基于 Petri 网结构的多处理机实时操作系统*

任爱华，杜悦冬

(北京航空航天大学 计算机科学与工程系，北京 100083)

E-mail: renah@buaa.edu.cn

http: www.buaa.edu.cn

摘要：多处理机环境下的实时系统具有并发事件驱动性质，其软件结构展现了多重同步点以及生产者与消费者之间的关系，这导致了复杂的控制结构.对于此类系统软件的开发缺少标准的方法和工具，造成了软件低效、程序结构不清晰、开发成本高、维护困难的现象的出现.根据 Petri 网易于描述并行/并发现象的特点，采用它来解决多处理机软件的描述问题，介绍了一种以 Petri 网图形方式在多处理机系统环境下进行程序设计的方法.该方法基于两种程序设计级别：任务级和作业级.前者负责描述基本操作，由单一控制线程完成；后者用于并行/并发程序建模，由整个多处理机系统来执行.在作业级程序设计中，用户采用面向对象 Petri 网来描述并行程序结构，以建立系统模型.该方法以一种接近于程序员的思维方式去设计并发软件.提供了一种可靠的并行结构的程序.阐述了支持此种程序设计方法的操作系统结构及其实现原理.

关键词：多处理机操作系统；Petri 网；面向对象技术；并发系统建模

中图法分类号：TP316 **文献标识码**：A

实时概念已十分广泛地应用于许多软件系统，所涉及的范围从具有微秒响应时间的嵌入式控制器，到具有秒甚至分响应时间的庞大系统.然而，在这些系统中，的确显露出一些共性，即实时系统是一个与其环境之间维持着及时交互关系的软件系统[1].

实时系统的用户从多处理机系统中获得的许多优势是单处理机系统所无法给予的，但也使实时系统的设计者们面临着一些特殊问题.尤其是在一些视觉设备或复杂仪表设备系统的集成应用中，对数据采集和分析以及数据通信方面的性能要求非常高，且与环境有很强的交互作用，经常是通过一个复杂的专用外围环境来取得这种交互.由于要把不同仪器中的处理机以及系统计算资源连接起来，因而导致了多处理机结构，而这些仪器通常又是不同的商业产品，所以形成了非对称多处理机结构.由于处理此类信息的性质以及紧迫的实时需求，在系统构件之间的通信中需要提供相当宽的带宽.

从软件角度来看，这些系统具有并发事件驱动性质，其软件结构展现了多重同步点以及生产者与消费者之间的关系，控制结构复杂，常常提出实时要求.此外，这是一种多学科软件，在应用开发的不同阶段通常由不同研究小组开发.

随着微处理机技术的不断发展，这些要求在硬件实现上渐趋简单，且功能更强.然而，随着应用的不断深入，所涉及的领域也愈加复杂，微处理机性能的改进速度虽然很快，但仍不能满足应用的复杂程度的需求，有时必须使用多处理机结构，利用一个或一组处理机来控制每一个子系统，这样可以获得较强的模块化能力和高度免除故障能力以及较强的计算能力.当应用中同时要求分散结

* 收稿日期：2000-01-07；修改日期：2000-03-23

基金项目：国家自然科学基金资助项目(69883002)

作者简介：任爱华(1957－)，女，辽宁盖县人，博士，副教授，主要研究领域为软件开发环境，分布式系统，并行计算，Petri 网应用；杜悦冬(1977－)，男，河北阳原人，硕士，主要研究领域为分布式系统，Petri 网应用.

构以及高速数据传输时,分布式多机配置可能效率不高,这时就需要采用紧密耦合多处理机结构.

目前,多处理机结构可用标准部件来搭建,也能获得所要求的性能,然而这些系统的软件开发十分复杂,常常形成开发的危机阶段[2~4].这是由于在这些系统的软件开发中缺少标准的方法和工具,通常要为每个应用提供专门的解决办法,从而导致了不规范的软件,其结构不清晰,这使得软件的维护既困难又昂贵.

以往对这类软件的开发经常采用的策略是在每个处理机上运行各自独立的管理程序,通过高级监控程序来管理共享资源,以便实现远程任务之间的通信.尽管这种解决办法是基于程序员所熟悉的软件技术,但软件并非很有效,因为每台处理机各自独立的程序设计不能提供具有可靠并行结构的程序.

