion optimization[J]. Circuits and Systems for Video Technology, 2008, 18(5): 596-608.

[3] 雷霆, 胡潇, 周学海. 系统级能耗优化的实时电压调度方法研究[J]. 系统工程与电子技术, 2006, 28(1): 153-158.
LEI Ting, HU Xiao, ZHOU Xue-hai. Real time dynamic voltage scheduling for system level energy reduction[J]. Aerospace Electronics Information Engineering and Control, 2006, 28(1): 153-158.

[4] TIWARI V, MALIK S, WOLFE A. Power analysis of embedded software:a first step towards software power minimization[J]. IEEE Transactions on VLSI Systems, 1994, 2(4): 437-445.

[5] TIWARI V, MALIK S, WOLFE A. Compilation techniques for low energy: an overview[C]//Proceedings of the IEEE Symposium on Low Power Electronics. San Diego: IEEE, 1994: 38-39.

[6] SINHA A, CHANDRAKASAM A P. JouleTrack-A Web based bool for software energy profiling[C]//Design Automation Conference. [S.l.]: [s.n.], 2001: 220-225.

[7] TAN K T, RAGHUNATHAN A, JHA N K. EMSIM: an energy simulation framework for an embedded operating system[C]//Proc ISCAS 2002. [S.l.]: [s.n.], 2002: 464-467.

[8] LI Tao, LIZY K J. Run-time modeling and estimation of operating system power consumption[C]//Proceedings of the ACM International Conference on Measurements and Modeling of Computer Systems. San Diego, USA: ACM, 2003: 160-171.

[9] 赵霞, 郭耀. 基于模拟器的嵌入式操作系统能耗估算与分析[J]. 电子学报, 2008, 36(2): 209-215.
ZHAO Xia, GUO Yao. Estimation and analysis of embedded operating system energy consumption[J]. Acta Electronica Sinica, 2008, 36(2): 209-215.

编 辑 税 红