基于 BIP 框架的 DPU 系统建模与验证*

黄崇迪[1a,b,c], 万 海[1a,b,c], 顾 明[1a,b,c], 陈 睿[2]

(1. 清华大学 a. 软件学院; b. 信息系统安全教育部重点实验室; c. 清华信息科学与技术国家实验室(筹), 北京 100084; 2. 北京控制工程研究所,北京 100190)

摘 要: DPU(data process unit,数据处理单元)是嵌入式系统中的一个典型组件,被广泛应用于太空领域,它在层次化的嵌入式系统架构中起到承上启下的作用。保证这类安全攸关系统可靠性的主要方法包括冗余容错、测试和仿真。近年来,形式化方法作为确保可靠性的一种重要补充,得到了广泛的关注。BIP(behavior interaction priority)是一个通用的系统级形式化建模框架,支持层次化和模块化,包含一套支持建模、模拟和验证的工具集。给出了一种基于 BIP 框架对 DPU 进行系统级建模与验证的一般方法,总结了一套使用 BIP 框架对 DPU 建模应遵循的原则及技巧。以航天领域一个真实 DPU 系统为例,系统地对方法、原则和技巧进行了介绍。通过该方法,找出了使用传统方法难以发现的错误。实践表明,该方法具有很好的可复用性和可扩展性,是确保系统可靠性的有益补充。

关键词: 数据处理单元; 行为—交互—优先级框架; 形式化方法; 建模; 验证

中图分类号: TP301.1 **文献标志码**: A **文章编号**: 1001-3695(2012)08-2961-06

doi:10.3969/j.issn.1001-3695.2012.08.041

Modeling and verifying DPU system based on BIP framework

HUANG Chong-di[1a,b,c], WAN Hai[1a,b,c], GU Ming[1a,b,c], CHEN Rui[2]

(1. *a. School of Software, b. Key Laboratory for Information System Security Ministry of Education, c. Tsinghua National Laboratory for Information Science & Technology(TNList), Tsinghua University, Beijing* 100084, *China*; 2. *Beijing Institute of Control Engineering, Beijing* 100190, *China*)

Abstract: DPU (data process unit) is a typical component for embedded systems. It's widely used in space application. It obtains data from sensors in lower level, processes data and then sends result back to master computer in higher level. There are redundancy, testing and simulation to ensure the reliability of the DPU system. BIP (behavior interaction priority) is a general formal modeling framework for embedded system. It supports hierarchy and module structure, contains a toolset including modeling, simulation and verification tools. This paper presented a set of general methods and principles for modeling DPU system using BIP framework and verifying properties on the BIP model. It took a real DPU system from space as example and succeeded in finding some bugs, which was difficulty for traditional way. The method was reusable and extendable. It's useful to ensure the reliability of the system.

Key words: data process unit(DPU); BIP framework; formal method; modeling; verification

0 引言

DPU(data process unit,数据处理单元, 也被称为 data processing unit 或 data handle unit)是嵌入式实时系统中的一个典型组件,被广泛应用于太空领域。DPU 是系统的核心模块,在层次化的系统框架中位于中间位置,是上层主控系统(master computer)和下层传感器交互的桥梁。DPU 接收上层主控系统的指令,根据指令从下层传感器获取数据并作处理,最后将处理结果发送回主控系统。

对于应用于太空领域的 DPU 系统而言,如何确保其鲁棒性和响应速度是一个安全攸关的问题。一般从三个方面判断 DPU 是否正常工作:操作是否能在指定时间内完成;操作是否按照指定的协议执行;从 DPU 返回的数据是否满足需求。本文以一个实际太空应用里的 DPU 系统为案例进行研究。

目前,确保 DPU 这类嵌入式系统可靠性的方法从硬件的角度来说主要有采用高可靠器件及冗余容错设计[1];从系统的角度来说主要有仿真和测试[2,3]。冗余容错设计硬件本文不作讨论。测试和仿真并不完备,它们只能针对输入和操作条件的一个子集对系统进行评价。此外,系统中有些错误只有当某些特定的事件按一定顺序发生后才会出现,不可重复。使用测试的方法来找这类错误需要尝试大量测试用例,由于不可重复性,这些测试用例的价值很有限。

形式化方法作为确保嵌入式系统可靠性的方法,在嵌入式系统领域得到越来越多的重视。形式化方法是仿真和测试的重要补充。

收稿日期: 2011-12-16; **修回日期**: 2012-01-22 **基金项目**: 国家自然科学基金资助项目(91018015);国家"975"重大基础研究计划资助项目(2010CB328003);国家自然科学基金委,中法多年期合作项目(60811130468)

作者简介: 黄崇迪(1987-),男,广东江城人,硕士研究生,主要研究方向为系统建模与模型检测;万海,男,湖南郴州人,博士,主要研究方向为系统建模与定理证明(huangcd.thu@gmail.com);顾明,女,教授,主要研究方向为操作系统、中间件技术、分布式应用系统支撑平台、电子商务等;陈睿,男,山西岚县人,工程师,主要研究方向为嵌入式软件和可信软件.

