实时嵌入式系统的 WCET 分析与预测研究综述

王颖洁[1,2] 周宽久[1] 李明楚[1]
(大连理工大学软件学院 辽宁 大连 116621)[1] (大连大学信息工程学院 辽宁 大连 116622)[2]

摘 要 在实时嵌入式系统设计中,为了保证系统的安全运行,需要验证系统是否满足时限,即任务必须在截止期之前完成,否则实时系统将失败。目前衡量实时嵌入式系统实时性的重要指标是任务的最坏情况执行时间(Worst Case Execution Time,WCET)。文章首先综述了 WCET 分析以及研究 WCET 分析的主要方法。分析了在当前多核平台上、复杂处理器架构下 WCET 分析存在的主要问题,并根据当前 WCET 分析存在的问题展开讨论,分别针对时序分析、微系统结构分析和多核多任务调度策略等方面分析了国内外的研究进展。最后提出了一种基于深度学习的自适应实时 DVFS 算法,该算法可以进行动态电压和频率调节(DVFS),以达到节能的目的;同时还能够动态修正程序的 WCET 值,为未来嵌入式系统中的 WCET 分析与预测提供指导方法。

关键词 最坏情况执行时间,时序分析,模型检验,调度策略,动态电压和频率调节
中图法分类号 TP311 **文献标识码** A

Survey of WCET Analysis and Prediction for Real-time Embedded Systems

WANG Ying-jie[1,2] ZHOU Kuan-jiu[1] LI Ming-chu[1]
(School of Software Technology,Dalian University of Technology,Dalian,Liaoning 116621,China)[1]
(College of Information Engineering,Dalian University,Dalian,Liaoning 116622,China)[2]

Abstract For the safe operation of real-time embedded system,verifying whether the system meet the duration limitation is necessary. That means the task must be completed before the deadline,otherwise the real-time system will fail. At present,the worst-case execution time (WCET) is one of the important indicators to measure the real-time embedded system. This paper firstly introduces WCET analysis itself and main methods for this analysis. Secondly,the main problems of WCET analysis under complex processor architecture on current multi-core platforms are investigated. Thirdly,the research progress in WCET analysis is discussed for timing analysis,micro-system structure analysis and multi-core and multi-task scheduling strategy based on the current problems. Finally,adaptive real-time DVFS algorithm based on deep learning is proposed,which can perform dynamic voltage and frequency adjustment (DVFS),achieve the target for energy saving,and dynamically correct the WCET value of the program to provide guidance for the analysis and prediction of WCET in future embedded systems.

Keywords Worst case execution time (WCET),Timing analysis,Model verification,Scheduling strategy,Dynamic voltage and frequency scaling (DVFS)

1 引言

在设计实时嵌入式系统时,既要保证计算结果的正确性,又必须满足实时要求,即要求在规定的时限内产生结果[1]。实时系统通常划分为硬实时系统和软实时系统两类。硬实时是指如果不能在规定的时限完成,将引发致命的或灾难性的后果。软实时是指偶尔不能在规定的时限完成,只会造成系统性能的降低,而不会产生永久性的影响[2]。由于硬实时系统对于执行时间的要求非常苛刻,因此在设计此类系统时,通常要在系统实际运行之前对系统的时间特性进行完整的验证,以确保在系统运行过程中规定的时间特性不会被破坏。这一工作被称为时间特性验证[3]。实时系统的时间特性必须在系统级进行验证,只有系统的每一个部分以及整个系统都满足其实时性要求时,这个系统才满足了实时性要求。

2 WCET 分析方法

为了保证系统的安全运行,需要验证系统是否满足时限,即任务必须在截止期之前完成,否则实时系统会失败。衡量系统实时性的最重要的参数是任务的最坏情况执行时间(Worst Case Execution Time,WCET),WCET 为任务的实时调度、任务的优先级仲裁、资源的冲突仲裁以及任务间通信提供依据,是确保系统安全运行的可信基础[4]。WCET 分析方法包括动态测量、静态分析和混合方法 3 种。

动态测量是指在目标环境中实际地运行任务,通过测量

本文受国家自然科学基金(61572097),中央高校基本科研业务费专项资金(DUT19ZD104)资助。
王颖洁(1977—),女,博士生,讲师,CCF 会员,主要研究方向为软件可信性、计算机体系结构,E-mail:wyj_dut@163.com;**周宽久**(1966—),男,博士,教授,CCF 会员,主要研究方向为软件可信性、软件测试,E-mail:zhoukj@dlut.edu.cn(通信作者);**李明楚**(1963—),男,博士,教授,博士生导师,主要研究方向为密码学、图论、密码学。

任务的执行时间，加以各种统计方法来估计程序的 WCET。理论上，通过测量的方法来获取程序的 WCET，必须向任务输入所有可能的参数，保证程序运行完所有可能的执行路径，而实际程序的输入参数空间往往非常巨大，输入所有参数来运行程序通常是不可行的，即使可行也会耗费相当长的时间[5]。因此，在有限样本情况下，动态度量很难保证所得到结果的安全性，尤其对于现代高性能处理器。

静态分析方法根据程序的流信息，针对目标处理器的特性估算出程序的 WCET。由于程序流信息通常非常复杂，而处理器尤其是现代处理器（例如高速缓存和流水线）的特性也很复杂，相应地静态分析和计算也变得非常复杂。但是程序静态分析方法由于能够保证所得到的结果是安全的，而且能够不运行程序就获得结果，从而成为 WCET 分析研究的主流技术[2]。但是静态分析方法往往会高估程序的 WCET 实际值。

