

基于 GSTE 的验证在 UART 模块中的应用研究

王志才¹, 杨国武¹, 郑德生¹, 漆 美²

(1 电子科技大学 计算机科学与工程学院, 四川 成都 611731; 2 攀枝花学院, 四川 攀枝花 617000)

摘 要: 实现基于 VIS 平台的 GSTE 验证方法, 并将非阻塞性赋值算法引入 VIS 中, 完成了对 UART 的寄存器传输级模型的 FSM 的状态转移的验证和对发送的数据是否与输入数据一致的符号化验证。实验表明, 该方法能够完成测试用例的全覆盖, 说明了该方法在实际例子中所具有的重要意义。

关键词: 形式化验证; GSTE; UART; VIS; 非阻塞性赋值

中图分类号: TP311.56

文献标识码: A

文章编号: 1000-7180(2013)09-0022-04

Research of GSTE—Based Verification on UART

WANG Zhi-cai¹, YANG Guo-wu¹, ZHENG De-sheng¹, QI Mei²

(1 School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; 2 Panzhihua University, Panzhihua 617000, China)

Abstract: This paper presents an implemented version of GSTE and extends the non-blocking assignments on VIS platform. Through verifying the status transition of register transfer level model and consistency between output data and input data of UART, the method has been proven to be full coverage for the test case and be important significant in actual issue.

Key words: formal verification; GSTE; UART; VIS; non-blocking assignments

1 引言

VIS (Verification Interacting with Synthesis)^[1] 是一个用于硬件平台的验证和综合工具。

VIS 共包含以下三个模块: ①前端模块: 读取并且转换成由 BLIF—MV^[2] 描述的层次化系统, BLIF—MV 是采用 VL2MV^[3] 编译高级硬件描述语言, 如 Verilog HDL, 后所产的文件; ②验证模块: 仅仅提供对 CTL (Computation Tree Logic)^[4] 语言的验证, 而且公平约束 (Fairness Constraint) 必须为 Buchi 自动机; ③综合模块: VIS 通过与 SIS 的交互来简化系统的某一部分, 从而达到简化验证的目的。另外, VIS 还支持层次化的综合。

GSTE (General Symbolic Trajectory Evaluation)^[5] 是对 STE (Symbolic Trajectory Evaluation) 的扩展, 它将 STE—style 的模型检测技术应用到了所有 ω -regular

性质的同时并且保留了空间和时间上的高效。并且将 STE 的规范化语言一般化为断言图 (Assertion Graph), 它是一个每条边都标记前序词 (Antecedent) 和后序词 (Consequence) 的有向图, 而传统的 STE 断言便可以看做线性断言图。GSTE 作为一个强有力的形式化验证工具, 使用时态逻辑模型检测技术将传统的 STE 技术和工业验证进行了结合。

本文在 VIS 中实现 GSTE 算法的基本思想, 并增加了非阻塞性赋值的相关算法, 并对其进行优化完善, 通过对 UART (Universal Asynchronous Receiver/Transmitter) 的 FSM (Finite State Machine) 的状态转移的验证和串行输出数据与总线输入数据的逻辑关系验证, 说明算法的有效性和可行性。

2 UART 功能的布尔描述

首先介绍 UART 的部分功能, 然后简要的阐述

收稿日期: 2012-11-20; 修回日期: 2012-12-22

基金项目: 国家自然科学基金项目 (60973016)

了怎样去验证 UART 发射机串行输出数据和总线输入数据的逻辑关系,和验证 FSM 的状态转移关系.

UART 是通用异步收发器的缩写,它通常是更大系统中的一部分,其中主处理器控制传输,发射器从主机读取并行数据并将其转化为串行模式发射,接收机接收串行数据,并检查数据的传输^[6]. UART 发射机的结构包括控制器,数据寄存器(XMT_datareg),数据移位寄存器(XMT_shfreg)和对已发射的位进行计数的状态寄存器(bit_count). 控制 UART 发射机的状态机包括三个状态: idle, waiting, sending. 系统任何时候当 reset_产生负跳变时系统将复位并处于 idle 状态,当 Byte_ready 有效,驱动系统到 waiting 状态,当系统在 waiting 状态且 T_byte 有效,系统将被驱动到 sending 状态,并将 XMT_shfreg 的最低(有效)位置 0,发出传输的开始信号. 在紧接着的下一个时钟有效沿开始,系统通过 bit_count 计数器来控制串行数据发送,发送完成后系统转至 idle 状态^[7].

