· 体系结构与软件技术 ·　　文章编号: 1000-3428(2017)04-0052-08　　文献标志码: A　　中图分类号: TP302.1

基于总线负载的自适应频率调节系统

李瑞祥[1],林志涛[1],马　德[2]

(1. 浙江大学 超大规模集成电路设计研究所,杭州 310027; 2. 杭州电子科技大学 微电子 CAD 所,杭州 310027)

摘　要: 系统总线是片上系统中承载 IP 通信的模块,现有研究多通过静态预设的方法使总线频率在满足最高性能的需求下保持不变,造成性能浪费。针对该问题,提出一种动态功耗管理策略,实现自适应频率调节系统。通过监测总线主设备的负载状态,根据程序运行的历史结果预测下一时间段的总线负载需求,并对预测结果自适应配置最佳总线频率,从而在达到性能要求的前提下保持最小功耗。该策略能够快速建立稳定的频率调节模式,具有较好的稳定性和可靠性。实验结果表明,该策略相比静态管理策略可使总线功耗降低 45.6% ~ 50.7% 。

关键词: 动态预测;总线频率管理;片上系统;低功耗;动态频率调节

中文引用格式: 李瑞祥,林志涛,马　德. 基于总线负载的自适应频率调节系统[J]. 计算机工程,2017,43(4):52-59.

英文引用格式: Li Ruixiang,Lin Zhitao,Ma De. Adaptive Frequency Scaling System Based on Bus Load[J]. Computer Engineering,2017,43(4):52-59.

Adaptive Frequency Scaling System Based on Bus Load

LI Ruixiang[1],LIN Zhitao[1],MA De[2]

(1. Institute of VLSI Design,Zhejiang University,Hangzhou 310027,China;
2. Institute of Microelectronics CAD,Hangzhou Dianzi University,Hangzhou 310027,China)

【**Abstract**】The system bus is the module of IP communication in System on Chip(SoC). Existing researches are usually static methods,which maintain the highest frequency in need according to the demands of performance,thus resulting in much power consumption. In this paper,a kind of dynamic power consumption management strategy is proposed,and an adaptive frequency scaling system is realized. By monitoring the bus masters' load status,the bus load status in next timeslot is predicted on the basis of historical results of program running. Then according to the predicted bus load status, the optimal bus frequency is optimized. Thus,the power consumption is the minimum value when meeting the demand of performance. This strategy can quickly establish the stable frequency scaling mode,therefore it has good stability and reliability. Experimental results show that,compared with original static management strategy,the proposed strategy reduces the bus power consumption by 45.6% ~50.7%.

【**Key words**】dynamic prediction;bus frequency management;System on Chip(SoC);low power consumption;Dynamic Frequency Scaling(DFS)

DOI:10.3969/j.issn.1000-3428.2017.04.009

0　概述

随着集成电路工艺尺寸的减小,片上系统(System on Chip,SoC)芯片在集成度和性能方面得到不断的提高,与此同时系统复杂程度的提高以及便携式移动设备的广泛使用对芯片功耗也有了更高的要求。低功耗设计已经成为 SoC 设计中一个非常重要的环节[1]。

总线作为目前 SoC 内部功能 IP(Intellectual Property)互联的基本架构,是数据传输的主要通路,其带宽直接影响系统的整体性能。提高总线频率能够显著提高 SoC 的数据带宽,然而往往也会造成额外的功耗。为此,基于总线负载的时间局部性和规律性,本文提出一种自适应的频率调节系统。通过监测 SoC 总线主设备的工作状态,结合历史负载信息对未来一段时间内各主设备的工作状态进行预测,并根据预测结果调节系统总线频率,同时对上一次预测进行更正,从而自适应地满足系统对总线带宽的需求,尽可能降低系统功耗。

基金项目: 国家"863"计划项目"CMC 系列芯片的设计、开发与制造"(2012AA041701)。

作者简介: 李瑞祥(1990—),男,硕士研究生,主研方向为 SoC 低功耗设计与验证;林志涛,博士研究生;马　德,讲师。

收稿日期: 2016-04-25　　**修回日期:** 2016-05-26　　**E-mail:** 605980561@qq.com

1　相关研究

1.1　SoC 低功耗技术

当前数字集成电路主要采用 CMOS 制造工艺,其中功耗由动态功耗和静态功耗组成。针对静态功耗的低功耗设计方法有门控电源、多阈值电路等涉及物理设计的方法。动态功耗主要由电路中电容充放电形成的开关功耗组成,适时地降低 SoC 的工作电压和时钟频率可以降低系统的运行功耗。针对 SoC 的低功耗技术研究主要从系统级、行为级、RTL 级、门级和版图级来展开,层次越高优化的幅度越大[2]。针对动态功耗,常见的低功耗设计技术有动态电压频率调整(Dynamic Voltage and Frequency Scaling,DVFS)、门控时钟技术、门控电源、资源调度(如 Cache 配置)等,其本质在于功耗管理单元对电压、频率、资源配置等进行调节,使系统的性能和功耗达到平衡。DVFS 是其中运用较为普遍并具有出色表现的方法,它是系统级的低功耗方法,能够最大程度地降低功耗,它可以适当地同时降低电压和频率,使得动态和静态功耗都下降,在一定程度上大幅降低整个片上系统的功耗[3]。