如果利用形式化方法辅助开发多处理机系统的并发软件,在程序结构上则会更加可靠,从而可以提高软件性能,使软件易于维护,并能较好地满足用户需求.从可视化角度出发,考虑到Petri网是一种具有并行执行语义的自动机,能够方便、自然地表示成有向图,而且Petri网具有表达基于严格同步关系的实时进程交互的能力,满足描述多处理机实时系统软件的需求,并且利用Petri网的特性,人们还可以检查和分析并行结构中潜在的问题,因此,本文利用Petri网来解决多处理机软件的描述问题,以自然、简单和灵活的步骤来描述这类软件中固有的并发、同步以及通信现象,以一种接近于程序员的思维方式去设计这类软件,提供一种可靠的并行结构的程序.

但是,在用Petri网描述较大规模的系统时,网中的结点将增多,甚至会导致状态爆炸.解决这一问题的方法之一是扩充Petri网的描述能力,采用高级Petri网建立系统模型.考虑到软件的可维护性与可重用性问题,对扩展后的Petri网又进行改造,使之最终成为面向对象Petri网.在图1中展示了用Petri网描述多处理机程序的概念,并显示了处理机、进程以及Transition三者之间的关联.

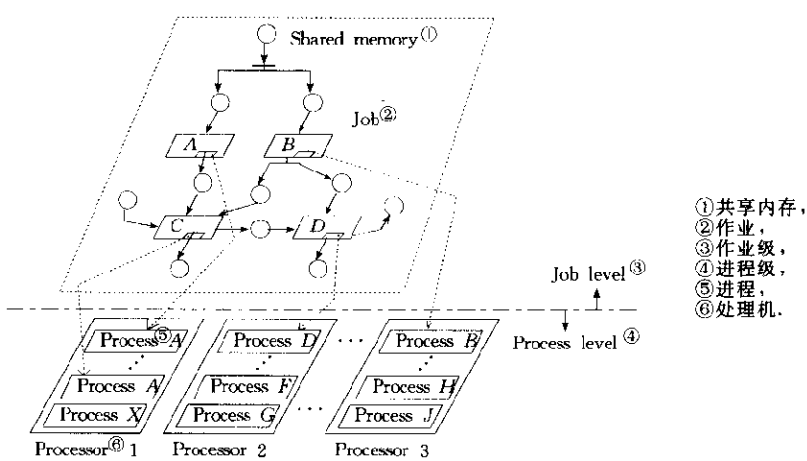
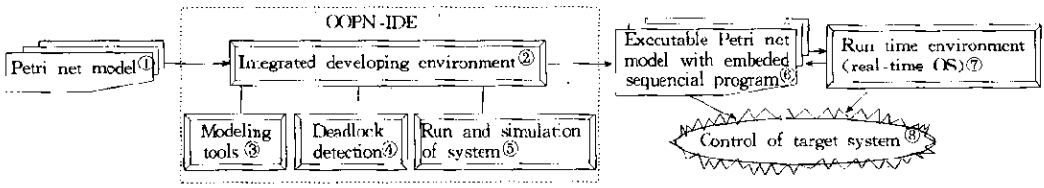


Fig. 1 Representation of a job through a Petri net
图1 用Petri网表示一个作业

为了验证本文提出的方法的可行性,目前我们开发出了可在WINDOWS'98,NT以及Solaris平台上运行的Java版的模型系统OOPN-IDE(object-oriented Petri net based integrated develop environment for concurrent software developing)[5].实时系统的开发环境与执行环境如图2所示,从系统观点来看,以Petri网行为控制实时系统的执行,所形成的实时系统的执行程序简单、有效,且具有通用性.



①Petri网模型,②集成开发环境,③建模工具,④死锁检测,⑤系统运作与系统模拟,
⑥可执行Petri网模型及其嵌入的顺序程序,⑦运作环境(实时OS),⑧目标系统控制.

Fig. 2 The developing and executing environment of the real-time system
图2 实时系统开发与运行环境

为实现本文阐述的多处理机实时操作系统(multi-processor real-time operating system,简称MRTOS),选用了共享存储器的单总线多处理机系统平台,以Linux源码为基础,修改其内核,使其成为实时系统并具有Petri网形式的作业处理方式.当用户对应用系统进行程序设计时,要求在两个级别上进行,即作业级和任务级.作业级使用扩展Petri网描述系统中的并发控制部分(也即应用系统的总体框架).任务级采用编程语言(例如C/C++、汇编等,由于直接固化在硬件平台上的Java虚拟机仍处在研制阶段,还没有看到现成的商品,因此在具体的应用中仍需使用非解释性语言编制目标系统程序).实现系统中的顺序控制部分.在此方法支持下的软件开发,要求用户了解Petri网,通过画图建立系统并发部分的Petri网模型,而用户的编程部分只涉及顺序的、无并发成分,即用户将顺序代码嵌入transition中.像这样将并发描述与具体编程分离,可使用户对应用系统更易于设计及维护.

