低功耗嵌入式系统的分析与应用①

Analysis and Application of Low-Power Embedded Systems

张 炜 韩 进 (山东科技大学 信息科学与工程学院 山东 青岛 266510)

摘 要： 低功耗嵌入式系统设计的能量消耗问题是近几年来人们在嵌入式系统设计中普遍关注的难点与热点,它严重制约着嵌入式系统的应用及发展。以减少嵌入式系统所耗电能为目标,首先分析了嵌入式系统功耗主要来源确定解决目标，而后根据嵌入式系统功耗分析给出低功耗嵌入式系统模型以及模型各层降低功耗的具体方法，从而使嵌入式系统能充分发挥降低功耗的作用。最后，通过具体嵌入式系统平台实现系统级低功耗控制，实验结果证明可以有效降低嵌入式系统的功耗。

关键词： 嵌入式系统 低能耗硬件设计 低能耗软件设计 DVS DPM

1 引言

随着嵌入式系统应用的迅速发展，嵌入式系统市场处于迅速增长趋势。嵌入式系统一般是由电池来供给电能的,减少电能消耗不仅能延长电池的寿命,降低用户更换电池的周期,而且能带来提高系统性能与降低系统开销的好处,甚至能起到保护环境的作用。因此，功耗作为一个关键的设计要素越来越成为系统设计所要考虑的重要方面，降低功耗和提高设备续航能力是嵌入式系统设计的热点。

研究微处理器的低功耗设计技术,首先必须了解它的功耗来源[1]。从高层次仿真得出的高性能 CPU 的功率分布可以看出：时钟功耗比例最大约占 50%，包括时钟发生、驱动器、时钟树、锁存器和所有时钟负担的器件；数据通路的功耗仅次于时钟，约占 25%，主要包括执行单元、总线和寄存器文件；其次片上存储器功耗占 15%，它主要由存储器的大小以及存储阵列的电路和物理结构所决定；而控制单元和 I/O 的功耗通常占整个芯片功耗的一小部分。

从运用 CMOS 工艺的嵌入式微处理器来讲，指令执行功耗主要包括动态功耗、静态功耗和短路功耗三部分，即 P=Pd+Ps+PI，其中 Pd、Ps 和 PI 分别表示动态、静态和短路功耗。动态功耗在 CMOS 电路中起决定作用，它大约占全部功耗的 70%到 90%[2]，根据 Pd=Ns*Co*Vdd*f 可知，减小工作电压和工作频率可以减小动态功耗。

由以上分析可以得出，降低时钟功耗、数据通路功耗、片上存储器功耗、指令执行功耗成为嵌入式系统低功耗设计的主要目标。接下来我们将介绍相应的几种降低功耗方法：可变频率时钟、低耗能软件设计、DVS 策略、降低外围设备功耗的 DPM 策略。

2 嵌入式系统功耗分析

低功耗设计是一个复杂的综合性课题。就流程而言,包括功耗建模、评估以及优化等;就设计抽象层次而言,包括自硬件底层、操作系统层和应用程序层，如图 1 所示，其中目前低功耗设计大都在操作系统层实施。下面将根据低功耗系统设计模型各部分讨论相应的低功耗设计技术及具体应用。

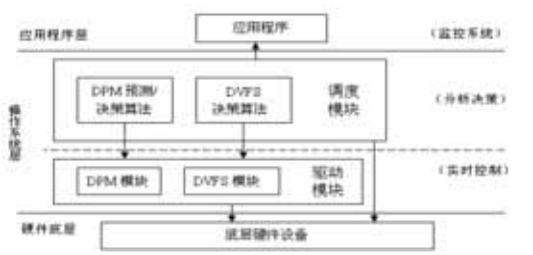

图 1 低功耗系统设计模型

① 基金项目:山东省研究生教育创新计划(SDYY08016);山东科技大学教育教学研究"群星计划"(qx0801119)
收稿时间:2009-03-01

2.1 低耗能硬件设计

低功耗硬件设计是嵌入式系统降低功耗的重要内容。因此，我们需要从系统内部结构设计、系统时钟设计和低功耗模式等几方面采用特定的方法来实现系统硬件节能设计。例如：门控时钟和可变频率时钟、低功耗单元库、低功耗状态机编码、cache 低功耗设计等方法。