1.2　DVFS 设计原理

典型的 DVFS 系统的工作流程如下[4-5]:

1)采集与系统负载有关的信号,计算当前系统负载。

2)根据当前负载和过去的负载情况,预测系统在下一时间段需要的性能。

3)根据预测的结果相应地调控频率和电压。

目前针对 DVFS 系统的研究通常从硬件结构和算法模型 2 个方向进行:

1)硬件结构方面的研究。文献[6]分析一种动态电压和频率管理体系,该体系通过动态电压与频率管理(Dynamic Voltage and Frequency Management,DVFM)模块实现 DVFS 技术。该模块主要包括动态电压调节(Dynamic Voltage Scaling,DVS)模块和动态频率调节(Dynamic Frequency Scaling,DFS)模块,两者相互协调共同实现 DVFS 功能,但该方法调节粒度过粗,无法在运行不同程序的情况下保持好的效果。有的研究采用片外附加的 DVFS 模块来实现更好的效果,例如文献[7]设计了一个外部频率电压控制模块,更好地提升了 DVFS 的效果,但该方法增加了额外的面积。

2)算法模型方面的研究。DVFS 算法方面的研究是近年来 DVFS 研究领域最常见的方向,不同的学者提出了许多不同的 DVFS 算法设计方案。文献[8]和文献[9]重点关注了 DVFS 设计方案中的调频间隔问题,并提出了间隔可变的 DVFS 方案。文献[10]从程序应用负载需求的预测方法切入,将预测方法划分为 3 种类型:Interval-based,Inter-tasks,Intra-tasks。其中,Interval-based[11]算法周期性地统计一个模块的忙碌程度(使用率),并根据当前时间段的忙碌程度对下一时间段的电压和频率进行相应配置。时间间隔的长度根据实际情况分为可变和不可变 2 种,可变的方案可以优化调节粒度,但会增加设计的复杂性并可能会带来额外的功耗代价。

1.3　常见的 DFS 设计方法

对于 SoC 的低功耗研究,动态电压调节在片上需要设计专用的电压调节单元,同时在芯片的物理设计时需要在各个电压点上进行时序收敛,这些问题都是芯片逻辑设计阶段无法准确验证的,引入了很大的工作量和设计风险,所以,在 SoC 系统中通常用的不多。而 DFS 通过数字逻辑控制时钟动态的调节,逻辑设计相对简单,可重用性也较好,被广泛应用在系统芯片的低功耗设计中。

近些年许多 SoC 系统的 DFS 设计方法被提出,文献[12]提出一种自适应的动态功耗管理系统 DPM,它能够根据硬件监测模块对 SoC 工作负载状况的反馈,按照事先配置好的低功耗策略调整 SoC 的工作频率,从而在满足性能的需求下降低系统功耗。这种方法能够自适应地调整工作频率,不需要软件干预管理,减少了软件干预管理过程带来的额外功耗代价。但由于该方法基于已发生的负载状态调整工作频率,因此频率调节具有滞后性,当系统负载状态值频繁变化时,频率的调整会出现无法与负载需求变化相匹配的情况。文献[13]提出一种基于时间片预测的动态频率调节方法 Iteration-prediction,根据当前时间片的系统负载状态和历史负载状态预测得出下一时间片的负载状态,这种调节较文献[12]的方法没有滞后性,并且调节粒度更细,频率变化曲线和系统负载需求曲线拟合程度较高。但这种方法的预测模型较为简单,它通过累计迭代计算多次项、得到历史系统负载状态彼此之间的关联性以预测负载状态的变化趋势,从而调整工作频率。这种预测模型不能充分识别系统负载状态变化的规律性,预测准确率较低,无法获得理想的低功耗设计效果。下面详细分析这两种算法。

1)DPM 算法

DPM 算法的体系架构由 3 个方面构成:工作状态,工作点,转换策略。工作状态反映 SoC 当前的负载状态,例如各主设备的负载状况、总线/存储器访问冲突等,是 DPM 的监测对象。工作点是 SoC 系统当前的运行状态,一般由工作频率、工作电压等表征,对于本文研究的参考需要,只考虑工作频率的不同。任何一个时间点,系统都工作在一个确定的工作点上。当监测到的工作状态发生变化时(例如总线上某个主

设备负载超过阈值),系统根据事先配置好的转换策略从当前工作点转换到另一个合适的工作点上(例如提高总线频率),以便达到性能与功耗的平衡。例如当系统监测到总线冲突大于 S、主存访问延时大于 T 时,系统判断主存的延时是由于总线带宽不足引起的,如果当前工作点的总线频率为 5 MHz,则将工作点转换到总线频率为 10 MHz 的工作点。工作点的转移如图 1 所示。

