第 35 卷 第 4 期 计 算 机 工 程 2009 年 2 月
Vol.35 No.4 Computer Engineering February 2009
·博士论文· 文章编号：1000—3428(2009)04—0004—03 文献标识码：A 中图分类号：N945

非经典切片优化的同步运行时检验方法

高新岩[1]，吴尽昭[1]，乔 瑞[1]，闫 炜[2]
(1. 中国科学院成都计算机应用研究所，成都 610041；2. 电子科技大学计算机学院，成都 610054)

摘 要：利用现有的同步 IP 核来构建全局异步局部同步系统是未来片上系统设计的一个重要发展方向。在整个设计流程中，正确的接口设计和同步问题是至关重要的。该文提出一种改进的抽象时序图与基于计算切片优化技术的谓词检测方法相结合的同步验证技术。该技术可以使待检查的全局状态空间的规模指数级缩减，使验证效率得到提高。

关键词：非经典切片；计算切片；全局异步局部同步系统；谓词检测；偏序迹

Non-classical Slicing Optimization Runtime Verification Approach for Synchronization

GAO Xin-yan[1], WU Jin-zhao[1], QIAO Rui[1], YAN Wei[2]
(1. Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu 610041;
2. School of Computer, University of Electronic Science and Technology, Chengdu 610054)

【Abstract】 Reusing the existing IP cores to compose globally asynchronous locally synchronous system is gaining increasing importance for the upcoming system-on-chip designs. The correct interface design and synchronization issue become a crucial step for the whole development process. This paper proposes a verification technique based on modified abstract timing diagrams and predication detection method with non-classical computation slicing optimization technique for synchronization. It is very effective due to exponential gains in reducing the global state space.

【Key words】 non-classical slicing; computation slicing; globally asynchronous locally synchronous system; predication detection; partial order traces

1 概述

当前，随着微电子技术的飞速发展，数亿只晶体管已经可以集成到一个单芯片上构成一个功能复杂的系统，电路设计师面临着创建如此复杂的系统在很高时钟频率上工作时稳定与可靠性所带来的挑战，特别是如果整个系统仅采用单一同步时钟进行时钟分布时，由于元件间互连延迟的影响，这个困难将会更加突出。

综合考虑，人们提出了一种可以有效地减轻时钟分布和时序收敛影响的方法，即全局采用异步时钟模式、局部组件采用同步时钟模式的方法。这种方法最早在 1982 年由 Chapiro[1]提出，该方法的核心思想是将同步设计与异步设计方法的优缺点进行互相折中与权衡。

通常，一个 GALS 系统由若干局部自治的同步模块通过异步通信协议互连构成，GALS 的系统架构非常适合于 SoC 系统实现，因为现有的 IP 核大都是成熟的同步模块，只需设计异步互连系统连接各个模块便可构造出整个系统。

不同时钟域或局部同步模块间通过异步协议传送数据时需要时序的同步，数据进入另一个时钟域时或者数据在接收寄存器采样时恰好发生改变都可能会引起亚稳定态或传送失败。因此很有必要去建立一套有效的异步通信协议和模块边界的高速同步机制，即正确的接口设计是未来 SoC 系统发展的重要课题，分析和验证这些协议和同步技术同样也是一项具有挑战性的任务，本文主要研究这个问题。

本文将基于非经典切片技术——计算切片优化[2]的谓词检测方法应用于解决时序同步问题，并且给出将时序验证问题转换为谓词检测问题的算法和转化规则，通过该规则和转换方法，只需少量修改，便可使用现有的偏序迹执行检测工具如 Vijay K. Garg 等人开发的分析器(Partial Order Trace Analyzer, POTA)[3]进行运行时检测。

2 相关工作

限于篇幅，这里只介绍与本文内容有关的分布式计算及其谓词检测的相关概念。

2.1 谓词检测

下面介绍谓词检测相关的基本概念。

简单地，对于异步分布式程序，检测其某个有限执行迹是否满足一个给定的谓词(如，使用调试器对分布式程序进行调试时，若当前所有进程的执行状态满足某个断点条件，则系统停止执行，这便是谓词检测的一个应用实例)的问题，称之为“谓词检测”问题。所谓谓词就是一个定义在状态集合上的一个布尔函数，通常用谓词来描述系统的性质。

谓词检测不仅是分布式系统的一个重要问题，同时，也是一种极为有效的运行时检验方法，采用数学技术分析所刻画的性质是否满足，比形式化方法的开销低得多，因为它仅检查一条特殊的执行迹的正确性而不需要证明所有可能的执行迹的正确性。