时钟是惟一在所有时间都充放电的信号,而且很多情况下引起不必要的门的翻转,因此门控时钟和可变频率时钟的应用对于降低功耗最为有效。常用的可变频率时钟技术根据系统性能要求,配置适当的时钟频率以避免不必要的功耗。可变频率时钟比门控时钟技术更加有效,但需要系统内嵌时钟产生模块 PLL ,增加了设计复杂度。例如，在具有低功耗特性的嵌入式芯片上进行的时钟系统设计方案[3]如图 2 所示，片上时钟系统通过两个数字锁相环 CPUPLL 和 SysPLL 来稳定 16MHz 的输入时钟，分别送到不同的倍频器和分频器。经 CPUPLL 的时钟信号作为处理器内核时钟，经 SysPLL 的时钟信号作为处理器内核之外的系统时钟、存储器时钟和外设时钟。由于处理器芯片不支持电压动态调节，如：i.MX1，采用通过配置片内数字锁相环的方法可实现内核频率动态调节。

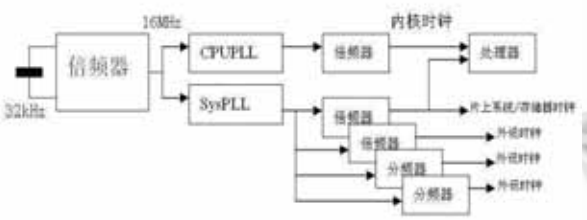

图 2　可变频率时钟

2.2 操作系统层

系统级低功耗设计一般是在操作系统层实现[4]。因为操作系统管理系统所有软硬件资源，并获取系统的各种状态信息，控制硬件设备的状态。因此，在操作系统中实现全局能耗控制是最佳选择。操作系统层面分成功耗驱动模块和功耗调度模块。

2.2.1 功耗驱动模块[5]

驱动模块实现相对简单，主要是对硬件操作。功耗模式转换和频率调节都是通过片上时钟系统控制寄存器进行设置。因此，实质上是对寄存器的设置。从睡眠模式或停止模式进入运行模式相对容易，只需向系统发出中断信号，系统唤醒进入运行模式；而从运行模式到睡眠模式或停止模式相对复杂，其关键代码如下：

```
enter doze-mode://清除全局时钟控制寄存器 GCCR 的//MPEN 位，关闭 CPUPLL
    mov r1, #0x0021B000
    mov r2, #0Xfffffffe
    ldr r3, [r1, #0x0]
    and r2, r2, r3
    str r2, [r1, #0x0] //将 CPU 设置为等待中断状态；
    mcr p15, 0, r1, c7, c0, 4
enter stop-mode://清除 GCCR 的 MPEN，UPEN 位，关闭//CPUPLL 和 SysPLL
    mov r1, #0x0021B000
    mov r2, #0xFFFFFFFC
    ldr r3, [r1, #0x0]
    and r2, r2, r3
    str r2, [r1, #0x0] //将 CPU 设置为等待中断状态；
    mcr p15, 0, r1, c7, c0, 4
change frequency: //设置 SysPLL 控制寄存器的 PD，MFD，//MFI 和 MFN 位，调节工作频率
    mov r1, # 0x0021B00C
    mov r2, [频率设置对应的值]
    ldr r3, [r1, #0x0]
```

2.2.2 功耗调度模块

其实现的关键技术在嵌入式 Linux 操作系统有具体体现[4]。在 Linux 操作系统中，任务的调度主要是由进程调度模块 schedule()完成。schedule()掌握系统内所有进程的运行状态，并对其执行的优先级进行管理调度。因此，系统级实现功耗控制，需要对嵌入式 Linux 内核的进程调度(或任务调度)模块 schedule()全面改写，将 DPM 和 DVS 策略加入其中。

① DPM 策略[6-8]