BIP 框架[4]是一个通用的系统级形式化建模框架,支持层次化结构,并包含一套支持建模、模型转换、仿真、验证和代码生成的工具集。使用 BIP 语言对系统建模之后,既可以对模型进行仿真,也可以用模型检测工具检查模型,在验证系统无误的情况下还可以使用代码生成工具将 BIP 模型转换成可执行代码,替换原有的程序。BIP 框架已在嵌入式系统中得到了较广泛的应用,包括对实时系统的建模[5,6]、对同步系统的建模[7]。同时,也可以将其他建模语言的模型转换成 BIP 模型[8]。

本文工作流程如图 1 所示。首先根据功能结构对系统进行层次化的分解,据此分别建立 DPU 软件系统和硬件环境模型。将软硬件模型组合得到完整的 BIP 模型,再将待验证属性建模成监控自动机与 BIP 模型组合,得到可验证的 BIP 模型。在此基础上,使用 BIP 编译器将模型编译成 C ++ 代码,并与 BIP 引擎一起编译成可执行程序进行仿真。此外,也可以应用 DFinder 工具对 BIP 进行死锁检测。

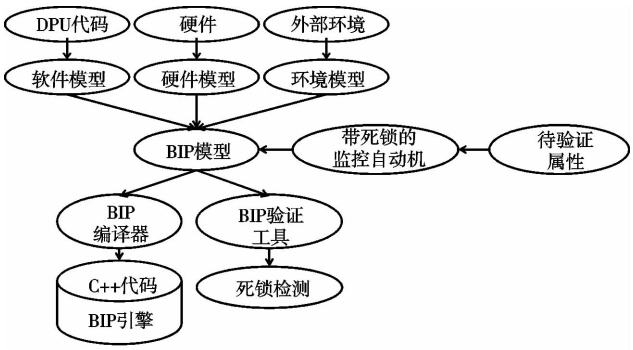

图1 BIP建模框架

本文主要的贡献如下:

a)将形式化方法应用到 DPU 系统的可靠性保障过程中,使用一套形式化的工具和方法,成功找到系统的两处问题。

b)给出了使用 BIP 框架对 DPU 进行系统级建模的核心流程,针对 DPU 系统以及外部环境模型建立统一的 BIP 模型。

c)总结了一套使用 BIP 框架对 DPU 系统建模应遵循的原则及技巧,包括如何对系统进行分解与组合、如何选择恰当的抽象层次、如何描述待验证属性以及如何验证系统是否满足需求。

d)在实际应用中,提出了用双重锁解决中断调度的策略。

1 BIP 框架介绍

BIP 是一套针对复杂系统的组合式建模框架。其基本元素是描述模块行为(behavior)和接口的原子组件。组件之间通过交互(interaction)和优先级(priority)的定义形成更大的组件。

1.1 BIP 基本概念

原子组件(atomic)由增加了变量和端口的 Petri 网或者有限状态自动机模型描述。变量用于存储内部数据;端口是迁移的标签,用于定义与其他模块交互的接口。端口可以关联数据,迁移是状态的跳转,包含端口、约束和动作。约束是布尔表达式描述的使能条件,动作是作用在变量上的运算。BIP 使用 C 语言的语法定义变量和动作。

图 2 是一个恒温器的原子组件。自动机的初始状态为 Off {temp = 30};在 Off 状态下,温度每单位时间增加 1;在 On 状态下,温度每单位时间降低 2;在 Off 状态下,温度如果温度大于等于 40,将触发 turnOn 动作,打开恒温器;在 On 状态下,如果温度小于等于 20,将触发 turnOff 动作,关闭恒温器。优先级 tick < turnOn 保证了一旦温度大于等于 40,turnOn 动作马上被执行。

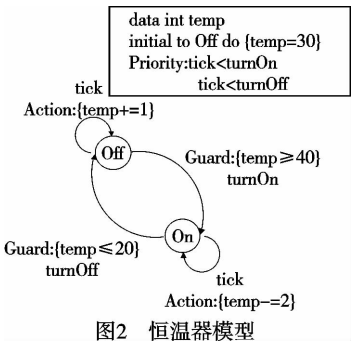

图2 恒温器模型

组件之间的同步和数据交换通过连接子(connector)描述。连接子包含端口集合和交互集合。BIP 定义了两种类型的交互:强同步(rendezvous)表示必须所有端口都参与交互;广播(broadcast)表示交互存在一个发起端口和一系列接收端口。端口集包括完备和不完备两种端口。完备端口是触发器,能够主动发起广播。图 3 上面的连接子是强同步类型,唯一允许的交互是{tick1,tick2,tick3}(这里,tick1、tick2、tick3 必须同时参与交互);下面的是广播类型,out 是触发器,允许的交互包括{out},{out,in1}, {out,in2}, {out,in1,in2}。