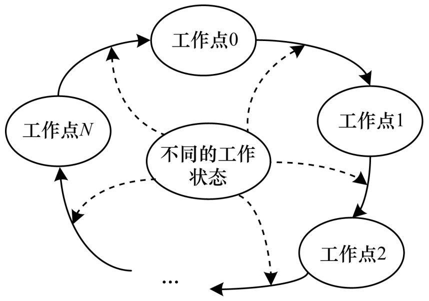

图 1 DPM 工作点的转移

2)Iteration-prediction 算法

Iteration-prediction 算法较 DPM 算法的区别在于使用基于时间片的预测方法进行工作频率的配置。它通过监测时间片内 SoC 系统的关键统计信息得到相应负载状态 $S(n)$,同时根据负载状态之间的关联性预测出下一时间片的负载状态以调整工作频率。负载状态的预测算法公式为:

$$S(n+1)=\sum_{k=0}^{n-1}\phi(k)S(n-k)$$

其中,$\phi(k)$ 为迭代系数,$0\leqslant\phi(k)<1$ 且 $\sum_{k=0}^{n-1}\phi(k)=1$。

由于累计迭代计算为多次项,对预测精度需求与实际计算量进行折中考虑,以历史监测信息为基础,采用三阶近似处理,即:

$$\begin{aligned}S(n+1)&=\phi(0)S(n)+\phi(1)S(n-1)\\&\quad+\phi(2)S(n-2)\\&=0.6S(n)+0.25S(n-1)\\&\quad+0.15S(n-2)\end{aligned}$$

其中,$\phi(0)$,$\phi(1)$,$\phi(2)$ 为迭代系数,通过最小二乘法拟合得到。

1.4 自适应频率调节系统

面向特定的应用,作为主要功能载体的总线主设备(master)对总线带宽的需求在时间上呈现明显的局部状态,主设备工作状态集合为总线负载状态。根据总线负载状态动态调节总线频率,使得总线带宽与实际负载相适应,可以避免性能过剩造成的系统功耗上升。

与 DPM 算法不同,本文采用基于动态预测模型的频率调节方法,能够预先调整总线频率,可解决 DPM 中频率调节较负载需求变化滞后、无法与负载需求变化良好拟合的问题。与 Iteration-prediction 算法相比,本文基于动态预测器的预测模型能够更好地识别总线主设备负载状态的变化规律,并且能够根据反馈信息对预测模型进行校正,从而使总线频率的调节更符合系统的负载需求。

2 自适应频率调节系统整体架构

2.1 监测对象的定义

程序在运行过程中驱动不同的总线主设备执行操作完成特定功能,如处理器(CPU)访存读取指令、DMA 拷贝数据、LCD 读取像素数据进行显示等。面向特定应用,由主设备总线带宽需求表征的工作状态具有一定的规律性。该系统基于总线负载状态调节系统总线频率。为了利用这种规律性进行预测,必须确认一个稳定量来衡量总线主设备的带宽需求,以作为主设备工作状态的监测对象,该监测对象不能受总线工作频率变化的影响。一般来说,数字逻辑是主设备的主要的控制逻辑,其执行流程不会因为频率的改变而发生较大变化。因此,以固定的总线周期数作为监测周期。衡量主设备工作状态有很多指标,如发出请求的数目、请求的平均处理时间、总线的有效占有时间等。主设备的总线有效占有时间,即访问请求的总延迟周期数,集中体现了这一段时间内主设备的总线请求数目和从设备的访问情况。在程序循环运行的过程中,即使总线频率发生变化,以固定的总线周期数作为监测周期,主设备的总线有效占有时间的变化规律是相对稳定的,可以作为评判主设备工作状态的监测对象。

另一方面,用于预测的历史负载信息包含系统中多个主设备的历史工作状态,主设备越多,或者状态定义粒度越细,历史负载信息可能的数目越大,下文所述预测器的硬件成本也越高。因此,本文采用总线的有效占有时间为指标判断忙碌和空闲两级状态,以表示单个主设备的工作状态。设主设备的状态阈值为 TH 个总线周期,在 T 个总线周期的监测周期内,当其总线的有效占有周期数大于 TH 时,该主设备状态为忙碌状态,记为 B。相反该主设备为空闲状态,记写为 I。系统整体的总线负载情况由系统内 N 个主设备工作状态表示。对于简单应用,考虑到一段时间内多个主设备同时处于忙碌状态的情况较少,为了缩减可能的状态数,可以根据主设备的总线有效占有周期数的大小,从多个处于忙碌状态的主设备中仲裁出一个最忙碌的主设备,其余主设备记为空闲状态,则系统总线负载状态有 $N+1$ 项。

2.2 总线负载状态预测

在 SoC 系统中,总线主设备工作状态变化的规律性是程序运行规律性的体现。应用场景越单一,状态迁移的规律性越强,越不容易被破坏。例如在

一个数据加密转发应用中[14]，CPU 执行各项配置操作后，DMA 将原始数据从其他接口模块拷贝至内存中，安全模块读取数据，加密后将加密数据返回内存，由输出设备（MAC，USB 等）读取并输出。各主设备的操作虽然在时间上并不完全隔离，比如 CPU 配置穿插于各个阶段，但程序基本流程循环往复，这些主设备的工作状态的变化也会呈现规律性。利用系统主设备工作状态的历史信息，可以对下一阶段主设备的工作状态进行预测。