DPM 策略在低功耗嵌入式 linux 系统实现上分为观察器和控制器两部分，如图 3 所示。监测资源需要对驱动程序作一些修改,即在驱动程序发送执行命令前和硬件完成服务并通过驱动程序告诉内核设备就绪

后这两个时间点插入一个 NotifyEvent()函数调用。

观察器根据得到的系统资源访问历史纪录采用了 Timeout 算法[9,10]计算出优化策略，该算法实现简单，预测准确性也较高，统计表明只要合理的设计 Timeout，这种假设的可信度为 95%[11]。控制模块将通过电源管理机发出控制命令给相应的硬件完成服务并通过驱动程序告诉内核设备就绪后这两个时间点插入一个 NotifyEvent()函数调用。具体的代码插入点根据不同类型的设备和对服务开始与结束的不同定义而不同。

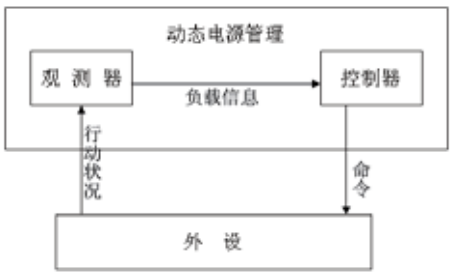

图 3　DPM 模型

但在动态电源管理实现过程中容易出现一个唤醒信号发送给一个等待队列中的进程后,该进程不能够立即被调度执行造成事件丢失的情况。为了解决这种情况,L_{low} 被设置成比 L_{max} 稍小的值,当事件数量到达 L_{low} 时,即使事件处理进程被唤醒后不能马上转入执行,由于 $L_{low}<L_{max}$,事件列表还没有全满,此时即使有新的事件产生也不会造成事件丢失,这样降低了事件丢失的可能性,对系统的影响也降到了最小,因为只有当事件列表快满时才会调用事件处理进程。

② DVS 策略

嵌入式系统降低功耗多采用动态电压缩放技术[12]，即系统运行时可以通过设置可编频率寄存器控制处理器的工作频率。实验观察发现系统的运行负荷具有明显的非平稳特性，短时间内可能具有很高的执行负荷，但绝大部分时间维持轻负荷状态。DVS 技术则根据嵌入式系统这一特点在系统负荷较高时将处理器设置为最高执行速度，保证系统的计算能力，而在系统负荷较轻时动态降低处理器的工作频率，降低处理器的执行功耗，从而实现系统计算性能与功耗的优化控制,如图 4 所示。DVS 的预测通过采用了基于时间间隔 AVGn 算法[13]的 cpu_dvs 函数实现，该函数读取 CPU 使用信息，按照 AVGn 算法来估算系统的运行负荷，并根据返回给 cpu_scan 函数的结果实现具体的动态电压缩放。

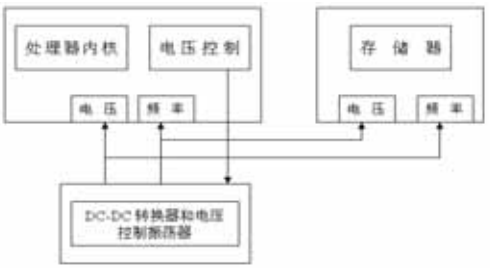

图 4　DVS 模型

cpu_dvs 函数设计核心代码：

```
util=(newstat.user-oldstat.user)+(newstat.system-oldstat.system)+(newstat.irq-oldstat.irq)+(newstat.softirq-oldstat.softirq)+(newstat.nice-oldstat.nice)+(newstat.iowait-oldstat.iowait);
util=(100*util)/(util+newstat.idle-oldstat.idle);
/*计算运行比例*/
oldstat=newstat;          /*保存上一次的 cpu 使用信息*/
forecast=(weight*prev_forecast+util)/(weight+1);      /*预测系统负荷*/
prev_forecast=forecast;          /*保存预测信息*/
if(forecast<hyst_min){
/*根据预测判断系统的行为并返回估算结果*/
decision.type=DECREASE;
return(&decision);}
else if(forecast>hyst_max)  {
decision.type=INCREASE;
return(&decision);}
else return NULL;
else return NULL; }
```

