基于 SystemC 的嵌入式系统设计的描述模型*)

栾 静[1,2] 顾君忠[1]
(华东师范大学计算机系计算机应用研究所 上海 200062)[1]
(新疆师范大学数理信息学院 乌鲁木齐 830054)[2]

摘 要 系统建模是嵌入式系统设计的关键步骤,其好坏直接影响着设计的质量和产品的上市时间。已有多种建模方案,但每种都有其局限性。本文提出一种由5层模型组成的嵌入式系统设计的模型框架。它能够从需求描述开始,建立CDM功能模型,经一致性验证,满足设计要求后映射到SystemC抽象模型上。利用SystemC的硬件描述特征和仿真库,增加设计细节,分层细化模型并进行验证,最后达到软硬件的协同设计和综合实现的目的。

关键词 嵌入式系统,分层抽象模型,SystemC

Specified Model of Embedded System Design Based on SystemC

LUAN Jing[1,2] GU Jun-Zhong[1]
(ICA, Computer Department, East China Normal University, Shanghai 200062)[1]
(School of Math-physics and Information Science, Xinjiang Normal University, Urumqi 830054)[2]

Abstract System modeling is an essential design in embedded system. It will directly affect design quality and time on market. Nowadays, there are some modeling solutions, but each of them has its limits. In this paper a new modeling frame of embedded system design is presented that consists of 5 level abstraction models. The suggested solutions make up of specification the product's requirements, establishing CDM model, consistently verifying these constructed components to satisfy the requirements of design, and mapping them into the SystemC abstraction models. With SystemC hardware properties and simulation kernel, the models can be gradually refined and verified by adding details. The suggested approaches get the aim of the hardware-software co-design and synthesis implementation.

Keywords Embedded system, Hierarchical abstract models, SystemC

1 引言

随着嵌入式系统在各个领域的广泛应用,系统设计变得越来越复杂,且涉及多个学科。研究嵌入式系统从需求描述、设计、测试到编码的整个设计过程的模型系统及建模方法变得越来越重要。面对日益激烈的市场竞争,系统设计周期的长短成为产品占领市场的一个重要因素。所以提高设计层次,为系统建立相应的多层模型,同时实现组件的可重用,成为亟待解决的问题。在这一过程中,系统建模是关键步骤。目前已有多种建模方法,但仔细研究发现,这些建模方法都有其局限性,因而只能实现系统的部分行为。如:UML建模方法,方便进行需求描述、系统功能设计和行为描述,但UML不能描述系统硬件,无法对模型进行验证。再如VHDL和Verilog硬件描述语言,能很好地反映硬件设计特征,但在设计细化阶段,用C或C++描述的系统结构和功能特性必须转换为VHDL和Verilog,从而造成系统设计与软硬件设计之间的间隙,使系统软硬件综合变得复杂,延长研发周期。

SystemC[1]硬件建模语言,是在C++的基础上增加描述硬件的类库,不仅有支持硬件操作的信号、时序和接口,在不同抽象层次上对设计进行建模,而且还带有仿真内核,容易搭建仿真平台,对设计各阶段进行验证。SystemC已经成为事实上的系统级设计语言的标准,但是它缺乏对系统需求进行规范描述。因此为了能够较全面地进行系统设计,有必要建立一种从需求描述到功能实现的设计环境。本文提出了一个基于SystemC的嵌入式系统的模型设计方案,旨在构建一个设计与测试平台,能够从需求描述开始,对设计功能、性能和约束进行描述,自动生成CDM(CoDesign Model)模型[6],在验证其合理性后,自动映射到SystemC建模语言中,进行不同抽象层次的设计细化。SystemC能够使用单一语言形式设计软硬件,克服了多种语言描述系统软硬件带来的不一致性,并在多个抽象层次上完成建模和验证,可缩短设计时间和提高设计质量。