本文提出一种总线负载状态的预测机制，如图 2 所示。

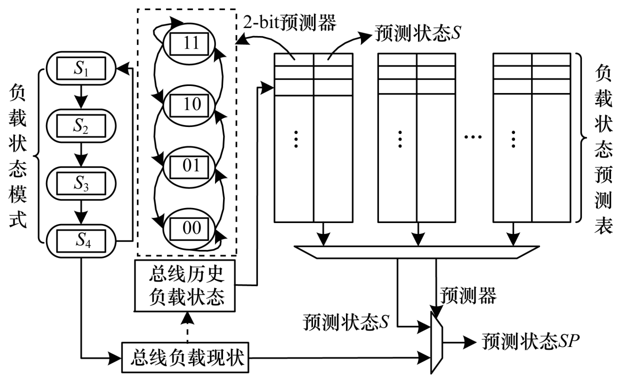

图 2　总线负载预测机制

每一个总线负载状态设计一个负载状态预测表，该表由总线历史负载状态进行索引，则每一表项的内容为该历史负载状态下预测的下一阶段负载状态和对应的 2-bit 预测器。在硬件代价可接受的条件下，预测状态和预测器可以是多组的。在一个监测周期后，根据得到的现阶段总线负载状态选取对应的预测表，由若干个监测周期的总线历史负载状态索引出预测状态及其预测器，进而仲裁出最终的总线负载预测状态。在 2-bit 预测器中，“2’b00”和“2’b01”表示预测状态可信度较低，“10”和“11”则表示预测状态可信度较高。预测状态可信度较高时，最终的预测状态为预测表中记录的预测状态，否则预测现阶段总线负载状态将延续至下一个监测周期。

负载状态预测表在每一个监测周期后根据实际总线负载状态进行更新，本文采用的一种更新策略如表 1 所示。其中，M_n，M_{n+1} 表示相邻的 2 个监测周期，当预测负载状态 S 与下一监测周期的负载现状相吻合时，预测状态无需改变，预测器则往高可信度方向前进一步；当预测负载状态 S 与下一监测周期的负载现状相异时，如果预测器处于高可信度状态，说明此次预测失败可能是因一次干扰或者出现概率较低的总线负载模式造成的，预测状态无需改变，但预测器往低可信度方向前进一步。如果预测器已经处于低可信度状态，说明原来的总线负载模式已经被打破，此时若预测器状态值为 01，则降低为 00；若预测器状态值为 00，则预测状态需要更新为与现状一致的状态，并将预测器状态置为 10。

表 1　负载状态预测表更新策略

M_n		M_{n+1}		
预测状态	2-bit 预测器	负载状态	预测状态更新值	2-bit 预测器更新值
S	P	S	S	$P==2'b11?2'b11:P+1$
	2’b00	S'	S'	2’b10
	2’b01	S'	S	2’b00
	2’b10	S'	S	2’b01
	2’b11	S'	S	2’b10

2.3　自适应频率调节

为保证一定的性能，针对总线带宽需求，系统内各个总线主设备往往存在一个最低性能要求下的阈值指标。如视频播放，每秒的最低帧数确定，则解码模块或者显示模块的最低总线带宽需求也确定了。在满足该阈值指标的条件下，频率尽可能地降低，有利于降低系统功耗。因此，系统在一个时间段内的最佳频率与在这个时间段中主设备的工作状态密切相关。通过预测下一个时间段中主设备的状态，结合各个主设备的阈值指标，可以实现总线频率的动态自适应调节。为每一个总线负载状态定义一个最佳频率，当预测器所预测状态具有高可信度时，将系统频率调节至该最佳频率。同时，各总线负载状态的最佳频率点也是伴随预测器更新而动态调整的。一个监测周期后，根据负载现状和当前频率，若对应指标低于最低性能要求的阈值，则提高该负载状态的最佳频率，当指标高于阈值时，则降低该负载状态的最佳频率。

3　自适应频率调节系统的硬件实现

基于总线负载的自适应频率调节系统框架的核心思想是：通过总线负载的历史信息预测下一阶段的负载状态，根据预测状态提前调节总线频率，使系统性能与负载相适应，避免性能过剩造成不必要的能耗[15]。不失一般性，本节将介绍该自适应频率调节系统的一种硬件实现方式来说明其在 SoC 系统中的应用。如图 3 所示，该系统主要包含寄存器组、监测单元、负载状态预测单元和动态时钟管理单元。其中，寄存器组用于配置各主设备的状态判定阈值、总线负载状态的最佳频率判定阈值以及监测周期的大小等；寄存器组增加了频率调节系统的灵活性和适应性，有利于提高其在不同应用背景下降低功耗的效果；监测单元的功能主要是监测程序运行过程中各主设备总线负载状况，作为判断系统总线整体负载状态的依据，主要包括主设备在监测周期内的

总线请求数目、总线操作的延时等;状态预测单元和动态时钟管理单元是系统的核心部分,下文将进行详细描述。

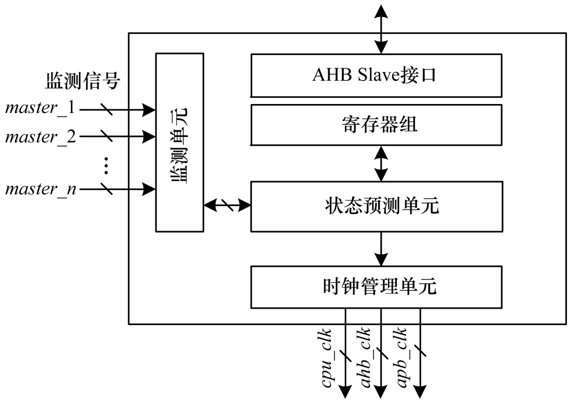

图 3 自适应频率调节系统结构

3.1 负载状态预测单元

负载状态预测单元由 4 个部分组成:控制模块,负载状态预测表,预测表更新模块和时钟控制信号生成模块,如图 4 所示。控制模块通过内部状态机控制该系统的基本流程。一个监测周期结束后,控制模块根据监测信息及寄存器组中配置的阈值生成当前总线负载状态,结合其维护的历史总线负载状态寄存器,从预测表中索引出对应的预测状态及预测器。时钟控制信号生成模块根据读出的预测信息生成对应的时钟控制信号,输出给动态时钟管理单元。同时预测表更新模块对前一次监测周期的预测结果进行比对,并更新相应表项。