当前静态 WCET 分析主要从 3 个方面展开研究：控制流分析、底层分析和计算方法[5-6]。

(1)控制流分析。控制流分析从程序的源代码或目标代码出发，通过分析程序的逻辑结构、语法、语义等信息来获取程序所有可能的执行路径（执行流）。控制流分析不考虑程序的执行环境的硬件特性，分析研究主要集中在程序控制流信息的提取、程序逻辑结构的表示、控制流信息的表示与转换、循环上界的确定、不可行和隐藏路径发现等。

(2)底层分析。底层分析，又称为执行时间建模，主要考虑缓存、流水线等处理器体系结构特性对程序执行时间的影响。现代计算机体系结构多采用上述特性来提高处理器性能，而这些特性的引入使得处理器执行指令的行为变得越来越不可预测，这恰恰与实时系统可预测性的重要特征相违背。执行时间建模是对目标处理器的各种加速特性进行建模。

(3)计算方法。WCET 分析的最后阶段是结合控制流分析和底层分析来进行计算，得到最终的 WCET 估计值。计算方法大致可分为 3 类：基于树（Tree-Based）方法、基于路径（Path-based）方法和隐藏路径枚举技术 IPET 方法。

混合方法就是将静态分析和动态测量相结合，该方法既包括静态分析也包括动态测量，是两种方法的混合。该方法有两种，一种是在测量的基础上分析得到 WCET，此时测量的是程序片段，比如基本块；另一种是在程序静态分析的基础上进行测量，例如首先找到可能导致 WCET 的程序路径，然后测量该路径的 WCET。

3　WCET 分析存在的问题

WCET 分析存在的问题[7]如下。

(1)高性能微处理器的使用增加了时序分析的难度

高性能处理器采用深层流水线、缓存、乱序执行和分支预测技术来提高处理器的性能。过去，指令执行时间是常量。现在，随着采用深层流水线、缓存和其他各种预测概念的微处理器的出现，导致各个指令的执行时间产生很大差异，指令执行时间成为了变量，同一指令的执行时间会因指令的出现位置不同而不相同，增加了时序分析的难度。

(2)微系统结构分析

执行时间界限能否确定、确定的界限是否精确，很大程度上取决于计算机系统结构的属性。由于计算机系统结构组件之间有相互依赖性，导致无法对单个组件进行分析，也无法以简单方式实施执行时间惩罚，而需要进行整体结构分析，这样带来了更高的复杂性并增加了资源需求。并且由于多核处理器存在 L2 Cache 和总线共享，造成多核访问冲突，同时针对多个线程相互交互，某一线程访问共享变量将导致处理器耗费一定时间维护 L1 Cache 数据的一致性，这都给多核嵌入式软件 WCET 分析带来不确定性。

(3)多核多任务调度

多核实时嵌入式系统的任务执行过程必须遵循某种任务分配策略。当系统中有多个处理器可以执行同一个任务时，该策略决定将任务分派给其中的一个处理器去执行。在动态调度算法的模型中，为了描述任务分配这一动态行为，任务与处理器之间的映射关系必须能够动态地建立和调整，而建立这种动态映射关系模型是很困难的。由于实际应用存在多个任务并行运行，同时多个任务之间存在相互通信，因此研究多核嵌入式软件的 WCET，必须解决基于多核、多任务的调度问题，即确定各个任务分别在哪个核上运行、何时开始运行。

4　时序分析

从本质上说，WCET 分析就是搜寻一条穿过程序的最长路径。这可以看作构建加权图并查找其中最长路径的问题：图的节点模拟基本块等程序片段，所谓基本块即直线代码的最长序列；图的边模拟可能的控制流；节点权重是程序片段执行时间的上限，而边权重是其遍历计数的界限。遵循这一方案，近年来出现了一种针对静态时序分析的类标准系统结构，如图 1 所示。

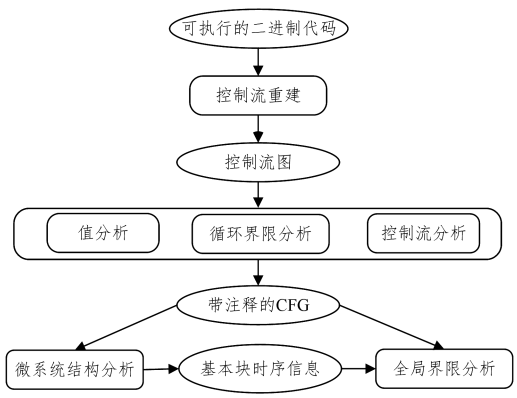

图 1　静态时序分析过程示意图

在图 1 中，控制流重建读取要分析的二进制可执行文件，重建其控制流并将其转换为中间程序表示形式；值分析以静态方式确定寄存器和内存位置中存储的值，循环界限分析需要此类信息来确定算术指令的执行时间，从而对有效地址进行安全近似，以便进行数据缓存分析；循环界限分析决定了图的边权重，它标识程序中的循环，并尝试确定循环迭代数的界限。同样，递归也必须有界；控制流分析可以更准确地确定穿过程序的可行路径的集合，以便缩小时序界限；微系统结构分析决定了图的节点权重；全局界限分析被称为路径分析，其最终决定图中的最长路径[7-8]。

4.1　形式化方法和模型检验技术

形式化规范方法是由具有精确语义的形式语言书写的程序功能描述，是设计和编制程序的出发点，也是验证程序是否正确的依据。形式化验证方法能够以最小的代价提前发现和

修改错误,目前主要有两种方法:模型检验和定理证明,分别用来处理有限状态空间模型问题和无限状态空间模型问题[9]。

模型检验技术作为形式化方法之一,从广义上讲,就是自动检测模型是否具有某种特定性质的方法。模型检验要求用户建立一个能够描述系统的模型,并描述出系统需求,将这些要求输入机器中进行自动的验证,寻找能够导致错误的反例,以检验系统中是否存在错误。如果没有发现错误,用户可以对模型进行提炼(如引入更多的设计信息来增强系统的真实度),重新进行验证。模型验证的算法一般基于全状态搜索,搜索每一个状态是否满足需求。通过可达性分析的方法来分析系统中是否存在死锁、最短路径等。模型检验的具体过程如图 2 所示[9]。