2 系统设计方法及分层模型描述

2.1 系统设计方法

嵌入式系统的设计方法主要是基于两种基本原理:形式化规范(formal specification)和设计重用(design reuse)。形式化规范是对系统进行全面的分析,对内在的规律进行抽象,用数学的方法描述模型的结构。如Petri Net,FSM都是这种形式。设计重用主要是为了缓解片上系统复杂度的不断增加与产品加快上市时间的压力之间的矛盾,设计领域迫切要求设计过程转移到高层次的抽象上,并在设计早期对产品进行验证,重用一些设计模块。某种形式的知识产权IP(intellectual property)复用机制已经成为SoC设计策略的一个有机组成部分。目前系统级设计方法大致有如下三类,不同程度地使用了设计重用技术。

*)基金项目:上海市科技发展基金项目(No. 025007014)。栾 静 副教授,博士研究生;顾君忠 教授,博士生导师。

• 自顶向下的系统级综合方法：从系统功能描述开始，根据系统的行为生成系统构造，通过不断精化细节逐渐达到RTL/ISS实现模型。如文[2]使用的就是系统级综合方法。

• 基于平台的设计方法：把系统的行为映射到预定义的系统构造上，而不直接从系统行为产生系统构造。文[3]采用这种方法。

• 自底向上的基于组件的设计方法：该方法聚焦点是组件重用。对已存在的不同层次的计算或通信组件被装配和包装，生成预定义的平台。如TIMA[4]实验室所采用的方法。

上述方法不能完成从需求描述到系统实现的整个设计过程，因为每种方法仅侧重于设计过程的某些部分。本文开发了以需求驱动的嵌入式系统软硬件协同设计与测试平台iCDMdt (Integrated CDM Design & Test)，其模型结构如图1所示。图中椭圆框表示模型，矩形框表示操作。箭头表示验证和精化方向。该平台能完成从需求描述到用SystemC描述的RTL级模型的实现，并在每层上进行验证，从而提高系统设计效率。

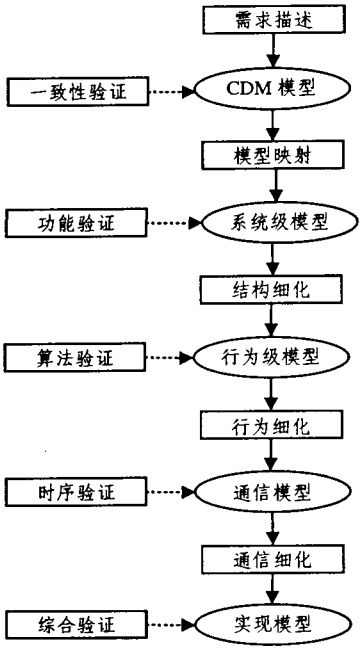


图1 系统抽象模型及细化

2.2 抽象模型描述

在分层抽象模型中，需求描述不依赖于具体的模型，它根据IEEE 830标准规范，对设计系统的综合信息、功能要求、性能要求和约束条件进行全面的描述，并自动生成设计需求说明书和CDM模型。CDM模型是德国Darmstadt大学VLSI系统实验室提出的一个嵌入式系统软硬件协同设计的模型，该模型具有良好的形式化定义，用它设计产品的系统级功能和各个模块，修改灵活。CDM模型不依赖于软硬件描述语言，因而不能具体到实现级。考虑到设计的功能细化和约束条件，CDM模型必须转化为能用软硬件设计语言描述的模型。通过对各类软硬件描述语言的比较，本文将CDM模型转化成SystemC系统级描述模型，并自动生成可执行的系统级代码，该代码能反映模块间执行的时间因果关系。对系统级模型细化，对功能行为加以描述，就形成行为级模型。行为级模型表示了系统的结构特征，模块间通信由抽象的通道来完成。将行为模型中抽象的通信语义精化，用实际的通信协议来代替，就形成了通信模型。最后是系统实现模型，它完成软件、硬件和通信接口的综合，是精确周期的计算模型。

