

一种基于 SAT 的运算电路查错方法

陈云霁¹⁾ 张 健²⁾ 沈海华¹⁾ 胡伟武¹⁾

¹⁾(中国科学院计算技术研究所计算机系统结构重点实验室 北京 100080)

²⁾(中国科学院软件研究所计算机科学国家重点实验室 北京 100080)

摘 要 基于 SAT 的运算电路查错方法将被验证系统中系统规范成立与否的问题转换为布尔公式和数学公式的混合形式 E-CNF, 通过采用了标志子句技术的 E-SAT 求解器进行求解. 实验表明该方法自动化程度高, 能处理大规模的运算电路, 有较强的查找错误能力.

关键词 形式验证; 模型检验; SAT; E-CNF; 标志子句

中图法分类号 TP301

A SAT-Based Arithmetic Circuit Bug-Hunting Method

CHEN Yun-Ji¹⁾ ZHANG Jian²⁾ SHEN Hai-Hua¹⁾ HU Wei-Wu¹⁾

¹⁾(Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

²⁾(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract E-CNF is hybrid of Boolean formula and mathematic formula. SAT-based arithmetic circuit bug-hunting method translates the verification problem into E-CNF, and solves E-CNF through E-SAT solver. E-SAT solver is an extension of complete SAT solver, with tag clause technique. Experiments show that SAT-based arithmetic bug-hunting method is powerful in finding bugs in arithmetic circuits.

Keywords Formal verification; SAT; E-CNF; Tag clause

1 引 言

运算电路的形式验证和一般的硬件形式验证有较大的不同. 原因有几个方面: (1)一般电路的系统规范只包含命题(时态)逻辑公式, 而运算电路的系统规范还包含多位布尔变量组成的代表整数或浮点数的字变量和字变量的数学公式; (2)运算电路常常包含一些常见模块如 ALU、CSA 等, 这些模块的字级描述比较简单, 但位级描述十分复杂; (3)运算电

路的输入变量和寄存器数通常较多(如双精度浮点数之间的运算就至少有 128 位输入变量), 容易导致状态爆炸. 因此常用的硬件形式验证的方法, 如基于二元决策图(BDD)的模型检验方法^[1]和基于 SAT 的有界模型检验方法(Bounded Model Checking)^[2], 难以用于运算电路的形式验证.

为适应运算电路的特点, Clarke 等人对基于二元决策图的模型检验进行字级(Word Level)扩展, 提出了基于决策图的字级模型检验方法^[3]. 该方法支持代表整数或者浮点数的字变量以及字变量间的

收稿日期: 2006-01-17; 最终修改稿收到日期: 2007-11-08. 本课题得到国家“九七三”重点基础研究发展规划项目基金(2005CB321600)、国家“八六三”高技术研究发展计划项目基金(2002AA110010, 2005AA119020, 2005AA110010)、中国科学院知识创新工程重大项目基金“高性能通用 CPU 芯片研制”(KGCX1-SW-09)和国家自然科学基金(60603049, 60673044)资助. 陈云霁, 男, 1983 年生, 博士, 助理研究员, 主要研究方向为高性能计算机体系结构和硬件验证. E-mail: cyj@ict.ac.cn. 张 健, 男, 1969 年生, 博士, 研究员, 博士生导师, 主要研究领域包括自动推理、约束求解、程序分析和软件测试. 沈海华, 女, 1971 年生, 博士, 副研究员, 主要研究方向为计算机体系结构、微处理器设计与验证. 胡伟武, 男, 1968 年生, 博士, 研究员, 博士生导师, 主要研究领域为高性能计算机系统结构、并行处理、VLSI 设计等.

数学运算,并采用更适合表示整数或者浮点数的决策图,如 *BMD^[4]、HDD^[5] 和 *PHDD^[6] 等.因此基于决策图的字级模型检验在运算电路验证中取得了一些成果.如 Clarke 等人使用基于 HDD 的字级模型检验工具 WSMV 实现了基 4 的 SRT 除法核心算法的验证^[3].Chen 等人使用基于 *BMD 的字级模型检验工具 ACV 实现了相同的基 4 的 SRT 算法的验证^[7].借助于状态分割,王海霞使用基于 *PHDD 的字级模型检验工具 ArithSMV 实现了龙芯的浮点加法运算电路的验证^[8].

然而基于各种决策图的字级模型检验方法存在一些固有的困难.首先,基于决策图的验证方法都对决策图的变量序非常敏感,往往需要人工的干预来选择适合的变量序.其次,无论采用哪种决策图,决策图的大小都是变量数的指数级函数.对于输入变量动辄达数百的工业级运算电路,状态爆炸依然是一个挑战.第三,基于决策图的方法处理错误电路比处理正确电路的复杂度(无论时间还是空间上)高,不利于查找设计的错误.

