基于时间 Petri 网的嵌入式系统中断建模与验证

周宽久[1,2] 常军旺[1] 侯 刚[1] 任龙涛[1] 王小龙[1]
(大连理工大学软件学院 大连 116620)[1] (大连理工大学软件评测中心 大连 116620)[2]

摘 要 嵌入式系统为中断驱动系统,但中断触发的随机性和不确定性导致中断缺陷很难被追踪发现,并且一旦发生中断故障,往往会使整个嵌入式系统陷入崩溃。因此必须保证中断系统软件的可信性,但是目前缺乏有效的中断系统资源冲突检测方法。针对上述问题,文中首先提出了一种基于时间 Petri 网的中断系统建模方法,其能够对中断的并发性和时间序列进行有效建模。然后,为方便后续形式化验证,将时间 Petri 网模型转化为与之等价的时间自动机模型,并提出一种符号编码方法对时间自动机进行形式化编码,将系统模型与所需验证性质编码为一阶谓词逻辑公式,从而能够通过 SMT 对时间自动机的不变属性进行 BMC 验证。最后,通过 SMT 求解器 Z3 进行实验,实验结果证明了所提方法的有效性。

关键词 中断建模,有界模型检测,时间自动机,可满足性模理论,时间 Petri 网
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.09.038

Interrupt Modeling and Verification for Embedded Systems Based on Time Petri Nets

ZHOU Kuan-jiu[1,2] CHANG Jun-wang[1] HOU Gang[1] REN Long-tao[1] WANG Xiao-long[1]
(School of Software, Dalian University of Technology, Dalian 116620, China)[1]
(Software Evaluation & Test Center, Dalian University of Technology, Dalian 116620, China)[2]

Abstract The embedded systems are interrupt-driven systems, but the triggered methods of interrupts are with randomness and uncertainty. The behavior of interrupt can be quite difficult to fully understand, and many catastrophic system failures are caused by unexpected behaviors. Therefore, interrupt-driven systems need high quality tests, but there is a lack of effective interrupt system detection methods at present. In this paper, firstly a modeling method of interrupt system was proposed based on time Petri nets, which has ability of describing concurrency and time series. Then the time Petri nets were transformed into timed automata for model checking. Consequentially, a symbolic encoding approach was investigated for formalized timed automata, through which the timed automata could be bounded model checked (BMC) with regard to invariant properties by using Satisfiability Modulo Theories (SMT) solving technique. Finally, Z3 was used in the experiments to evaluate the effectiveness of our approach.

Keywords Interrupt modeling, Bounded model checking, Timed automata, Satisfiability modulo theories, Time petri nets

1 引言

中断在嵌入式系统开发过程中起着不可或缺的作用,中断的运用使嵌入式系统中的突发事件处理、实时处理、同步处理等变得更加高效。中断触发的随机性和不确定性给测试带来了很大难度,使得这些中断错误很难通过软件动态测试或静态代码分析检查出来。同时,中断和硬件、环境以及应用密切相关,使得中断软件设计要比顺序软件困难得多,且在实现过程中很容易引入诸如开关中断时机错误、现场保护错误、中断嵌套错误等软件错误。一旦中断造成了程序错误,就会导致一系列不易追踪的严重软件故障,这些故障对系统来说常常是致命的。因此,本文考虑在软件设计阶段对嵌入式系统的中断行为进行形式化建模,并通过数学方法严格推导中断运行流程,有效解决上述问题。

目前,嵌入式系统的形式化建模方法主要有 FSM 和 Petri 网两种,半形式化方法有 UML。但 FSM 不能描述中断并发行为,UML 模型后期又难以验证,因此,两者都有一定局限性。相比而言,Petri 网是一种比较完善的形式化建模方法,具有精确的形式化定义、严格规范的推导方法、较好的工具支持,特别适合描述系统的控制流、并发特性和异步行为。近年来,学术界提出了许多将 Petri 网应用于嵌入式系统建模与验证的方法。Rammig 等人[1]提出了一种可以动态修改的嵌入式实时系统建模方法。Gu 等人[2]提供了一种基于时间 Petri 网的实时系统建模与分析的集成方法。Costa 等人[3]给

到稿日期:2013-10-12 返修日期:2014-03-21 本文受国家自然科学基金:航天多核嵌入式软件可信性验证与系统原型(61272174)资助。
周宽久(1966—),男,博士,教授,主要研究领域为嵌入式系统建模、可信软件;**常军旺**(1989—),男,硕士生,主要研究领域为软件可靠性、形式化验证;**侯 刚**(1982—),男,博士生,讲师,主要研究领域为复杂性理论、模型检测,E-mail:hg.dut@163.com(通信作者);**任龙涛**(1988—),男,硕士生,主要研究领域为软件可靠性、形式化验证;**王小龙**(1989—),男,主要研究领域为软件可靠性、模型验证。

