基于分布函数的WCET快速估计

周国昌[1] 郭宝龙[2] 高 翔[1] 王 健[2] 闫允一[2]
(中国空间技术研究院西安分院 西安 710000)[1] (西安电子科技大学 西安 710071)[2]

摘 要 在实时软件系统中,软件时间性能的分析与评估技术是一个重要的课题,然而随着CPU的结构越来越复杂,采用传统的模拟底层硬件执行的方法越来越困难。而基于分布函数的最坏执行时间(Worst Case Execution Time,WCET)估计方法从概率角度出发,可以绕过复杂的底层硬件建模,估计程序的最坏执行时间。首先对TI TMS320C6713 DSP汇编代码进行基本块的划分,以基本块为结点构建程序流图;然后用贝塔分布模拟每条指令的运行时间并采用改进的计划评审技术(Program Evaluation and Review Technique,PERT)确定贝塔分布相关参数,指令叠加后用正态分布模拟每个基本块的执行时间;最后利用基于路径的方法得到整个程序的最坏执行时间。实验结果表明此方法是可行的和合理的。

关键词 WCET,DSP,PERT,实时软件
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.029

Fast Estimation of WCET Based on Distribution Function

ZHOU Guo-chang[1] GUO Bao-long[2] GAO Xiang[1] WANG Jian[2] YAN Yun-yi[2]
(China Academy of Space Technology(Xi'an),Xi'an 710000,China)[1] (Xidian University,Xi'an 710071,China)[2]

Abstract In the real-time software systems,the software-time performance analysis and evaluation techniques are important. But with the structure of CPU being more and more complex,traditional methods based on underlying hardware simulations are becoming increasingly difficult. Based on distribution function,the worst-case execution time(WCET) estimation method that is from a probabilistic perspective,can bypass the complex underlying hardware modeling and estimate the worst-case execution time. This article first divided TI TMS320C6713 DSP assembly code into basic blocks to build the program flow diagram. Then the run time of each instruction was simulated based on beta distribution whose parameters are determined according to the improved program evaluation and review technique(PERT). The instruction execution time was superimposed based on normal distribution to simulate the run time of each basic block. Finally, the worst execution time of the whole program was gotten by using a path based approach. Experimental results show that this method is feasible and reasonable.

Keywords WCET,DSP,PERT,Real-time software

1 介绍

在实时嵌入式软件系统中,系统对时间具有更加严格的要求,系统的正确性不仅与其程序的逻辑有关,还与其时间特性有关,只有在规定时间内完成规定的任务时系统才是有效的,否则会导致该系统性能的降低甚至系统的失败。因此,事先获取系统中每个任务的最坏执行时间(WCET)[1]对实时系统的时序分析等具有重要的意义。

由于现在的处理器为了提高性能采用了很多技术,主要包括高速缓存、流水线、分支预测等,这使得事先估计代码的执行时间成为很困难的事情,主要原因是无法仅通过分析汇编代码得到Cache命中率、流水线执行情况。尤其在DSP中,不同的CMD文件配置可能导致Cache命中率不同。除此之外,流水线技术的使用也导致了很多不确定的因素,主要包括流水线是否被打断等。

文献[2]对WCET估计的各种方法、工具进行了详细的叙述,并讨论了如何将WCET估计方法、工具应用于具体的开发过程。文献[3]对实时操作系统(Real-time Operating System,ROS)WCET的估计方法以及目前存在的一些问题进行了介绍。由于计算机存储系统对WCET的估计有很大的影响,因此文献[4]定量地分析了3种指令缓存方法对WCET估计的影响;文献[5]提出了一种基于基本块的指令预取(Basic Block based Instruction Prefetching,BBIP)机制,用于改善针对实时应用的WCET分析的界;文献[6]提出了一种基于Petri网的参数化WCET估计方法,用于估计对NAND闪存进行操作的闪存转换层(Flash Translation La-