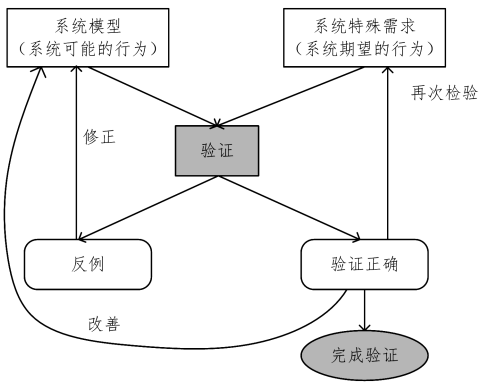

图 2 模型检验过程图

文献[10]中提出的模型检测方法,是在不影响程序真实的 WCET 情况下,通过改变程序模型来降低程序状态空间,但这个方法先假设一个理想的数据缓存,即认为所有的访问都是命中的,但通常实际情况并非如此。文献[11]在多核处理器的共享存储器中加入仲裁机制,使得存储器存储同步,这样就可以增加系统的可预测性。该多核结构就可以使用现有的单核 WCET 分析工具进行分析。

4.2 时间自动机理论

为了能够准确地描述实时系统的时间性质,引入时间自动机(Timed Automata,TA)理论,利用时间自动机来描述系统的实时过程。TA 能够通过一种有效的方式描述带时钟变量的系统状态转换图,能够描述系统随时间变化的动态行为,对于实时系统特别是硬实时系统的模型检验有特别重大的意义,成为实时系统行为模型和相关验证工具的理论基础。

从状态转换系统的定义可知,状态转换系统可以抽象地描述系统的运行过程,状态之间的转换构成了系统的全部功能。对于我们研究的实时系统而言,状态转换系统中的激励事件带有时间约束,此时可以通过时间约束描述状态内部和状态之间转换的过程和性质,只有时间约束满足时状态才可以进行转移,时间在状态转移的过程中流逝[9]。

时序分析是要保证程序执行时序逻辑的正确性,对并发程序进行时序分析是目前一个重要问题,有关时序分析的研究进展如表 1 所列。

表 1 时序分析的研究进展

研究类别	时间/年	研究内容
时序分析研究	2006	时序约束是判断实时系统运行是否正确的重要规约。为了减小测试时由于对系统进行插装而产生的对实时系统行为的影响,文献[12]提出了一种混合式监控方法。它对系统的时间干扰比纯软件方式小,并支持对系统的完全测试。此外,文献[12]还提出一种基于 WCET(Worst-Case Execution Time)分析技术的目标系统时间补偿方法,在精确地计算插入断言对目标系统的时间影响基础上,给出时间补偿
	2017	概率硬实时系统的时序分析需要提供概率最坏情况的执行时间(pWCET)估计。pWCET 分布可以描述为超越函数,它给出了任务执行时间将超过任何给定执行时间预算的概率的上限。文献[13]介绍了一种更有效的多路径程序静态概率时序分析(SPTA)。该分析估计了多路径程序的 pWCET 分布在失效时随机替换缓存的时间贡献;引入了简单的程序转换,以减少路径不确定性的影响,同时确保可靠的 pWCET 估算;提出了最坏情况执行路径(WCEP)的约简方法,以减少分析的路径集,提高了时间估计的严密性。但该方法与使用 LRU 缓存的确定性系统的分析结果不具有可比性
时序分析框架	2010	文献[14]提供了一个集成的时序分析框架,可以捕获共享缓存和共享总线的时序影响。这个时序分析框架可以应用于类似的多处理器架构
	2017	文献[15]提出了一种可扩展的框架 MRTA,用于根据对不同硬件资源的需求进行多核响应时间分析。MRTA 框架可扩展到不同的多核架构,具有用于公共互连的各种仲裁策略,以及本地存储器的不同类型和布置。MRTA 框架提供了一个通用的时序验证框架,该框架在硬件配置(公共互连、本地存储器、内核数量等)中是参数化的,因此可以在架构设计阶段通过不同的硬件配置来保证不同的实时性能水平,也可以在开发和集成阶段验证特定系统的时序行为
符号化 WCET 分析	2006	符号化 WCET 分析是用符号表达式表示任务的最大执行时间,表达式中包含了参数。通过在运行时刻快速确定表达式值,符号化 WCET 分析可以更精确地估算 WCET。文献[16]提出了一种针对其分支直接依赖于输入数据的程序的符号化 WCET 分析方法。符号化 WCET 公式直接依赖于输入参数,使得运行时的 WCET 估算更加简单直接
	2017	文献[17]提出一种用于参数化 WCET 分析的新技术,该技术是基于 WCET 公式的符号计算新方法。符号 WCET 计算有许多优点:它大大减少了分析系统参数的时间;它是模块化的,很容易单独分析系统的不同模块;它允许在线有效地计算 WCET,从而可以很容易地在自适应系统中使用。该方法的主要局限是不能指定与不同参数之间的约束,这可能不利于公式的化简

5 微系统结构分析

微处理器体系结构设计正在经历从单核处理器到多核处理器的巨大转变。多核处理器通常采用共享 Cache 设计,多个核心上的任务对共享 Cache 的细粒度访问造成了系统时间行为的复杂性和不可预测性,这给 WCET 分析带来了巨大挑战[3]。在现代计算机系统结构中,使用推测技术成为运算的常规模式,而且效果很好。推测技术包括使用缓存对数据复用情况进行推测;利用分支预测对比较结果进行推测;使用流水线推测是否缺少数据依赖关系等。