出了一种基于 Petri 网的嵌入式系统协同设计方法。Zhang 等人[4]提出了一种基于 Petri 网的分布式实时嵌入式系统调度模型。

本文提出了一种基于时间 Petri 网的嵌入式系统中断资源冲突检测方法。首先，本文提出了一种基于时间 Petri 网的中断系统建模方法，其能够对中断的并发性和时间序列进行有效建模。然后，为方便后续形式化验证，将时间 Petri 网模型转化为与之等价的时间自动机模型，并提出一种符号编码方法对时间自动机进行形式化编码，将系统模型与需要验证的性质编码为一阶谓词逻辑公式，从而能够通过 SMT 对时间自动机的不变属性进行 BMC 验证。最后，通过 SMT 求解器 Z3 进行上述方法的实验，实验结果证明了本文提出方法的有效性。

2 中断的时间 Petri 网建模

2.1 中断行为分析

中断是指当出现需要时，CPU 暂时停止当前程序的执行转而执行处理新情况的程序和执行过程（interrupt service routine，ISR）。即在程序运行过程中，系统出现了一个必须由 CPU 立即处理的情况，此时，CPU 暂时中止程序的执行转而处理这个新的情况的过程就叫做中断。有 3 种方式可以触发中断：(1)硬件中断，一些外部硬件通过在中断请求线上改变电压水平触发中断；(2)软件中断，正在执行的程序通过执行一条特殊的指令或写入中断相关的内存映射寄存器触发中断；(3)异常，当内部硬件检测到故障发生时触发中断，如段错误。

以上 3 种类型的中断触发模式是不同的，但是中断响应过程基本相同。一个完整的中断处理过程应该包括：中断请求、中断仲裁、中断响应、中断服务和中断返回。采用时间 Petri 网建模的过程中，上述步骤均需要进行处理。

中断请求：当中断请求发生时，由中断源向 CPU 发出中断请求信号。外部设备发出中断请求信号要具备以下两个条件：1)外部设备的工作已经告一段落；2)系统允许该外设发出中断请求。如果系统不允许该外设发出中断请求，那么可以将这个外设的请求屏蔽。因此，Petri 网建模过程中需要考虑与中断控制相关的寄存器状态。

中断仲裁：中断请求具有随机性，有时多个中断源会同时提出中断申请。但 CPU 每次只能响应一断源请求，这就必须根据各中断源工作性质的轻重缓急，预先安排一个优先级顺序。当多个中断源同时申请中断时，即按此优先级顺序进行响应，CPU 在处理完高优先级中断后，再响应低优先级中断申请。因此，在建模时需要在时间 Petri 网的基础上引入优先级，实现对并发状态的按优先级响应。

中断响应：当 CPU 发现有中断请求时，需要中止现行程序，保存程序断点，并自动执行中断处理程序。中断响应是解决中断发现和接收的过程，是由硬件装置自动完成的。因此，在 Petri 网建模时，此过程不需要外加控制信号参与建模。

中断服务：中断服务是由中断服务例程 ISR 完成的，是中断事件的核心内容。中断服务例程一般顺序执行，因此，可采用顺序的 Petri 网模型进行建模。但在中断服务例程运行中，有可能出现比现行中断优先级更高的中断请求，则 CPU 需要先停止现行的低优先级中断处理，转去响应高优先级中断，当高优先级中断处理完成后再继续处理低优先级中断，这种情况称为中断嵌套。因此，我们需要在 Petri 网建模过程中引入层次化的表示结构实现对中断嵌套问题的建模。

中断返回：对于硬件中断和软件中断，一旦中断服务例程完成，被中断的程序就会从中断点恢复重新执行。对于异常触发的中断，当中断服务例程完成时，触发异常的程序不能正常恢复，但程序计数器会被设定得到某个固定的地址位置，从而引导程序重新正确执行。因此，在中断的时间 Petri 网建模中，我们需要设计不同的中断处理返回状态。

2.2 带有优先级的时间 Petri 网

P. M. 给出了时间 Petri 网的基本概念和定义详细说明[5]，这里仅引入与本文研究内容相关的一些定义。

定义 1 一个时间 Petri 网（TPN）是一个元组 $\langle P, T, Pre, Post, M_0, efd, lfd\rangle$，其中：$P=\{p_1, p_2, \cdots, p_m\}$ 是一个有限的非空库所集合；$T=\{t_1, t_2, \cdots, t_n\}$ 是一个有限的非空变迁集合；$Pre \subseteq P\times T$ 是一个有限的非空输入弧集合，它定义了库所和变迁之间的流关系；$Post \subseteq T\times P$ 是一个有限的非空输出弧集合，它定义了变迁和库所之间的流关系；M_0 是 Petri 网的初始标识；efd 为允许变迁触发的最早时间，lfd 是允许变迁触发的最迟时间。