到稿日期:2015-03-03 返修日期:2015-07-13

周国昌(1978—),男,研究员,主要研究方向为计算机体系结构星载ASIC设计研究;**郭宝龙**(1962—),男,教授,主要研究方向为图像处理、模式识别、电路与智能系统;**高 翔**(1982—),男,高级工程师,主要研究方向为ASIC可靠性研究;**王 健**(1990—),男,硕士,主要研究方向为智能信息处理与控制;**闫允一**(1979—),男,副教授,主要研究方向为复杂电子系统可靠性设计以及智能信息处理。

yer,FTL)的最坏运行时间。这些基于硬件结构的 WCET 分析在一定程度上提高了 WCET 的分析精度。此外,一些研究者从编译器入手进行了 WCET 估算的研究。如文献[7]对 WCET 分析工具及其与编译器之间的接口和相互作用进行了介绍,并将两者进行了集成。

目前,存在许多 WCET 分析方法,WCET 分析方法按是否运行程序分类,主要包括 3 种[8]:静态分析法、动态度量法和混合分析法。

动态度量法[9]就是通过运行程序动态地测试程序的执行时间,进而分析得到 WCET。由于可能无法获得源代码从而无法通过分析源码估计 WCET,因此,文献[10]给出了一种动态分析程序 WCET 的方法,该方法通过给执行路径标记时间戳并重构程序的控制流图,进而推导出 WCET。然而动态度量法存在很大的弊端,首先该方法需要目标机,需要实际运行程序并需要大量的实验,耗费人力、物力、财力,并且在航天、航空领域,很多情况下不能模拟真实的目标机环境或者目标机昂贵,且对实时性要求较高,不可能让未经检验的程序在真实的目标机上运行[11]。除此之外,该方法很难保证测量得到的结果是安全的,即无法保证不低估最坏执行时间,因为很难保证所做的实际测量满足程序 WCET 的条件,尤其对于现代处理器来说,此点更难做到。

混合分析法即将静态分析法和动态度量法相结合的方法,该方法具体分为两种情况,一种是首先对程序进行静态分析,即运用静态分析法进行分析,然后根据静态分析法的结果对程序进行运行测试,最终确定程序的 WCET;另一种是首先对程序采用动态度量法进行测试分析,然后在动态度量法的基础上进行静态分析,进而确定程序的 WCET。文献[12]提出了一种 WCET 混合分析方法,该方法使用了参数回归以辨识用于 WCET 计算的 IPET(Implicit Path Enumeration Technique)模型的参数,并保证通过此方法估算出来的 WCET 不会小于任何通过实际运行得到的时间。但无论采用哪种方法,混合分析法在包含静态分析法和动态度量法的优点基础上,也包含它们各自的缺点,并且混合分析法比较复杂。

静态分析法是通过理论分析估算得到程序的 WCET,该方法不仅可以在不运行程序的情况下获得计算结果,而且能够保证得到的结果是安全的,因此,这是目前最常用的方法。静态分析方法可以进一步分为两类:一类经过分析程序得到一个确定的 WCET 的估计值;而另一类通过分析程序得到 WCET 的估计表达式,该方法被称为参数化的 WCET 估计法。例如,文献[13]在以往研究的基础之上,提出了一种更高效的参数化 WCET 分析方法。文献[14]中介绍的方法首先将整个工程分解成各个组件,用公式表达组件的 WCET,进而利用模型驱动工程(Model Driven Engineer,MDE)的方法,由各个子组件 WCET 的公式推导出由组件构成的整体的 WCET。第二类分析方法较为复杂,并且应用的场景可能受到一定程度的限制。

传统静态分析方法一般由 3 个部分组成[11]:(1)程序的控制流分析,主要分析确定程序可能的执行路径、最大循环次数和不可行路径等;(2)底层分析,主要考虑目标机器的特征,比如高速缓存[15]、流水线[16]、分支预测等,目的是确定每条汇编语句或者每个基本块执行的时间;(3)WCET 计算,根据前两项信息,运用具体的计算方法对程序的 WCET 进行计算,主要计算方法分为基于路径的方法、基于树的方法和隐含路径枚举的方法。

