异步FIFO的模型检验方法

罗 莉 欧国东 刘 彬 徐炜遐 窦 强
(国防科技大学计算机学院 长沙 410073)

摘 要 跨时钟域(Clock Domain Crossing,CDC)设计和验证是SOC系统芯片设计的关键问题。讨论了异步FIFO的模型检验方法,利用模型检验工具SMV,建立了异步FIFO的有限状态机模型,使用时序逻辑LTL对该模型和属性进行了描述和验证。实验结果达到要求,同时表明该方法是行之有效的。与传统的模拟和仿真等验证方法相比较,模型检验具有能够自动进行、验证速度快、不用书写测试激励等优点。

关键词 CDC(Clock Domain Crossing),异步FIFO,LTL,符号模型检验,SMV
中图法分类号 TN402 **文献标识码** A

Symbolic Model Checking of Asynchronous FIFO

LUO Li OU Guo-dong LIU Bin XU Wei-xia DOU Qiang
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract Clock domain crossing(CDC) is an important issue in SoC design and verification. We presented the symbolic model checking of asynchronous FIFO, proposed a finite state machine to model the asynchronous FIFO, then, used SMV to analyze and check its specification described by linear temporal logic. Result shows the design is correct and the method is effective. Compared with simulation and emulation, model checking can save time, run automatically, and does not need test bench.

Keywords CDC(Clock Domain Crossing), Asynchronous FIFO, Linear temporal logic, Symbolic model checking, SMV

1 引言

跨时钟域(Clock Domain Crossing,CDC)的数据传输电路被称为跨时钟设计,其功能的正确性是SoC(System on Chip,SoC)系统芯片各跨时钟域部件间正确传输数据的基础。为了满足高性能和低功耗需求,SoC系统芯片的多个部件通常工作在不同频率的异步时钟域中,如众核高性能处理器、高清视频处理器、基带通讯模块等多个IP芯片,全局异步、局部同步的NOC芯片都用到多个时钟域和跨时钟设计。

信号从一个时钟域传输到另外一个时钟域会产生多种问题,比如亚稳态、数据丢失、数据不一致等问题。对跨时钟域设计进行功能验证是SoC系统芯片验证工作中的难点,亚稳态等现象很难在RTL验证流程中体现出来,但如果将跨时钟域设计的功能错误遗留到FPGA验证阶段甚至流片后才被发现,则会严重影响产品的上市时间。

CDC设计的验证方法主要有模拟验证和形式化检验两种。模拟验证很直观,但是编写测试激励、追溯错误非常耗时,而且模拟和静态时序分析都不能完全验证数据被正确传输。为了克服模拟方法的不足,设计者求助于各种形式化验证方法,如定理证明、等价性检验和模型检验等方法。形式化验证方法就是根据逻辑设计的结构和功能描述,用类似数学的方法或定理证明的方法来证明逻辑电路设计的正确性。

CDC设计通常采用异步FIFO进行跨时钟域的数据传输,而且异步FIFO中包括了同步器、格雷码和二进制编码转换逻辑、指针递增逻辑、空满信号产生逻辑等CDC设计中典型的电路结构,所以本文以异步FIFO为例,讨论跨时钟域设计的设计规范描述和模型检验方法,利用模型检验工具SMV,建立一个异步FIFO的有限状态机模型,使用时序逻辑LTL对该模型和属性进行描述和验证,实验结果达到要求,同时也表明利用符号化模型检验方法验证跨时钟域设计是行之有效的,达到了预期目标。

2 CDC设计和验证

不正确的CDC设计可能产生3种情况,一种是产生亚稳态(Metastability),该路径终点寄存器的建立或保持时间违例(Setup/Hold Timing Violation),从而引起该寄存器的输出端进入亚稳定状态;另一种是数据保持时间(hold time)不足,产生数据丢失,例如来自一个快时钟域的信号不能够被慢时钟域的时钟采样到;最后一种是数据聚合(Re-convergence),一组被同步的控制信号再次汇聚时将丢失或重复采样。

