并发实时系统的串行非实时仿真框架*

宋 羽,王 琨,张 宝
(中国兵器工业第 203 研究所,西安 710065)

摘 要:为了解决并发实时系统在通用计算机上的仿真问题,文中分析了并发实时系统的一般特性,归纳了仿真验证的硬件架构选择,通过分离并发进程计算与运行顺序,按照时标调度进程,提出了一种将并发问题串行化的通用仿真框架,该框架适用于一般性的并发实时系统仿真,可以有效加快仿真程序开发,提高程序质量。

关键词:武器系统;并发实时;仿真;框架
中图分类号:TJ765.4 **文献标志码**:A

Serial Non-realtime Simulation Framework of Concurrent Real-time System

SONG Yu, WANG Kun, ZHANG Bao
(No. 203 Research Institute of China Ordnance Industries, Xi'an 710065, China)

Abstract: For solving the simulating problem of concurrent real-time system on common computer, the general characteristics of concurrent real-time system were analyzed, two types of hardware architecture selection were concluded, and a framework was proposed for serializing concurrence through separating algorithm from execution order and scheduling process according to time-stamp. The framework is applicable to general simulation of concurrent real-time system. It can speed up development of simulation program effectively, and improve program's quality.

Keywords: weapon system; concurrent real-time; simulation; framework

0 引言

武器系统大多是并发实时系统。武器系统开发中,仿真是一种重要验证手段。文中分析了并发实时系统的一般特性,归纳了仿真的两种硬件架构,并提出一种基于串行非实时计算机的并发实时系统仿真通用框架。

1 并发实时系统的一般特性

并发实时系统具有并发性和实时性。并发指的是可以在同一个芯片上多核、同一个处理器上的多个进程或在物理分离的多处理器上执行。实时性具有相对意义,通常来说,当不能满足定时约束或时限就被认为是致命错误时,就说这个定时约束是强的[1],武器系统的时间约束通常是致命的,因此武器系统一般为强实时系统。

一种典型的具有两个模块的并发实时系统如图 1 所示。该图描述了并发实时系统一种复杂形式,A 与 B 既是数据的生产者,也是数据的消耗者。在图中,A 或 B 可以是运行在不同处理器上的不同进程,也可以是运行于同一个处理器上的不同进程。虚线代表控制流,例如时钟或是外部事件中断;实线代表数据流。程序的运行由控制流激活。中间的小方框代表数据存储缓冲。由于 A 或 B 有各自独立的启动时刻,数据计算后不会立刻消失,而是放在存储缓冲区中。程序与数据的存在时间是不同的。

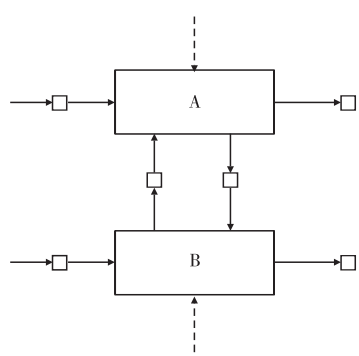

图 1 典型的并发实时系统模块图

当 A 和 B 运行时间发生交叉时,如果允许在运行过程中间交换数据,A 所使用的数据 d_{BA} 将在 t_{B2} 时刻发生变化(如图 2),当 A 在 t_{B2} 前后两次用到 d_{BA},则在一次运算过程中将使用两个不同数据,这有可能导致运行错误,因此设计者要采用必要的措施消除这种不确定性,使数据仅在进程的开始时刻进入程序,一旦进入,在一次运算过程中不再改变,直至当次运

* **收稿日期**:2011 - 04 - 07
作者简介:宋羽(1981 -),男,安徽巢湖人,工程师,研究方向:系统工程,控制工程,软件工程。

算结束,文献[3]阐述了并发控制的几种方法。这些并发控制方法的数据传输时序可以表达为图2形式,B在A的运行期间产生新的数据,而A直到B运行结束后下一个运行周期才会使用 d_{BA}。