总体上说,传统的静态分析方法是在 WCET 计算过程中,把每条汇编指令的执行时间作为确定值来对待的,然后通过一定的方法得到程序的 WCET,但此值也是确定的。可以看到,传统的方法没有考虑到程序执行时间的不确定性,根据现在处理器的特点,这种方法显然是存在很大弊端的,因为现在处理器为了加快运算的速度采用了很多先进的技术,比如高度缓存、流水线、分支预测等,使得程序执行时间的不确定性更加明显。

因此,程序的运行时间不是单一的某个值,而是某个范围的区间值。目前针对此问题的解决方法主要是用概率法求解程序的 WCET,即 pWCET[17],此概念是由 Guillem Bernat 在 2003 年提出来的,主要目的是解决某段代码或者某个函数等的 WCET 的概率分布问题。例如,文献[18]提出的采用极值统计理论求解程序执行时间区间的方法,其大概思想为:多次运行程序,得到很多个程序运行时间值,然后用 Gumble 概率函数逼近,求取函数极值,即为 WCET 值。分析此文献,容易看出,此种方法不是纯粹的静态分析,因为它是对多次运行程序得到时间值进行函数拟合逼近,求极值得到 WCET,同样具有动态度量法的局限性。文献[9]也采用极值统计方法估算程序最坏执行时间,但同样具有文献[18]的缺点。

文献[11]提出了一种新的用概率法估计程序执行时间的方法。然而,该方法需要较多的人工预处理,不适用于大型工程。同时,该方法借鉴了计划评审技术(PERT)[19]的思想,然而,该文献采用传统的 PERT 方法,该方法本身具有很大的弊端,主要在于贝塔分布参数估计方法的不精确性。

本文在文献[11]的基础上,提出了一种新的基于分布函数的程序最大执行时间(WCET)的估计方法,在本文算法中采用改进的 PERT 方法确定贝塔分布的相关参数,并借鉴编译原理中基本块的概念[20]分析构造程序流图,减少人工预处理,自动化估算得到 WCET。实验结果证明本文算法能比原算法得到更接近实际情况的最坏执行时间,且能较好地适用于大型工程代码。

2 基于分布函数的 WCET 实现

基于分布函数的 WCET 实现的基本思想为:从概率角度出发,宏观上把握程序的执行时间,进而得到程序执行时间的区间。具体为:首先,根据 TI DSP 汇编代码的特点并结合编译原理中基本块的概念构建程序流图;然后,采用贝塔分布模拟每条指令运行时间,根据概率定理,指令叠加后用正态分布模拟每个基本块的执行时间;最后,采用基于路径的方法得到整个程序的最大执行时间,即 WCET。

2.1 构造程序流图

根据基本块概念[20]并结合 TI DSP 汇编指令[21]的特点,对 DSP 汇编代码进行分析得到各个基本块,并确定基本块之间的拓扑关系,进而构建程序流图。

一个基本块是指在静态分析时汇编代码中一段顺序执行的代码序列，其只包含一个进入基本块指令和一个退出基本块指令，进入基本块的语句就是它的第一条代码语句，退出基本块的语句就是它的最后一条代码语句。

基本块入口语句具体判定准则为：(1)程序的第一条语句；(2)条件跳转指令或者无条件跳转指令能够跳转到的语句；(3)条件跳转指令后的语句；(4)无条件跳转指令后的语句。若满足上面条件之一，则判定为基本块入口语句。