交互是一个四元组$\{pl,g_p,u_p,d_p\}$。其中:pl 是参与交互的端口集;g_p 是交互的使能条件;u_p 和 d_p 是两个执行的动作,分别记为上动作(up action)和下动作(down action)。交互中,上动作先于下动作执行。对层次化的连接子而言,动作的执行顺序是:低层连接子的上动作→高层连接子的上动作→高层连接子的下动作→低层连接子的下动作。若涉及数据计算,可以认为上动作将低层数据向上传送,在高层连接子计算,之后通过下动作将结果返回给低层。图 4 是一个求最大值并将结果返回给交互方的连接子。在低层,上动作求出 a 和 b 的最大值并将其向上传送;顶层的上动作继续求最大值,下动作将新的最大值向下传送;最后在底层的下动作更新。交互完成后,变量 a、b、c 被更新为它们中的最大值。

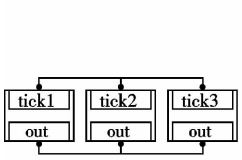

图3 广播与同步

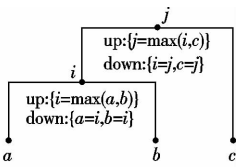

图4 层次连接子

优先级是定义在两个交互上的偏序关系。有多个交互能够执行的时候,优先级最高的将被执行。优先级描述系统的调度策略,通过优先级可以实现复杂的逻辑关系。图 5 是一个利用优先级实现的 FIFO 调度算法,原子组件从 SLEEP 状态进入 WAIT 状态时得到当前时间。前两条优先级描述了较早进入 WAIT 状态的组件有优先执行的权利;后两条优先级描述了后执行的任务必须等待先执行的任务执行完成之后才能执行。

在已有组件之上定义交互和优先级可以构成复合组件(compound)。通过组合的方式,可以在保持系统结构的前提下得到整个系统的 BIP 模型。

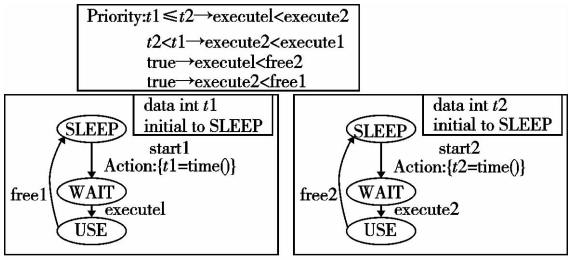

图5　FIFO调度

1.2　BIP 工具集

BIP 工具集包含建模、代码生成、执行、分析(静态的和运行时的)和模型转换等工具。图 6 是工具集的一个概述,从图中可以看出,BIP 工具集不仅可以用于对软/硬件系统进行建模,也可以将其他语言的代码转换成 BIP 模型。有了 BIP 模型之后,可以使用工具进行仿真和验证,也可以使用代码生成工具生成可执行代码。生成的代码可以直接替换原有的程序,以获得更强的鲁棒性。

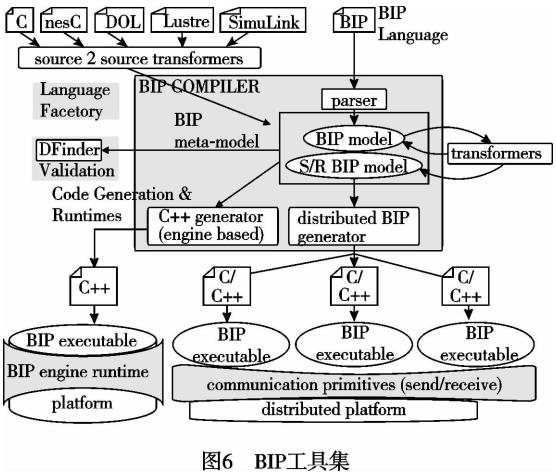

图6　BIP工具集

2　DPU 系统介绍

DPU 是嵌入式系统中的一个常用模块,本文中的 DPU 来自某太空领域中的探测系统,其作用是接收上层主控系统(master computer)的指令,然后从下层传感器中获取数据,对数据进行处理并将处理结果返回给主控系统。除了需要按照要求完成指定任务以外,DPU 的运行还有严格的时间约束。