经过建模、编译后生成的系统代码,最终将加载到多处理机系统上,由其上的多处理机实时操作系统调度执行.

整个系统软件设计需要实现两大部分:(1) Petri网的编辑、编译以及运行工具(在OOPN-IDE工具中,已完成开发[3]);(2) 多处理机系统上的实时操作系统(MRTOS,可直接执行Petri网描述的作业).

总之,经过编译系统处理生成的网模型既是MRTOS操作系统处理的作业控制方案,也是用户开发的实时系统的控制方案,与MRTOS一起构成整体控制系统.

1 MRTOS软件结构

由于对时间的考虑直接影响到对实时系统的设计,因此实时系统有软实时与硬实时之分.本文针对硬实时情况(即在指定期限内系统必须有回应),对MRTOS进行了研制,在设计中考虑了可靠性、实时性、可维护性、可重用性以及方便用户程序开发和方便系统的总体调试几个问题.

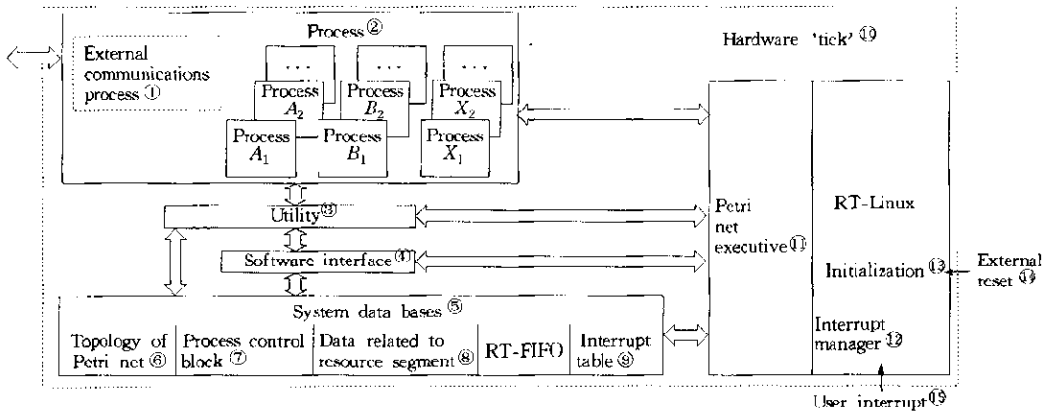
1.1 MRTOS体系结构

操作系统MRTOS的软件结构如图3所示.考虑到系统功能的可扩展性,基于普通Linux的核心控制,在中断控制硬件上开发一层实时内核[5],并把普通Linux作为这个实时内核的扩展部分,只有当实时内核中没有实时进程时才去执行普通Linux.

这样,整个系统分为实时与非实时两部分,非实时部分由普通Linux及其非实时进程组成,实时部分由实时内核及其实时进程组成.实时进程与非实时进程之间通过RT-Linux提供的RT-FIFO缓冲区进行通信.在实际应用中可根据需要对非实时部分进行剪裁.操作系统的实时控制软件结构如图3所示.

操作系统的实时内核包括面向对象Petri网的执行程序、通信机制、进程管理、中断管理以及

实用程序等内容.



①外部通信进程,②进程,③实用程序,④软件接口,⑤系统数据基,⑥Petri网结构,⑦进程控制块,⑧资源段相关数据,⑨中断表,⑩硬件"激励",⑪Petri网执行程序,⑫中断管理,⑬初始化,⑭外部复位,⑮用户中断.

Fig. 3 The software structure of MRTOS

图3 MRTOS软件结构

1.2 MRTOS 的任务调度

在硬实时系统中,任务往往是周期性的,一般使用优先级驱动的调度方法,其优点是实现简单、速度快、可调度性分析容易、瞬时过载的行为可预测.在任何时候,具有最高优先级的任务首先执行,而一个正在处理机上执行的任务如果不被具有更高优先级的任务强占其处理机,则当前的这个任务一直会执行到自动放弃该处理机为止.MRTOS 的任务调度直接关系到实时和多处理机两个方面,应采用优先级为基础的强占式调度策略(比如最终期限调度算法,按照任务指定的必须响应的期限进行调度),并且能够合理地将负荷分担到各个处理机上,发挥多处理机的优势.由于MRTOS所接收的作业是用 Petri 网描述的,其任务的调度由 Petri 网的行为来具体控制.