基本块出口语句具体判定准则为：(1)下一个进入基本块的语句之前的那条语句；(2)汇编代码的终止语句。若满足上面条件之一，则判定为基本块出口语句。

综上分析，基本块划分的步骤为：

1)根据基本块入口语句和出口语句的判断准则，求出所有入口语句和出口语句；

2)每个入口语句对应一个基本块，该基本块由该基本块入口语句到基本块出口语句之间的语句序列组成。

基本块关系确定方法为：

1)若基本块 i 的出口语句为转移语句(条件或者无条件转移语句)，并且该转移语句的目标地址等于基本块 j 的入口语句对应的地址，则 i 可以到达 j；

2)若基本块 i 的出口语句为条件转移语句，并且该出口语句对应的地址加 4 等于基本块 k 的入口语句对应的地址，则 i 可以到达 k；

3)若基本块 i 的出口语句为无条件转移语句，并且该出口语句对应的地址加 4 等于基本块 k 的入口语句对应的地址，则 i 可以到达 k，但此时基本块 k 在基本块 i、j 之后执行；

4)若基本块 i 的出口语句不为转移语句(条件或者无条件转移语句)，并且该基本块的出口语句对应的地址加 4 等于基本块 m 的入口语句对应的地址，则 i 可以到达 m。

通过上面的判断方法，便可以确定各个基本块范围及其之间的拓扑关系，并用数据结构中基于邻接表的有向图进行描述存储，将基本块作为有向图中的各个节点，得到程序流图。比如，对于冒泡排序程序，可以得到如图 1 所示的程序流程图，具体为 main 函数所对应的程序流程图，除此之外，其还调用一个库函数。

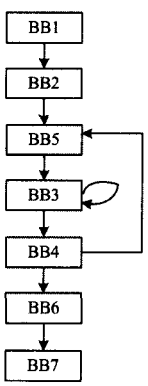

图 1　冒泡排序主程序流程图

其中 BB1－BB7 为分析冒泡排序对应 DSP 汇编代码得到的各个基本块，由图可以看出构造的程序流程图符合冒泡排序程序实际的逻辑关系。

2.2　估算基本块的 WCET

估算基本块的 WCET 的基本思想为：采用贝塔分布模拟每条 DSP 汇编指令运行时间，并考虑指令的并行情况，指令叠加后用正态分布模拟每个基本块的执行时间。具体步骤如下。

1) 计算每条指令运行时间的期望值 μ_i 和方差 σ_i^2

采用贝塔分布模拟每条 DSP 汇编指令运行时间并用改进的计划评审技术(PERT)确定贝塔分布的期望值和方差，即采用 Perry-Greig 期望值近似公式[22]计算每条 DSP 指令运行时间的期望值 μ_i，采用 Person-Turkry 方差近似公式[22]计算每条 DSP 指令运行时间的方差 σ_i^2，具体计算公式分别为：

$$\mu_i = 0.630m_i + 0.185(b_i + a_i)$$

$$\sigma_i^2 = 0.630(m_i - u_i)^2 + 0.185[(b_i - u_i)^2 + (a_i - u_i)^2]$$

其中，b_i 表示每条指令运行时间峰值，a_i 表示每条指令运行时间谷值，其中峰值表示指令最长运行时间，谷值表示指令最短运行时间，峰值由指令类型确定，由于 TI DSP 中指令在理想情况下执行时间基本为单周期的，因此谷值确定为 1，m_i 表示每条指令运行时间的最可能值，根据多次实验分析发现对于大部分情况，一条指令的执行时间为 1 个 CPU 周期，故取最可能值为 1。

2) 筛选并行指令

在基本块中可能存在并行执行的指令，需要对指令并行性进行分析，从而使得估算的基本块执行时间更加准确。根据上一步骤获得的每条指令运行时间的期望值 μ_i 和方差 σ_i^2，并结合指令并行情况，筛选得到最终参加指令叠加运算的指令信息，具体为：

$$F_j = \max(t_1, \cdots, t_k)$$

其中，t_k 表示基本块中每个并行执行块中第 k 条指令的执行时间，$t_k = A_k + 2S_k$，A_k 为并行执行块中第 k 条指令对应基本块中指令的执行时间的期望值，S_k 为并行执行块中第 k 条指令对应基本块中指令的执行时间的方差；k 取值范围为 1，2，…，8，因为在 TI DSP 中最多只能有 8 条指令并行执行；max 为取最大值操作；F_j 表示基本块中每个并行执行块的执行时间，j 的取值范围为 1，2，…，M 表示基本块中包含的并行执行块个数。