5.1 多核共享缓存分析

当前多核结构通常采用共享 Cache 来提高处理器性能以

及 Cache 利用率，处理器核通过共享片上互连访问共享 Cache。共享 Cache 会导致并行线程间干扰，运行在一个核上的任务可能会破坏运行在另一个核上的并行任务在共享 Cache 中的某些数据。同时，多个处理器核通过片上互连访问共享 Cache 时的竞争使用也使得线程的执行可能受到干扰而延迟。为了获得多核实时系统中线程的有效 WCET 估值，必须考虑并行任务间在共享 Cache 和互连上的干扰。

多核处理器体系结构很难甚至无法满足实时系统的实时限制和对 WCET 的可预测性要求。陈芳园等[18]从多核中的共享资源入手，分析多核中的片上共享资源（共享 Cache、片上互连）和片外共享资源（片外存储）对 WCET 分析的影响，探讨了各种干扰下的 WCET 分析方法。其介绍了两种多核 WCET 分析模型：多核静态 WCET 分析模型和多核混合 WCET 分析模型；同时，针对嵌入式实时应用提出了多核设计原则。在进行可预测实时多核设计时应尽量遵循如下原则：

（1）结构与应用相关，即针对特定的嵌入式硬实时系统进行多核结构设计，在提高最坏情况的性能的同时保证可靠的、精确的可预测性。

（2）简化设计中的部件结构，在保证可预测性的前提下将那些与性能无关的特性取消。例如，在以存储系统性能界定系统性能的系统中可以尽量简化流水线结构。

（3）尽量减少系统中共享资源的干扰，可以在不影响性能的前提下尽量避免构建共享 Cache 或存储。

在将应用程序集映射到目标体系结构中时尽量避免引入应用程序间的共享干扰[18]。

多核共享缓存分析的研究进展如表 2 所列。

表 2　多核共享缓存分析的研究进展

研究类别	时间/年	研究内容
片上共享资源带来的问题	2009	随着片上多核处理器在嵌入式实时系统中的应用，片上共享资源给任务的 WCET 分析带来诸多挑战，使得对多核共享资源冲突问题的研究变得非常重要。文献[19]针对共享 Cache 和共享总线的多核结构，对可能访问共享总线的存储访问总是考虑最大的等待时间。这种方法消除了总线对 WCET 分析的影响，但在任务实际执行过程中，Cache 未命中对互连的访问请求会产生不同的延迟。这种考虑最大互连竞争延迟的方法过于保守，WCET 估值精度较低
	2014	文献[20]把目前已有的研究分为面向共享资源冲突分析和面向共享资源冲突约束两大类。对于面向共享资源冲突分析问题，探讨了不同共享资源冲突产生的原因，概括和比较了典型的冲突分析方法的优势和局限性；对于面向共享资源冲突约束问题，给出了其主要的研究内容，并评述和分析了几种主流的冲突约束方法
共享 Cache 干扰带来的问题	2012	程序的 WCET 通常受到硬件体系结构的影响，Cache 则是其中最为突出的因素之一。在共享 Cache 的多核处理器中，线程在共享 Cache 中的指令可能被其他并行线程的指令替换，从而导致线程间在共享 Cache 上的干扰。多核结构下 WCET 估值需要考虑并行线程间在共享 Cache 上的干扰。针对当前典型的共享 Cache 和共享总线的多核结构，文献[21]提出了一种迭代的 WCET 估值分析方法。该方法考虑共享总线对共享 Cache 访问的时序影响，基于该时序分析线程间在共享 Cache 上的干扰，从而得到较精确的 WCET 估值
	2012	现代处理器通常提供缓存锁定功能，可以静态和动态地应用缓存锁定功能以可预测的方式管理缓存。选择要锁定在指令高速缓存（I-Cache）中的指令会对系统性能产生巨大影响。文献[22]介绍如何将缓存锁定技术应用于共享 I-Cache，以最大限度地减少多任务嵌入式系统的最坏情况 CPU 利用率（WCU）；分析并比较了 3 种不同的策略来执行 I-Cache 锁定：静态锁定、半动态锁定和动态锁定；利用嵌入式应用的预知信息提出了不同的算法。实验结果表明，与以前的技术相比，所提出的算法可以减少 WCU
	2016	文献[23]通过优化核-bank 映射关系来使硬实时多核系统中的 bank 冲突延迟最小化，即在对 bank 冲突延迟进行分析的基础上，首先通过优化核-bank 之间的映射关系来消除 bank 冲突；若无法消除，则需要寻找能使 bank 冲突延迟最小化的核-bank 映射关系解，并为此设计了一种基于多核总线请求时间序列的 bank 冲突延迟求解算法。最后设计了能够对总线访问延迟进行消重的多核硬实时任务 WCET 估算方法
	2017	文献[24]提出了一种新的缓存分析方法，目标是查找缓存访问组的最坏情况行为以及由循环内部访问引起的未命中数的上限。该方法分两步：首先以完全精确的方式分析访问的每个固定大小的邻域，并且汇总缓存未命中路径信息；然后，对访问的缓存未命中路径执行各种分析，以获得基本块的最坏情况配置文件，这些配置文件指示基本块中的最大未命中数和引起未命中的封闭环的最小迭代次数。实验表明，与以前的方法相比，WCET 值的精度明显提高，而且该方法的扩展性很好

5.2　分支预测技术

一个系统的时间可预测性不仅仅取决于应用程序，同时取决于为应用程序提供服务的实时操作系统。实时操作系统以控制密集型代码为主，其代码结构和执行特性与普通应用程序有很大区别。因此，对实时操作系统进行静态 WCET 分析，并探讨传统技术在分析实时操作系统代码时的精度等问题就显得尤为重要[3]。