MRTOS 的任务调度采用两种优先级:一种是绝对优先级,依此优先级把任务分组;另一种是动态优先级,依此优先级来控制每个任务组内部的任务调度.优先级的分配根据进程的相对重要性以及对时间要求的紧迫程度来安排,MRTOS 的优先级机制易于更换,从而可适应于任何特定的调度策略.

处理机的分配可以是静态或动态的,依赖于分配给每个任务的这组处理机.如果有几台处理机符合条件,则分配算法很简单,只需评估一下每台处理机的工作负荷,然后把进程分配到负载最轻的处理机上.一旦进程分配到处理机上运行,则该进程将一直由该处理机调度运行,直到进程结束为止.

1.3 Petri 网的行为执行程序

Petri 网用于描述各进程的控制方案和进程之间的通信方案,Petri 网模型应存放在共享存储器中,并根据所规定的网的行为规则来执行网模型.为了执行 Petri 网,在每台处理机中都存有一个专门用于此目的的执行程序.该执行程序在每个时间片 T_i 用完时中断运行的进程,并且解释上一个时间片期间 Petri 网所发生的变化.

每个处理机中的执行程序的基本活动如下:

· 保留上下文.当时间片 T_i 用完时,中断处理机对当前进程的执行,并保留其上下文.如果该进程已结束,则此项工作可以省略.

· 传送 Token.把在前一时间片执行期间产生的 Token 送到 Petri 网的适当的 place 中,同时

并入到 Petri 网数据结构中.在该阶段系统将产生一个 place 列表,记载所产生的活动变化,即指 place 从空变为含有 Token,或者反之从有 Token 到空这样的活动变化.

• 激活 Transition.检查活动变化表,看是否有与之相关的 Transition 已成为有效的,如果有,则将该 Transition 插入到所选处理机上的就绪 Transition 表中,即 Transition 进入激活状态,如果被激活的 Transition 是"控制 Transition",这说明该 Transition 没有任何与之关联的程序代码,仅仅是控制 Token 的传递工作,那么,执行程序立即完成控制 Token 的传递,该控制 Transition 的工作则到此结束.

• 选择下一个进程(调度).检查该处理机的就绪进程表,选择拥有最高优先级的进程,恢复该进程的上下文(如果是刚激活的 Transition,则要建立新的进程上下文),并把下个时间片 T_i+1 的控制期转给该进程.该时间片完成后,执行程序又重复开始这 4 项工作.

1.4 资源管理

为实现实时功能,应尽可能地简化中断处理、内存管理和 I/O 管理,其方法是把它们都作为资源进行统一调度.

1.5 实用程序

MRTOS 环境为任务的开发提供了一组实用程序,任务程序员利用这些实用程序既可控制任务行为,也可实现该任务与外界环境之间的接口以及与作业中其他进程之间的接口.实用程序分为 4 组:通信与同步、资源管理、进程控制以及中断管理.表 1~表 3 列出了部分实用程序,其他实用程序还包括 RTLinux 提供的实用程序[6].

Table 1 Communication utilities

表 1 通信实用程序

	Initial and final transferences tokens①
Initial_Message	Obtain the data associated to the initial tokens transferred during the firing of the transition②
Final_Message	Token associates data to the final tokens that will be generated at the task termination③
	Transference of tokens during the process execution④
Send_Token	Send a token through the indicated intermediate arc⑤
Wait_Token	Receive a token from the place linked to the indicated intermediate arc. If there is no token available, the task is suspended until one arrives⑥
Receive_Token	Receive a token through the indicated intermediate arc. Task execution continues even when there are no tokens available⑦

①进程起始与终止时对 Token 的传输,②接收 Transition 引发期间传输的初始 Token 相关数据,③接收任务结束时产生的终止 Token 相关数据,④进程执行期间对 Token 的传输,⑤通过中间弧发送一个 Token,⑥从与中间弧连接的 place 中接收一个 Token.若此处没有 Token 存在,则任务挂起,直到有 Token 到达时,任务才被激活,⑦通过中间弧接收 Token.若 Token 不存在,任务继续执行.