图 7 是系统的结构框图,共包括七个模块,一个主控系统、一个 DPU,一个四通道双向多路选择器和四个传感器(为描述方便,将四个传感器分别命名为传感器 1 ~4)。传感器 1 通过计数器与 DPU 相连;传感器 2 ~4 及主控系统通过多路选择器与 DPU 交互。多路选择器始终与 DPU 相连并受 DPU 控制,DPU 通过选通不同端口来与主控系统或传感器 2 ~4 通信。DPU 可以处理串口中断和同步中断,同步中断由主控系统发送的同步信号触发。每当有数据到达 DPU 串口时,将触发串口中断。串口中断比同步中断优先级高。

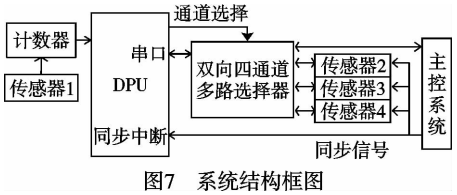

图7　系统结构框图

从传感器 1 读取数据的方法与从其他三个传感器获取数据的方法不一样。传感器 1 与 DPU 之间有内存共享,DPU 可以通过直接读内存的方式读取其数据。但由于传感器 1 会持续写内存,因此在读取之前,需要先锁住内存块然后再去读取(读取之后内存块将自动解锁)。而 DPU 获取其他三个传感器数据时需要通过多路选择器发送取数指令,然后等待传感器准备数据,数据准备好后通过多路选择器读取数据。DPU 系统运行时序如图 8 所示。

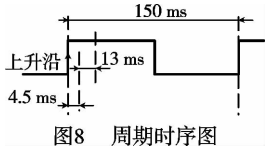

图8　周期时序图

系统的主要功能如下:

a)收集数据。主控系统以 150 ms 为周期发送同步信号,信号上升沿触发 DPU 同步中断,DPU 进入同步中断处理程序,在 4.5 ms 内读取传感器 1 的数据。

b)数据上传。上升沿触发之后 13 ms,主控系统通过多路选择器向 DPU 发送取数指令,多路选择器触发串口中断,在串口中断处理程序中,DPU 需在 13 ms 内完成以下操作:

(a)锁存并读取传感器 1 的数据;

(b)对传感器 1 的数据进行处理(前后两次的值做差);

(c)选通传感器 2 ~4,发送取数指令并等待接收数据;

(d)将所有数据打包后发送回 DPU;

(e)选通主控系统,确保能够接收到下一周期主控系统发送的取数指令。

由于系统运行在太空中,环境比较恶劣,各个部件之间的通信很容易出现信号丢失的现象。对于 DPU 而言,同步中断和主控系统发送的串口中断都有可能丢失。在信号可能丢失的情况下依然可以最大限度地正确处理主控系统的指令,是 DPU 程序最重要的功能。出于容错的考虑,DPU 在同步中断处理程序和串口中断处理程序都会读取传感器 1 的数据。对主控系统而言,最理想的情况是返回前后两次同步中断时取得的数据之差;之后是前后两次串口中断取得的数据之差。如果信号丢失比较严重,前两者都不能保证的话,那在串口中断不丢失的情况下,应该返回后一个周期串口中断时的数据和前一个周期同步中断的数据的差。

对于 DPU 程序,需要满足如下需求:

a)主控系统向 DPU 发送取数指令时,DPU 应该已经选通多路选择器的主控系统端口。

b)能够根据要求的协议与主控系统进行交互。

c)传感器 1 提供锁存和读取两个功能,这两个功能的调用顺序应该符合要求。

d)能够对传感器数据进行正确的处理。

e)能够在 4.5 ms 的时间内对传感器 1 的数据进行处理。

f)能够在 6 ms 的时间内获取传感器 2 ~4 的数据。

g)能够在 13 ms 的时间内完成从接收主控系统的取数指令到发送数据的全过程。

3　DPU 系统建模方法

DPU 系统建模的关键问题是保证模型和系统的一致性。本文使用自顶向下分解和自底向上组合的方式,先根据结构和功能将系统分解成小的独立子模块,然后对这些子模块分别建

立子模型,最后通过定义模型之间的交互和优先级,把子模型重新组合成一个完整的系统模型。为了保证一致性,在分解、建模和组合三个方面都需要遵循一定的原则。

系统的结构分解需要遵循如下原则:

a)在硬件和软件两个层次分别进行划分。从硬件层次来看,系统有七个组件:四个传感器、多路选择器、主控系统和DPU,每个硬件作为一个单独的模块。从软件层次来看,根据程序结构和功能进行划分,除了针对每一个函数定义一个组件以外,还有一个串口组件、调度器组件(用来实现中断和优先级)、共享变量组件。

b)选择恰当的抽象级别。例如传感器 1 实际上是一个微处理器,在硬件上由计数器和传感器两部分构成。但模型里面只关注锁存和读取两个动作,因此使用一个原子组件来描述传感器 1 的行为。

c)根据功能进行分解。若串口通信组件的数据是双向的,从 DPU 的角度来看,组件包含发送数据和接收数据两个独立功能,而且两个功能互补重叠,因此应该分别建立发送和接收两个模块的模型。