由于现代处理器使用了流水线、乱序执行、动态分支预测、Cache 等性能提高机制以及多核之间的资源共享，使得系统的最坏执行时间分析变得非常困难。因此，国际学术界提出时间可预测系统设计的思想，以降低系统的最坏执行时间分析难度。已有研究主要关注硬件层次及其编译方法的调整和优化，而较少关注软件层次，即时间可预测多线程代码的构造方法以及到多核硬件平台的映射。文献[25]提出一种基于同步语言模型驱动的时间可预测多线程代码生成方法，建立了基于 AADL（Architecture Analysis and Design Language）的时间可预测多核体系结构模型，作为研究的目标平台；最后，给出多线程代码到多核体系结构模型的映射方法，并给出系统性质的分析框架。

许多高性能嵌入式处理器都引入了多级缓存、硬件预取及软件预取等机制，为使支持软件预取的硬实时任务具有执行时间的可预测性，文献[26]提出一种支持软件预取的缓存 WCET 分析方法。该方法对多级缓存抽象解释模型进行了软件预取语义扩展，分析了软件预取对任务的最坏情况下性能和能耗的影响，但该分析模型并没有扩展到嵌入式多核结构上，没能对多核多任务、多核多线程的应用场景进行定时分析。文献[27]提出一种基于程序基本块的指令预取方法。该方法以基本块为粒度执行指令预取，避免了传统指令预取技术引入的无效预取；通过简化最坏情况下的指令访问命中/缺失情况的判定，简化任务 WCET 分析过程并优化 WCET 评估值。

分支预测是流水线处理器实现高性能的重要特征。但

是,当过于保守地建模时,它可能导致过度悲观的最坏情况执行时间(WCET)界限。文献[28]介绍了局部动态分支预测器的分支错误预测数量的界限。边界可以容易地集成到WCET问题的整数线性编程公式中。动态分支预测很难在WCET分析中建模,一个分支可能干扰另一个分支的预测。然而,静态分支预测是一种处理器技术,用于克服实时系统的可分析性问题。文献[29]提出将静态分支预测用于减少最坏情况执行时间(WCET),可将其应用于支持此类预测的任何处理器,但要求有WCET工具支持。其还描述了如何用静态估算方法获得最大WCET减少量。

文献[30]首先提出了将ICache、分支目标缓冲区和指令预取缓冲区同时集成的分析方法。在WCET分析的模型检验工作中,首先解决的是动态分支预测机制。在此基础上,其还对动态分支预测和Branch Target Buffer(BTB)对嵌入式控制系统WCET估计的影响进行了评估。文献[31]研究了减少任务的分支错误预测次数以使其WCET(最坏情况执行时间)最小化的问题,并提出了一种混合分支预测方法。该方法包括基于静态分支分析器和动态分支预测器。基于概要文件的静态分支分析器使用数据依赖分析来找到分支之间的精确关联,并识别所有对任务的WCET没有任何影响的分支。动态预测器使用相关信息进行在线预测。

6 多核多任务调度策略

随着多核处理器的广泛使用,研究和探讨适应多核处理器并行系统的任务调度算法就显得尤为重要。在多核处理器内部,系统中的共享资源是有限的,线程间需要争夺共享资源的使用权。如果没有很好的调度机制,某些线程可能占用大部分乃至全部的系统资源,导致其他线程的请求得不到服务,最终对系统的性能造成影响。好的调度规则和调度策略能够最小化并行程序的全局完成时间,降低系统资源消耗,提高系统工作效率[32-33]。如果存在一个调度策略能够在多个并行线程间公平有效地分配共享资源,则可以缓解由于对共享资源的争夺所带来的负面影响,从而改善系统性能。因此,在片上处理器系统中对共享资源的分配调度成为一个值得研究的热点问题[34-35]。

对于多核处理器来说,把任务分配到各个处理核心将会产生两类开销:一种是处理核心执行该任务的计算开销,另一种是由于有依赖关系,导致任务被分配到不同处理核心而产生的任务之间的通信开销[36]。为了提高多核处理器的性能,必须达到两个目标:1)最小化不同处理核心上的任务间的通信开销;2)平衡不同处理核心上的任务执行代价。这两个目标看起来是矛盾的:一方面,当所有任务被分配到同一个处理核心时,核心间有依赖关系的任务之间的通信将被消除,但是这将导致某些处理核心上的负载过重,其他核心却处于空闲状态,核心间的负载不平衡;另一方面,把任务均衡地分配到各个核心将会最大化处理核心的利用率,但是这会加重核间任务的通信开销。因此在多核处理器中,任务分配算法的目的就是建立一个这样的任务调度:它的通信开销和运算开销的总和是最小的。在一个多核处理器系统中进行任务调度的本质是建立一组从有序的任务集合到处理核心集合的有效映射关系[37]。