传统设计的验证是通过模拟验证来完成的,然而随着电路复杂性的日益增加,模拟验证需要大量的 CPU 时间,并且穷举的模拟验证很难保证设计的正确性. 本例中将根据 UART 协议对 UART 发射机串行输出的数据 Serial_out 和数据总线 Data_Bus 上的数据是否一致进行符号化验证,及通过判断输入总线和输出总线的逻辑关系来判断数据是否一致,从而达到对数据总线的全覆盖的测试. 除此之外还验证上段提及的 UART 发射机的 FSM 的状态转移关系.

具体方法为按照芯片协议输入激励,然后再判断输出端的数据或者内部寄存器是否满足要求. 例如在 UART 发射机输出开始信号 0 以后的下一个时钟,通过 $\text{Serial_out} \odot \text{Data_Bus}[0]$ 来判断输出数据 Serial_out 是否与 Data_Bus[0] 相等,再下一个时钟,将通过 $\text{Serial_out} \odot \text{Data_Bus}[1]$ 来判断输出数据的正确性,以此类推直到验证完 8 个发射的串行数据,其中 $\odot$ 表示布尔逻辑中的同或. 对于 FMS 的状态转移的验证,则通过 $\text{state}[2;0] = s$ 来判断当前所处的状态是否为 s(s 表示 UART 发射机中的某一个状态,如 waiting). 所以,当需要对 UART 发射的第一个有效数据和芯片当前的状态进行验证时,只需要系统满足如下布尔表达式:

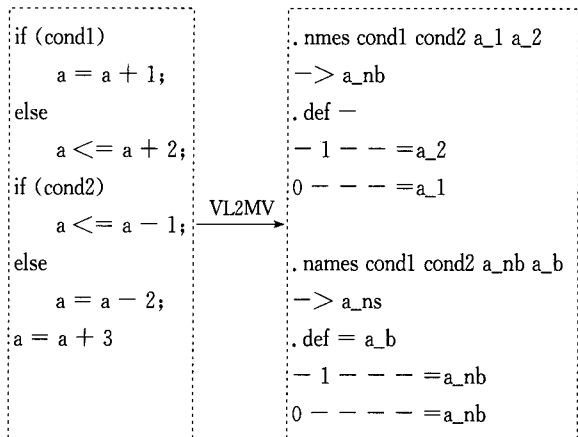
$(\text{Serial_out} \odot \text{Data_Bus}[0]) \& \text{state}[2;0] = \text{sending} \& \text{next_state}[2;0] = \text{sending}$

3 非阻塞性赋值算法

介绍 VL2MV 中的非阻塞性过程赋值语句的编译过程,阐述怎样将非阻塞性赋值算法加入到 VIS 中.

在非阻塞性过程赋值语句中,对目标的赋值是非阻塞的,但可以被预定在将来某个时刻执行. 当语句执行时,计算右侧的表达式,然后在预定的时刻将右侧的值赋给左式的目标,并继续执行下一条语句. 及在当前时刻结束前或者被预定输出的时刻结束前的最后一刻,对左边的目标进行赋值.

VL2MV 首先编译每个非阻塞性赋值语句的右侧表达式作为右值,对于同一个寄存器变量,可能存在有多个右值存在的情况,这时编译器将根据语句的执行顺序来决定由哪个右值来作为最终的赋值. 如图 1 是文献[3]中的一段 Verilog HDL 代码到 BLIF-MV 的转化关系.



a_1 为语句 $a \leq a + 2$ 的右值,

a_2 为语句 $a \leq a - 1$ 的右值.