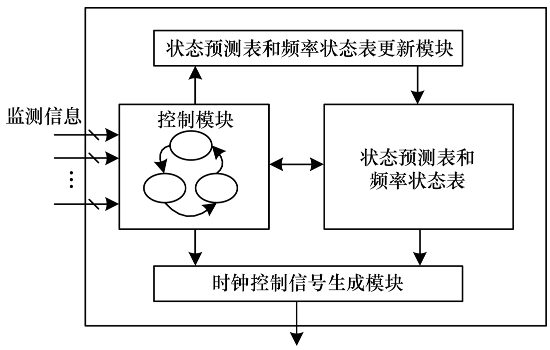

图 4 负载状态预测单元结构

3.2 动态时钟管理单元

动态时钟管理单元如图 5 所示,它包含 PLL、时钟切换单元、时钟分频器、动态变频单元等,它接收来自总线负载状态预测单元输出的时钟频率控制信号,生成相应频率的时钟提供给总线。动态时钟管理单元提供了丰富的频率选择,为了使系统总线时钟能够快速变换,本文采用时钟切换单元和动态时钟变换逻辑实现时钟的无毛刺切换。

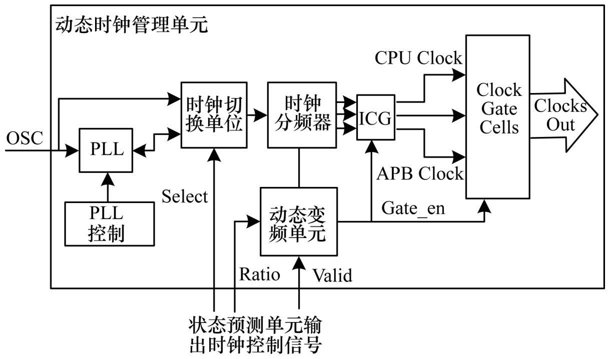

图 5 动态时钟管理单元结构

4 实验结果与分析

本节首先在一个基于 CK-CPU 的 SoC 平台集成上述的自适应频率调节系统,该平台包含丰富的通用接口和常用的加解密模块。然后运行一个加密应用程序,分析频率调节系统的表现与监测周期长度之间的关系。之后在该平台上实现 DPM 和 Iteration-prediction 频率调节方法,运行同样的加密应用程序,比较它们与本文系统的性能功耗表现差异,以验证本文系统的适应性和有效性。

4.1 自适应频率调节系统的实验结果分析

以一个加密应用为例对自适应频率调节系统进行验证。该应用每秒须处理外部输入的数据量约64 KB,一次处理大小为 256 Byte 的数据包[16]。在实验之前,通过对程序的分析可知,该应用主要涉及的总线主设备包含处理器、DMA 和加密模块。其中处理器主要负责程序流程控制和功能模块的配置,DMA 通过接口单元将片外数据加载进片上存储及将加密后的数据重新输出至片外存储中,加密模块则负责对数据进行加密操作。针对特定应用,程序涉及的总线主设备往往较少,且总线负载状态模式也比较少。监测周期的大小影响频率调节的粒度和总线负载状态预测的准确率,进而影响自适应频率调节系统的调节效果。在自适应频率调节系统设置 4 级可调频率,选取监测周期 $T=512,1\,024,2\,048,4\,096$ 个总线周期对同一程序进行实验。表 2 为不同监测周期下单位时间内总线负载状态的预测准确率。由表中和程序运行的相关信息分析可知,当$T=512$时,监测周期偏小,主设备繁忙时段往往被分割在多个监测周期,不利于性能与功耗的平衡。同时历史总线负载状态发生重名的几率大大增加,降低了状态预测的准确率。这些不利因素使得系统性能不足,只完成了单位时间内规定任务量的 74% 。监测周期 $T=1\,024$ 时调节系统能够监测到每一个数据包的处理流程,进而

发现该应用驱动总线主设备的规律,因此,预测准确率较高。

表 2　不同监测周期下调节系统的预测准确率　　%

T	预测次数	预测正确次数	准确率
512	2 985	2 519	84.4
1 024	2 166	2 105	97.2
2 048	1 437	1 280	89.1
4 096	828	616	74.4

由表 2 可知,预测错误主要出现在前期预测表的建立过程中,随着程序循环运行次数的增加,预测表建立过程带来的损失可以忽略。监测周期增大,使得监测周期内各主设备的具体状态更加混杂,同时监测周期跨越不同数据包的处理流程的范围更大,都使得预测变得更加困难。下面将详细分析不同监测周期下调节系统的具体运行情况。