目前,多核多任务调度研究进展如表3所列。

表3 多核多任务调度研究进展

研究类别	时间/年	研究内容
多核处理器任务和线程的调度问题	2007	针对多核处理器任务和线程的调度问题,文献[38]提出了一个包含两步操作的任务分配算法,合并通信代价较高的任务为任务簇,直到任务簇的数量小于多核处理器中处理核的数量,然后将这些任务簇分配到不同的核上。这个算法充分利用了多核结构的特点,降低了任务执行过程中的通信开销。但是该算法没有考虑任务之间的优先关系
	2008	文献[39]提出一个调度算法,调度连续的程序到具有较快通信链路的节点中。它通过利用分布在不同节点的操作数的拷贝,减少对于通信资源运送源操作数和竞争资源的循环次数。但是这个算法的时间复杂度较高
	2009	文献[40]提出了一个基于复制的调度算法,选择性地复制前驱任务节点,然后根据任务间的通信关系进行任务分组,最后把这些任务组映射到各个处理核心上。该算法有效地提前了任务开始执行的时间,降低了任务间的通信,但没考虑在任务分配过程中多核间的负载平衡
	2013	文献[41]提出了基于独立实时域的实时优化方法:通过虚拟化技术把处理器分为“实时域”和“非实时域”,实时任务和非实时任务运行在不同的核心上。该方法充分利用多核处理器的各个核心,高效地调度实时任务和非实时任务运行。但该方法是一种特殊的定制,在可扩展性以及广泛的适用性上还存在很多缺陷
能耗问题	2015	能耗问题已成为实时系统调度研究的热点问题。针对周期任务,考虑通用的功耗模型,结合动态电压缩放技术和动态功耗管理技术,文献[42]提出了基于平均空闲时间分配的低功耗调度算法。该算法是两阶段的调度算法,离线阶段计算静态运行速度,回收静态空闲时间;在线阶段回收动态空闲时间,调节处理器的运行速度,并适时地利用动态功耗管理技术关闭处理器,以降低处理器能耗
	2017	针对实时系统周期性任务模型,郭锐锋等[43]利用动态电压调节及动态电压管理技术,考虑通用的功耗模型,使用关键速度,提出一种根据不同任务最坏执行时间比例来分配空闲时间的低功耗算法。该算法分为两个阶段,第一阶段计算离线状态的静态速度,第二阶段在线回收并分配动态空闲时间,降低处理器运行速度来节省能耗
混合关键级调度问题	2014	文献[44]对低关键级任务采取了积极的处理方法,基于同构多处理器平台构建两类队列,一类队列容纳回收的空闲时隙,另一类队列为任务队列,包括就绪任务队列和被抛弃的低关键级任务队列。针对这两种任务队列的特性采取不同的调度方案:就绪任务队列采用混合关键级局部调度,被抛弃的低关键级任务则对空闲时隙进行分配。此调度方法在保证高关键级任务截止时限的同时,能够使混合关键级系统的可接受任务集数目获得明显提升
	2015	文献[45]面向通信竞争的混合关键级汽车电子系统的调度问题进行研究,通过建立多DAG(有向无环图)异构计算模型和异构网络模型,提出了公平策略的多DAG动态任务与消息调度算法F_MDDTMS,以降低多DAG的调度长度;接着提出了关键级策略的多DAG动态任务与消息调度算法C_MDDTMS,以确保高关键级DAG应用的实时性;最后综合F_MDDTMS和C_MDDTMS,提出混合关键级策略的多DAG动态任务与消息调度算法MC_MDDTMS,既确保混合关键级系统中高关键级DAG应用的实时性,又使得低关键级DAG应用得到积极的处理

近期出现的混合关键级调度问题成为嵌入式领域极富挑战性的课题之一。现代实时嵌入式系统,例如航空电子设备以及汽车系统的实际应用,正向着功能不断增加但所占空间更小、重量更轻、成本更低、能耗更少的方向发展。因此使用

冗余硬件分层设计来实现任务之间隔离执行的传统方法，正向着不同关键级的多种功能集成到一块物理平台上，共享硬件资源的研究方向转变。但这种以提高资源利用率为目的的共享可能会使低关键级（Low Criticality，LO）任务对高关键级（High Criticality，HI）任务的正确执行产生干扰，导致 HI 任务错过截止时限，从而造成非常严重的后果。例如从高效利用处理器计算能力的角度，可以将汽车的防抱死制动任务与导航任务放在同一个处理器上执行，若导航任务未能按预设时间执行完毕，防抱死任务很可能无法得到及时执行，从而导致重大安全事故[46-47]。

混合关键级（Mixed Criticality，MC）系统能够同时保证高效的资源利用与高关键任务的正确执行。此类系统的实时调度需同时满足两个目标：一是在相对保守的时间属性设定下，满足认证标准对涉及安全关键功能的验证要求；同时在较乐观的时间属性设定下，满足高效利用计算资源的设计要求。这类混合关键级系统的调度问题，无法使用传统的实时调度算法解决。系统依据标准认证进行关键级分级，任务获得自己的关键级属性，高关键级任务的时间参数值随着关键级增加渐趋保守；在系统处于相对低关键级执行阶段时，基于优先级调度所有任务，在系统处于相对高关键级执行阶段时则优先确保高关键级任务的截止时限[48]。

7　基于深度学习的自适应实时 DVFS 算法

实时嵌入式软件不仅要求功能正确，保证实时性，还需要满足低功耗要求。例如由于太空飞行器存在粒子反转现象，而采用加固计算机将带来散热问题，因此在满足时间要求的情况下应尽可能降低电压和 CPU 工作频率。预测 WCET 不仅判断软件是否满足系统实时性要求，还需要调整 CPU 工作电压和频率，使运行时间接近规定时限以节省能耗。在实时系统中提供最佳低功耗和高性能处理器的一种有前景的技术是动态电压和频率调节（DVFS）。我们研究基于深度学习的实时嵌入式系统功率优化问题：假设处理器频率设置较低并根据任务要求逐渐增加。在实时系统中，每个任务都指定了时间阈值，由于嵌入式系统软件的复杂性、Cache 引入、多核问题会导致 WCET 估算异常复杂，同时 WCET 估算会远高于实际运行时间，因此根据 WCET 选择 CPU 以及确定 CPU 工作频率往往导致实际执行时间远小于时间阈值，这对降低功耗是不利的。RT-DVFS（Real-time Dynamic Voltage and Frequency Scaling）实时动态电压和频率调节技术[49]利用实际工作负载中的这些变化来动态调整电压和频率，以降低处理器的功耗。当实际执行时间 AET 接近时间阈值时，该技术会增加工作频率和电源电压以提高运行速度；当任务执行时间 AET 远低于时间阈值时，其会降低工作频率和电源电压以降低功耗，但无论如何降低功耗都需要以满足 AET 小于时间阈值为前提。本文提出一种自适应实时 DVFS 算法，如图 3 所示，其可以进行动态电压和频率调节（DVFS），以达到降低功耗的目的，并动态修正 WCET 值[50-51]。递推最小二乘由于算法简单、使用方便而在系统辨识领域得到广泛应用，我们采用递推最小二乘方法根据软件时间复杂度、空间复杂度、输入数据规模等信息预测软件执行时的 CPU 电压和工作频率。