基于 SAT 的有界模型检验方法的基本思想是从初始状态开始,在小于等于 k 次迁移的状态空间中寻找使规范不成立的反例.通过给定 k ,线性时态逻辑公式的模型检验问题被转化为 CNF 形式的布尔公式,通过完全的 SAT 求解工具求解.由于 SAT 求解算法的特点,该方法对变量序相对不敏感,更重要的是它一旦找到错误就不需要继续遍历整个状态空间.因此它基本无需人工干预,能处理较大规模的电路,适合查找错误,避免了基于决策图的方法的固有困难.但由于传统的有界模型检验方法不支持包含数学公式的系统规范,所以难以用于验证运算电路.因此,针对运算电路验证,中科院计算所龙芯验证组提出了结合字级模型检验方法和基于 SAT 的有界模型检验方法的思想^[8].

基于 SAT 的运算电路查错方法将被验证系统中系统规范成立与否的问题转换为布尔公式和数学公式的混合形式 E-CNF.我们还实现了一个能对 E-CNF 进行混合约束求解的工具 E-SAT.为了提高效率,E-SAT 引入了标志子句的方法.实验数据表明标志子句方法提高了 E-SAT 处理复杂 E-CNF 的速度.

和一些混合约束求解方法如文献[9-10]和文献[11]相比,我们的方法针对字级模型检验,处理的 E-CNF 中允许变量同时在数学子句和布尔子句中

出现,适合有大量变元的复杂非线性数学公式.多值 SAT 求解器 CAMA^[15]能在多维多值域的解空间中进行约束求解,但是它需要对值域进行独热编码(one-hot-encoding),而对运算电路的值域进行独热编码代价太大,此外它也没有对非线性运算的专门支持.针对 RTL 的求解器 HDPLL^[16]是和本文工作最相近的工作.HDPLL 针对 RTL 中常见的数据通路进行了专门的支持,能进行混合约束求解.HDPLL 中数学公式通过基于上下界算术(interval arithmetic)的方法进行处理,较好地提高了效率.但在上下界算术方法中,数学公式包含的任何一个变量被赋值都需要重新计算数学公式的上下界,在很多情况下这种策略并非最优.此外上下界算术方法计算乘法的上下界时需要将乘法展开成大量 ITE(If-Then-Else)结构,处理复杂的非线性数学公式效率低.此外 HDPLL 的系统规范不支持乘法和除法,更不支持幂运算.

基于 SAT 的运算电路查错方法避免了基于决策图的方法的固有困难,自动化程度更高,能处理的运算电路规模更大,查找错误效率更高.在龙芯 2 号微处理器功能部件验证中,该方法在较短时间内发现了一些仿真验证难以发现的高质量的 bug,显著地缩短了龙芯 2 号的设计-调试周期,提高了设计者在流片前的信心,对龙芯 2 号的验证起到了重要作用.最重要的是,在多种不同的运算电路(多媒体功能部件和各种定点、浮点运算功能部件等)上的成功使我们合理相信该方法能够广泛运用于运算电路验证.

本文第 2 节简要介绍基于决策图的字级模型检验方法;第 3 节简要介绍完全 SAT 求解器中常用的 DPLL 算法;第 4 节给出 E-CNF 的定义和运算电路的模型检验问题转换成 E-CNF 的算法;第 5 节介绍 E-SAT 求解器;第 6 节介绍龙芯 2 号微处理器中功能部件验证的例子;第 7 节进行分析和总结.

2 基于决策图的字级模型检验

字级模型检验系统是对模型检验系统进行字级扩展后产生的验证系统,主要用于运算电路的验证.它必须支持包含字变量和字变量间的数学运算的系统规范.

以往字级模型检验系统都是基于决策图的.为表示字变量和字变量间的数学运算,必须对二元决

策图的数据结构进行改进. Clarke 等人首先提出将二元决策图终端节点的值域从 $\{0,1\}$ 扩展到整数域, 也就是 MTBDD^[7]. 但大部分数学公式用 MTBDD 表示规模是指数级的. 为此, HDD^[5] 和 PHDD^[6] 等决策图被提出, 其中 *PHDD 被认为是这些决策图中表达能力最强的.