图1 Verilog HDL 编译为 BLIF-MV 文件

从图 1 中的 BLIF-MV 文件可以看出,当 cond1 为 0 的时候, a 的右值 a_nb 等于 a_1, 当 cond2 为 1 的时候 a 的右值 a_nb 为 a_2.

VIS 在创建芯片模型的迁移关系的时候,是不支持非确定性的. 也就是说芯片模型在输入信号和当前寄存器状态一定的时候,芯片模型的下一个寄存器状态必须是一定的. 所以对于图 1 中的例子,当 cond1 为 0, cond2 为 1 的时候,那么 a 的右值 a_nb 是不确定的. 但是 BLIF-MV 中的每一个关系表达式却是有先后之分的,及越往前显示的关系优先级越高或者说越容易被选中匹配,所以 VIS 在读取 BLIF-MV 文件的时候,须增加相应的预处理功能,如图 2 是将图 1 中的 BLIF-MV 经过非阻塞性

赋值算法预处理后的结果.

```
.names cond1 cond2 a_1 a_2 -> a_nb
.def 0
- 1 - - =a_2
0 0 - - =a_1

.names cond1 cond2 a_nb a_b -> a_ns
.def = a_b
- 1 - - - =a_nb
0 0 - - - =a_nb
```

图2 VIS 预处理 BLIF-MV 文件后
非阻塞性赋值算法如下:

PreDealTransitionOfNames (names) begin

①. def $\rightarrow$. def 0; // 将. def 中所有的'-'修改为'0'

② $T_1, T_2, T_3 \dots T_n$; // 初始化 $T_1, T_2, T_3 \dots T_n$, 其中 T_1 表示 BLIF-MV 中 names 中的第一个关系, T_2 表示第二个关系, 依次类推.

③ $T_1 \rightarrow T$; // T_1 的值赋给中间变量 T

④for ($i = 2$; $i \leq n$; $i++$) begin

⑤ $T_i - T \rightarrow T_i$

⑥ $T_i \cup T \rightarrow T$

⑦end

⑧end function

经过预处理后, 当 cond1 为 0, cond2 为 1 的时候, 那么 a 的右值 a_{nb} 将确定为 a_2 .

4 UART 性质描述

根据文献[5]中的 GSTE 理论和第二节提及的需要验证的 UART 模块的功能, 对其性质用断言图进行描述, 如图 3 所示. 图中的每条边的时间跨度为一个时钟, 边上的 ant (Antecedent, 前序词) 和 cons (Consequence, 后序词) 在表 1 中定义, 这里布尔表达式的描述与 VIS 的 CTL 验证输入保持一致, 该图描述了 UART 发射机所需要具有的状态转移关系和输出数据与输入数据之间的逻辑关系. 与文献[5]中的 GSTE 理论稍有不同的是, 本文中每条边迁移出来的状态存储在边的终止节点上. 表 1 中 * 表示与, + 表示或, $\leftrightarrow$ 表示同或; 当变量的值为 1 时, 表示变量成立, 同理如果为 0 表示不成立. 例: $\text{Byte_ready} = 0 * T_byte = 1$ 表示布尔表达式: $! \text{Byte_ready} \& T_byte$, 而 $\text{state}[2:0] = 4$ 等价表示为 $\text{state}[2] \& ! \text{state}[1] \& ! \text{state}[0]$, 即验证当前芯片模型所处的状态为 sending. 在表 1 的 ant 中没有给出的变量则表示该变量可以为任意的值. 例: 表 1 中 ant 边并没有对 Data_Bus 的输入进行说明, 从而 Data_Bus 看做任意的输入或符号输入.