定义 2 一个带有优先级的时间 Petri 网（PrTPN）是一个元组 $\langle P, T, Pre, Post, M_0, efd, lfd, Pr, \Sigma^{\epsilon}, L\rangle$，其中：$\langle P, T, Pre, Post, M_0, efd, lfd\rangle$ 是一个原始时间 Petri 网；$Pr \subseteq T\times T$ 是优先级关系，具有非自反性、非对称性和可传递性；Σ 是动作或标记的有限集合，但不包括 $\in$；$L: T\to\Sigma^{\epsilon}$ 是标记函数的一个函数调用。

对于 $f, g: P\to N^+$，$f \geqq g$ 表示对于每一个 p，$(\forall p\in P)(f(p)\geqq g(p))$ 和 $f\{+|-\}g$ 对应于 $f(p)\{+|-\}g(p)$。一个标记是一个函数 $m: P\to N^+$，$t\in T$ 使能如果 m iff $m\geqq efd(t)$，$\epsilon_N(m)$ 表示 Petri 网 N 在 m 情况下使能的迁移集合。$(t_1, t_2)\in Pr$ 可被写为 $t_1 > t_2$ 或 $t_2 < t_1$（t_1 的优先级高于 t_2）。

定义 3 标识 M 是 Petri 网中托肯分布的集合。对于 $p\in P$，其标识为 $M(p)$，表示为多重集合。对于一个特定的标识 M，一个库所 p 称为被标识的，当且仅当 $M(p)\neq\phi$。

2.3 中断建模

根据上述嵌入式系统的中断行为分析，我们设计出 3 种类型的时间 Petri 网场景对中断进行建模。

(1)中断顺序执行：在某一时刻只有一个中断请求发生，且在中断响应过程中没有高优先级中断请求。在这种情况下，中断服务例程顺序执行，因此，可采用顺序时间 Petri 网模型对其进行建模（见图 1）。IRQ 是中断请求状态，IRP 是中断返回状态，剩下的库所表示寄存器、全局变量、共享资源等。

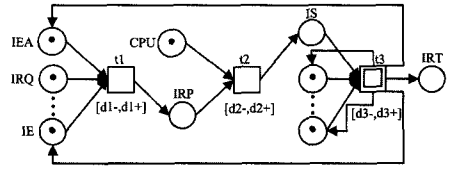

图 1 中断顺序执行的时间 Petri 网模型

(2)中断嵌套执行：在某一时刻只有一个中断请求发生，且在中断响应过程中有高优先级中断请求发生。在这种情况下，当前正在执行的中断被打断，转去执行高优先级中断的请求；当高优先级中断执行完成后，返回被中断的低优先级中断

继续执行。因此，可采用层次化的时间 Petri 网模型进行优先级中断(见图 2)。

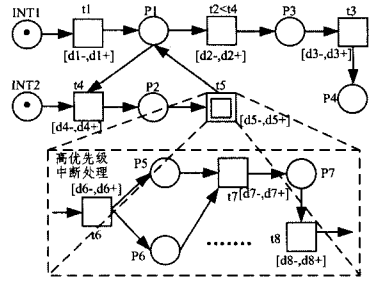

图 2　中断嵌套执行的层次化时间 Petri 网模型

(3)中断并发请求：在某一时刻多个中断请求同时发生。在这种情况下，需要根据中断优先级对中断请求进行仲裁，响应高优先级中断，屏蔽低优先级中断(或者将低优先级中断加入到中断请求队列，待高优先级中断执行完毕后再进行处理)。针对这种场景，可采用并发的时间 Petri 网模型进行建模(见图 3)。

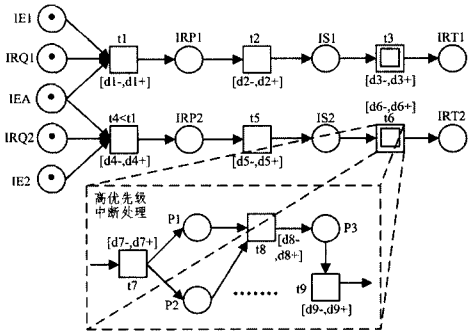

图 3　中断并发请求的时间 Petri 网模型

3　时间 Petri 网转化为时间自动机

目前已经提出了很多解决从时间 Petri 网转化为时间自动机的方法。Cassez 等人[6]提出了一种从时间 Petri 网到时间自动机的结构化转化方法，该转化方法为时间 Petri 网的每个迁移建立一个对应的时间自动机。Lime 等人[7]运用时间状态空间的搜索性，提出了一种基于扩展的状态类图(SCG)去计算所谓的状态类时间自动机(SCTA)。为了分析和验证时间 Petri 网模型的不变属性，Guoyin 等人[8]给出了一种从时间 Petri 网到时间自动机的转化方法，并进行了形式化验证。但是上述方法缺少时间 Petri 网优先级的描述及其转化方法。基于上述方法，我们对时间 Petri 网进行优先级扩展并提出了一种新的转化方法来实现优先级到时间自动机的转化。