*PHDD 表示从一组有序布尔变量到整数或浮点数的映射函数. *PHDD 结点可用三元组 (s, w, f) 表示, 代表函数 $(-1)^s \times 2^w \times f$. 符号位 s 表示函数是否取负值, w 为边的权值, 可以为任意整数. 因此 *PHDD 只用整数结点而不需有理数结点即可表示从布尔变量到浮点数的映射函数. *PHDD 中每个布尔变量 x 有三种可能的分解方式, 但决策图中该变量的所有结点都必须使用同一种分解类型. 这三种分解方式分别是: Shannon (S) 分解、Positive Davio (PD) 分解和 Negative Davio (ND) 分解, 如公式 (1) 所示.

$$f = \begin{cases} (1-x) \cdot f_{\text{low}} + x \cdot f_{\text{high}}, & \text{S} \\ f_{\text{low}} + x \cdot f_{\text{high}}, & \text{PD} \\ f_{\text{low}} + (1-x) \cdot f_{\text{high}}, & \text{ND} \end{cases} \quad (1)$$

其中, f 为当前节点对应的函数, f_{low} 和 f_{high} 为 f 的 0-边和 1-边结点对应的函数. *PHDD 的结构使得它易于表示整数和浮点数, 并且用它表示大部分整数运算函数, 包括加减乘除, 只需要线性的节点数.

虽然 *PHDD 能够较为精简地表示整数和浮点数及其运算, 但它无法摆脱决策图方法所固有的困难. 实际验证中也需要更有力的查错方法来提高验证效率.

3 DPLL 算法

有界模型检验使用完全的 SAT 求解工具. 多数完全 SAT 算法核心部分是 DPLL 算法^[13] (图 1). 函数 *decide_next_branch* 选择一个未赋值变量对其赋值, 如果无未赋值变量, 则找到了输入布尔公式的一组解. 函数 *deduce* 通过布尔约束传播 (Boolean Constraint Propagation) 过程进行推导, 试图推导出其它变量的赋值并检查是否存在冲突. 若从决策变量赋值可能推导出的新的变量赋值都已经推导出且不存在冲突, SAT 算法跳回循环开始处. 函数 *decide_next_branch* 重新选择一个未赋值变量对其赋值. 若存在冲突, 函数 *analyze_conflicts* 分析冲突的原因以指导回溯. 函数 *back_track* 根据冲突分析的结果进行回溯. 如果无法再回溯, 则说明该布尔公

式不可满足.

```
while (1) {
  if (decide_next_branch()) {
    while (deduce() == conflict) {
      blevel = analyze_conflicts();
      if (blevel == 0)
        return UNSAT;
      else back_track(blevel);
    }
  }
  else return SAT;
}
```

图 1 DPLL 算法

虽然 SAT 的求解是一个 NP 难问题, 但近年来的研究引入了多种优化方法, 如惰性数据结构、变量选择策略、跳跃性回溯、冲突驱动学习和对称性等. 它们使得完全算法的 SAT 求解工具已经能处理相当规模的问题.

4 E-CNF 及其转换算法

运算电路的系统规范包含数学公式, 因此无法直接利用 SAT 求解器进行可满足性判定. 因此我们必须对 SAT 求解器及其输入进行字级扩展.

4.1 E-CNF 的定义和语义

进行运算电路验证, 必须有适当的表示布尔公式和数学公式的混合形式的方法. 因此我们需要将 CNF 扩展为 E-CNF.

定义 1. E-CNF 及其相关的定义为

- (1) 布尔变量: $V ::= v_1 | v_2 | \dots | v_n$;
- (2) 文字: $L ::= V | \neg V$;
- (3) 字变量: $Word ::= \{V\}$;
- (4) 常数: $C = \mathbb{R}$ (\mathbb{R} 为实数域, 下同);
- (5) 数学项: $A\text{-term} ::= C | V | Word | A\text{-term} \circ$
 $A\text{-term} (\circ \in \{+, -, \times, \%, \text{powerof}2\}, \text{下同});$
- (6) 数学公式: $A\text{-formula} ::= A\text{-term} \sim A\text{-term}$
 $A\text{-term} (\sim \in \{<, >, \leq, \geq, =, \neq\}, \text{下同});$
- (7) 数学子句: $A\text{-clause} ::= A\text{-formula}$;
- (8) 布尔子句: $B\text{-clause} ::= \{L\}$;
- (9) 子句: $clause ::= A\text{-clause} | B\text{-clause}$;
- (10) E-CNF: $E\text{-CNF} ::= \{clause\}$.

定义 2. E-CNF 及其相关的语义为

- (1) 布尔变量的指派: $Assign : V \rightarrow \{0,1\}$
- (2) 文字: $F : L \rightarrow \{0,1\}$

$$F(v) = Assign(v)$$

$$F(\neg v) = 1 - Assign(v)$$

(3) 字变量: $W: Word \rightarrow N$ (N 为整数域, 下同)

$$W(v_{n-1}, v_{n-2}, \dots, v_0) = \sum_{i=0}^{n-1} 2^i \text{Assign}(v_i)$$

(4) 数学项: $E: A\text{-term} \rightarrow \mathbb{R}$

$$E(c) = c$$

$$E(v) = \text{Assign}(v)$$

$$E(w) = W(w)$$

$$E(A\text{-term}_1 \circ A\text{-term}_2) = E(A\text{-term}_1) \circ E(A\text{-term}_2)$$