图 6 为监测周期 $T=1\ 024$ 个总线周期时截取的自适应频率调节系统建立稳定调节模式附近时间段内的运行效果。可以看出,在 1 ~ 15 监测周期,调节系统的总线负载状态预测时有出错,导致总线频率与实际负载不匹配,如 9,11 监测周期。16 监测周期以后,调节系统已经发现了稳定的总线负载状态模式,即预测准确率趋近 100% 。同时,根据设定的阈值参数,调节系统自动为不同状态设置了最佳频率。由该图可以看到,自适应频率调节系统形成的稳定调节模式基本与总线实际负载相匹配。

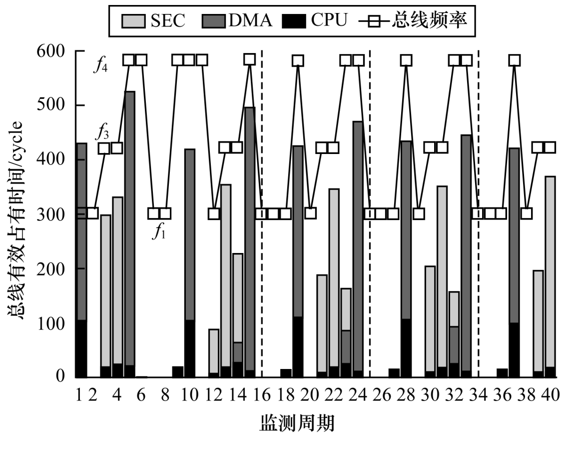

图 6　频率调节系统运行效果($T=1\ 024$)

当监测周期 $T=2\ 048$ 个总线周期时,如图 7 所示,自适应频率调节系统仍能够形成稳定的调节模式。具体的频率对应关系与监测周期为 1 024 时相同。当监测周期 $T=4\ 096$ 时,如图 8 所示,调节系统不能形成稳定的调节模式,预测准确率较低,造成总线频率与实际负载不匹配。

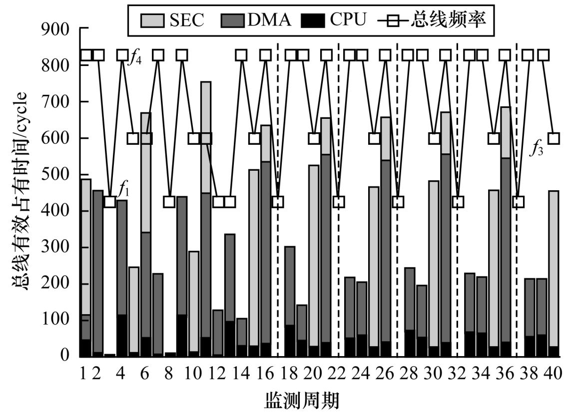

图 7　频率调节系统运行效果($T=2\ 048$)

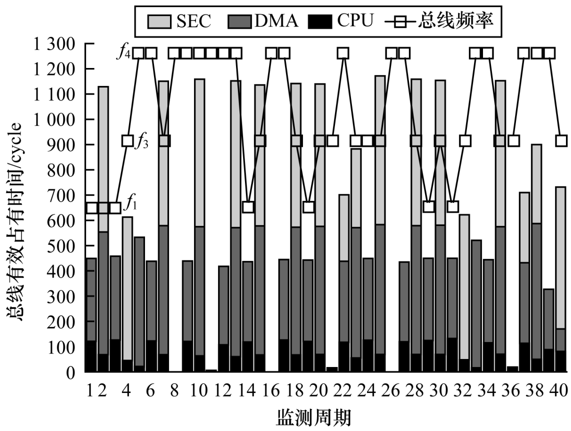

图 8　频率调节系统运行效果($T=4\ 096$)

该自适应频率调节系统的目的是通过自适应调节系统总线频率降低系统的功耗。表 3 即为在 SoC 平台下运行加密应用时不同监测周期下的功耗对比,其中,性能对比系数是按照完成规定的任务量所需时间来衡量的,以最高频下的性能为基准。恒定频率作为对照组,没有使用该自适应频率调节系统,而是采用先在最高频下运行,在单位时间内所有加密数据处理完成后将频率降至最低的策略。$T=512$ 时性能对比系数只有基准的 0.21,系统未能完成单位时间内设定的任务量,不具有参考意义;$T=1\ 024$ 时,系统较好地达到了性能和功耗的平衡;$T=4\ 096$ 时,调节粒度过粗,频率频繁处于高频阶段,因而性能较高但功耗较大。

表 3　功耗对比

T	功耗/mW	节省比例/%	性能对比系数
512	0.871	–	0.21
1 024	1.015	50.8	0.31
2 048	1.056	48.7	0.43
4 096	1.121	45.6	0.61
恒定频率	2.062	–	–

4.2　3 种方法的性能功耗比较

在同样的 SoC 平台上分别实现了 DPM 和 Iteration-prediction 频率调节方法以及本文提出的基

于负载预测的动态调频方案，系统总线频率调节分为 4 档（10 MHz，5 MHz，2.5 MHz，1.25 MHz），其他条件相同。实验选择的加密应用程序每隔单位时间需要完成一定的任务量。如果在截止时刻限定的任务量没有完成，则表示性能过低不符合系统的性能需求；如果过早地完成规定的任务量，会导致任务完成到截止时间这一段时间的性能没有得到有效地利用，造成性能的浪费。因此，理想的情况为在满足应用的实时性需求下，频率性能的利用率尽可能地高，实现频率调节与系统负载需求变化趋势完美拟合。使用这 3 种不同的频率调节模型分别运行同样的加密应用程序。