定义 4　时间自动机[9]是一种增加了时钟变量的特殊的有限状态自动机，其中，时钟变量可用实数域变量进行表示。时间自动机用一个六元组模型$\langle S, S_0, \Sigma, X, I, T\rangle$表示，其中：$S$ 是时间自动机的有限状态集合；$S_0 \in S$ 是时间自动机的初始位置集合；Σ 是时间自动机的有限字符表；X 是时间自动机的时钟集合；$I: S \rightarrow C(X)$是位置到位置不变量的映射，$C(X)$代表位置上的变量约束；T 是时间自动机的状态迁移集合。

优先级：我们只考虑静态优先级，时间自动机的定义通过迁移的局部反自反、非对称和可传递的优先级关系进行扩展。该语义遵守如下规则：一个迁移能够发生当且仅当该时刻没有更高优先级的迁移发生。

带优先级的时间 Petri 网到时间自动机的具体转化过程包括如下步骤：

步骤 1　定义标号集 Σ，Σ 是时间 Petri 网中所有变迁的集合。

步骤 2　定义变量集 V，V 包含时间 Petri 网中所有变量的集合。

步骤 3　定义全局时钟 c，满足 $c \in X$。对于评估系统性能和统计系统运行过程中消耗的时间，全局时钟发挥着重要作用。

步骤 4　为了对系统状态的局部变迁进行计时，并且保证变迁在模型定义的最早和最迟触发时间间隔$[efd, lfd]$内发生，定义局部时钟 x，满足 $x \in X$。

步骤 5　每一个库所对应于位置集合中的一个元素，根据 Petri 网中库所结合 P 定义位置结合 L。

步骤 6　模型中每一个变迁用 t 表示，时间自动机边的数量由 t 的输出有向弧的数量决定，所有边具有相同的时间约束、变量约束和响应行为映射。可以根据变迁集合 T 和节点函数 N 定义边的集合 E。

步骤 7　如果时间 Petri 网的边存在时间或者变量约束，在转化的过程中要定义相应的守卫函数 G 进行表示，并在转化的时间自动机边上添加相应的 G。

步骤 8　如果边集合 E 发生变化，则定义对应边到响应行为映射 $A(e)$，如果全局时钟 c 或局部时钟 x 发生变化，则定义输入库所$(\forall p_i \in {}^{\cdot}t)$对应位置的不变函数 $I(l_i)$。

步骤 9　如果在原时间 Petri 网中库所 p_i 在初始标记 M_0 下处于使能状态，则在转化的时间自动机中定义相应位置为初始位置 l_i，所有初始位置 l_i 构成初始位置集合 L_0。

步骤 10：如果对于时间 Petri 网中任意两个变迁 t_i 和 t_j 存在优先级语义 $t_i > t_j$ 或 $t_j < t_i$(即 t_i 的优先级高于 t_j)，其中 t_i 和 t_j 对应的迁移约束条件分别是 g_i 和 g_j，时间自动机 $\overline{t_j}$ 依据上述步骤方法添加上对应的约束条件 $g_j \wedge \neg g_i$。

4　时间自动机的模型检测

在模型验证领域已有许多符号编码方法被提出。最初的想法是把一个状态转移系统转换成无量词逻辑公式并采用 SAT 求解技术[10]对公式进行有界模型检测，该方法由 Biere 等人[11]提出。一个最近的研究[12]采用类似的方法来分析和验证一个状态转移系统，但基于 SMT 求解技术。为了避免在有界模型检测过程中对变量进行布尔编码以及对时间自动机模型中的时钟进行预处理，Xiaoliang[13]给出一个利用 SMT 工具对时间自动机进行有界模型检测的方法。Weiqiang 等人[14]研究了一种专门针对 STM(State Transition Matrix)模型设计的符号编码方法，并采用 SMT 求解器 Yices 对模型的不变属性进行了有界模型检测。基于上述方法，我们提出一种带有优先级的时间自动机编码方法。

通过下述描述来展示我们的符号编码方法，对于一个时间自动机模型 M，它由 n 个时间自动机 $A_0, A_1, \cdots, A_n$ 组成，并给出有界模型检测的边界值 K。除此之外，基于 SMT 模型验证理论，我们通过可达性分析算法对实时中断系统进行

安全属性和活性属性的验证。我们采用线性时序逻辑 LTL (Linear Temporal Logic)来描述时间自动机的不变属性，例如 $f=EFp|AGp$。然后，我们将时间自动机模型和要验证的属性转化为一个无量词的合取范式(用 $BMC(M,f,K)$ 表示)，并用 SMT 求解技术对其进行有界模型检测。最后，我们在第 5 节采用 SMT 求解器 Z3 进行实验并验证我们提出方法的有效性。