(5) 数学公式: $Z: A\text{-formula} \rightarrow \{0, 1\}$

$$Z(A\text{-term}_1 \sim A\text{-term}_2) = E(A\text{-term}_1) \sim E(A\text{-term}_2)$$

(6) 子句: $B: clause \rightarrow \{0, 1\}$

$$B(A\text{-clause}(A\text{-formula}_n)) = Z(A\text{-formula}_n)$$

$$B(B\text{-clause}(l_0, l_1, \dots, l_{n-1})) = \bigvee_{i=0}^{n-1} F(l_i)$$

(7) E-CNF: $D: E\text{-CNF} \rightarrow \{0, 1\}$

$$D(E\text{-CNF}(clause_0, clause_1, \dots, clause_{n-1})) = \bigwedge_{i=0}^{n-1} B(clause_i)$$

例 1. 考虑如下 E-CNF($abcd$ 为 4 个布尔变量, $w_1(b, a)$ 和 $w_2(d, c)$ 分别为 ba 和 cd 组成的字变量):

$$\varphi = (a \vee b \vee c) \wedge (\neg c \vee d) \wedge ((w_1(b, a) - w_2(d, c) + 1 \leq 0)).$$

它的一组成真指派为 $a=0, b=0, c=1, d=1$. 此时 $w_1(b, a)$ 为 $0 \cdot 2^1 + 0 \cdot 2^0 = 0$, $w_2(d, c)$ 为 $1 \cdot 2^1 + 1 \cdot 2^0 = 3$. 易知 φ 的 3 个子句都为 1.

4.2 E-CNF 的转换算法

我们可以通过图 2 中的转换算法将运算电路验证问题转换成 E-CNF 的形式. 该算法通过引入辅助变量的方法达到了线性时间复杂度. 它是一般布尔公式转化为 CNF 算法^[12]的扩展. 图 2 中的 \circ 和 \sim 符号分别表示字变量间的运算操作符和比较操作符.

```

hybrid-to-ecnf( $a, v_a$ )
{
  if (cached( $a, v$ )) return clause( $v_a \leftrightarrow v$ );
  case
    atomic( $a$ ): return clause( $v_a \leftrightarrow a$ );
     $a = b \circ c$ :
       $C_1 = \text{hybrid-to-ecnf}(b, v_b)$ ;
       $C_2 = \text{hybrid-to-ecnf}(c, v_c)$ ;
      assert(cached( $a, v_a$ ));
      return clause( $v_a \leftrightarrow v_b \circ v_c$ )  $\cup$   $C_1 \cup C_2$ ;
     $a = b \sim c$ :
      assert(cached( $a, v_a$ ));
      return arithclause( $v_a = b \sim c$ );
  ecase;
}

```

图 2 从混合公式到 E-CNF 的转换算法

函数 hybrid-to-ecnf 输入为布尔函数 a 和布尔

变量 v_a , 返回表示布尔公式 $a \leftrightarrow v_a$ 的 E-CNF. 函数首先检查子句集中是否包含对应于布尔函数 a 的子句 (即已经用变量 v 表示过布尔函数 a), 如果包含, 在子句集中只需增加表示布尔公式 $v_a \leftrightarrow v$ 的子句. 接着, 函数检查 a 是否是变量, 如果是, 在子句集中增加表示布尔公式 $a \leftrightarrow v_a$ 的子句即可. 若布尔函数 a 是另外两个布尔函数 b 和 c 的布尔运算结果, 那么引入两个辅助变量 v_b 和 v_c 分别表示函数 b 和 c , 然后递归调用函数 hybrid-to-ecnf, 产生表示布尔公式 $b \leftrightarrow v_b$ 的一组 CNF 子句 $C1$ 和表示布尔公式 $c \leftrightarrow v_c$ 的一组 CNF 子句 $C2$, 最后将 $C1$ 、 $C2$ 以及表示布尔公式 $v_a \leftrightarrow (v_b \circ v_c)$ 的子句一起, 作为表示布尔公式 $a \leftrightarrow v_a$ 的 CNF 子句返回. 和文献[12]中的算法唯一不同的是, 当操作符是比较操作符时, 函数 arithclause 生成并返回数学子句和对应的数学公式. 因为每个数学公式都可以转换为 $t \sim 0$ 的形式, 所以为方便处理, arithclause 会自动将返回的数学公式都改写成上述形式.