如果是单根控制信号,最简单的方法是采用两级同步器,问题2和3可采用异步FIFO解决,异步FIFO中包括了同步

到稿日期:2011-04-12 返修日期:2011-07-26 本文受核高基重大专项(2011ZX01028-001-001)资助。

罗 莉(1971-),女,博士,副研究员,主要研究方向为计算机体系结构等,E-mail:luoli_hi@sina.com;欧国东(1977-),男,博士,讲师,主要研究方向为高性能微处理器体系结构设计;刘 彬(1982-),男,硕士生,主要研究方向为计算机体系结构;徐炜遐(1963-),男,硕士,研究员,主要研究方向为计算机体系结构等;窦 强(1972-),博士,研究员,主要研究方向为计算机体系结构等。

器、格雷码和二进制编码转换逻辑、指针递增逻辑、空满信号产生逻辑等，我们重点分析和阐述异步 FIFO 的形式化验证方法。

CDC 设计和验证的相关工作中，文献[1]Sarwaryt 和 Verma 主要分析了基于 enable 的同步器的设计和验证。文献[2]提出了 CDC 的覆盖率，利用时序数据流图传输 CDC 错误到可观察变量，定义了 CDC 电路特性的覆盖率模型。文献[3]提出用有限状态机形式化验证 CDC 设计，为缓解模型检验的空间爆炸问题，提出两种优化策略。文献[4]说明怎样去综合断言逻辑，相比后端 CDC 设计的形式化验证时间消耗较少。文献[5]说明的是商业工具 spyglass，这是被业界广泛采用的 CDC 验证工具，该工具提供了模板匹配的方法，但没有考虑到电路特性描述的完整性问题。文献[6,7]提出了采用断言进行 CDC 设计的验证。

我们设计的异步 FIFO 信号转换逻辑图如图 1 所示，异步 FIFO 的时钟是写时钟和读时钟，在写时钟域，写数据由写信号使能，读信号控制数据从 FIFO 流出。读写指针采用 Gray 编码同步到不同的时钟域。

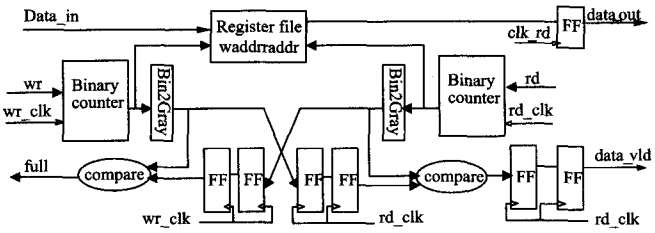

图 1　异步 FIFO 的同步信号转换逻辑图

3　异步 FIFO 的模型检验

3.1　符号模型检验工具 SMV

模型检验的基本思想是使用时序逻辑形式化描述设计规范，利用有限状态机表示电路实现的状态及状态间的迁移关系，使用二叉判定图(Binary Decision Diagram，BDD)表示上述状态机，通过遍历 BDD 来检验电路实现是否符合设计规范，如果不符合则给出反例。

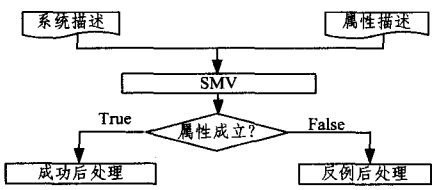

图 2　SMV 工作原理

SMV(Symbolic Model Verifier)是卡内基——梅隆大学的 McMillan 博士开发的符号化模型检测工具，目前，SMV 已成为非常流行的有限状态系统验证工具。SMV 的工作原理：首先用 SMV 规范语言建立各个模块的有限自动机模型和系统全局状态的 Kripke 模型，用 LTL(Linear Temporal Logical)公式或 CTL(Computational Tree Logic)公式描述系统待检测的性质。本文选用 LTL 公式对电路特性进行形式化描述。系统描述和 LTL 逻辑公式提交到 SMV 系统运行，SMV 从系统描述文件中提取有序二叉决策树(OBDD)表示的状态迁移系统，利用基于 OBDD 的搜索算法确定系统是否满足 LTL(CTL)描述的系统性质，最终给出 True 或 False 的结果，并在某属性为 False 的情况下给出导致该结构的反例，SMV 工作原理如图 2 所示。