从系统级到实现级模型是在SystemC环境下的逐步精化的过程，上层模型增加一定的设计细节，就可达到下层模型。利用SystemC的仿真内核，验证平台可同时生成。验证平台和设计模型结合，可进行不同抽象层次模型的测试验证，并可修改不适应的部分，最后用可执行的程序代码实现系统设计。

3 抽象模型的实现

该部分主要从模型的建模能力(计算模型)、模块间的数据传输和反映模块执行序列的调度等方面进一步描述各层模型的特征。

3.1 CDM模型

CDM模型是一个有向图，如图2表示。图中的每个结点(Process)表示处理模块，代表系统的一个功能行为，结点之间的有向边代表处理之间的通信，边上传输的信息用令牌(Token)表示，它是处理发生的条件。

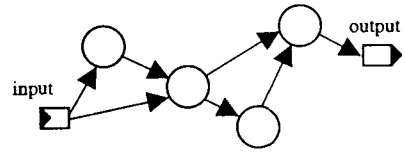


图2 CDM模型

定义1 CDM模型是一个二元组 $G=(P,E)$，其中：$P=P_p \cup P_I \cup P_o$，$E=\{e \mid e=(p_i, p_j) \wedge p_i \neq p_j\}$，$P_p$：操作处理的集合，$P_I=\{p \mid \neg\exists(p_i, p) \in E\}$，$P_o=\{p \mid \neg\exists(p, p_i) \in E\}$，$G$中的一个令牌可定义为三元组 $m=(e, t, C)$ 其中 $e \in E$ 和 $t \in R^+$，R^+ 表示时间点 t 的集合，C 表示令牌存在的一个条件。m 表示在时刻 t，当 C 中的所有条件都得到满足时，边 e 上有令牌产生。

计算：每个处理模块由一个核心程序(Core)和壳(Shell)组成。Core通过一些子处理完成模块的计算功能。Shell监视输入/输出的参数，对内一旦需要的令牌全部到达，就激活相应的子处理；对外将处理后的输出令牌送到后续模块中。Shell能独立地执行，通过接口与核心程序或其它模块进行通信。

数据传输：模块间传输的信息抽象成令牌(Token)，不区分数据/控制流，仅表示信息传送，且传输信息需要一定的延迟时间。

调度规则：使用ASAP(As Soon As possible)调度方法，并假定资源是无限制的。

CDM模型根据自定义类库和一定的映射规则，在iCDMdt环境中由模型映射编译器自动将其转化为SystemC支持的系统级可执行代码，形成系统级模型。

3.2 系统级模型的实现

系统级模型是一个无时间要求的功能模型，数据通过模块的端口Port以Signal的形式传输，没有引入通道的概念。使用Port和Signal，可以隔离模块，使各模块可以由不同人员并行设计。如图3(a)所示。

计算：SystemC提供的基本计算单元有：

• Module：划分设计功能的构造实体。有两种基本形式：基本模块，它只包含进程、端口、函数和构造器；复合模块：

除基本模块的内容外，还可包含其它模块，允许表达分层模块构造。

• Process：是 SystemC 中基本行为单元，是在并发环境下的执行过程，能包含函数调用，但不能调用其它的进程，即进程没有分层结构。

• Function：类似于 C 语言中函数的定义，是特定功能的静态描述。

数据传输：通过模块中的端口(Port)和信号(Signal)来传输数据。端口是硬件信号机制，具有延迟更新的特点；变量是通过赋值语句得到，具有立即更新的特点，故变量不能通过端口传输。

执行序列：由于 SystemC 是在并发环境下执行，只支持动态调度规则。在系统级只涉及到静态敏感表的触发执行。静态敏感表在构造器中声明，是触发进程重新执行或活动的静态信号集。当信号集中任一信号的值发生变化时，进程就开始或重新执行。