5 E-SAT 求解器

5.1 SAT 求解器的字级扩展

为使 SAT 求解器能够处理 E-CNF, 我们在基于 DPLL 算法的 SAT 求解器 zchaff^[14] 之上进行了一些扩展, 实现了 E-SAT 求解器. 需要指出的是, 这些扩展也很容易在其它基于 DPLL 算法的 SAT 求解器上实现. 扩展算法主要需要对数据结构和布尔约束传播过程进行改进.

数据结构方面, 扩展算法为每个子句增加了一个区分布尔子句和数学子句的标志位和一个指向数学公式的域. 如果子句是布尔子句, 则数学公式域无意义.

当一个未被判断真伪子句所含布尔变量仅有一个未赋值时, 它就是一个单元子句 (unit clause). 布尔约束传播过程通过单元子句规则, 推导出所有可推导出的布尔变量赋值. 由于在数学子句中, 布尔变量没有真假属性, 只有赋值还是未赋值属性, 因此和布尔子句的判定不同, 只有当数学子句所含布尔变量都已经确定了值, 才可判断子句的真伪. 因此在布尔约束传播中, E-SAT 对布尔类型的单元子句处理方式不变, 而对数学类型的单元子句的处理方式有所不同. 布尔约束传播处理数学单元子句 $A\text{-clause}_i$ 的规则如下:

如果 $A\text{-clause}_i$ 所含布尔变量除 v' 外都已经赋

值(也就是一个单元子句),那么将所有已赋值布尔变量的值代入对应的 $A\text{-formula}_i$,并假设 $v'=1$,得到结果 $ResT$;再次将所有已赋值布尔变量的值代入 $A\text{-formula}_i$,并假设 $v'=0$,得到结果 $ResF$. 当 $ResT$ 为真、 $ResF$ 为假,则布尔变量 v' 赋值 1; $ResF$ 为真、 $ResT$ 为假,则布尔变量 v' 赋值 0; 当 $ResT$ 和 $ResF$ 都为假,发生冲突;当 $ResT$ 和 $ResF$ 都为真,则跳到下一单元子句.

例 2. 考虑某 E-CNF 中的一个 $A\text{-clause}$ 的 $A\text{-formula}$ 如下:

$$w_1(b,a) - w_2(d,c) \cdot w_3(c,b,a) + 1 \geq 0.$$

假设在布尔约束传播过程中,已知 $a=1, b=1, d=1$ 而 c 未知. 此时该数学子句是一个单元子句. 假设 $c=1$,代入后即为 $11-11 \cdot 111+1 \geq 0$; 假设 $c=0$,代入后即为 $11-10 \cdot 11+1 \geq 0$. 无论 c 的值如何,该子句都不成真,因此发现了冲突.

然而,简单地将 SAT 求解器扩展到字级,效率是很低下的. 大量信息被包含在数学子句中. 实际上,究竟运算电路是否符合规范,很大程度上依靠数学子句来判断. 但是数学子句只能在所包含布尔变量已经完全被赋值的情况下才能被计算,没有提前回溯的可能. 因此,对于有 n 个输入的运算电路,必须进行 2^n 次回溯. 如果不进行改进,即使是对 16 位的全加器这样简单的电路都难以完全验证^[8],更无法用于复杂的工业级运算电路.

5.2 E-SAT 的改进

E-CNF 是从被验证系统和系统规范中产生的,因而有以下几个特点: (1) E-CNF 中的数学公式常常很复杂,不但是非线性的,而且包含数百个布尔变量,计算一次值需要很长时间. (2) 原电路的输入完全决定 E-CNF 的值. 一旦确定输入变量的值, E-CNF 的值就被固定下来. (3) 数学子句只有在所包含布尔变量都已赋值的情况下才能进行判断. 实际上部分布尔变量被赋值就可能已经让对应的数学公式有确定的值,但求解器还必须做无谓的深度遍历直到所有布尔变量都被赋值. 但是用类似上下界算术^[16]的方法进行提前判断会带来过多的数学公式计算.

针对以上特点 E-SAT 进行了专门的改进. 其中的重点就是引入了标志子句的方法,以尽可能在部分布尔变量被赋值的情况下判断出 $A\text{-clause}$ 的真伪.

定义 3. 标志子句及其相关的定义:

C 为一个 $A\text{-clause}$, C 对应的 $A\text{-formula}$ f 为 $t \sim 0$, 若 w_i 为在 f 中出现的字变量, 则数学公式 $\partial t /$

$\partial w_i \geq 0$ 被称为 f 的标志式, 记作 $T\text{-formula}_i^f$, w_i 被称为该标志式对应的标志字变量, 指向 $T\text{-formula}_i^f$ 的子句 $T\text{-clause}_i^f$ 被称为 C 的标志子句, 而 C 被称为 $T\text{-clause}_i^f$ 的父子句.