基金项目：国家自然科学基金资助项目(60373113)；国家“973”计划基金资助项目(2004CB318000)
作者简介：高新岩(1976－)，男，博士，主研方向：形式刻画与验证；吴尽昭，研究员、博士生导师；乔 瑞、闫 炜，博士
收稿日期：2008-09-20 **E-mail：**Tyler_Gao@hotmail.com

谓词检测存在状态爆炸问题，即所有可能全局状态数随着进程数目的增加呈指数级增长。

2.2 计算切片

为了有效解决谓词检测问题，Vijay K. Garg 等人[2]首次提出计算切片的概念，计算切片作为一种有效的抽象机制，受到了 M. Weiser 在 1979 年[4]提出的经典程序切片思想启发而产生的非经典切片技术，理论与实验结果表明这是分布式计算中谓词检测中缩减状态空间的一种强有力的技术。

直观上，分布式系统中某个时刻系统的状态常由时间片来反映，时间片的概念通过一种称为切割(cut)的图形化方法来表示。

定义 1 切割

分布式计算的切割(cut)是一个集合 $C=\{c_1, c_2,\cdots,c_n\}$，其中，$c_i$ 是进程 P_i 中的切割事件，也就是对应于切割的一个局部状态。

从文献[5]可知，所有的一致切割集在≤关系下形成一个格。≤表示可达关系，$G\leq H$，表示 H 是一个由 G 可以到达的切割。

有限偏序执行迹作为分布式计算的一种抽象，用有向图对其进行建模。通常，关于某个规范的有限偏序执行迹的切片是一个子迹，该子迹包含了所有满足给定规范的全局状态。该方法的最大优点是，任何检验过程只需在感兴趣的子迹片段上进行就足够了，而没有切片时，不得不在全部的状态空间上进行验证。在多数情况下，子迹的规模比原始迹呈指数级地缩小了，并且对于多数常用的谓词类，都存在多项式时间的切片算法。

定义 2 计算切片

关于某个谓词的计算切片(computation slice)是包含(在原始计算中)切割都满足该谓词的最小有向图(包含最小一致切割数目)。

例如，图 1 是 1 个由 2 个进程 P_1 和 P_2 组成的分布式计算的偏序执行迹。

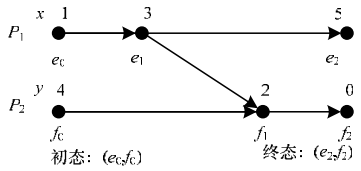
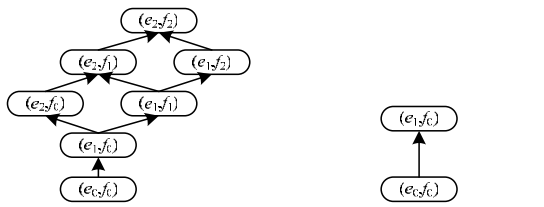


图 1 分布式计算的偏序执行迹

进程 P_1 和 P_2 分别包含私有变量 x 与 y，图中黑点表示事件，e_0,e_1,e_2,f_0,f_1,f_2 是事件标号，黑点上方的数字表示变量在该时刻对应的取值，该计算对应的一致切割构成的格即分布式计算的状态空间如图 2 所示。



(a)初态：(e_0,f_0)，终态：(e_2,f_2)　　(b)初态：(e_0,f_0)，终态：(e_1,f_0)

图 2 分布式计算及其切片的状态空间

假设，全局谓词是 $P=(x<4)\wedge(y>2)$，那么满足该谓词的子格如图 2(b)所示。若检测一个计算树逻辑(Computation Tree Logic, CTL)的正规子集[2]刻画的一个性质 $EF(P)$，即是否存在一个全局状态满足 P。没有计算切片时，不得不检测图 2(a)所有的 7 个全局状态，而引入了计算切片之后，只需要检测切片中的状态了，即只要检测图 2(b)中的 2 个状态即可。

3 同步问题

3.1 抽象接口

在进行基于 GALS 的 SoC 设计时，因为现有的 IP 核是供应商经过全面验证的，一般不宜修改。所以，接口设计者无法对接口的输入/输出信号施加任意的限制。图 3 表示了 GALS 系统中局部同步模块间简化的抽象接口的一般视图，这种简化主要根据从 IP 核可以获得的输入/输出信号的集合来进行的。