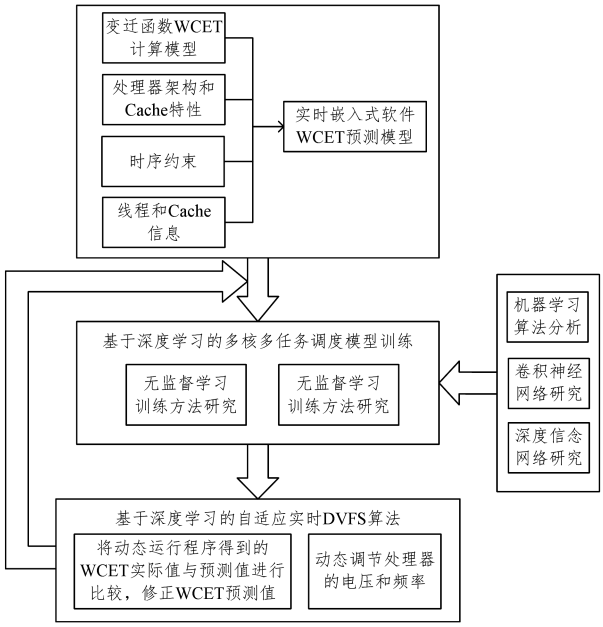

图 3　基于深度学习的自适应实时 DVFS 算法

结束语　在实时嵌入式系统中，衡量系统实时性的最重要的参数是任务的最坏情况执行时间 WCET，它为任务的实时调度、任务的优先级仲裁、资源的冲突仲裁以及任务间通信提供依据，是确保系统安全运行的可信基础。本文首先对 WCET 分析类型及主要方法进行总结，分析目前 WCET 研究存在的主要问题；然后针对 WCET 分析存在的主要问题展开讨论，在时序分析、微系统结构分析和多核多任务调度策略等方面对国内外研究进展进行综述，阐述嵌入式领域近期出现的研究热点——混合关键级调度策略问题；最后，提出了一种基于深度学习的自适应实时 DVFS 算法，该算法可以进行动态电压和频率调节（DVFS），以达到节能的目的，同时还能够动态修正程序的 WCET 值，为未来嵌入式系统中 WCET 的分析与预测提供指导方法。

参 考 文 献

[1]　吕鸣松，关楠，王义. 面向 WCET 估计的 Cache 分析研究综述[J]. 软件学报，2014，25(2)：179-199.

[2]　姬孟洛. 实时系统最差情况执行时间分析的研究[D]. 长沙：国防科学技术大学，2006.

[3]　吕鸣松. 实时系统最坏情况执行时间分析技术的研究[D]. 沈阳：东北大学，2010.

[4]　王海瑞. 基于极值统计的实时软件 WCET 估计研究[D]. 大连：大连理工大学，2007.

[5]　江华. 实时程序 WCET 分析模型与算法[D]. 武汉：华中科技大学，2008.

[6]　陈芳园. 基于多核处理器平台的实时系统 WCET 分析研究[D]. 长沙：国防科学技术大学，2011.

[7]　WILHELM R，GRUND D. How Much Time Does It Take to Calculate? [J]. ACM 通信，2014，57(2)：94-103.

[8]　姬孟洛，李书浩，秦杰，等. WCET 分析中面向对象程序多态性问题的解决方法[J]. 计算机科学，2006，33(11)：249-255.

[9]　张曦. 基于 WCET 分析技术的程序实时性模型检验方法研究[D]. 长沙：国防科学技术大学，2011.

[10]　WU L，ZHANG W. Bounding Worst-Case Execution Time for Multicore Processors through Model Checking[C]// Proc. 16th

IEEE Real-Time and Embedded Technology and Applications Symposium(RTAS'10). Work-in-Progress Session,2010:17-20.

[11] VALERO. Hardware support for WCET analysis of hard real-time multicore systems[C]// Proc. 36th International Symposium on Computer Architecture(ISCA 2009). 2009:57-68.

[12] 王馨,姬孟洛,王戟,等. 基于 WCET 分析的实时系统轨迹获取技术[J]. 软件学报,2006,17(5):1232-1240.

[13] LESAGE B,GRIFFIN D,ALTMEYER S,et al. On the analysis of random replacement caches using static probabilistic timing methods for multi-path programs[J]. Real-Time Systems,2017(2):1-82.

[14] CHATTOPADHYAY S,ROYCHOUDHURY A,MITRA T. Modeling shared cache and bus in multi-cores for timing analysis[C]// International Workshop on Software & Compilers for Embedded Systems. ACM,2010.

[15] DAVIS R I,ALTMEYER S,INDRUSIAK L S,et al. An extensible framework for multicore response time analysis[J]. Real-Time Systems,2017(5):1-55.

[16] 姬孟洛,齐治昌,王怀民. 包含依赖输入分支程序的符号化 WCET 分析[J]. 软件学报,2006,17(3):628-637.

[17] BALLABRIGA C,FORGET J,LIPARI G. Symbolic WCET Computation[J]. ACM Transactions on Embedded Computing Systems(TECS),2017,17(2):1-26.

[18] 陈芳园,丁亚军,张冬松,等. 面向 WCET 分析的实时多核体系结构研究[J]. 计算机工程与科学,2014,36(3):393-398.