3) 计算基本块执行时间均值 μ 和方差 σ^2，具体的计算公式分别为：

$$\mu = \sum_1^M \mu_j$$

$$\sigma^2 = \sum_1^M \sigma_j^2$$

其中，μ_j 表示对应 F_j 的指令运行时间的期望值，σ_j^2 表示对应 F_j 的指令运行时间的方差。

4)获得基本块执行时间 T

求得基本块执行时间的均值和方差后，给定时间 t，就可以计算 $P(T' \leqslant t)$，即基本块执行时间不大于某一给定值的概率。

对于正态分布 $X: N(\mu, \sigma^2)$，有如下事实：

$P(t \leqslant \mu + \sigma) = 84.15\%$

$P(t \leqslant \mu + 2\sigma) = 97.7\%$

$P(t \leqslant \mu + 3\sigma) = 99.85\%$

本文取 $T=\mu+2\sigma$ 作为基本块的执行时间。

对于冒泡排序程序，估算得到的各个基本块执行时间如表1所列，表1中只包括 main 函数对应的基本块，库函数的基本块没有列出。

表1 基本块执行时间结果

基本块标号	基本块起始地址	基本块终止地址	基本块执行时间(CPU Cycle)
BB1	000005c0	000005c0	6.76957
BB2	000005c4	000005fc	6
BB3	00000628	0000062c	7.76957
BB4	00000630	0000066c	10.7696
BB5	00000600	00000624	3
BB6	00000670	00000674	6.76957
BB7	00000678	0000067c	0

表1中，对于基本块 BB7，由于其对应汇编指令皆为 NOP 指令，因此其执行时间为0。

2.3 估算程序的 WCET

程序的最坏执行时间(WCET)不仅仅与基本块的执行时间相关，还与程序的执行路径和基本块执行次数有关，所以要准确得到程序的最坏执行时间，需把程序的执行路径、基本块执行时间和基本块执行次数相结合。基于此原因，将基本块执行时间和执行次数作为权值构造加权程序流图。

构造了加权程序流图，即加权有向图之后，便可以利用有向图的一些处理方法得到所需的信息。利用路径搜索算法，寻找一条总权值最大的路径，则其执行时间最长，并将此路径上的总权值作为程序的 WCET。具体思想为：首先，利用深度优先算法获得加权程序流图的各条路径；然后，用每条路径上各个基本块的执行时间乘以其对应基本块执行次数，累加求和，得到该条路径上的执行时间；最后，寻找所有路径上执行时间的最大值，即为程序的 WCET，其中基本块执行次数由人工估计得到。

3 实验结果及分析

为了检验本文提出的算法，利用 TI CCS 集成开发环境编写测试程序并得到所需的对应汇编代码，采用本文算法估算程序的最坏执行时间 WCET。同时为了说明本文方法的有效性和准确性，与 CCS 软仿真结果和文献[11]提出的方法进行比较，对于文献[11]，本文只实现其方案2的估算方法，因为按照文献[11]的描述，其方案2比方案1更好。

实验分别编写冒泡排序、选择排序等7种常见的排序程序进行测试，其结果如表2所列。其中大部分的输入采用一组逆排序数据，这样通过 CCS 软仿真的方法可以尽可能地逼近程序的最坏执行时间。

表2 排序程序测试结果(CPU Cycle)

函数名称 \ 方法名称	软仿真方法	文献[4]方法	本文方法
冒泡排序	479	372	538
选择排序	470	364	533
希尔排序	678	511	681
堆排序	854	711	935
快速排序	1258	933	1345
插入排序	1544	1092	1761
归并排序	1919	1469	2085