Table 2 Process control utilities

表 2 进程控制实用程序

Delay	Suspend the execution of a task during the indicated time. Allow the CPU to execute other ready tasks①
Terminate	Logically terminates a task, de-allocate all tokens its assigned resources, and send the final tokens②
Abort	Abnormally terminates a task after some error conditions. The final tokens are not dispatched and the operating system halts job execution③
Ask_PID	Dynamically modifies the assigned priority④

①在指定时刻挂起任务,同时允许 CPU 执行其他就绪任务,②从逻辑上结束一个任务,并去分配,收回它的所有资源,发送终结 Token,③在某些错误条件发生之后,任务异常结束,不发送终结 Token,操作系统停止作业的执行,④动态修改已分配的优先级.

Table 3 Interrupt management utilities
表 3 中断管理实用程序

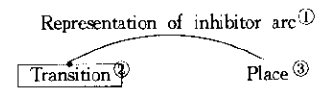
New_ISR	Assign a routine as an interrupt server①
Rel_ISR	Release an interrupt from being serviced by its current ISR②

①指派一个例程为中断服务程序 ISR,②释放一个由其当前 ISR 处理的中断.

2 扩展 Petri 网[7](EPN)

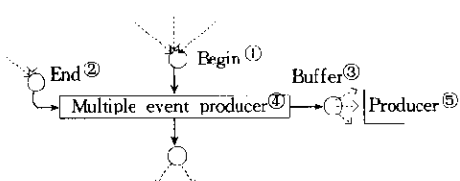
本文涉及的扩展 Petri 网,是指在标准 Petri 网的基础上添加 3 项内容:禁止弧、计时 Transition 以及中间弧,使 Petri 网的描述能力增强.

禁止弧. Transition 可以具有的一种特殊输入弧称为禁止弧,其作用是提供零测试能力. 具有禁止弧的 Transition,仅当禁止弧的起始端处的 place 无 Token 时,该 TRANSITON 才能引发. 换句话说,带有禁止弧的 Transition 能够引发,则意味着在禁止弧上无 Token 的消耗. 其图形表示如图 4 所示.

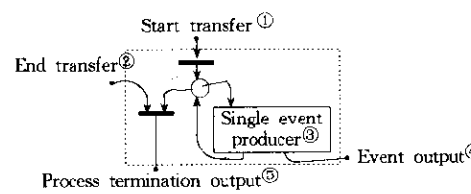


①禁止弧表示,②变迁,③库所.
Fig. 4 Representation of inhibitor arc
图4 禁止弧的图形表示

时控 Transition. 在 Transition 的定义中增加"活动/非活动"状态,即当 Transition 有效时(引发条件已满足),立即被引发,输入位置中的 Token 被移走,此时 Transition 被置为活动状态. 这种活动状态要持续一段时间,持续时间的长短取决于该 Transition 的函数参数,当这段时间过后,对应于每个输出弧都将产生一个 Token. 在此扩充下,Petri 网的状态由网标识(Marking)与每一个 Transition 的活动状态的组合来定义. 若用 Transition 表示多处理机软件的进程,则必须采用时控 Transition,用于描述进程的非零执行时间.



①开始,②结束,③缓冲区,④多重事件生产者,⑤消费者.
(a) Graphical representation
(a) 图形表示



①开始输入,②结束输入,③单一事件生产者,④事件输出,⑤进程终止输出.
(b) Standard Petri net model of the producer process
(b) 用标准Petri网建模的生产者过程

Fig. 5 Examples of a Petri net with intermediate arcs
图5 带中间弧的Petri网的例子

中间弧. 时控 Transition 只是在引发起始时刻接收 Token,而且仅在活动状态结束时才产生 Token. 为了改进这种约束,采用一种特殊弧,称为中间弧. 中间弧既可以是输入弧也可以是输出弧,Transition 在活动状态时通过中间弧来获取或产生 Token. 中间输入弧对 Transition 的有效与否不起任何作用,同样地,在 Transition 结束时,也没有 Token 在中间输出弧上产生. 中间弧的图形表示,是由 Transition 矩形框的短框两侧画出的弧来完成.

在 Petri 网中采用中间弧的主要目的是为 Petri 网表示的多处理机软件提供灵活性,使得 Petri 网中的时控 Transition 不仅仅是在 Transition 的活动开始和结束时传递 Token,而且在 Transition 的活动期间也可以传递 Token.

现在以两个处理机系统为例,执行如图 6(a)所示的扩展 Petri 网描述的例子,每台处理机上的执行程序执行该网时的相应动作如图 6(b)所示.