[19] PAOLIERI M,CAZORLA F J,BERNAT G,et al. Hardware support for WCET analysis of hard real-time multicore systems[C]// International Symposium on Computer Architecture. ACM,2009:57-68.

[20] 甘志华,古志民,安立奎,等. 基于 WCET 的多核共享资源冲突分析与约束研究[J]. 计算机科学,2014,41(8):19-24.

[21] 陈芳园,张冬松,王志英. 多核实时线程间干扰分析及 WCET 估值[J]. 电子学报,2012,40(7):1372-1378.

[22] LIU T,XUE C J. Instruction cache locking for multi-task real-time embedded systems[J]. Real-Time Systems,2012,48(2):166-197.

[23] 张吉赞,古志民. 多核共享缓存 bank 冲突分析及其延迟最小化[J]. 计算机学报,2016,39(9):1883-1899.

[24] NAGAR K,SRIKANT Y N. Refining Cache Behavior Prediction Using Cache Miss Paths[J]. Acm Transactions on Embedded Computing Systems,2017,16(4):1-26.

[25] 杨志斌,赵永望,黄志球,等. 同步语言的时间可预测多线程代码生成方法[J]. 软件学报,2016,27(3):611-632.

[26] 安立奎,古志民,付引霞,等. 支持软件预取的缓存 WCET 分析[J]. 北京理工大学学报,2015,35(7):730-736.

[27] 王恩东,倪璠,陈继承,等. 一种面向实时系统的程序基本块指令预取技术[J]. 软件学报,2016,27(9):2426-2442.

[28] PUFFITSCH W. Persistence-based branch misprediction bounds for WCET analysis[C]// ACM Symposium on Applied Computing. ACM,2015:1898-1905.

[29] CARMINATI A,STARKE R A,OLIVEIRA R S D. On the use of static branch prediction to reduce the worst-case execution time of real-time applications[J]. Real-Time Systems,2018:1-25.

[30] MANGEAN A,BÉCHENNEC J L,BRIDAY M,et al. WCET Analysis by Model Checking for a Processor with Dynamic Branch Prediction[M]// Verification and Evaluation of Computer and Communication Systems. 2017:64-78.

[31] SU X,WU H,YANG Q. An Efficient WCET-Aware Hybrid Global Branch Prediction Approach[C]// IEEE, International Conference on Embedded and Real-Time Computing Systems and Applications. IEEE,2016:195-201.

[32] 张冬松. 多核多处理器系统的节能实时调度技术研究[D]. 长沙:国防科学技术大学,2012.

[33] 姚鑫骅. 数控实时系统调度理论及应用研究[D]. 杭州:浙江大学,2006.

[34] 王磊,刘道福,陈云霁,等. 片上多核处理器共享资源分配与调度策略研究综述[J]. 计算机研究与发展,2013,50(10):2212-2227.

[35] 刘轶,张昕,李鹤,等. 一种面向多核处理器并行系统的启发式任务分配算法[J]. 计算机研究与发展,2009,46(6):1058-1064.

[36] 耿晓中. 基于多核分布式环境下的任务调度关键技术研究[D]. 长春:吉林大学,2013.

[37] 赵林祥. 基于多核处理器任务复制的分簇调度算法研究[D]. 长沙:湖南大学,2012.

[38] LIU Y,ZHANG X,LI H,et al. Allocating Tasks in Multi-core Processor based Parallel Systems[C]// 2007 IFIP International Conference on Network and Parallel Computing Workshops(NPC2007),2007.

[39] LEE L T,CHANG H Y,CHAO S W. A Hybrid Task Scheduling for Multi-Core Platform[C]// 2008 Second International Conference on Future Generation Communication and Networking Symposia. 2008:40-45.

[40] HATANAKA K,BAGHERZADEH N. Scheduling Techniques for Multi-Core Architectures[C]// 2009 Sixth International Conference on Information Technology:New Generations. 2009:865-870.

[41] 冯华,卢凯,王小平. 面向多核处理器的实时优化技术:基于独立实时域的实时优化方法[J]. 计算机科学,2013,40(9):159-162.

[42] 张忆文,郭锐锋,刘娴,等. 基于平均空闲时间分配的低功耗调度算法[J]. 小型微型计算机系统,2015,36(8):1907-1910.

[43] 郭锐锋,吴昊天,邓昌义,等. 一种 WCET 比例空闲时间分配的周期任务低功耗算法[J]. 小型微型计算机系统,2017,38(8):1856-1860.

[44] 黄丽达,李龙,李仁发,等. 混合关键级多任务调度中低关键级任务的积极处理[J]. 计算机工程与科学,2014,36(1):6-11.

[45] 刘樑骄,谢国琪,李仁发,等. 通信竞争的混合关键级系统多 DAG 动态调度策略[J]. 计算机研究与发展,2015,52(11):2608-2621.

[46] 黄丽达,李仁发. 事件触发关键级提升的实时任务可调度性分析[J]. 计算机研究与发展,2017,54(1):184-191.

[47] 赵庆玲. 混合关键度 CPS 系统中的资源共享协议和设计优化[D]. 杭州:浙江大学,2015.

[48] 黄丽达. 混合关键级调度的若干关键问题研究[D]. 长沙:湖南大学,2016.

[49] PILLAI P,SHIN K G. Real-time dynamic voltage scaling for low-power embedded operating systems[J]. Acm Sigops Operating Systems Review,2001,35(5):89-102.

[50] NAIK B V,DAS S,KAPOOR H K. RT-DVS for Power Optimization in Multiprocessor Real-Time Systems[C]// International Conference on Information Technology. IEEE,2015:24-29.

[51] WANG W,RANKA S,MISHRA P. Energy-aware dynamic slack allocation for real-time multitasking systems[J]. Sustainable Computing Informatics & Systems,2012,2(3):128-137.