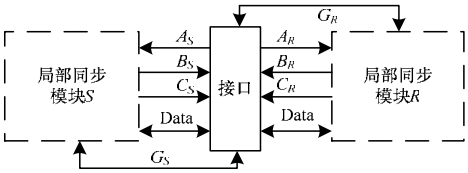


图 3 抽象接口的一般视图

假设数据通过总线传送，每个模块都有一个输出端口和一个输入端口，这里 A_S 与 A_R 分别是发送和接收模块的输入端口，B_S 与 B_R 则分别是发送和接收模块的输出端口，并且控制信号 A_S, A_R, B_S, B_R 用于通信握手协议，输入端口上的信号由模块的锁存器或触发器采样，信号 C_S 和 C_R 表示经过缓冲的时钟，信号 G_R 和 G_S 则是时钟发生器产生的输入时钟。

3.2 抽象时序图分析

人们为 GALS 系统设计多种接口模式，有些基于同步器的接口模式虽然适用性较强，但存在失效的问题。文献[6]提出了基于抽象时序图来替代基于信号转移图(Signal Transition Graph, STG)的接口同步的分析方法，在抽象时序图中，事件间因果关系及时序关系显式地表示在图上。

通常抽象时序图可以通过如下规则进行构建：

(1)对接口协议的每个感兴趣的信号分配一条水平线；

(2)沿着水平线表示时间的演进，即时间流从左至右；

(3)如果事件 x 比事件 y 早触发时间段 d，那么不论事件 x 与 y 是否为同一信号的事件，从 x 到 y 有一条标有延迟 d 的箭头；

(4)如果 2 个事件发生得足够接近，则可以认为是同时发生的。

对于所有的信号，只研究控制信号而忽略数据信号分析结果仍然是可靠的。

发送方发送消息的事件由经过缓冲的时钟活跃边沿同步触发，并且发送方和接收方时钟缓冲树分别有△S 和△R 延迟，发送方和接收方时钟发生器输入的时钟相位差△SR，接口电路在收到发送方消息发送事件后直到将该事件转发到接收方输入端口前的延迟为△*。

接收方时钟发生器产生的时钟周期为 TR，接收方输入端口上收到转发消息的事件时进行信号采样，该采样信号同接收方时钟的 2 个活跃边沿间的时间间隔分别为保持时间△H 和上升时间△SU。

发送方和接收方之间正确接口设计要求如下：

规则 1 每条从发送方的输出端口发送到接收方输入端口的消息，其信号的建立时间和保持时间约束必须满足。

规则 2 如果时钟产生信号由接口电路控制，那么信号的上升时间和下降时间及脉冲宽度约束也必须满足。

即满足如下不等式：

(1)$\triangle^* > \triangle SR + (\triangle R - \triangle S) + \triangle H$

(2)$\triangle^* < \triangle SR + (\triangle R - \triangle S) + (TR - \triangle SU)$

4 谓词检测

谓词检测过程主要通过辅助检测工具来协助完成，正确而有效地使用谓词检测工具，有 2 项任务是必须的，即设计的偏序迹表达和设计规范要求的谓词刻画。

如 POTA[3]是一个采用 Java 语言开发的内嵌计算切片功能模块的分析检测工具，图 4 是谓词检测工具的系统结构。

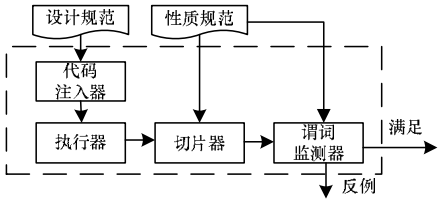


图 4　谓词检测工具的系统结构

该工具的输入是系统设计和性质刻画规范。其中，代码注入器模块主要用于在设计描述程序中适当位置插入监控代码，这些经过注入处理的代码便会在执行时输出读、写以及消息的发送、接收事件和用来确定事件间关系的向量时钟值。执行器输出事件的偏序迹和由 CTL 的正规子集描述的谓词性质一起输入到切片进行约简，约简后的迹片段再由谓词检测器检测而得到最终结果，如果性质成立则输出满足，否则给出反例。

4.1 偏序表达

对偏序执行迹作如下修改：

(1)每个感兴趣信号的行为用一个单独的信号进程来表示，并且该进程拥有一个时间变量来显示时间值。

(2)计算切片上每个事件的实际时间戳的值由每个信号根据当前的模拟运行时间步来计算得到。

(3)增加一个辅助函数：*Time t*(*Event e*)，用来返回每个事件的时间戳，该函数的输入参数是事件，返回值为该事件发生时的时间戳。改进的抽象时序图如图 5 所示。