3.3 行为级模型的处理

主要任务是在上层模型的基础上对计算单元进行详细描述和通道绑定。计算单元由一个或多个进程组成，是系统行为的描述，产生系统的构造模型。进程之间通过信号来传输数据，不同模块的信号端口必须与通道(Channel)绑定，通道表示数据的传输和并发进程之间的同步。行为级模块使计算单元与通信分离，使各个组件可以独立设计，并能最大限度地重用计算和通信组件。

计算：系统行为被映射到 method 和 thread 等类型的进程或类似于 C 的函数上，进程可由静态敏感表中的信号或动态事件触发，并发执行以实现相应的处理。函数可由进程或其它函数调用来执行。

通道数据传输：通道可以有多种类型的选择，若是模块内的进程间共享数据变量，则可以使用 mutex 通道，使各进程互斥访问；若模块内的进程间仅传输数据，则可直接用信号作为触发后续进程的事件；若是不同模块的进程间通信，总线协议可简化成阻塞型或非阻塞型读写访问 FIFO 通道。模型处理如图 3(b)所示。通道必须用接口与模块连接。

调度：当进程的静态敏感表中的信号发生变化时，立即执行相应的进程。对动态敏感表，它使用 event-wait-notify 来调度进程，当进程等待的事件被通告，进程就开始或重新执行。在进程执行过程中，更新输出信号，直到再没有信号或事件发生变化为止。

3.4 通信模型的细化

通信模型是在行为模型的基础上对通信组件进行细化。细化过程由两步组成：在总线功能级选择协议，将通信功能内嵌到实现组件的行为上，且与组件的功能相匹配。这一步需要 Master-Slave 通信库[8]的支持。

计算：进程之间通过内嵌的通信机制，将复杂的系统模型利用顺序通信的功能模块之间的互联来构建，屏蔽无用的实现细节，保留功能级定义的通信和执行顺序。

协议选择：有三类预定义的总线协议，非握手(no-handshake)协议，没有附加条件，只传输 Data 信息；握手许可(enable-handshake)协议，有 Data、Enable 和 Ready 信号；全握手(full-handshake)协议，有 Data、Req 和 Ack 等信号。

通道细化：抽象的通信通道被总线协议实例化后，就形成了总线协议通道。因为通道插入总线协议后端口已经改变，必须在模块和协议通道间插入适配器来完成接口之间的转换，如图 3(c)所示，即利用协议通道提供的协议接口来实现模块间的数据通信。最后适配器要被内嵌到实现行为的组件中，使抽象端口消失，模块通过相应端口连接到总线线路上。图 3(d)就是协议内连后的模型。

数据传输：模块间数据传输由上层绑定到通道的抽象数据转换成具体的 Req、Ack 和 Data 数据(全握手协议)，是精确周期的通信。模块内各事务间信息传输是与时间密切相关的，是在一定触发条件下完成的操作。