4.1 时间自动机模型 M 的编码方法

初始状态(s_0,v_0)可用 Init 表示，则时间自动机的全局初始状态可写为：

$$Init=Init_0\wedge Init_1\wedge\cdots\wedge Init_{n-1}$$

$Init_j$ 表示第 j 个时间自动机的初始位置，并且该时间自动机的所有时钟变量的初始值都被设置为 0，具体的公式表示如下：

$$Init_j=\bigvee_{s\in S_0^j}(s_0=s)\wedge\bigvee_{t\in X^j}(t_0=0)$$

$Const_k(j)$表示第 j 个时间自动机在第 k 个状态的变量约束，其定义如下：

$$Const_k(j)=\bigvee_{t\in X^j}(t_0\geqslant 0)\wedge\bigvee_{s_\mu\in S^j}\{(s_k=s_\mu)\rightarrow\bigvee_{s_v\in S^j\wedge s_v\neq S_\mu}((s_k\neq s_v)\wedge I^k(s_\mu))\}$$

其中，t_k 表示系统在第 k 个状态时钟变量的值，为了保证时钟变量为非负值，用 $t_k\geqslant 0$ 对时钟变量进行约束。sk 表示系统处于第 k 个状态的位置，公式的其他部分用于保证每个时间自动机在某一时刻只能处于一个位置以及其不变属性 $I_k(s\mu)$ 的成立。

基于上述 $Const_k(j)$编码，可得到全局状态(s_k,v_k)的不变约束，如下所示：

$$Const_k=Const_k(0)\wedge\cdots\wedge Const_k(n-1),k\in[0,K]$$

在时间自动机运行的过程中，会发生两种类型的迁移。其中一个是时间延迟迁移，因为时间一直在改变，一旦时间变量发生变化，整个时间自动机模型就会从一个状态迁移到另一个状态。所以定义系统时间延迟迁移如下：

$$T_\mu^j(i)=\neg\alpha_\mu^k\wedge(s_k=s_\mu)\wedge(s_{k+1}=s_\mu)\wedge\bigwedge_{t\in X^j}(t_{k+1}=t_k+d_k)$$

时间自动机是针对整个实时中断系统的建模，包括其所有的动态行为。当时间自动机执行一些动作序列之后，时间自动机的状态就会发生迁移。所以另一个类型的迁移是动作迁移，定义如下：

$$T_\mu^j=\alpha_\mu^k\wedge(s_k=s_\mu)\wedge(s_{k+1}=s_\mu')\wedge\bigwedge_{t\in\lambda_u^j}(t_{k+1}=0)\wedge\bigwedge_{t\notin\lambda_u^j}(t_{k+1}=t_k+d_k)$$

由于时间自动机设计模型 M 由 n 个时间自动机组成，当多个自动机进行迁移时，会发生互斥现象，并且，当一个时间自动机运行时，每个时间自动机每次只能迁移一步。为保证时间自动机上述性质的正确性，对于 $T_v=\langle s_v,\alpha_v,\sigma_v,\lambda_v,s_v'\rangle$，定义如下约束：

$$Exl_\mu^j(k)=\bigwedge_{T_v\in T\wedge T_\mu\neq T_v}\neg\alpha_v^k$$

在我们提出的符号编码方法中，我们使用 BMC 技术来缓解状态空间保证稳定，定义的边界值为 K。$Step_k(j)$表示第 j 个时间自动机的第 k 步迁移，其中 d_k 是时钟延迟。整个迁移过程定义如下：

$$Step_k(j)=\bigvee_{T_\mu\in T^k}(Exl_\mu^j(k)\wedge T_\mu^j(k)\wedge T_\mu^j\wedge d_k>0)$$

时间自动机的 k 部路径是一个有限的状态序列，可用一种更加清晰的形式来表示该过程：

$$(s_0,v_0)\xrightarrow{d_0}(s_1,v_1)\xrightarrow{d_1}\cdots\xrightarrow{d_{K-1}}(s_K,v_K)$$

其中，(s_0,v_0)是路径的初始状态，(s_k,v_k)表示经过时间 $d_0+d_1+\cdots+d_k$ 到达的第 k 个状态。$Step_k$ 表示系统的第 k 步迁移，由每个时间自动机第 k 步迁移 $Step_k(j)$的合取组成。每一步迁移用一个五元组 $T_\mu=\langle s_\mu,\alpha_\mu,\sigma_\mu,\lambda_\mu,s_\mu'\rangle$表示。在 k 步迁移过程中，$Step_k(s_k,v_k)\xrightarrow{d_k}(s_{k+1},v_{k+1})$定义如下：