对不同类型的子模块进行建模应遵循不同的原则:对软件建模需要读代码,根据代码的逻辑结构抽取模型;对硬件建模需要读硬件的说明书,在此基础上作适当的抽象;对环境建模除了对环境作抽象之外,还应该针对不同的环境建立不同的模型,以满足进一步分析验证的需求;对待检查属性的建模应该保证在非错误状态之外不应该对系统的运行造成任何影响。

子模型的组合是系统分解的逆过程,组合过程需要忠于各个模块之间的交互关系。必要时可以增加辅助子模块的方式来达到这个目标。在 DPU 内部,为了保证功能的独立和结构的清晰,增加了串口通信、数据交互、调度控制三个额外的模型。

根据上面的原则,本文首先对 DPU 系统进行自顶向下的结构划分,将系统从硬件层次划分成如图 9 所示的框图,图中连线表示模块存在交互。与图 7 相比,增加了一个全局时钟模块用于同步和计时,并对传感器 1 作了抽象。进一步对 DPU 模块从功能进行划分,根据代码结构将 DPU 分成主循环、同步中断处理和串口中断处理三个模块。同时,为了保证模型和系统在功能和结构上的一致性,还增加了调度器、串口和同步器三个辅助子模块,最终得到如图 10 所示的框图。

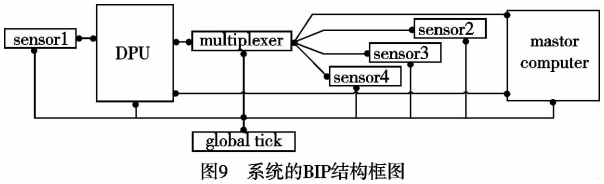

图9 系统的BIP结构框图

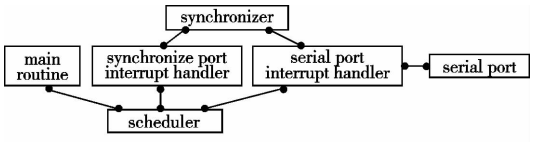

图10 DPU框图

3.1 DPU 模块的建模

对 DPU 模块的建模以汇编代码为基础。首先对 1 800 余行 DPU 代码进行分析,划分模块,画出每个模块的流程图;根据流程图给出模块的 BIP 描述,并适当添加辅助模块,以保持功能的一致性;最后通过定义各个模块之间的交互与优先级得到完整的 DPU 模块。

从代码中划分出来的模块有三个:

a)主循环模块。中断响应程序之外的系统循环自检函数。

b)同步中断处理模块。响应主控系统发送过来的时间同步中断。

c)串口中断处理模块。响应多路选择器发送过来的串口中断。多路选择器在向 DPU 发送数据之前,会发送一个中断信号,触发 DPU 的串口中断响应,并在 DPU 内存中设置标记位,DPU 在执行串口中断处理函数的时候不会响应其他的串口中断。

系统中各个模块之间存在优先关系,这种优先关系可以通过不同模块之间端口两两的优先级描述(如同步中断比串口中断优先级低,则可以定义所有同步中断模块的动作优先级低于所有串口中断模块优先级)。这种描述方式存在的问题是优先级的数量随着端口的数量呈指数级增长。为了解决这个问题,本文引入一个调度器组件,采用双重锁的方法实现优先级,如图 11 所示。以同步中断为例,lockSync、unlockSync 分别与同步中断模块中的进入和退出端口同步,而同步中断模块的其他端口都需要与 freeSync 同步。这样就实现了同步中断可以随时打断主循环、串口中断可以打断同步中断和主循环、串口中断不能被打断的优先级关系,并且将指数增长的描述降低为线性增长的描述。

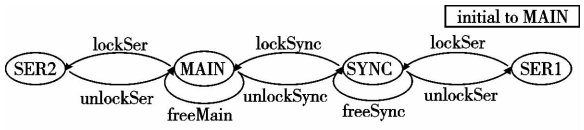

图11 调度器

此外,DPU 程序里面的所有变量都是全局共享变量,可以直接读写内存,而 BIP 不支持组件间的数据共享。有两种方式能够实现数据共享的功能:一种是直接在原有模块内部保存数据,其他模块需要访问时通过连接子进行读取;另一种是增加一个单独的数据共享模块,用于存放共享数据。两种方式各有优劣,前者会对原有模型造成污染;后者增加额外的组件。从保持系统结构的角度出发,本文采用后一种方式,在 DPU 里面增加了一个数据共享模块。数据共享模块将数据以内部变量的方式保存,针对每个变量,对外提供相应的 get 和 set 端口,从而模仿对内存的读取和写入操作。