执行顺序：将并发的事务执行转换成可串行化执行的序列，经过通道的互连，实行串行执行。

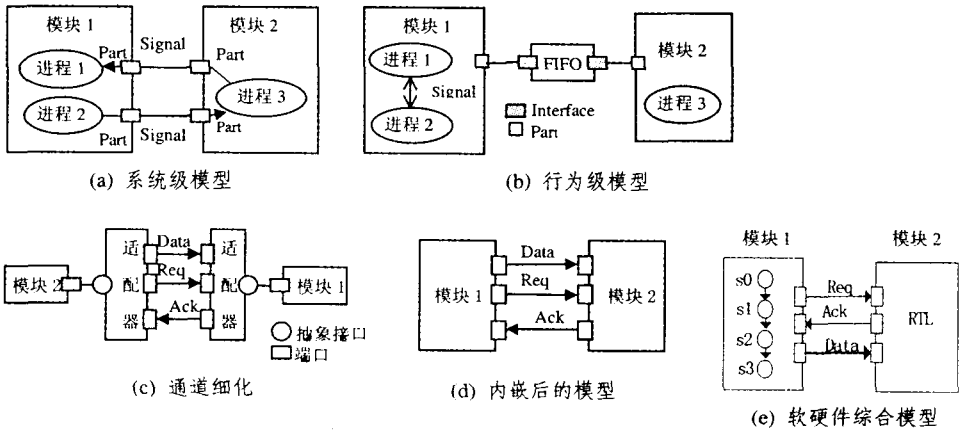
3.5 实现模型

主要任务有三个：将最底层的寄存器传输级的行为映射到定制的硬件组件上；将进程、函数等映射到可编程的处理器上，将硬件一端的总线接口和软件一端的总线驱动进行通信综合。这实质上就是要求模块内的每一步操作都受到时钟的严格控制，是一个时间精确的软硬件综合过程[5,7]。

计算：采用特殊的线程 clocked thread 实现，为了与硬件的行为相符合，它只对时钟的上升沿或者下降沿敏感。硬件综合成 RTL，软件细化到 CPU 可执行的指令。

数据传输：数据和控制信号已经与系统时钟、内存和 SOC 的引脚建立关联，是精确时钟/周期的数据传输模式。

执行序列：CPU 根据核心功能模块设定的调度策略执行。模型如图 3(e)所示。



(a) 系统级模型　(b) 行为级模型

(c) 通道细化　(d) 内嵌后的模型　(e) 软硬件综合模型

图 3　分层抽象模型的实现

表1是不同抽象模型在通信模式、计算时间和通信时间等方面的特性比较，由此可清楚知道各层模型的设计特点。

表1 不同抽象模型的特性

抽象模型	通信时间	计算时间	通信模式
CDM模型	Approximate	Approximate	Token
系统级模型	No	No	Port, Signal
行为级模型	No	Approximate	Event, channel
通信模型	Cycle accurate	Approximate	detailed bus chinnel
实现模型	Cycle accurate	Cycle accurate	wire

总结 在现有可供选择的嵌入式系统设计方案中，仍存在对同一任务，不同阶段使用不同的语言描述和工具进行的混合设计方法。在增大系统综合难度的同时，也容易造成系统的不一致。本文提出一个基于5层模型的iCDMdt设计平台，支持嵌入式系统数字产品的高层设计，能进行需求描述、功能建模、行为细化和通道细化，并在设计的每一抽象层次上进行验证。系统设计使用统一的软硬件描述语言，方便了软硬件的综合。利用组件和计算模型，通过逐步精化细节，有根据地调整控制策略后，实现从需求描述到真实系统的优化设计。利用该模型平台进行设计，可提高设计效率，并在多层验证基础上提高设计质量。

参考文献

1 System C. Functional Specification for SystemC2.0. [EB/OL] http://www.systemc.org, 2001
2 Keutzer K, et al. System level design: Orthogonalization of concerns and platform-based design. IEEE Transactions on Computer-Aided Design, Dec. 2000
3 Paulin P, et al. Stepnp: A system-level exploration platform for network processors. In IEEE Design and Test of Computers, Nov-Dec 2002
4 Cesario W, et al. Multiprocessor soc platforms: a component-based design approach. In IEEE Design and Test of Computers, Nov-Dec 2002.
5 Cai L, et al. Comparison of SpecC and SystemC Languages for System Design: [Technical Report CECS-TR-03-11]. UCI, May 2003
6 Boßung W, Huss S A, Klaus S. High-Level Embedded System Specifications Based on Process Activation Conditions. Journal of VLSI Signal Processing, Special Issue on System Design, Kluwer Academic Publishers, 1999, 21(3): 277～291
7 Cai Lukai, Gajski D. Transaction Level Modeling in System Level Design. CECS Technical Report 03-10 Mar 28, 2003
8 陈曦，徐宁仪. SystemC片上系统设计. 科学出版社，2004

（上接第74页）

务的时间。W_P 的分布函数为：

$$W_P(t)=1-\frac{\lambda_i}{\mu_i}e^{-\mu_i(1-\rho_i)t},t>0 \tag{6}$$

平均等待时间为：

$$\overline{W_P}=\frac{\lambda_i/\mu_i}{\mu_i(1-\lambda_i/\mu_i)}=\frac{\lambda_i T_i}{\mu_i-\lambda_i} \tag{7}$$