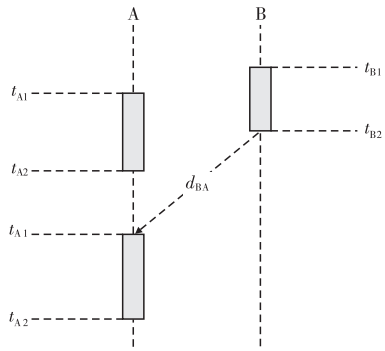

图2　并发程序间的数据传输时序

2　仿真验证硬件架构的选择

按仿真实验中所取时间标尺 t_1(模型时间)与自然时间标尺 t_2(原型)之间比例关系可将仿真分为实时仿真和非实时仿真两大类,$t_1/t_2=1$ 为实时仿真,否则为非实时仿真[2]。一般仿真验证硬件架构见图3,实时仿真中仿真程序与被验证对象同时运行,通过数据采集装置采集实际对象的输入,产生标准输出与实际输出对比;非实时仿真一般采用数据记录装置记录实际对象的输入输出,实际对象运行结束后仿真程序从输入记录中读取输入数据,并生成标准输出与实际输出对比。

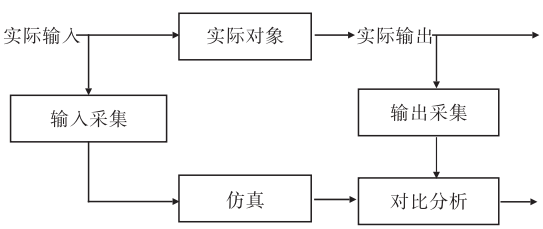

图3　仿真验证硬件架构

实时仿真对于仿真程序所运行的硬件要求较高,必须满足实时性要求;而非实时仿真可以运行于通用计算机上,具有较低的成本。文中的框架主要用于非实时仿真软件,通过统一调度模拟并发实时计算。

3　仿真软件框架

3.1　需要解决的问题

当采用通用计算机仿真并发实时系统时,有两个主要的约束,第一,通用计算机的运行方式是串行的,一次只能处理一件事;第二,在实时系统中,有硬时钟保证时间约束,而通用计算机一般没有这种资源。需要一种有效的规范化设计方法,将并发实时系统转换为串行非实时结构。

3.2　解决方案

对以上问题的一个解决方案是在仿真中使用单个调度程序统一调度进程,所有的进程运行时刻放在一个时间轴上排列,主控程序按时间顺序依次循环统一时间轴上的每个时间点,根据每个时间点上的程序标识选择进程,将并发的运行转化为串行的逻辑分支。该方法类似于数据库系统设计中所采用的时标排序并发控制方法[4],虽然数据库系统与仿真是两个完全不同的领域,但在它们所面临的并发问题具有相似性。

3.3　框架准备

在实现该框架前,需先收集以下信息:

1)每个处理器上运行的进程,其中数据记录应作为运行于记录装置上的一个并发进程;

2)每个进程之间交换的数据;

3)每个进程的运行时间,确认是否可忽略,对于不可忽略时间的进程确定运行时间;

4)每个进程的启动条件。

3.4　框架实现

该框架流程见图4,图中流程的详细说明如下:

1)生成进程和数据对应的时间列。

一般有以下几种方式:

a)采用数据记录中已有的时间列:该时间列通常来自于仿真对象或记录装置的内部时钟;

b)根据输入计算时间:适用于由外部输入触发的进程,例如在某个脉冲到来时才运行的程序;

c)固定的时间:适用于按照固定规律运行的进程,例如以固定周期运行的某个运算程序。

对于运行时间可以忽略不计的进程,仅仅为进程的运行建立时间列,对于运行时间不可忽略的进程,需要分解为"运行"和"数据更新"两个进程,每个进程的时间确定方法可以是以下两种方法之一:

①所有进程运行时刻使用实际进程开始时刻,接口数据的更新时刻使用发送方进程的结束时刻;

②所有进程运行时刻使用实际进程结束时刻,接口数据的更新时刻使用接收方进程的开始时刻。

2)由于不同硬件时钟可能存在误差,因此时间列需要同步。

3)按照时间排列运行次序。

有可能出现两个进程运行时间相同的情况,处理方法如下:

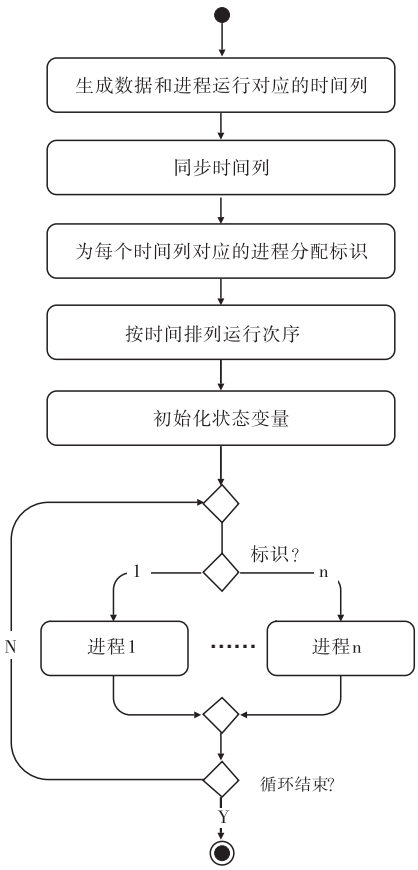

图 4 仿真流程框架

a)位于同一处理器并且时间相同:实际对象会按一定的调度方法串行处理同一时间出的多个进程,一般通过中断机制实现,仿真应模拟实际对象这种机制,按中断优先级排列同一时间出现的进程;

b)位于不同处理器并且时间相同:这可能是巧合,也可能是因为记录设备的时间分辨率不够导致不能区分具有细微时差的两个进程。此时无法确定进程的先后顺序,只有以任意顺序排列进程。这种不确定性可能导致仿真结果与实际的偏离,因此仿真者应确保记录设备有足够的时间分辨率或者允许仿真结果和实际输出的偏离。

4)初始化状态变量时要包括所有数据接口缓冲和模块自身状态。

5)仿真循环,循环中判断当前步骤对应的进程标识,选择运行不同的进程。

3.5 程序样例

以下是使用 Matlab 实现该框架一个样例:

```
%% = = = = = = = = = = = = = = = = = = = = %%
% 生成时间列,并分配标识
tIn = aTu; ProcIn = 0 * ones(size(tIn));
tA = aT0; ProcA = 1 * ones(size(tA));
tB = aT1; ProcB = 2 * ones(size(tB));
tRc = 0:0.01:10; ProcRc = 3 * ones(size(tRc));
tA = tB - t0AB; % 同步 A 和 B
tsSim = sortrows([tA, ProcA; tB, ProcB; tRc; ProcRc],1); % 按时间排列运行次序
% 初始化状态变量
.....
kIn = 1; kOut = 1; nCompute = length(tsSim);
% 仿真循环
for k = 1:nCompute;
procId = tsSim(k,2); t = tsSim(k,1);
switch procId
case 0; u = inputRec(kIn), kIn = kIn + 1;
case 1; ya = fa(u,t);
case 2; yb = fb(ya,t);
case 3; y(kOut) = yb, kOut = kOut + 1;
otherwise
end
end
%% = = = = = = = = = = = = = = = = = = = = %%
```

4 结论

文中的框架结构简单、易于扩展、适用范围广,可以有效加快仿真程序开发,提高代码质量;同时由于人的大脑类似低速串行处理器,通过转化模型,可以提高开发人员对于软件行为的清晰理解。

参考文献:

[1] Jane W S Liu. 实时系统[M]. 北京:高等教育出版社,2002.

[2] 何江华. 计算机仿真导论[M]. 北京:科学出版社,2001.

[3] 慕春隶. 嵌入式系统的构建[M]. 北京:清华大学出版社,2004.

[4] George Coulouris, Jean Dollimore, Tim Kindberg. 分布式系统概念与设计[M]. 金蓓弘,曹冬磊,译. 北京:机械工业出版社,2008.