$$Step_k=Step_k(j)\wedge\cdots\wedge Step_k(n-1),k\in[0,K-1]$$

基于 $Step_k$，可得到时间自动机的第 k 步路径公式，定义如下：

$$Path_k=\bigwedge_{k=0}^{K}Step_k$$

最后把整个时间自动机模型转化为无量词的一阶逻辑公式，如下：

$$[[M]]_k=Init\wedge\bigwedge_{k=0}^{K}Const_k\wedge Path_k$$

4.2 不变属性 f 的编码方法

时间自动机模型 M 的所有可达全局状态用 RM 表示，其不变属性是约束 M 所有可达状态的一个状态谓词 p。例如，对于可达性属性，可以用 LTL 语义进行描述，即 $f=EFp$，其表示在自动机运行的过程中，存在一条路径，并且在这条路径上存在一个状态使性质 f 成立。基于上述方法，可以得到有界模型检测公式$[[f]]_k=\bigvee_{k\in[0,K]}[[p]]_k$，其中$[[p]]_k$ 由一般的一阶逻辑公式组成。

安全性 $f=AGp$ 与可达性相反，其表达的性质为在时间自动机状态迁移的过程中，p 在所有路径上的所有状态都成立。利用这种性质可以对 f 取反，得到其对应的可达性。对此性质 $f'=EF\neg p$ 进行验证，如果性质 f'成立，则说明在运行路径中存在一条路径，这条路径上的一个状态满足性质 $\neg p$，这意味着性质 f 不成立。相应地，如果性质 f'不成立，则说明自动机运行 k 步之内，性质 f 成立。

当完成 M 和 f 的符号编码后，可得到用于有界模型检测的整个逻辑公式定义如下：

$$BMC(M,f,K)=(\bigwedge_{k=0}^{K}[[M]]_k)\wedge(\bigwedge_{k=0}^{K}[[\neg f]]_k)$$

5 实现和实验

SMT[15]工具和 SAT 工具可被用于验证一个问题是否满足一个命题逻辑公式，两者之间存在一定的差异。SAT[16]工具用于处理仅包含布尔变量的命题逻辑，而 SMT 可解决更加广泛的命题逻辑问题，包括整数、实数、数组、未解释函数等一系列问题。SMT 使有界模型检测变得更加方便，因为不再需要将所有的系统变量全部转化为布尔变量去验证，仅仅需要把命题逻辑公式用特定语言直接进行描述，然后输入 SMT 工具便可直接求解。本文选用 Z3 作为模型检测过程中的实验工具。

嵌入式系统的时间自动机模型用 M 表示，系统所需验证的属性用 f 表示，有界模型检测的最大步数用 K 表示，这些都是模型检测过程中的输入。对于 $k<K$，我们可以得到系统

模型对应的逻辑公式 $BMC(M,f,K)$，然后通过 SMT 求解器 Z3 进行求解。

ARM CortexTM-M3 是一个 32 位微控制器，应用于工业自动化和其他应用领域。它带有一个称为 SysTick 的系统计时器。计时器可用于触发 ISR，每毫秒执行一次。图 4 为基于该处理器平台的一个中断设计实例(只给出关键代码)。其中，主函数 main 用于完成系统正常的工作业务，ISR1 实现从某个初始值开始每毫秒倒计时，直到计数器达到 0，然后停止倒计时；ISR2 是一个外部的按键中断服务例程，可以实现计时器的初始值复位。外部中断的优先级高于定时器中断，通过上述两个中断来实现中断嵌套。