AVGn 算法[14]分析了多种简单或复杂的估计算法和平滑技术。其基本思想是指采用指数平滑平均值方法是预测即将到来的间隔的 n 个运行百分比的加权平均值。但 AVGn 算法存在一个问题就是逐档的改变频率导致系统不能及时的响应负载变化，需要大量深入细致研究。

2.3 应用程序层

在基于微处理器、微控制器的系统中,软件对系统的能量消耗同样有很大的影响。但是直到目前,还没有有效且精确的方法可用来评估软件设计对能量消耗所起的效应没有能量评估,但是引起 CPU 电源消耗的众多因素中,至少有两个受软件的影响极大：(1)存储系统:存储单元的读写和数据结构映射到多种内存条将影响多字并行装载使得存储器的存取耗能量高。代码压缩为指令分配特定的代码可减少存取的指令数和降低缓存失败的可能性,从而减少总线上的传输量,减少耗能量。采用这种方法,总线的耗能量将减少 35 %，且不会带来任何额外的硬件开销。(2)系统总线:系统总线上的转换动作在很大程度上依赖于软件，因此指令总线和地址总线上的转化顺序可以通过编译时计算出来并进行优化达到降低功耗的目的。

3　实验及结果分析

实验采用基于 ARM920T 内核的工作频率 60-200MHz 的 i.MX 嵌入式处理器。根据 i.MX 的 CPU 工作频率除正常 Run 模式外还设有 doze 模式、stop 模式和不支持电压动态调节的特点实验通过配置片内锁相环以及对片上时钟系统控制寄存器进行设置实现切换 CPU 运行模式，从而达到动态调频的目的。各模式时钟频率则通过改变时钟系统控制寄存器内容，根据公式 $f_{dpll}=f_{ref}*\frac{MFI+MFN/(MFD+1)}{PD+1}$[15] 计算出系统频率，其中：$f_{ref}$ 是系统的低频时钟频率，作为倍频的参考频率；MFI 是倍频因子的整数部分；MFN 和 MFD 分别是倍频因子的分子和分母；PD 是预设分频因子。MFI、MFN、MFD 以及 PD 构成了时钟系统控制寄存器内容。对于 linux 系统加载 DVS 策略和 DMP 策略具体思路为：由于 Linux 内核提供的 cpu_usage_stat 结构记录了处理器运行时间的分配情况，我们可以通过读取这些的参数并计算出当前系统的运行比例，即通过 cpu_scan 函数来实现具体的操作，该函数是处理器设备驱动的主要部分，它在固定的时间片内调用，时间片的大小可以根据需要在 5ms 到 100ms 之间选取，该函数通过调用 cpu_dvs 函数和 cpu_dmp 函数来评估系统的状态，这两个函数分别是可变电压技术和动态功耗管理的实现。

实验首先将具有 DPM 和 DVFS 策略的 linux 系统加载并启动运行内核功耗调度模块根据 CPU 处于 idle 模式时间的占用比里，对 CPU 的工作频率做出相应的调整，以适应系统负载变化的情况。通过系统对调频函数 cpu_scan 的调用情况，发现 CPU 在内核初始化的那一小短时间会出现多次改变 CPU 工作模式的情况。内核启动完毕，运行 Qtopia 程序，在整个启动过程中通过 klogd 和 syslogd 监控 CPU 工作频率的变换，其结果如下：

Feb 20 00:07:14 lubbock syslag.info syslogd started: BusyBox v0.60.1

Feb 20 00:07:25 lubbock auth.info login[19]:root login on 'tty'