标志子句包含的布尔变量为父子句的子集. 因此在父子句 C 中的布尔变量并未全部被赋值时, 标志子句 $T\text{-clause}_i^f$ 中的布尔变量可能已经全部被赋值. 当 $T\text{-clause}_i^f$ 已有确定的值, f 对于标志字变量 w_i 就是单调的. 因此分别假设 w_i 中未赋值布尔变量全部为 0 和全部为 1, 代入 f 可得到 f 的上下界. 通过将 f 的上下界和 0 进行比较, 我们有可能提前判断出 C 的值. 标志子句还可以拥有自己的标志子句, 以进一步提高提前判断原数学子句值的可能性.

例 3. 考虑如下 $A\text{-formula}$ ($w_1 w_2 w_3$ 是字变量, v_0 是一个不在 $w_1 w_2 w_3$ 中出现的布尔变量):

$$f: w_1 + w_2^2 - w_1 \cdot w_3 \cdot v_0 > 0.$$

改进前的方法直到 v_0 和 $w_1 w_2 w_3$ 中的所有布尔变量都被赋值后方可判断 f 的真假值. 改进后的方法首先求出三个标志子句, 分别指向: $f_1: 1 - v_0 \cdot w_3 \geq 0$, $f_2: 2w_2 \geq 0$ 和 $f_3: -v_0 \cdot w_1 \geq 0$. 而 $f_1 f_2 f_3$ 分别拥有各自的标志子句. f_1 唯一的标志子句是对应 w_3 的 $f_{13}: -v_0 \geq 0$. 若我们知道 v_0 为 0, 则 f_{13} 的值为真. 因此 f_1 对 w_3 来说是单调的. 分别假设 w_3 为其最小和最大可能值(也就是令 w_3 中所有未赋值的布尔变量为 0 和 1), 代入 f_1 , 即可判断出 f_1 为真. 再假设 w_1 为其最小和最大可能值, 代入 f , 我们就可以判断出原不等式的左端大于 0, 也就是说 f 值为真.

当我们从被验证系统和系统规范得到了原始的 E-CNF, E-SAT 会进行预处理: 如果数学公式很复杂, 就求出它的各个标志式并将标志子句加入子句列表; 如果标志式也很复杂, 则把标志子句的标志子句也加入子句列表. 标志子句只判断真假, 不参与布尔约束传播的推导. 拥有标志子句的子句通常处于睡眠状态, 不参与 E-SAT 的求解, 只有当它的某个标志子句有确定的值, E-SAT 才会激活它并尝试判断其真假值. 该策略既避免了过多不必要的数学公式计算, 又能尽早计算出数学子句的值.

6 龙芯 2 号微处理器浮点功能部件验证

龙芯 2 号微处理器是由中国科学院计算技术研究所设计的 64 位高性能通用处理器. 它的功能部件包括定点运算部件、浮点运算部件和多媒体部件. 我

们设计了两个字级验证工具,分别基于 *PHDD 决策图和 SAT. 它们的输入格式是 verilog(可综合部分). 两个工具互为补充,一起完全验证了龙芯 2 号的功能部件. 限于篇幅,我们在这里只介绍相对比较复杂的浮点加法和浮点除法部件的验证. 实验环境是双 Intel P4 2.8GHz CPU, 2GB DDR 内存, LINUX 操作系统的工作站.

龙芯 2 号的浮点加法部件实现浮点加减运算,并采用了双通路算法、交换操作数、提前数一算法、提前舍入算法和错位并行算法等技术以降低延迟. 验证的数据见表 1. *PHDD 列是采用基于 *PHDD 的字级模型检验方法的情况. SAT(before)列为基于 SAT 的方法没有采用标志子句技术时的情况, SAT(after)列则是采用标志子句技术时的情况. Time 列是基于 *PHDD 方法的运行总时间. Time1 和 Time2 列分别是运行总时间和求解中用于计算数学公式的总时间. 生成和预处理 E-CNF 所用的时间很短所以被我们忽略了. Mem 列是求解过程中内存用量的峰值. R 行是完全验证正确的设计所用的时间和内存. W1, W2 和 W3 行是对 3 个含有 bug 的

设计找出 bug 所用的时间和内存.

基于 *PHDD 的方法对变量序非常敏感,因此浮点加法运算电路如果不进行状态分割,很难避免状态爆炸. 我们根据执行的操作和两个操作数指数差的不同,将状态空间分割为多个子状态空间,分别手工设定变量序,分别进行验证(实际验证中分成了 113 个,表 1 中 *PHDD 栏时间为各个子问题运行时间的总和,内存为各个子问题占用空间最大值). 基于 SAT 的方法无需人工干预.