基于本文提出的时间 Petri 网建模方法(详细内容见第 2 节)，对上述实例进行建模，建模结果如图 5 所示。

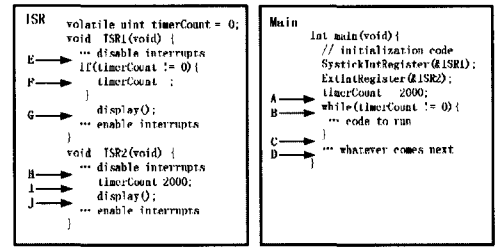

图 4　中断系统设计实例

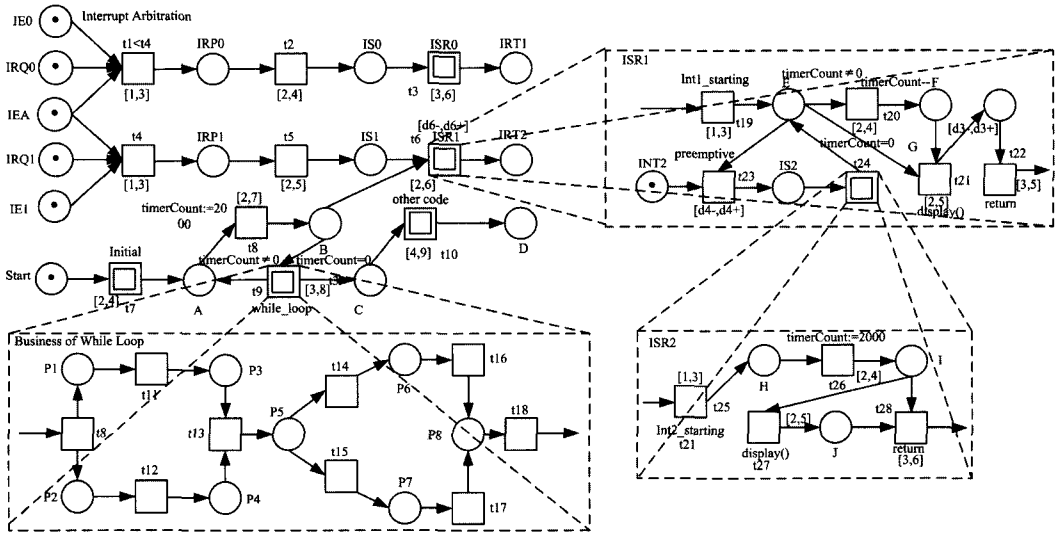

图 5　中断实例的时间 Petri 网模型

如何将时间 Petri 网模型转化为与之等价的时间自动机模型在第 3 节进行了详细描述。由于篇幅限制，我们仅仅给出时间自动机模型的主体转化结果，如图 6 所示。

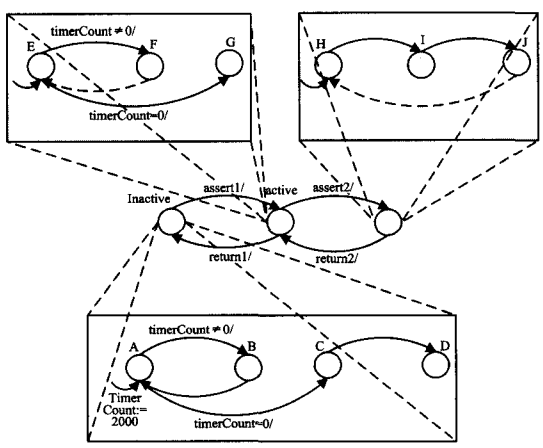

图 6　设计实例的时间自动机转化

采用 3 种类型的不变属性来验证设计实例的可满足性，在工业领域中，时间自动机设计通常都是要求满足这些不变属性。

可达性：我们想要知道对于某个时间自动机模型，是否存在一条有限的路径能够从 Sa 到达 Sa'，例如：

$$EPath_A(s_a^A,v_a^A)\rightarrow\cdots\rightarrow(s_{a'}^A,v_{a'}^A)$$

上述实例两个具体的可达性属性描述如下：

$$reachability1=A\rightarrow\cdots\rightarrow C,reachability2=E\rightarrow\cdots\rightarrow G$$

静态属性：是指不同的时间自动机模型之间存在某种相关性，这种相关性可以表述如下：如果时间自动机 A 处于状态 S_a，则可以推断出时间自动机 B 一定处于状态 S_b，例如：

$$\forall(s,v)\in R_M,(s_a^A,v_a^A)\Rightarrow(s_a^B,v_a^B)$$

当一个系统设计包含大量的时间自动机时，设计者很难把握整个设计的全貌，从而无法保证系统静态属性的可满足性，动态属性也是如此。

上述实例两个具体的静态属性描述如下：

$$static1=F\Rightarrow B,static2=I\Rightarrow B$$

动态属性：是指当一个时间自动机状态发生变化时，另一个时间自动机处于某个特定状态的不同时间自动机之间的关联。具体可描述如下：当时间自动机 A 从状态 S_a 转变为 $S_{a'}$ 时，时间自动机 B 一定处于状态 S_b，例如：

$$\forall(s,v)\in R_M,\{(s_a^A,v_a^A)\xrightarrow{d}(s_{a'}^A,v_{a'}^A))\}\Rightarrow(s_a^B,v_a^B)$$

上述实例一个个具体的动态属性描述如下：

$$dynamic=(E\xrightarrow{d}F)\Rightarrow B$$

接下来，我们通过第 4 节提出的符号编码方法对上述实例进行编码，并把生成的 BMC 逻辑公式转化为 Z3 求解器的输入语言 SMT-LIB。然后，使用 Z3 4.3.0(运行环境为 Windows 7，2.8GHz，4GB RAM)对上述实例属性进行检测。在验证的过程中我们故意引入一些逻辑错误来检验我们提出方法的正确性。表 1 是系统含有逻辑错误的实验结果，表 2 是

(下转第 219 页)

COMM 2009 conference on Data communication. 2009:51-62
[13] Garg S K, Ya R B. Network CloudSim: Modelling Parallel Application in Cloud Simulations [C]// Fourth IEEE International Conference on Utility and Cloud Computing. 2011:105-113
[14] Buyya R, Ranjan R, Calheiros R N. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities[C]// International Conference on High Performance Computing & Simulation. 2009:1-11

(上接第 209 页)