Feb 20 00:07:49 lubbock daemon.warn klogd: 199.15MHz to 99.27MHz

Feb 20 00:07:51 lubbock daemon.warn klogd: 99.27MHz to 199.15MHz

Feb 20 00:08:53 lubbock daemon.warn klogd: 199.15MHz to 99.27MHz

Feb 20 00:08:56 lubbock daemon.warn klogd: 99.27MHz to 199.15MHz

Feb 20 00:08:57 lubbock daemon.warn klogd: 199.15MHz to 99.27MHz

Feb 20 00:08:59 lubbock daemon.warn klogd: 99.27MHz to 199.15MHz

Feb 20 00:09:50 lubbock daemon.warn klogd: 199.15MHz to 99.27MHz

Feb 20 00:09:51 lubbock daemon.warn klogd: 99.27MHz to 199.15MHz

通过实验可以看出，没有实现动态调频功能之前该过程耗时达 20 秒左右系统负载较大.启用了动态电源管理功能后 CPU 的工作频率在该程序的启动过程中会在 100Hz 和 200Hz 间切换约 8 次而且较多的时间是工作在 200Hz 的频率下，整个启动过程的时间也控制在 22 秒钟左右,从而满足在系统响应速度较小的情况下降低功耗的目的。该实验如果代替 i.MX1 使用带有高频模式的处理器可以在缩短用户响应时间和延长电池使用时间之间取得最佳平衡。

4　总结

论文根据嵌入式系统功耗分析给出了低功耗嵌入式系统模型以及具体的设计方法，并使用基于 ARM920T 内核的 i.MX 的嵌入式开发平台，实现了

DVS 策略和采用可变频率时钟的降低嵌入式系统功耗技术。实验表明，该方法能在保证程序所需性能的要求下，有效降低程序执行的能耗，在性能和能耗之间实现合理的权衡；此外无须对已有的 linux 系统作较大的修改即可将 DVS 策略嵌入系统中，充分体现了操作系统层实现系统级低功耗的易实现性。

参考文献

1 Vivek T, Deo S, Suresh R. Reducing power in high-performance microprocessors. 35th Design Automation Conference. 1998.732 - 737.

2 Chandrakasan A, Sheng S, Brodersen RW. Low- power CMOS design. IEEE Journal of Solid-State Circuits, 1992,27(4):472 - 484.

3 Semiconductor F. Power Network Design for i.MX1, i.MXL, and i.MXS, 2006.2 - 6.

4 吴慧良,史烈,陈小平,周向军.Linux 下休眠功能的软件实现.计算机工程, 2002,28(4):235 - 236.

5 朱俊星,姜新建,陈爱杰,等.基于 SKYPERTM32 的 IGBT 驱动电路的设计.国外电子元器件, 2007,(11):52 - 54.

6 Luca B. A survey of design techniques for system-level dynamic power management. IEEE Transactions on VLSI Systems, 2000,8(3):299 - 316.

7 Paleologo GA, et al. Policy optimization for dynamic power management. SIGDA.Proc. of Design Automation Conference. New York: Academic Press, 1998.173 - 178.

8 Tajana S, Luca B, Peter G, Giovanni DM. Dynamic power management for portable system. Sigcomm. Proc. of the 6th Annual International Conference on Mobile Computing and Networking. New York: Academic Press, 2000.11 - 19.

9 Benini L, Hodgson R, Siegel L. System-Level power estimation and optimization. IEEE-SSCS. Proc. of the 1998 international Symposium on Low Power Electronics and Design. New York: ACM Press, 1998.173 - 178.

10 Lu YH, Simunic T, Giovanni DM. Software controlled power management. IEEE-CS.Proc. of the 7th International Workshop on Hardware Software codesign. USA: IEEE Computer Society Press, 1999.157 - 161.

11 Lu Y, Micheli GD. Adaptive hard disk power management on personal computers. IEEE Great Lakes Symposium on VLSI. USA: IEEE Computer Society Press, 1999.50 - 53.

12 吴琦.嵌入式操作系统功耗管理技术研究.成都:电子科技大学, 2006.

13 Govil K, chan E, Wasserman H. Comparing algorithms for dynamic speed-setting of a low-power CPU. SIGACT. International Conference on Mobile Computing and Networking. New York: ACM, 1995. 13 - 15.

14 Grunwald D, Levis P, Farkas KI, et al. Policies for dynamoic clock scheduling. In: SIGOPS. 4th Symposium on Operating System Design and Implementation. San Diego. Berkeley: USENIX Association, 2000.40 - 41.

15 Semiconductor F. MC9328MX1 i. MX Integrated Portable System Processor Reference Manual. 2003.189 - 193.