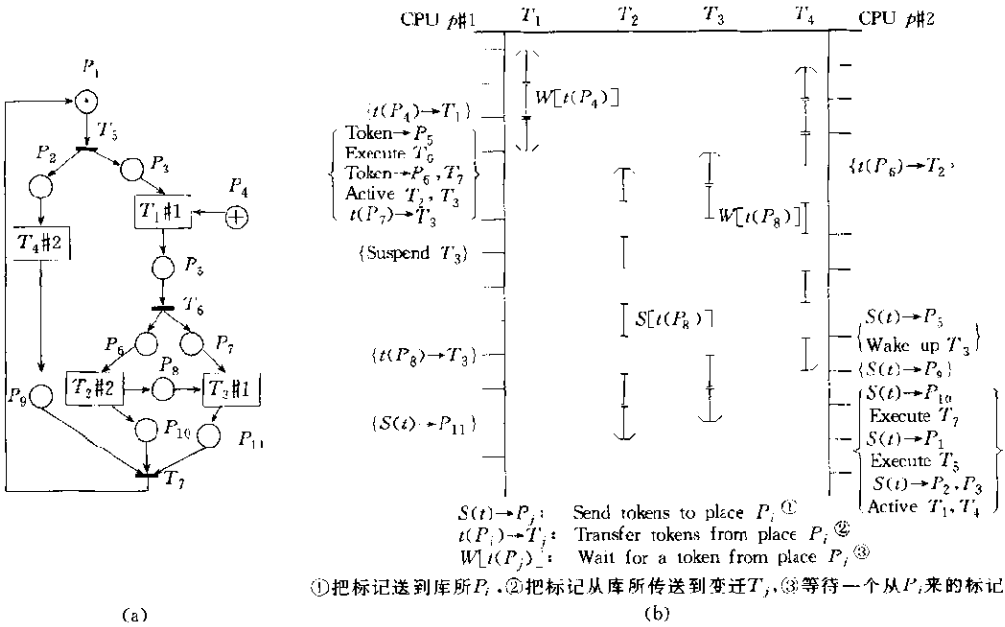


Fig. 6 Examples of the operation of two executive
图6 两个执行程序操作的例子

使用EPN描述多处理机软件,使程序员摆脱了对应用系统并发部分的编码工作,尤其是省却了并发进程之间通信的复杂细节,而让用户集中精力于系统功能的开发上,并使用户在Petri网模型的制导下,在Transition中嵌入顺序程序代码,与EPN构架一起形成应用系统的程序.该方法简化了并发程序开发过程,提供了可靠的程序结构.但该模型的表示在软件的可维护性与可重用性上仍然存在着问题.因此,我们将EPN表示改造成面向对象Petri网的表示,以解决软件的可维护性与可重用性问题.

3 面向对象Petri网(OOPN)[8]

EPN虽然在建模能力上比标准Petri网有所增强,但在系统的维护上仍有待改进,并且在软件的可重用性上还很困难.为此,我们采用面向对象Petri网来解决软件的可维护性及可重用性问题.本节所介绍的面向对象Petri网简称为OOPN,是并发软件开发工具OOPN-IDE所采用的面向对象Petri网模型.

改造EPN的方法是,用方框圈住对象的内部(EPN描述的子网),表示封装与抽象,对象的外部接口部分由"消息队列"(用椭圆表示,其作用类似于用圆表示的place)、"门"(用粗线表示,其作用类似于用条线表示的控制Transition)以及它们之间的流关系(用弧线表示)给出.各对象内部分别可以有若干实体(用黑点表示,即Token),各Token所处的place表示各自的当前状态.Token在网中的流动代表了网的行为,所以把此类Token看成是网的实体.

在OOPN中定义了两种对象类型:简单对象与复合对象.在简单对象中,不包含并发部分,它只用于表示顺序行为与静态性质;而在复合对象中则允许并发,因为复合对象由具有独立行为的简单对象构造而成,其控制分布在各个简单对象中,并且根据系统要求可以使这些简单对象的顺序行为同步.复合对象的图形表示如图7所示,其中子对象的细节没有给出.OOPN把目标系统组织成一个个并发对象,并建立起对象之间的消息传递关系.一个对象有其内部结构和外部结构,内部结

构定义操作的实现细节,而外部结构定义对象职责和通信信息.对象职责包括对外提供的服务,通信信息包括消息类型.通过用门来连接相关对象的外部结构,从而建立起对象之间的通信.这种构造可使同步约束从每个对象内部分离出来,更便于对象的重用,也为系统死锁分析方法奠定了基础[9].