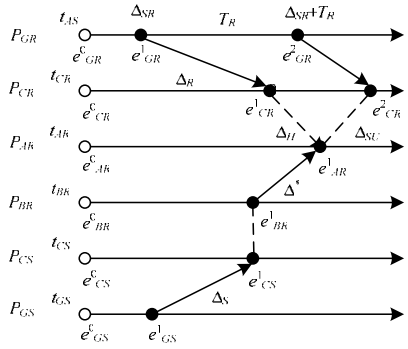


图 5　改进的抽象时序图

4.2 向量时钟算法

为了正确地表达事件间的偏序关系，通常采用向量时钟更新机制来实现。如下是修改后的算法——向量时钟与时间戳算法的伪码实现：

输入 进程 P_j 的向量时钟 $\boldsymbol{v}[1,2,\cdots,n]$

输出 更新后的进程 P_j 的向量时钟

//初始化信号进程

01. $(\forall i: i \neq j: v[i] := 0) \wedge (v[j] := 1)$;

//更新进程 P_j 的每个事件 e 的向量时钟

02. 如果 e 是相对事件，则:

03.　　v[j] := v[j]+1;

04. 如果 e 是发送事件，则:

05.　　发送 v 到进程 P_k;

06. 如果事件 e 是事件 f 的接收事件，则:

07.　for i := 1 to n do{ v[i] := max(v[i], f .v[i]);};

08. 获取事件 e 的实际时间戳 timestamp;

09. 如果 e 是相对事件，则:

10.　输出三元组(e, v, timestamp);

11.结束

这里的时钟向量采用大小为 n 的向量来表示，其中，n 为系统的进程数目，每个进程在收到一个相关事件后将自己的向量时钟值加 1，每个进程在自己发出的消息中附加一个自己向量时钟的拷贝，接收消息的进程根据消息中包含的时钟向量的值更新自己的时钟的相应值。

4.3 谓词刻画

通过以上分析可知，正确的接口设计一般要求满足设计规则 1 和规则 2，对于不同的接口模式和接口技术，首先要对各项延迟指标分类，再采用抽象时序图进行分析，第 1 类是已知的但不可控制的延迟。包括$\triangle R$, $\triangle S$, $\triangle SU$ 和$\triangle H$ 这几个信号。第 2 类是未知的延迟，如$\triangle SR$ 则属于这一类。第 3 类是可控延迟，$\triangle^*$属于这一类延迟，在可暂停时钟模式下，TR 也属于这类延迟。上述设计要求用谓词 P 表达为

$$P=(\triangle^*>\triangle SR+(\triangle R-\triangle S)+\triangle H)\wedge(\triangle^*<\triangle SR+(\triangle R-\triangle S)+(TR-\triangle SU))$$

表达式中的延迟时间通过调用辅助函数得到。即设计要求就是要检验性质 $AG(P)$在所有的路径上都满足。

5 结束语

本文介绍了如何通过谓词检测手段对同步问题进行运行时验证的基本方法，同步行为采用改进的时序图进行刻画，系统的设计需求则通过本文提供的转换规则转换成全局谓词，然后利用谓词检验工具进行验证。验证过程得益于计算切片优化技术，使得状态空间得到指数级的缩减，时空效率大大提高，并且这种技术跟经典的程序切片技术是正交的，为了使未来的验证工具验证能力更加强大，可以将 2 种切片技术有机结合，用计算切片技术快速寻找错误，然后再使用程序切片技术对程序约简后再进行分析和错误定位。

参考文献

[1] Chapiro D M. Globally-asynchronous-locally-synchronous Systems [D]. Palo Alto, CA, USA: Stanford Univ., 1984.

[2] Mittal N, Garg V K. Computation Slicing: Techniques and Theory[C]//Proc. of the Symposium on Distributed Computing. Lisbon, Portugal: [s. n.], 2001: 78-92.

[3] Sen A, Garg V K. Formal Verification of Simulation Traces Using Computation Slicing[J]. IEEE Transactions on Computers Archive, 2007, 56(4): 511-527.

[4] Weiser M. Programmers Use Slices When Debugging[J]. Communications of the ACM, 1982, 25(7): 446-452.

[5] Johnson D, Zwaenepoel W. Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing[J]. Journal of Algorithms, 1990, 11(2): 462-491.

[6] Chakraborty S, Mekie J, Sharma D K. Reasoning About Synchronization Techniques in GALS Systems: A Unified Approach[C]//Proc. of Workshop on Formal Methods in GALS Architectures. Pisa, Italy: [s. n.], 2003.