3.2　异步 FIFO 的有限状态机模型

在模型检验中，异步 FIFO 实现的状态转换关系被映射为有限自动机，它可以由五元组 $F_{FIFO}=\langle Q,\Sigma,\delta,q_0,f\rangle$ 表示，其中 Q 表示状态集合；Σ 表示输入变量取值集合；δ 表示状态迁移函数，即 $\delta:Q\times\Sigma\rightarrow Q$；$q_0$ 表示初始状态；f 表示最终状态集合。

3.3　脉冲信号采用 Gray 码的同步转换器

异步 FIFO 设计的 FSM 描述 F_{FIFO} 如图 3 所示，其中初始状态 $q_0=\{\text{empty}\}$ 表示该数据寄存器没有新的数据；状态{full}表示该数据寄存器有满的数据，状态{al_empty}表示该数据寄存器有几乎空的数据，状态{al_full}表示该数据寄存器有几乎满的数据，结束状态为 $f=\{\text{false}\}$，进入该状态表示数据传输错误；状态{data}表示该寄存器中存有稳定的待传输数据。F_{FIFO} 中的状态转换关系如表 1 所列，其中 rd_en 和 wr_en 分别为读、写使能信号。

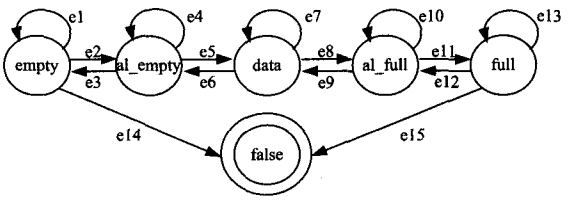

图 3　异步 FIFO 设计的 FSM 描述

表 1　FSM 描述中的各个状态转换关系

状态转换关系	当前状态	下一状态	wr_en	rd_en
e1	empty	empty	0	0
e2	empty	al_empty	1	0
e3	al_empty	empty	0	1
e4	al_empty	al_empty	0	0
e5	al_empty	data	1	0
e6	data	al_empty	0	1
e7	data	data	0	0
e8	data	al_full	1	0
e9	al_full	data	0	1
e10	al_full	al_full	0	0
e11	al_full	full	1	0
e12	full	al_full	0	1
e13	full	full	0	0
e14	empty	false	0	1
e15	full	false	1	0

抽象待检验的异步 FIFO 系统模型 LTL 的描述如下：

e1:assert G(empty&~wr_en&~rd_en->empty);

e2:assert G(empty&wr_en&~rd_en->al_empty);

e3:assert G(al_empty&~wr_en&rd_en->empty);

e4:assert G(al_empty&~wr_en&~rd_en->al_empty);

e5:assert F(al_empty&wr_en&~rd_en->data);

e6:assert F(data&~wr_en&rd_en->al_empty);

e7:assert G(data&~wr_en&~rd_en->data);

e8:assert F(data&wr_en&~rd_en->al_full);

e9:assert F(al_full&~wr_en&rd_en->data);

e10:assert G(al_full &~wr_en&~rd_en->al_full);

e11:assert F(al_full &wr_en&~rd_en->full);

el2:assert F(full&～wr_en&rd_en－＞al_full)；

el3:assert G(full&～wr_en&～rd_en－＞full)；

el4:assert G(～(full&～wr_en&rd_en))；

el5:assert G(～(full&wr_en&～rd_en))。

3.4 数据总线的同步器异步 FIFO

根据异步 FIFO 的特点，从安全性和存活性两方面制定电路特性，对于空满信号，其安全性有两个：

(1)当 FIFO 为空(FIFO Empty)时，写指针经过跨时钟域传输后和读控制逻辑生成的 empty 一定为真，当 FIFO 满(FIFO Full)时，读指针经过跨时钟域传输后和写控制逻辑生成的 full 一定为真。

(2)empty 和 full 不能同时为真。

由于经过同步器传输的指针信号有不确定的延时，因此空满信号的存活性为：

(1)当 full 为真且 FIFO 非满(FIFO ～Full)时，full 最终将会撤消。

(2)当 empty 为真且 FIFO 非空(FIFO ～Empty)时，empty 最终将会撤消。

对于表征电路状态的格雷码读写指针，需要检查的安全性是：每次变化前后指针向量的汉明距离(Hamming Distance)为 1，即变化前后相邻的两个 n 位指针向量，有且仅有 1 位不同。