图6(a)中给出的EPN网可用OOPN表示为如图8所示的形式.

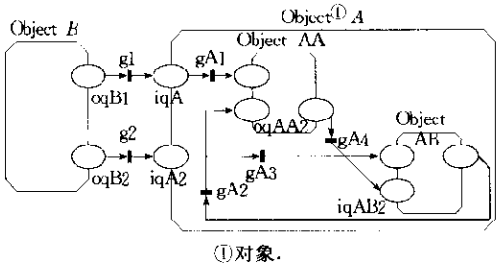


Fig. 7 The composite object A and object B constitute the system

图7 复合对象A及对象B构成的系统

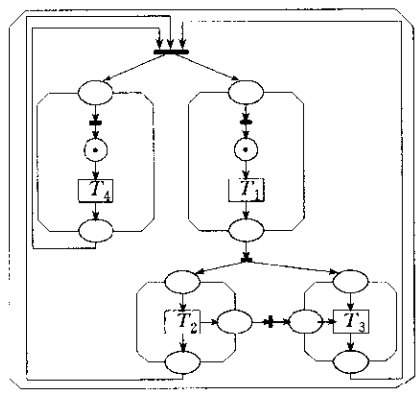
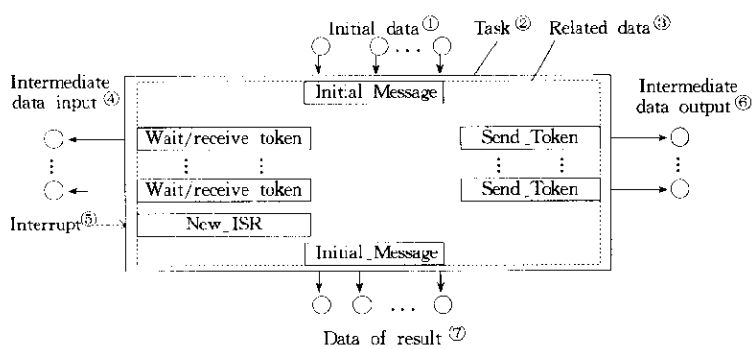


Fig. 8 Equivalence of EPN of Fig. 6 in OOPN

图8 OOPN表示的图6中的EPN网

4 MRTOS为任务开发提供的编程环境

在MRTOS环境中,任务是基本的软件单元,是单一控制线程表示的顺序程序,在单一处理机上完成全部执行,所以任务的规格说明、设计、编码和测试可以采用传统方法进行.为了建立每个任务的功能规格说明,MRTOS环境提供了与时控Transition相对应的形式化框架.该框架以实用程序为基础,定义任务的功能结构,这组实用程序构成操作系统的界面.采用这种模式,任务的开发相对独立,不受其他任务开发的限定.任务程序员编写代码的方式与在单处理机环境下相同,系统实用程序为OOPN的Transition抽象描述提供了应遵循的一般准则.在任务程序的开始处调用分配内存实用程序,主要为变量和栈区安排内存,并且调用Initial_Message实用程序,从而取得与初始Token相关的数据.任务结束时则调用Final_Message,产生分布到终止弧上的数据,随后调用Terminate实用程序,结束任务.图9表示了时控Transition框架与实用程序调用之间的功能关系.



①初始数据,②任务,③相关数据,④中间数据输入,⑤中断,⑥中间数据输出,⑦结果数据.

Fig. 9 Structure of a task

图9 任务结构

为了能够用OOPN网方便地描述程序,OOPN-IDE提供了交互式图形建模工具以及自动生成OOPN网的编译工具.

5 结 论

本文涉及的以面向对象 Petri 网为基础的多处理机操作系统,允许用户以一种接近于人的思维方式描述系统软件,通过图形方式建立并发系统模型.文中提出的两级程序设计方法(作业级:图形方式的并发 Petri 网模型;任务级:顺序程序设计)使得多处理机系统软件易于开发和维护,并提供了软件可重用机制.

该方法的验证模型 OOPN-IDE[10]目前运行在 Java JDK2.0 平台上.针对具体目标系统的开发,将 OOPN-IDE 中的网的执行部分以及系统的进程及中断管理部分移植到目标机上.目前,我们已在奔腾 III 双 CPU 机型上实现了目标系统原型,以电梯调度为例对系统进行了验证[11].

在解决多处理机系统的应用问题中,以 Petri 网为基础,把 Petri 网直接用于控制,涉及到实时性、多处理机系统的并行计算,以及以图形方式描述需求并以此生成并发程序,这些问题的圆满解决,其任务是艰巨的.将这种方法用于分布式系统上是我们下一步的研究目标.