则子事务在LDBObject的平均停留时间为：

$$\overline{W_L}=\overline{W_p}+T_i=\frac{\lambda_i T_i}{\mu_i-\lambda_i}+T_i=\frac{1}{\mu_i-\lambda_i} \tag{8}$$

由于事务 T 经过GDBAgent后被划分为 m 个子事务，而这 m 个子事务又是根据冲突类原则排成 r 个队列的，因此在每个LDBObject处的子事务个数 $0\leqslant k_i\leqslant m$ 不同，所以事务 T 在LDBObject处的停留时间为：

$$\overline{W_M}=\max\{k_i\ \overline{W_L},i=1,2,3,\cdots,r\} \tag{9}$$

所以一个事务 T 从到达到离开服务系统共用时间为：

$$W=\overline{W_G}+\overline{W_M}=\frac{1}{\mu-\lambda}+\max\{k_i\ \overline{W_L},i=1,2,3,\cdots,r\} \tag{10}$$

5 模型分析

从以上结论可以看出，当GDBAgent在单位时间内处理的事务越多即 μ 越大、单位时间到达的事务数 λ 越小，在GDBAgent处停留的时间 $\overline{W_G}$ 越短。理想状态是一旦事务到达GDBAgent马上获得服务，不需要排队等待，这就要求系统为每个事务分配一个GDBAgent，但这在实际系统中是不现实的，所以整个系统中需要GDBAgent的数目是根据实际系统而定。事务到达后根据所有GDBAgent队列中事务个数，排在当前事务最少的GDBAgent队列中，如果整个系统中的GDBAgent排队事务都足够多时，可以申请系统启动一个新的GDBAgent。

在GDBAgent将事务分解为多个子事务时，这些子事务分别排成多个队列，各个队列中的事务数目 k_i 是不相同的，而事务在二重队列中的逗留时间是所有队列中处理子事务所需要的时间与子事务数目的积的最大值，即 $\overline{W_M}=\max\{k_i\ \overline{W_L},i=1,2,3,\cdots,r\}$。所以在事务分解时，将各个子事务均匀分布在各个LDBObject队列中，且各个LDBObject队列中事务个数最少时 $\overline{W_M}$ 最小。因此，一个全局事务分解后是否平均分布到各个局部数据库对象上将直接影响整个事务处理的效率，理想状态是各个子事务均匀分布到不同的局部数据库对象。同时为了提高整个系统工作效率，可以将多个全局事务请求并发执行，使得各个LDBObject不会出现空闲状态。但这同时又增加了全局事务处理的难度，这个任务只能由事务执行器来完成。

结束语 本文介绍的分布式数据系统基于对象服务的思想设计，将全局事务分解为多个子事务分散到各个局部数据库对象上同时进行，提高了并发执行的速度，并且所有数据库都被抽象为对象，可以实现跨平台异构分布式数据库系统构建。通过对系统的排队模型分析得出了影响系统执行效率的主要因素。所以在实际系统中可以根据需要增加GDBAgent的个数，提高GDBAgent的工作效率，事务执行归类，提高局部数据库对象执行速度等途径来提高系统效率。根据该方案设计和实现的数据库系统已经成功应用到某分布式电力监控系统中。

参考文献

1 [美] Henning M, Vinoski S. Advanced CORBA Programming with C++[M]. 北京：清华大学出版社，2000
2 Silberschatz A. Database System Concepts(Fourth Edition)[M]. 北京：机械工业出版社，2003
3 舒后. 分布式数据库系统的并发控制方法讨论[J]. 微型机与应用，2002，12：14～16
4 田俊峰，王凤先. 一种基于C/S模式的分布式数据库系统及其排队模型[J]. 计算机工程与科学，2002.24(2)：88～91