异步 FIFO 电路特性的线性时序逻辑 LTL 的描述如下：

FIFO_ property 1:assert G(FIFO Empty－＞empty)；

FIFO_ property 2:assert G(FIFO Full－＞full)；

FIFO_ property 3:assert G((empty&FIFO ～Empty)－＞F(～empty))；

FIFO_ property 4:assert G((full& FIFO ～Full)－＞F(～full))；

FIFO_ property 5:G(Hamming_Distance(wptr,wptr_reg)＝1)

FIFO_ property 6:G(Hamming_Distance(rptr,rptr_reg)＝1)

FIFO_ property 7:assert G(～(empty&full))

其中，Hamming_Distance()函数返回两个输入变量的汉明距离，wptr_reg,rptr_reg 分别为变化前的 wptr,rptr 值。

3.5 实验结果

表 2 异步 FIFO 深度宽度不同对应的验证时间

FIFO 深度	FIFO 宽度	模型检验时间(s)
32	8	0.015625
	16	0.03125
64	16	0.015625
	32	0.03125
128	32	0.015625
	64	0.046875
256	64	0.015625
	128	0.046875
512	128	0.015625
	256	0.046875

实验的硬件环境采用的 CPU 是 AMD Athlon(tm) 64 X2 Dual core Processor 4400＋ 2.31GHz，内存 1.0G 的 DDR2，操作系统：Microsoft Windows XP，模型检测工具：美国卡内基大学的 SMV(Symbolic Model Verifier)。实验结果如表 2 所列。

根据实验结果分析发现，本文描述的系统和异步 FIFO 的几个关键属性，经 SMV 检验工具验证，设计是正确的；改变异步 FIFO 的深度和宽度，验证得出不同的用户时间和系统时间，模型检验具有能够自动进行、验证速度快、不用书写测试激励的优点。当系统模型不满足系统要求时，模型检验器会自动生成不满足系统规格的反例，这些反例反映了模型中的瑕疵，告诉设计者如何进一步修改模型以满足所需求的属性。

结束语 跨时钟域设计被越来越多地应用到 SoC 系统芯片中，对跨时钟域设计进行功能验证是 SoC 系统芯片验证工作中的难点。本文重点分析了异步 FIFO 的模型检验方法，建立了一个异步 FIFO 的有限状态机模型，利用符号化模型检验工具 SMV 对该模型和属性进行了描述和验证，结果显示其功能正确，所设计的异步 FIFO 已成功用于 FT 系列 CPU 芯片上；同时也表明，与传统的模拟和仿真等验证方法相比较，模型检验具有能够自动进行、验证速度快、不用书写测试激励的优点，节省了大量时间。

参 考 文 献

[1] Clifton C, Leavens G T, Chambers C, et al. MultiJava: modular open classes and symmetric multiple dispatch for Java[J]. ACM SIGPLAN Notices, 2000, 35(10): 130-145

[2] Shaker S, Verma. Critical clock-domain-crossing bugs. Electronics Design, Strategy, News[EB/OL]. http://www.highbeam.com/doc/1G1-177439329.html, 2008-03-03

[3] Feng Yi, Zhou Zheng, Tong Dong, et al. Clock Domain Crossing Fault Model and Coverage Metric for Validation of SoC Design [C]//Design, Automation&Test in Europe Conference& Exhibition. San Jose, 2007: 1385-1390

[4] Feng Yi , Yi Jiang-fang, Liu Dan, et al. Property Generation Method for Model Checking on Clock Domain Crossing Design [J]. Acta Elecironica Sinica, 2008, 36(5): 886-892

[5] Li Bing, Kwok C K-K. Automatic Formal Verification of Clock Domain Crossing Signals[C]//Design Automation Conference in Asia and South Pacific, 2009. ASP-DAC, 2009: 654-659

[6] SpyGlass® Clock-Reset Rules Reference , Atrenta , Inc. Mar. , [OL]. http://www.atrenta.com, 2009

[7] Clifford E. Cummings, "Clock Domain Crossing(CDC) Design & Verification Techniques Using SystemVerilog [C] // SNUG-2008. Boston

[8] Ly T, Hand N, Kwok C K-K. Formally Verifying Clock Domain Crossing Jitter Using Assertion-based Verification [C] // Design& Verification Conference & Exhibition. DVcon, Mar 2004: 1-5