为了更加形象地看出3种方法所测出的数据的区别，现将其绘制成为图2。

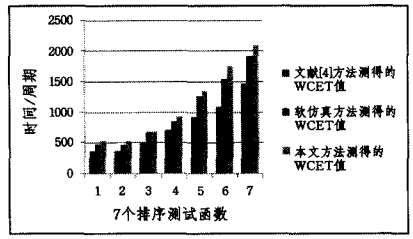

图2 排序程序测试结果

为了进一步验证方法的可靠性，我们对在 DSP 上的两个实时程序进行测试，包括基于 8×8 矩阵的 DCT(离散余弦)变换和采样点为64位的 FFT(快速傅里叶)变换两个程序。

表3 DCT 和 FFT 测试结果 (CPU Cycle)

函数名称 \ 方法名称	软仿真方法	文献[4]方法	本文方法
DCT	235742	204882	258299
FFT	673161	606510	744228

由图2和表3可以看出，本文方法估计得到的程序 WCET 值比较接近软仿真方法逼近的程序 WCET 值，并且本文方法得到的 WCET 值较文献[11]方法更具安全性，因此本文估算方法更加合理。

由实验结果可以看出，本文方法测量得到的数据与真实值比较接近，且相对仿真所得到的数据偏大，这个误差是由多方面原因引起的。首先，改进的 PERT 方法虽然精确度相对传统 PERT 较高，然而在3个参数乐观时间 a、悲观时间 b 和最可能时间 m 理想的情况下，会有0.5%的误差，而3个参数也是凭借实验估计得到，很难达到理论值，不可避免地存在误差，所以对整个工程来说，实验误差不可避免，由实验数据可以得到相对误差在15%范围以内。其次，实验数据的误差与人工估计循环模块的循环次数非常密切，汇编程序下有大量的循环模块，因此循环次数的准确度是非常重要的，在循环次数比较模糊的情况下，出于软件安全运行方面的考虑，最好让循环次数偏大。除此之外，我们知道程序在 DSP 硬件上运行时，有多种因素使实际时间比仿真时间长一些，比如中断等原因会使实际的时间更长，出于安全性考虑，可在测得的时间基础上乘以一个大于1(比如1.05到1.1之间)的因子。

结束语 实时性分析对于实时系统有重要的意义，而程序执行时间估算是实时性分析研究的重要课题。本文提出的基于分布函数的 WCET 快速估计方法属于概率法求解 WCET 的一种，本方法采用静态的方式分析估算程序的最坏执行时间，绕过复杂的底层建模和动态度量法的弊端，可用于航天领域中程序 WCET 的估算，实验结果证明该方法可以获得较准确的程序最坏执行时间且获得的 WCET 值是安全的。

参考文献

[1] Puschner P, Burns A. A review of worst-case execution time analysis[J]. Real-Time Systems, 2000, 18(2/3): 115-128

[2] Wilhelm R, Engblom J, Ermedahl A, et al. The worst-case execution-time problem-overview of methods and survey of tools [J]. ACM Transactions on Embedded Computing Systems,

2008,7(3):1-53

[3] Lv M,Guan N,Zhang Y,et al. A Survey of WCET Analysis of Real-Time Operating Systems[C]//Proceedings of the 2009 International Conference on Embedded Software and Systems. 2009:65-72

[4] Metzlaff S, Ungerer T. Impact of Instruction Cache and Different Instruction Scratchpads on the WCET Estimate[C]//2011 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS). 2012:1442-1449

[5] Ni F,Long X,Wan H,et al. Using Basic Block Based Instruction Prefetching to Optimize WCET Analysis for Real-Time Applications[C]//Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. 2012:459-466

[6] Yoo J, Lee J, Hong S. Petri Net-Based FTL Architecture for Parametric WCET Estimation via FTL Operation Sequence Derivation[J]. IEEE Transactions on Computers, 2013, 62(11): 2238-2251

[7] Puschner P, Prokesch D, Huber B, et al. The T-CREST approach of compiler and WCET-analysis integration[C]//2013 IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC). 2013:1-8

[8] Ji Meng-luo,Qi Zhi-chang. An Overview of Worst Case Execution Time (WCET) Analysis[J]. Computer Science, 2006, 33(10):238-241(in Chinese)
姬孟洛,齐治昌. 实时系统程序最差情况执行时间(WCET)的分析[J]. 计算机科学,2006,33(10):238-241