正确系统的实验结果。

表 1 系统含有逻辑错误的实验结果

属性	步数	结果	时间(秒)
reachability1	22	unsat	2.23
reachability1	23	sat	2.31
reachability2	20	unsat	2.12
reachability2	21	sat	2.21
static1	24	unsat	2.45
static1	25	sat	2.48
static2	25	unsat	2.66
static2	26	sat	2.74
dynamic	23	unsat	2.34
dynamic	24	sat	2.47

表 2 系统错误改正实验结果

属性	步数	结果	时间(秒)
reachability1	50	unsat	4.45
reachability2	50	unsat	4.52
static1	50	unsat	4.68
static2	50	unsat	4.49
dynamic	50	unsat	4.78

如果实验结果输出为“unsat”，则增加有界模型检测的步数 k，直到输出结果为“sat”或达到预先设定的检测步数最大值 K。如果满足，说明在 k 步之内找到了一个反例，该公式模型(包括对变量的解释)便检测出系统设计的漏洞或错误。模型检测工具可以返回一个反例，通过对反例的解读可以得到性质不成立的原因，为系统的修正提供重要线索。反之，如果 SMT 实例不可满足，则表明受验证的系统或模型运行到 k 阶段时是安全的、没有错误的。

结束语 本文提出了一种基于时间 Petri 网的嵌入式系统中断行为建模方法，该方法能够对中断嵌套和并发行为进行有效描述。此外，为了通过模型检测技术对中断模型进行验证，提出了一种从时间 Petri 网模型到时间自动机模型的转化方法。在模型检测过程中，提出了一种把时间自动机转化为一阶逻辑公式的符号编码方法，最后通过 SMT 模型检测方法对其进行有界模型检测。根据不同的建模粒度，验证的属性是不同的。

本文的研究仍然存在一些细节工作有待提高，比如在模型编码和验证的过程中，我们转化的 BMC 公式并不是最优的，同时，我们也没有为时间 Petri 网模型到时间自动机模型的转化提供严格的数学证明，在未来的工作中我们会进行更加深入的研究。

参 考 文 献

[1] Rammig F, Rust C. Modeling of dynamically modifiable embedded real-time systems [C] // The Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2003(WORDS 2003 Fall). IEEE, 2003:28-28
[2] Gu Z, Shin K G. An integrated approach to modeling and analysis of embedded real-time systems based on timed petri nets[C]// Proceedings 23rd International Conference on Distributed Computing Systems, 2003. IEEE, 2003:350-359
[3] Costa A, Gomes L. Petri net splitting operation within embedded systems co-design[C]// 2007 5th IEEE International Conference on Industrial Informatics. IEEE, 2007, 1:503-508
[4] Zhang H, Ai Y. Schedule modeling based on Petri nets for distributedreal-time embedded systems [J]. Jisuanji Gongcheng/Computer Engineering, 2006, 32(18):6-8
[5] Merlin P M. A study of the recoverability of computing systems [D]. University of California, Irvine, 1974
[6] Cassez F, Roux O H. Structural translation from time Petri nets to timed automata[J]. Journal of Systems and Software, 2006, 79(10):1456-1468
[7] Lime D, Roux O H. Model checking of time Petri nets using the state class timed automaton[J]. Discrete Event Dynamic Systems, 2006, 16(2):179-205
[8] Guoyin Z, Ming L, Aihong Y, et al. Methodology of modeling and formal verification based on extended Petri net[J]. Applicaiton Research of Computers, 2010, 27
[9] Alur R. Timed automata [C] // Computer Aided Verification. Springer Berlin Heidelberg, 1999:8-22
[10] Barrett C, Sebastiani R, Seshia S, et al. Handbook of Satisfiability [M]. Fairfax: IOS Press, 2009
[11] Biere A, Cimatti A, Clarke E, et al. Symbolic model checking without BDDs[M]. Springer Berlin Heidelberg, 1999
[12] Veanes M, Bjørner N, Raschke A. An SMT approach to bounded reachability analysis of model programs [M] // Formal Techniques for Networked and Distributed Systems-FORTE 2008. Springer Berlin Heidelberg, 2008:53-68
[13] Wang Xiao-liang. Bounded model checking of timed automata based on Yices[J]. Computer Engineering and Design, 2010, 31(1):126-129
[14] Weiqiang K, Shiraishi T, Katahira N, et al. An SMT-Based Approach to Bounded Model Checking of Designs in State Transition Matrix[J]. IEICE transactions on information and systems, 2011, 94(5):946-957
[15] Barrett C, De Moura L, Stump A. Design and results of the first satisfiability modulo theories competition (SMT-COMP 2005) [J]. Journal of Automated Reasoning, 2005, 35(4):373-390
[16] Le Berre D, Simon L. The essentials of the SAT 2003 competition[C] // Theory and Applications of Satisfiability Testing. Springer Berlin Heidelberg, 2004:452-467