图 9 是 DPM 运行加密应用程序时对频率的调节效果。从图中可见频率调节多次出现与系统负载需求不匹配的情况，例如 6，11 和 16 周期系统处于空闲状态，工作频率却被调到最高。这是因为 DPM 的频率调节基于已发生的负载状态调整工作频率，所以频率调节具有滞后性，图 9 中系统的负载需求变化较为频繁，DPM 的频率调节无法及时跟上负载状态的变化，导致多次出现频率调节与负载变化的不匹配，频率曲线与负载变化曲线拟合情况较差，所以无法有效降低功耗。

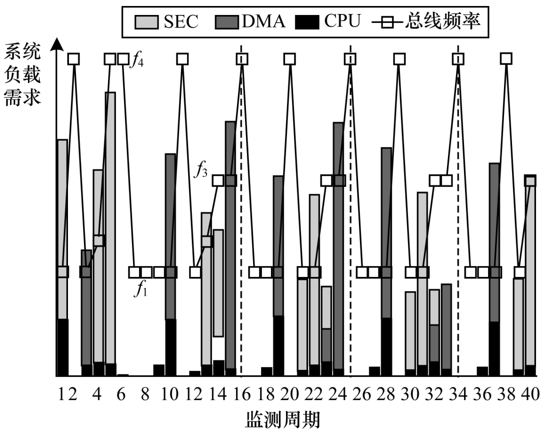

图 9 DPM 频率调节运行效果

图 10 是 Iteration-prediction 运行加密应用时的频率调节效果。从图 10 可知，频率调节与负载变化的匹配程度较图 9 更好，但由于系统负载需求变化较为频繁，整个时间段依然出现了多次频率调节与系统负载需求不匹配的现象，预测的准确率不高。由于它采用三阶迭代计算得出下一负载信息，在系统负载需求变化较为平缓的应用中能够有不错的表现，但对于变化较为频繁的应用，它无法充分地识别系统负载状态的规律性，因此无法建立稳定的预测模型。即使经过程序的充分运行，它也无法建立起准确稳定的预测模型，因为这种预测模型没有反馈机制，不具备动态校正学习的能力。对比而言，从图 6可以看出自适应频率调节系统能够动态地调整预测模型参数，在初始阶段，预测模型没有稳定建立，会有较多的预测错误出现，但随着程序的运行，预测模型较好地识别了系统负载变化的规律，预测模型被稳定建立，因此，后期预测准确率明显提升。

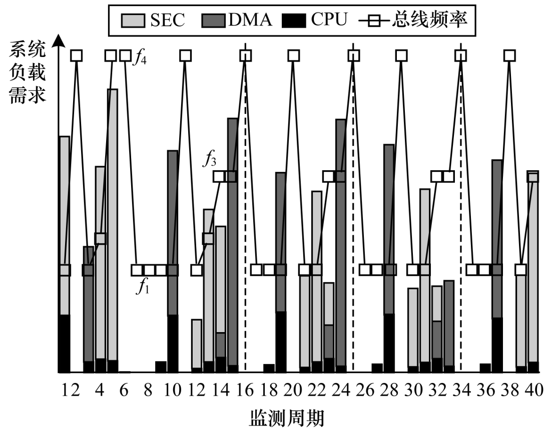

图 10 Iteration-prediction 频率调节运行效果

下面讨论 3 种方法的功耗与性能表现。从表 4 可以得到 3 种方法的性能与功耗表现，性能以完成规定任务量的有效时间长短进行衡量，时间用频率为 10 MHz 的 CPU 时钟周期数来表征，因此时钟周期的绝对时间长度为 1×10^{-7} s。对于该加密运算应用，性能要求平均每秒完成规定的任务量 S，表中展示了使用这 3 种方法和不使用任何 DFS 算法在恒定频率情况下完成任务量 S 所花费的时钟周期数，从中可知，根据性能需求，完成任务量 S 花费的周期数不能超过 1×10^7。

表 4 3 种模型的功耗性能对比

方法	功耗/mW	节省比例/%	完成任务量周期数
DPM	1.368	33.7	93 582 160
Iteration-prediction 算法	1.260	38.9	9 934 517
本文方法（T=1 024）	1.015	50.8	9 973 110
恒定频率	2.062	–	3 127 170

从表 4 可知，DPM、Iteration-prediction、本文方法和恒定频率下完成任务花费的周期数都小于 1×10^7，因此，它们都符合系统应用的性能要求。在相同时间内完成相同任务量 S 的情况下，DPM 算法的功耗节省比例最少，只有 33.7%，本文的自适应频率调节系统的功耗节省比例最高，达到 50.8%，Iteration-prediction 算法介于两者之间，为 38.9%。DPM 算法由于没有采用预测机制，因此频率的调节较负载需求变化相对滞后，无法与系统负载需求变化曲线良好匹配，因此功耗偏高。Iteration-prediction 虽然使用了预测机制，但它的预测模型为人为事先设定，且粒度较粗，无法很好地动态识别系统负载需求变化的规律，并且不能随着程序的运行根据反馈