[9] Zhang Bao-min,Wu Guo-wei,Yao Lin. Program worst-case execution time extreme value statistics estimation method[J]. Computer Engineering and Applications,2010,46(26):67-71(in Chinese)
张保民,吴国伟,姚琳. 程序最坏执行时间极值统计方法[J]. 计算机工程与应用,2010,46(26):67-71

[10] Marref A,Betts A. Accurate Measurement-Based WCET Analysis in the Absence of Source and Binary Code[C]//Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. 2011:127-135

[11] Hu Ming-hua,Tang Ming-duan. Worst case execution time estimation based on distribution function[J]. Computer Engineering and Design,2006,27(16):3045-3047(in Chinese)
胡明华,汤铭端. 基于分布函数的程序执行时间的静态预估[J]. 计算机工程与设计,2006,27(16):3045-3047

[12] Lisper B, Santos M. Model Identification for WCET Analysis[C]//Proceedings of the 2009 15th IEEE Symposium on Real-Time and Embedded Technology and Applications. 2009:55-64

[13] Bygde S, Ermedahl A, Lisper B. An Efficient Algorithm for Parametric WCET Calculation[C]//Proceedings of the 2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. 2009:13-21

[14] Leveque T,Borde E,Marref A,et al. Hierarchical Composition of Parametric WCET in a Component Based Approach[C]//Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. 2011:261-268

[15] Lv Ming-song,Guan Nan,Wang Yi. Survey of Cache Analysis for Worst-Case Execution Time Estimation[J]. Journal of Software,2014,25(2):179-199(in Chinese)
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

[16] Wu Guo-wei, Li Zhang. Precise program worst-case execution time analysis method[J]. Computer Engineering and Applications,2010,46(18):60-64(in Chinese)
吴国伟,李张. 一种精确程序最坏执行时间分析方法[J]. 计算机工程与应用,2010,46(18):60-64

[17] Bernat G, Colin A, Petters S. pWCET: a Tool for Probabilistic Worst-Case Execution Time Analysis of Real-Time Systems: YCS-2003-353[R]. England,UK:University of York,2003

[18] Petters SM. How much worst case is needed in wcet estimation[C]//International Workshop on Worst Case Execution Time Analysis. 2002

[19] http://wiki.mbalib.com/wiki/PERT

[20] Aho V A,Sethi R,Ullman J D. 编译原理[M]. 北京:机械工业出版社,2008

[21] T. I. TMS320C6000 CPU and instruction set reference guide[Z]. 2000

[22] Wu Su,Luo Yan-sheng. Application of a New Module of PERT in Project Management[J]. Construction Technology, 2007, 36(12):50-53(in Chinese)
吴苏,罗延生. 一种新 PERT 模型在项目管理中的应用[J]. 施工技术,2007,36(12):50-53

(上接第 156 页)

[13] Holzmann G J. The SPIN model checker: Primer and reference manual[M]. Reading: Addison-Wesley,2004

[14] Baier C,Katoen J P. Principles of model checking[M]. Cambridge: MIT press,2008

[15] Harel D. Statecharts: A visual formalism for complex systems[J]. Science of Computer Programming,1987,8(3):231-274

[16] Mikk E,Lakhnech Y,Siegel M,et al. Implementing statecharts in PROMELA/SPIN[C]//Proceedings 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques, 1998. IEEE,1998:90-101

[17] Latella D,Majzik I,Massink M. Automatic verification of a behavioural subset of UML statechart diagrams using the SPIN model-checker[J]. Formal Aspects of Computing,1999,11(6): 637-664

[18] Dong Wei,Wang Ji,Qi Zhi-chang. Slicing UML Statecharts for Model Checking[J]. Acta Electronica Sinica, 2002, 30(12): 2084-2089(in Chinese)
董威,王戟,齐治昌. UML Statecharts 的切片模型检验方法[J]. 电子学报,2002,30(12):2084-2089