为了描述方便,还增加了串口中断模块,用于描述 DPU 是否响应多路选择器触发的串口中断。最终,DPU 模块共包含六个子组件:主循环模块、同步中断响应模块、串口中断响应模块、调度模块、数据共享模块和串口中断模块。

3.2 外部环境的建模

对外部硬件设施的建模需要根据硬件说明书分析功能并作抽象,因此需要结合工程师的描述和待验证的需求去对这些设备建模。硬件组件包括传感器 1 ~ 4、多路选择器和主控系统,其中传感器 2 ~ 4 和多路选择器与属性验证关系不大,不作介绍。下面主要介绍传感器 1 和主控系统的建模。

3.2.1 传感器 1 的建模

传感器 1 实际上是一个微处理器,硬件设备比较复杂。由于实验仅关注锁存和读取两个功能,因此根据相关建模原则作简化。根据说明文档:在任何时候 DPU 都可以锁存或者读取

传感器的数据,但是只有在锁存之后再读取才是正确的流程,读取之后数据将自动解锁。为了区别正确的读取和错误的读取,模型需要有两个状态,由此设计了如图 12 所示的自动机模型。模型初始化于 IDLE 状态,可以通过 lock 操作转到 LOCK 状态,表示传感器的数据已经被锁定;为了响应 DPU 的误操作,在 IDLE 和 LOCK 状态下分别可以执行 read 和 lock 操作,执行之后模型状态不变。

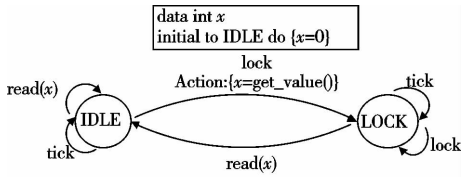

图12　传感器1

3.2.2　主控系统的建模

主控系统在每周期的 0 时刻触发同步中断,45 ms 的时候向多路选择器发送取数指令,然后等待 DPU 的数据,1 500 ms 的时候开始新的周期。据此可以建立一个简单的主控系统模型。

最终,本文建立了一个包含 13 个原子组件和三个复合组件的 BIP 模型,约 1 100 行代码,包括学习 BIP 语言在内,整个建模共花了三周时间。

4　待验证属性的建模

第 3 章描述了对 DPU 系统和外部硬件环境建立统一的 BIP 模型的过程。为了验证属性,还需要对模型进行检测,检查模型是否满足给定的属性,有两种方法:

a)通过在模型中加入打印语句,观察系统执行信息。BIP 模型可以通过 BIP 编译器编译成 C ++ 代码,并与 BIP 引擎代码一起编译成可执行的仿真程序,通过仿真程序的输出信息,可以得到系统运行的轨迹,检查 DPU 程序是否存在问题。

b)将待验证属性建模成带死锁的监控自动机,在原有模型外部观察系统的运行,形成带死锁的 BIP 模型。监控自动机的设计原则是在系统进入错误状态之前不能影响系统的执行。与一般描述原子组件的自动机相比,监控自动机有一些特点:首先需要与待观察的事件进行交互,为了保证不影响原有系统,交互的方式应该是广播,待观测事件作为完备端口参与交互;自动机是对属性的描述,并且应该包含一个错误状态,当属性不满足时进入错误状态,错误状态是一个死锁,从而使得属性一旦不能满足,整个系统都将被锁死。

本文选择方法 b)。对属性建模有两个基本原则:a)监控自动机不应该影响原有系统;b)监控自动机应该能够观察所有需要的事件(即在必要的情况下,应该向外引出原有系统的端口)。

不影响原有系统包括两层含义:监控自动机不应该改变系统的内部状态,即对原有系统内部数据,自动机只能读取,不能更改;自动机不应该阻塞原有系统的执行,一般情况下,自动机应该通过接收广播的方式观察事件,对于那些需要同步的事件,则需要确保对应的端口在错误状态之外的所有状态都是无条件使能的。如图 13 所示,自动机监控的 lock 和 read 事件通过接收广播的形式与系统相连;tick 是用于时钟同步的,需要在错误状态以外的所有状态添加无条件的自环迁移并带上 tick 标签,以保证在正常状态下监控自动机不会阻塞系统的时钟。