龙芯 2 号的浮点除法模块核心部分采用重叠基 4 的 SRT 算法,此外还使用了在线舍入等优化技术. 浮点除法的验证在功能部件验证中有一定的特殊性. 它采用不流水的循环算法,不能使用一般运算电路中的去除锁存技术,需要分别验证循环初始、循环体和最后处理的正确性,通过数学归纳法保证总结果的正确性. 由于其它模块相对比较简单,表 2 中的数据都是验证循环体也就是 Overlapped Radix-4 SRT 求商模块的情况. 各行各列的含义同表 1. 基于 *PHDD 的方法手工指定了变量序,基于 SAT 的方法无需人工干预.

表 1 浮点加法功能部件验证时间

采用 *PHDD 的结果		采用 SAT(before)的结果			采用 SAT(after)的结果		
Time/s	Mem/MB	Time1/s	Time2/s	Mem/MB	Time1/s	Time2/s	Mem/MB
3929.19	73.51	>24h			>24h		
1927.33	100.67	25.44	1.98	4.33	10.07	2.10	5.11
490.98	59.37	18.35	1.07	4.78	7.45	3.90	5.67
2536.23	102.45	90.43	3.06	4.56	1.33	0.59	5.25

表 2 Overlapped Radix-4 SRT 求商模块验证时间

采用 *PHDD 的结果		采用 SAT(before)的结果			采用 SAT(after)的结果		
Time/s	Mem/MB	Time1/s	Time2/s	Mem/MB	Time1/s	Time2/s	Mem/MB
818.21	44.17	>24h			>24h		
1749.51	56.89	5.84	0.42	1.03	3.14	1.09	1.25
1569.39	52.34	4.50	0.20	1.25	0.57	0.18	1.34
1983.83	60.67	20.48	1.32	1.59	3.89	1.28	1.92

实验数据表明,对正确设计,基于 *PHDD 的方法在验证正确设计时比基于 SAT 的方法有效. 对有 bug 的设计,基于 SAT 的方法相比基于 *PHDD 的方法在时间和内存上都有着明显的优势. 而基于 *PHDD 的方法验证有 bug 的设计所用时间和内存一般都超过了验证正确设计所用的时间和内存(浮点加法验证中,验证错误电路时间较短是因为找到错误后就停下来没有处理剩余的子空间). 需要特别指出的是,基于 SAT 的方法能够在较短时间内发现一些传统仿真验证难以发现的高质量的 bug. 如求商循环部分的第一个错误设计中的 bug,类似于 Pentium 除法错,引进了标志子句后,基于 SAT 的

方法只用了 3.14s 就给出了反例,而 INTEL 上百万组仿真向量也没能发现.

从 Time1 列和 Time2 列的时间对比可以看出,复杂非线性运算电路验证中,计算数学公式的值所用时间是不可忽略的. 如果数学公式包含的任何一个变量被赋值时都去重新计算数学公式的上下界,有可能大大增加计算量,反而影响效率. 对比 SAT(before)和 SAT(after)中的 Time1 列,可以看出采用了标志子句技术后,基于 SAT 的方法速度有了明显的提高. 更重要的是,标志子句方法并没有太多增加数学公式的计算量.

7 分析和总结

基于 SAT 的运算电路查错方法无需手工指定变量序或进行状态和模块分割,自动化程度较高. SAT 求解算法对问题规模相对不敏感,因而该方法能处理较大规模的运算电路.更重要的是,它能有效地查找设计错误.标志子句技术使得 E-SAT 求解器能提前判断子句的值,避免了无谓的深度遍历,进一步提高了该方法处理复杂运算电路的速度.和比较适合于线性数学公式的基于上下界算术的优化方法相比,标志子句技术无需增加太多数学公式计算,更适合非线性数学公式.该方法在龙芯 2 号微处理器中的工业级运算电路上的成功,使我们有理由相信它能够广泛运用于运算电路验证.

对正确的电路,基于 SAT 的运算电路查错方法还不是很成熟.一个原因是验证正确电路时 E-SAT 求解器必须遍历整个状态空间,因此效率上肯定不如验证有错误的电路.另一个重要原因是父子句比较操作符两端的值相差越大,通过标志子句方法提前判断出父子句真值的概率越大;相差越小,提前判断出父子句真值的概率越小,必须进行深度遍历.上下界算术和标志子句方法的结合是未来的研究方向之一.

致 谢 感谢审稿专家的建议.他们细致的工作对本文裨益良多.严俊和刘生也提出了宝贵的建议,在此一并致谢!