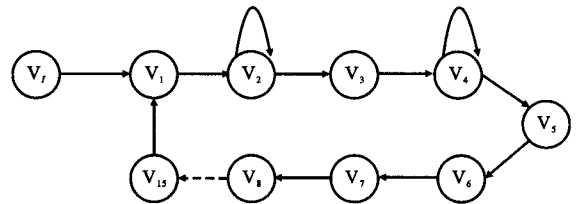


图3 UART 功能验证的断言图

表1 图3中边的 ant 和 cons 的布尔描述

edge	ant	cons
$v_1 \rightarrow v_1$	$\text{reset}_- = 0$	S
$v_1 \rightarrow v_2$	$\text{reset}_- = 1 * \text{Load_XMT_datareg} = 1 * \text{Byte_ready} = 0$	$\text{state}[2:0] = 1 * \text{next_state}[2:0] = 1$
$v_2 \rightarrow v_2$	$\text{reset}_- = 1 * \text{Byte_ready} = 0$	$\text{state}[2:0] = 1 * \text{next_state}[2:0] = 1$
$v_2 \rightarrow v_3$	$\text{reset}_- = 1 * \text{Byte_ready} = 1$	$\text{state}[2:0] = 1 * \text{next_state}[2:0] = 1$
$v_3 \rightarrow v_4$	$\text{reset}_- = 1 * \text{Byte_ready} = 1$	$\text{state}[2:0] = 1 * \text{next_state}[2:0] = 2$
$v_4 \rightarrow v_4$	$\text{reset}_- = 1 * T_byte = 0$	$\text{state}[2:0] = 2 * \text{next_state}[2:0] = 2$
$v_4 \rightarrow v_5$	$\text{reset}_- = 1 * T_byte = 1$	$\text{state}[2:0] = 2 * \text{next_state}[2:0] = 2$
$v_5 \rightarrow v_6$	$\text{reset}_- = 1 * T_byte = 1$	$\text{state}[2:0] = 2 * \text{next_state}[2:0] = 4$
$v_6 \rightarrow v_7$	$\text{reset}_- = 1 * \text{Byte_ready} = 0 * T_byte = 0$	$\text{Serial_out} = 0 * \text{state}[2:0] = 4 * \text{next_state}[2:0] = 4$
$v_7 \rightarrow v_8$	$\text{reset}_- = 1 * \text{Byte_ready} = 0 * T_byte = 0$	$\text{Serial_out} = 1 \leftrightarrow \text{Data_Bus}[0] = 1 * \text{state}[2:0] = 4 * \text{next_state}[2:0] = 4$
$v_i \rightarrow v_{i+1}$	$\text{reset}_- = 1 * \text{Byte_ready} = 0 * T_byte = 0$	$\text{Serial_out} = 1 \leftrightarrow \text{Data_Bus}[i-7] = 1 * \text{state}[2:0] = 4 * \text{next_state}[2:0] = 4$
$v_{15} \rightarrow v_1$	$\text{reset}_- = 1 * \text{Byte_ready} = 0 * T_byte = 0$	$\text{Serial_out} = 1 * \text{state}[2:0] = 4 * \text{next_state}[2:0] = 1$

在表 1 中 S 表示全集, i 从 8 开始并终止于 14, $\text{state}[2:0] = 1$ 用于判断状态是否为 idle, 同理 2 为 waiting, 4 为 sending.

在该断言图中从 v_1 到 v_6 为完成对 UART 芯片

的初始化, 及系统从 idle 状态迁移到 sending 状态, 并发送了传输的开始信号 0. 以 v_2 为例, $v_2 \rightarrow v_2$, ant: $\text{Byte_ready} = 0$, 那么系统在其驱动下从 v_2 上的当前状态所产生的所有下一步状态将并到 v_2 节点上,

而 $v_2 \rightarrow v_3$, ant: Byte_ready = 1 那么在其驱动下产生的下一步状态将会并到 v_3 节点上. 系统从 v_7 节点开始发送有效数据, 通过边上的 cons 来验证数据是否有效, 例如: 验证第一个有效数据是否跟 Data_Bus[0] 相等, 只需要在 $v_7 \rightarrow v_8$ 的边上标明 cons: Serial_out = 1 $\leftrightarrow$ Data_Bus[0] = 1, 程序将会检查 v_7 上面的状态是否满足于 cons 从而判断系统是否出错. 如果没有发生任何的错误, 当所有节点上的状态达到不动点及状态不在变化, 系统将会停机, 从而完成对系统的验证. 如果出错将会打印出反例^[8]及从初始状态到错误状态的错误路径, 工程人员通过错误路径便可以快速定位到出错的代码.