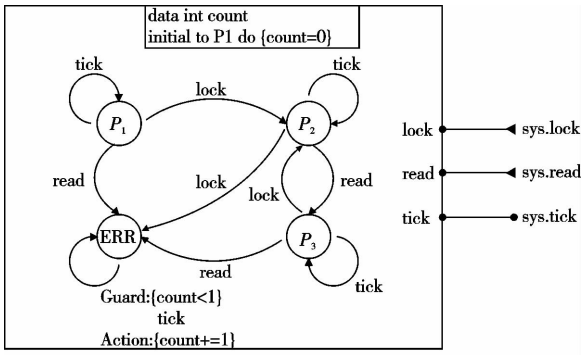

图13　读写顺序检测

DPU 程序在理想的条件下运行是永远不会出错的,实际场景下主控系统和 DPU 之间的通信可能出现信号(同步中断或取数指令)丢失。为了检测可能的错误,本文根据不同的外界环境建立了三个不同的主控系统模型,用于覆盖各种可能出现的情况:

a)主控系统 1,即 3.2.2 节的模型。这是一个完美的模型,信号不会丢失。

b)主控系统 2。模型在主控系统 1 的基础上进行修改,使之能够再现信号丢失的情况,这是真实环境下的系统。

c)主控系统 3。主控系统 2 包含一种无法处理的情况,即同一周期里主控系统发出去的同步中断和取数指令都丢失了,这种情况下属性的验证毫无意义。因此,实验设计主控系统 3,在真实场景的基础上稍作改进,用内部变量作约束,确保同一周期内两个信号不会同时丢失。图 14 是这一模型的自动机描述,注意到从 INIT 到 SYNC 的两个迁移的约束条件不互斥,当两个约束条件同时满足(count ==0 && signal)时,两个迁移都能进行,这种情况下 BIP 引擎会随机挑选一个迁移执行,从而实现了不确定性。

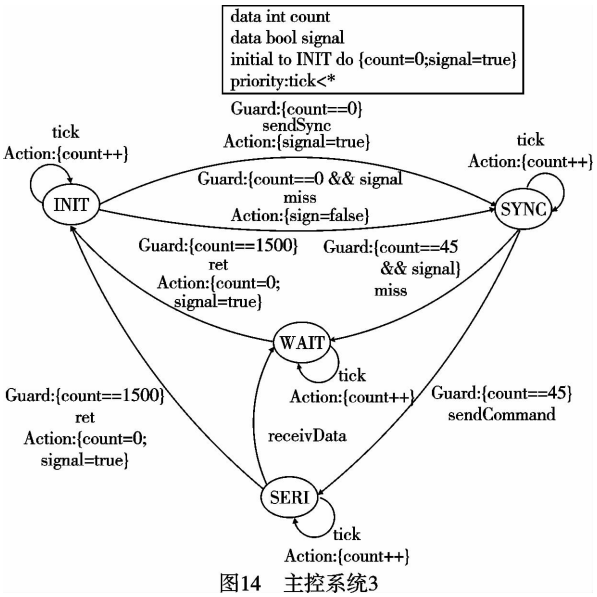

图14　主控系统3

需要验证的属性一共有五条:

a)对传感器 1 的操作应该符合规范。不能不锁存直接读取,也不能连续两次锁存或者读取。

b)返回到主控系统传感器 1 的值应该符合需求。优先返回前后两次同步中断函数获得的值之差;不能满足的情况下,应该返回前后两次串口中断函数获得的值之差;最差的情况下,应该返回后一次串口中断函数获得的值和前一次同步中断

函数获得的值之差。

c)同步中断处理模块消耗的时间在规定范围之内。

d)串口中断处理函数消耗的时间在规定范围之内。

e)从主控系统发出一次同步中断信号到接收到 DPU 返回数据之间的时间间隔在规定范围之内。

下面以属性 1 为例进行说明。属性 1 是对传感器 1 的断言。需要观测的事件是传感器 1 的锁存(lock)和读取(read)。在三种情况下属性不满足:最开始时没有锁存就读取;连续读取两次;连续锁存两次。据此,可以用自动机来描述该属性。图 13 是监控自动机模型。锁存和读取事件通过接收广播的方式监控。时钟动作 tick 与系统同步,在正常状态下无条件使能,错误状态下监控自动机的 tick 只能执行一次,从而锁死系统时钟。

5 属性的模拟验证

首先通过仿真的方式验证结果,利用 BIP 编译器将整个 BIP 语言描述的 DPU 系统和监控自动机一起编译成 C ++ 代码,再用 Linux 下的 G ++ 编译器将这些代码连同 BIP 引擎代码一起编译成一个可执行的仿真程序。

实验针对三种不同的主控系统模型分别进行仿真测试。为了保证测试能覆盖尽可能多的情况,对监控自动机作了修改:当监控自动机检查到错误状态时,不跳到死锁状态,而是输出错误日志,然后正常跳转继续监控。对每个主控系统分别进行 10 min 的仿真测试,程序大约运行 6 700 个周期。运行完成以后,根据日志查找错误并分析原因。表 1 是测试结果。

表 1 仿真结果

属性	主控系统 1	主控系统 2	主控系统 3
1	√	√	√
2	√	×	×
3	√	√	√
4	√	√	√
5	√	√	√
时间/min	10	10	10
周期	6 627	6 627	6 900