References:

[1] Selic, B. Turning clockwise: using UML in the real-time domain. Communications of the ACM, 1999,42(10):46~54.

[2] Fathi, E. T., Krieger, M. An executive for task-driven multimicrocomputer systems. IEEE Micro Chips, System Software and Application, 1983,3(5):32~41.

[3] Grasso, P. A., Forward, K. E., Dillon, T. S. Operating system for dedicated common memory multi microprocessor system. IEE Proceedings on Digital and Computer Techniques, 1982,129(5):200~206.

[4] Heath, W. S. Software design for real-time multiprocessor VMEbus systems. IEEE Microcomputer, 1987,7(6):71~80.

[5] Ren, Ai-hua, Niu, Jin-zhong, Sun, Zi-an, *et al*. OOPN integrated developing environment for concurrent programs. Computer Science, 1999,6(supplement):126~131 (in Chinese).

[6] Barabanov, M. A Linux based real-time operating system [MS. Thesis]. Socorro, New Mexico: New Mexico Institute of Mining and Technology, 1997. http://www.rtlinux.org.

[7] Vallejo, F., Gregorio, J. A., Harbour, González, M., *et al*. Shared memory multimicroprocessor operating system with an extended Petri net model. IEEE Transactions on Parallel and Distributed Systems, 1994,5(7):749~762.

[8] Ren, Ai-hua, Niu, Jin-zhong, Zhang, Yong-ming. Object-Oriented Petri net based method for concurrent program modeling. Journal of Beijing University of Aeronautics and Astronautics, 1998,24(4):491~494 (in Chinese).

[9] Lee, Yang Kyu, Park, Sung Joo. OPNets: an object-oriented high-level Petri net model for real-time system modeling. Journal of Systems Software, 1993,20(1):69~86.

[10] Niu, Jin-zhong. Integrated development environment of parallel program based on object-oriented Petri net [MS. Thesis]. Beijing: Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, 1999 (in Chinese).

[11] Ren, Ai-hua. Research on the concurrent software developing method based on object-oriented Petri net [Ph.D. Thesis]. Beijing: Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, 2001 (in Chinese).

附中文参考文献:

[5] 任爱华,牛锦中,孙自安,等.OOPN 集成开发环境.计算机科学,1999,6(增刊):126~131.

[8] 任爱华,牛锦中,张永鸣.一种基于面向对象 Petri 网的并发程序建模方法.北京航空航天大学学报,1998,24(4):491~494.

[10] 牛锦中.基于面向对象 Petri 网的并发软件集成开发环境的研究与实现[硕士学位论文].北京:北京航空航天大学计算机科学与工程系,1999.

[11] 任爱华.基于面向对象 Petri 网的并发软件并发方法研究[博士学位论文].北京:北京航空航天大学计算机科学与工程系,2001.

The Multi-Processor Real-Time Operating System Based on Petri Net Model*

REN Ai-hua, DU Yue-dong

(*Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, Beijing* 100083, *China*)

E-mail: renah@buaa.edu.cn

http://www.buaa.edu.cn

Abstract: In the multi-processor environment, real-time systems have a concurrent event driven nature. The software structure presents multiple synchronization points and producer-consumer interrelations, which give rise to a very complex control structure. The lack of standard methods and tools for software development in these systems results in poorly specified software with an unclear structure that is very difficult and expensive to maintain. In order to develop this kind of software with high efficiency and maintainability, the Petri net is adopted to represent multi-processor system software since the Petri net is a very powerful description tool for the parallel and concurrent program. In this paper, a methodology for programming multiprocessor real-time systems is discussed. This methodology is based on two programming levels: the task level, which involves programming the basic actions that may be executed in the system as units with a single control thread, and the job level, on which parallel programs to be executed by the complete multiprocessor system are developed. The model that has been chosen for the representation of the system software is based on an objected oriented Petri net, which facilitates job-level programming. In this research, in order to support a reliable program structure, a natural, simple, and flexible procedure has been sought to describe the concurrence, synchronization, and communication phenomena inherent in this kind of software in a way that is very close to how the human programmer conceives of them. The structure and implementation of an operating system for the proposed methodology are described in this paper.

Key words: multiprocessor operating system; Petri net; object-oriented technique; concurrent system modeling

* Received January 7, 2000; accepted March 23, 2000

Supported by the National Natural Science Foundation of China under Grant No. 69883002