此断言图包含了 UART 发射端状态转移的所有可能性, 因此能够验证全覆盖的状态转移. 而对于 Data_Bus 与 Serial_out 的关系, 是通过符号化方法来验证的, 即在验证过程中不需要具体的测试用例, 输入信号由一个形式化符号来代替并保存于整个验证过程中(默认为信号端口的名字), 最后直接验证信号之间的布尔关系, 即可达到测试数据的全覆盖性.

5 UART 验证结果

通过在 VIS 平台实现的 GSTE 工具, 采用图 3 所示的断言图对 UART 发射机模块进行验证, 验证文件为文献[7]中的 UART_Transmitter_Arch. v, 验证结果如表 2. 表中反例的有无, 是指对应的例子是否应该出现反例, 跟使用的工具无关.

表 2 UART 验证结果

测试用例	寄存器数	反例	GSTE	VIS's CTL
UART	47	无	1.8s	N/A
UART_ERR	47	有	0.5s	N/A
UART_ERR2	47	有	0.5s	N/A

表 2 中 UART_ERR, UART_ERR2 为自行对 UART_Transmitter_Arch. v 用例的错误修改, 其中 UART_ERR 涉及状态机的修改, UART_ERR2 涉及发射数据的修改.

根据实验结果, 原生的 VIS 并不支持对这三个例子的验证, 而 GSTE 工具快速的完成了对例子的验证, 当验证的测试用例出现反例的时候, GSTE 工具正确地给出了错误路径, 包含从开始状态到错误状态的每一步所输入的激励信号和其导致的寄存器状态.

6 结束语

文中通过在 VIS 中实现非阻塞性赋值算法, 在验证的过程中符号化的保存了输入信号, 最终在 VIS 平台实现了 GSTE 验证方法并完成了对 UART 发射机功能的符号化验证, 展现了形式化验证中数据无关性的优点, 而该方法同样也适合于其它的集成电路芯片模型. 下一步将去调研模型的抽象和划分, 以及布尔表达式高效的基础数据结构, 来解决超大规模的集成电路的芯片验证问题.

参考文献:

- [1] Robert K Brayton, Gary D Hachtel. VIS: A system for verification and synthesis[J]. Lecture Notes in Computer Science, 1996(1102):428-432.
- [2] Robert King Brayton. BLIF—MV: an interchange format for design verification and synthesis[C]// Memorandum UCB/ERL M91/97. University of California at Berkeley, 1991.
- [3] Szu Tsung Cheng, Robert K Brayton. Compiling verilog into automata[C]//Memorandum UCB/ERL M94/37. University of California at Berkeley, 1994.
- [4] Michael Huth, Mark Ryan. Logic in Computer Science [M]. 2nd ed. UK: Cambridge University Press, 2004.
- [5] Yang J, Seger C J. Generalized symbolic trajectory evaluation[R]. Intel SCL Technical Report, 2000.
- [6] 李洋洋, 吴武臣, 王龙伟, 等. 基于断言的验证方法在 UART 模块中的应用研究[J]. 微电子学与计算机, 2010, 27(1):151-153.
- [7] Michael D Ciletti. Verilog HDL 高级数字设计 [M]. 李镭, 张雅绮, 译. 北京: 电子工业出版社, 2005.
- [8] 李义年, 曹占涛, 郑德生, 等. 基于广义符号轨迹赋值模型验证的反例的产生[J]. 计算机科学与技术, 2010(9):245-248.

作者简介:

王志才 男, (1986—), 硕士研究生. 研究方向为形式化验证.

杨国武 男, (1966—), 博士生导师. 研究方向为逻辑综合和形式化验证.

郑德生 男, (1983—), 博士研究生. 研究方向为形式化验证和嵌入式软件设计.

漆美女, (1959—), 副研究员. 研究方向为计算机应用技术.