参 考 文 献

- [1] McMillan K. Symbolic model checking: An approach to the state explosion problem[Ph. D. dissertation]. Pittsburgh, PA, USA: Carnegie Mellon University, 1992
- [2] Biere A, Cimatti A, Clarke E, Zhu Y. Symbolic model checking without BDDs//Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of System, LNCS 1579, 1999: 193-207
- [3] Clarke E M, Khaira M, Zhao X. Word level model checking—Avoiding the pentium FDIV error//Proceedings of the 33rd ACM/IEEE Design Automation Conference. Las Vegas, Nevada, USA, 1996: 645-648
- [4] Bryant R E, Chen Y A. Verification of arithmetic functions with binary moment diagrams//Proceedings of the 32nd ACM/IEEE Design Automation Conference. San Francisco, California, USA, 1995: 535-541
- [5] Clarke E M, Fujita M, Zhao X. Hybrid decision diagrams: Overcoming the limitations of MTBDDs and BMDs//Proceedings of the 1995 IEEE International Conference on Computer Aided Design. San Jose, California, USA, 1995: 159-163
- [6] Chen Y A, Bryant R E. *PHDD: An efficient graph representation for floating point circuit verification//Proceedings of the 1997 International Conference on Computer Aided Design. San Jose, California, USA, 1997: 2-7
- [7] Chen Y A, Bryant R E. ACV: An arithmetic circuit verifier//Proceedings of the 1996 International Conference on Computer Aided Design. San Jose, California, USA, 1996: 361-365
- [8] Wang Hai-Xia. Research on formal methods in arithmetic circuit verification[Ph. D. dissertation]. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 2004(in Chinese)
(王海霞. 运算电路的形式化验证方法研究[博士学位论文]. 中国科学院计算技术研究所, 北京, 2004)
- [9] Audemard Gilles, Bertoli Piergiorgio, Cimatti Alessandro, Kornilowicz Artur, Sebastiani Roberto. Integrating Boolean and mathematical solving: Foundations, basic algorithms, and requirements//Proceedings of the Artificial Intelligence, Automated Reasoning, and Symbolic Computation. Marseille, France, 2002: 231-245
- [10] Zhang Jian, Wang Xiao-Xu. A constraint solver and its application to path feasibility analysis. International Journal of Software Engineering and Knowledge Engineering, 2001, 11(2): 139-156
- [11] Ji Xiao-Hui, Huang Zhuo, Zhang Jian. On the integration of constraint programming and optimization. Chinese Journal of Computers, 2005, 28(11): 1790-1797(in Chinese)
(季晓慧, 黄拙, 张健. 约束求解与优化技术的结合. 计算机学报, 2005, 28(11): 1790-1797)
- [12] Plaisted D, Greenbaum S. A structure-preserving clause form translation. Journal of Symbolic Computation, 1986, 2(3): 293-304
- [13] Davis M, Logemann G, Loveland D. A machine program for theorem proving. Communications of the ACM, 1962, 5(7): 394-397
- [14] Moskewicz M, Madigan C, Zhao Y, Zhang L, Malik S. Chaff: Engineering an efficient SAT solver//Proceedings of the 38th Design Automation Conference. Las Vegas, NV, USA, 2001: 530-535
- [15] Liu Cong, Andreas Kuehlmann, Moskewicz Matthew W. CAMA: A multi-valued satisfiability solver//Proceedings of the 2003 IEEE International Conference on Computer Aided Design. San Jose, California, USA, 2003: 326-333
- [16] Parthasarathy Ganapathy, Iyer Madhu K, Cheng Kwang-Ting. Forrest brewer: RTL SAT simplification by Boolean and interval arithmetic reasoning//Proceedings of the 2005 IEEE International Conference on Computer Aided Design. San Jose, California, USA, 2005: 297-302



CHEN Yun-Ji, born in 1983, Ph. D. . His research interests include high performance computer architecture and verification of hardware.

ZHANG Jian, born in 1969, Ph. D. , professor, Ph. D. supervisor. His research interests include automated reason-

ing, constraint solving, program analysis and software testing.

SHEN Hai-Hua, born in 1971, Ph. D. , associate professor. Her research interests are computer architecture, microprocessor's design and verification.

HU Wei-Wu, born in 1968, Ph. D. , professor, Ph. D. supervisor. His research interests include high performance computer architecture, parallel processing and VLSI design.

Background

This Research is supported by the National Basic Research Program of China (973 Program), under grant No.2005CB321600, by the National Natural Foundation of China for Distinguished Young Scholars, under grant No. 60325205, by the Basic Research Foundation of the Institute of Computing Technology, Chinese Academy of Sciences, under grant No.20056020, by the National High-Tech Research and Development Plan of China, under grant Nos.2002AA110010, 2005AA110010 and 2005AA119020, and by the Knowledge Innovation Program of the Institute of

Computing Technology, Chinese Academy of Sciences, under grant No.20056240.

The authors mainly focus on design and verification of microprocessor Godson2. Conventional simulation method cannot cover the total state space of many module of modern microprocessor, so they dedicated on combination of formal verification and conventional verification methods. Till now, they have fully verified all function units of a modern microprocessor.