通过分析输出运行轨迹,发现了原 DPU 代码中关于待验证属性 2 的两个错误:

a)如图 15 所示,如果最近两个周期中,前一个周期的同步信号(*S*3)丢失,其他信号正常,DPU 程序认为本周期信号正常,应该用前后两次同步中断程序获取的值做差作为结果返回,即(*S*5 - *S*3),但由于 *S*3 时刻的信号已经丢失,因此内存中的 *S*3 实际上是 *S*1 的值,亦即 DPU 返回的是(*S*5 - *S*1),是隔了两个周期的差值。根据属性需求,正确的做法应该是返回前后两次串口中断程序获取的值之差,即(*S*6 - *S*4)。错误的原因是源程序仅用一个状态位描述同步中断信号,而没有保留上一周期的信息。解决的方法是在代码中增加一个状态位描述上一周期同步中断信号。

b)另一个错误则出现在前一周期的串口中断信号和本周期的同步中断信号同时丢失的情况下。期望的返回值应该是本周期的串口中断程序获取的值与前一周期的同步中断程序获取的值的差,但由于连续两个中断信号丢失,而 DPU 系统缺乏计时机制,对此无法作出判断,因此 DPU 程序会误认为根本没有信号丢失。这是一个程序无法修复的错误(对应图 16 中,如果 *S*4 和 *S*5 同时丢失,那么 DPU 认为信号没有丢失,因此返回的是 *S*3 - S1,但实际应该返回的是 *S*6 - *S*3)。

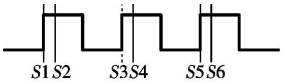

图15 错误1

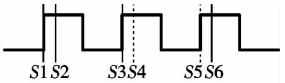

图16 错误2

6 结束语

DPU 是嵌入式系统中的一个典型组件,被广泛应用于太空领域。DPU 在层次化的系统框架中位于中间位置,负责接收上层主控系统发送过来的指令,根据指令从下层传感器获取数据并作处理,最后将处理结果发送回主控系统。

DPU 的正常工作对整个太空单位非常重要。一般而言,从三个方面判断 DPU 是否正常工作:操作是否能在指定时间内完成;操作是否按照指定的协议执行;从 DPU 返回的数据是否满足需求。

本文以一个实际太空应用中的 DPU 系统为例进行研究。使用 BIP 框架对 DPU 系统进行建模以及仿真测试。首先采用自顶向下的方法对 DPU 系统进行层次结构的划分,对子模块分别建立扩展的自动机模型;然后将描述待验证属性的自动机与各个子模块层层组合成完整可验证的 BIP 模型;最终使用 BIP 工具生成仿真程序,通过仿真程序的执行,检查出 DPU 代码中存在的问题。

从建模过程中学到了一个重要的建模原则:模型应该忠于系统,但是不需要体现所有的细节。以串口为例,最开始,尝试搭建一个完全忠于硬件结构的串口模型,随后发现模型过于复杂,且没有体现 DPU 系统的重点。因此根据串口模块的说明书及实际的需求,搭建了一个简单可用的串口模型。

实验介绍的对 DPU 系统进行建模和验证的方法是对现有仿真与测试方法的一个补充和扩展。

目前针对本案例的工作在两个方向延伸:一方面笔者尝试使用更好的验证技术和工具对 BIP 模型进行验证;另一方面笔者将基于 BIP 进行更多的案例研究。

参考文献:

[1] 伏洪勇,林宝军,陈福恩. 星载数据处理单元的可靠性研究[J]. 军民两用技术与产品,2008(2):37-39.

[2] 范进,石雅缪,金声震,等. 基于软件无线电结构的雷达数据处理单元设计[J]. 核电子学与探测技术,2010,30(1):88-92.

[3] 张建泉,席隆. 数据处理单元仿真测试系统的设计与实现[J]. 电子测量技术,2006,29(2):54-55.

[4] The BIP toolset[M]. [S. l.]:Verimag, 2011.

[5] SIFAKIS J, BASU A, BOZGA M. Component-based construction of heterogeneous real-time systems in BIP[C]// Proc of the 30th International Conference on Applications and Theory of Petri Nets. Paris: Springer-Verlag, 2009.

[6] BASU A, BOZGA M, SIFAKIS J. Component-based construction of real-time systems in Bip[C]//Proc of the 21st International Conference on Computer Aided Verification. 2009: 33-34.

[7] CHKOUR M Y, ROBERT A, BOZGA M, *et al*. Translating AADL into BIP-application to the verification of real-time systems[M]//Models in Software Engineering. Berlin: Springer-Verlag, 2009: 5-19.

[8] BOZGA M, SFYRLA V, SIFAKIS J. Modeling synchronous systems in BIP[C]//Proc of the 7th ACM International Conference on Embedded Software. New York: ACM Press, 2009: 77-86.