信息校正预测模型,因此预测准确率较低,频率的调节无法有效地匹配负载需求的变化。同时它每个监测周期要用硬件模块进行 3 次乘法运算和 2 次加法运算,一定程度上增加了预测模型产生的额外功耗。本文方法的频率调节曲线能够较好地与负载需求曲线相拟合,因此有着更好的降低功耗效果。

上述实验结果表明,针对特定应用,设置合理的监测周期和状态阈值,本文自适应频率调节系统能够利用程序运行的规律性,自适应地调节总线频率,使其与总线负载相匹配,从而降低 SoC 系统的整体功耗。针对这一加密应用,3 种方法都在相同的时间完成了相同的任务量 S,较 DPM 和 Iteration-prediction 方法,本文自适应频率调节系统的频率调节与负载需求变化拟合程度更好,功耗更低,因此,有着更好的低功耗设计效果。

5　结束语

本文根据程序运行时主设备状态变化的规律性,设计一种基于总线负载的总线频率调节系统。根据监测得到的总线历史负载状态预测下一时间段的总线负载状态,利用预测得到的结果自适应地进行总线频率配置,使总线频率在不同的主设备状态下能够自适应调整至相应的最佳频率,避免性能浪费。实验结果证明,这种调节系统结构简单,易于实现,在较小的硬件代价下实现了总线频率的自适应动态调节,一定程度上降低了功耗。未来将进一步改进算法,例如通过人工神经网络或者统计机器学习的方法,重点着眼于历史负载之间的关系,建立更优的模型,提升预测的准确率,从而使频率调节和负载需求具有更高的拟合度。

参考文献

[1] Affes H, Auguin M. SoC Power Management Strategy Based on Global Hardware Functional State Analysis[C]//Proceedings of 2015 Euromicro Conference on Digital System Design. Washington D. C., USA: IEEE Press, 2015:614-620.

[2] Nebel W, Mermet J. Low Power Design in Deep Submicron Electronics[M]. [S. l.]: Kluwer Academic Publishers, 1997.

[3] Kimura H, Sato M, Imada T, et al. Runtime DVFS Control with Instrumented Code in Power-scalable Cluster System[C]//Proceedings of 2008 IEEE International Conference on Cluster Computing. Washington D. C., USA Press: IEEE Computer Society, 2008:354-359.

[4] 卢春鹏. 动态电压与频率调节在降低功耗中的作用[J]. 单片机与嵌入式系统应用, 2007, 7(5):12-14.

[5] Huang Jiangwei, Chen Tianzhou, Qian Jie, et al. Partitioning the Program into Different Regions Using Dynamic and Static Approach with Kernel-assisted in Power Management for Embedded System [C]//Proceedings of 2006 IEEE International Conference on Information Reuse and Integration Heuristic Systems Engineering. Washington D. C., USA: IEEE Press, 2006:341-344.

[6] Nakai M, Akui S, Seno K, et al. Dynamic Voltage and Frequency Management for a Low-power Embedded Microprocessor[J]. IEEE Journal of Solid-State Circuits, 2005, 40(1):28-35.

[7] Albea C, Puschini D, Lesecq S, et al. Advanced Coupled Voltage-frequency Control for Power Efficient DVFS Management [C]//Proceedings of the 38th Annual Conference of the IEEE Industrial Electronics Society. Washington D. C., USA: IEEE Press, 2012:2168-2173.

[8] Najibi M, Salehi M, Kusha A A, et al. Dynamic Voltage and Frequency Management Based on Variable Update Intervals for Frequency Setting [C]//Proceedings of IEEE/ACM International Conference on Computer-Aided Design. New York, USA: ACM Press, 2006:755-760.

[9] Salehi M E, Samadi M, Najibi M, et al. Dynamic Voltage and Frequency Scheduling for Embedded Processors Considering Power/Performance Tradeoffs [J]. IEEE Transactions on Very Large Scale Integration Systems, 2011, 19(10):1931-1935.

[10] Kim S K, Kim Y J. GPGPU-Perf: Efficient, Interval-based DVFS Algorithm for Mobile GPGPU Applications [J]. Visual Computer, 2015, 31(6-8):1-10.

[11] Keramidas G, Spiliopoulos V, Kaxiras S. Interval-based Models for Run-time DVFS Orchestration in Superscalar Processors[C]//Proceedings of the 7th Conference on Computing Frontiers. New York, USA: ACM Press, 2010:287-296.

[12] Hsieh H C, Wen Shuian, Liao Cheyu, et al. Low Power Design and Dynamic Power Management System for VLIW DSP Subsystem [C]//Proceedings of 2011 International Symposium on Intelligent Signal Processing and Communications Systems. Washington D. C., USA: IEEE Press, 2011:1-5.

[13] 孙大鹰, 徐　申, 徐玉珉, 等. 应用于低功耗嵌入式处理器的功耗动态管理策略设计[J]. 东南大学学报(自然科学版), 2013, 43(4):695-700.

[14] 刘　曼. 基于 SoC 的高速数据流加密传输的方法实现[J]. 中国集成电路, 2009, 18(7):32-33.

[15] 王延升, 刘雷波. SoC 设计中的时钟低功耗技术[J]. 计算机工程, 2009, 35(24):257-258.

[16] 甘庆晴, 王晓明. 云环境下一种高效的密钥聚合加密方案[J]. 计算机工程, 2016, 42(2):33-37, 44.

编辑　金胡考