Word Level Model Checking - Avoiding the Pentium FDIV Error

Author: E. M. Clarke、M. Khaira和X. Zhao

摘要

Pentium中广为人知的除法错误强调了对算术电路进行形式验证的重要性。基于二进制决策图（BDDs）的符号模型检查技术已成功用于验证控制逻辑。然而，缺乏适当的表示方法来处理将布尔向量映射到整数的函数，这阻碍了该技术用于验证算术运算。

我们开发了一种新的验证算术电路的技术。这种新技术，称为字级模型检查，已成功用于验证基于Pentium使用的SRT算法的除法和平方根计算电路。该技术使得可以同时处理电路中的控制逻辑和数据路径。状态变量的总数超过600（远大于其他符号模型检查器之前处理的任何电路）。

1 引言

证明算术运算的正确性一直是一个重要问题。最近，由于Pentium中的除法错误而受到广泛关注，这个问题的重要性得到了强调。为了验证这类电路，必须表示和操作将布尔向量映射到整数值的函数。我们使用混合决策图（HDDs）来表示算术电路验证中出现的整数函数。对于代表数据位的状态变量，我们的表示方式类似于二进制矩阵图（BMD），而对于代表控制信号的状态变量，它表现得像多终端BDD（MTBDD）。通过使用这种表示，我们能够处理同时具有控制逻辑和宽数据路径的电路。

我们为将布尔向量映射到整数的函数提供的表示，使我们能够扩展时态逻辑模型检查，以便它可以处理算术电路。在传统的模型检查系统中，规格用命题时态逻辑表达，电路设计和协议被建模为状态转换系统。使用一种高效的搜索程序自动确定规格是否由转换系统满足。这种方法的主要缺点是，如果被验证的系统有许多可以并行转换的组件，可能会发生状态爆炸。由于使用了BDDs，可以验证的转换系统的大小已经大幅增加。尽管这种符号模型检查技术在验证控制逻辑方面取得了成功，但它不能直接用于验证算术电路。

在字级模型检查系统中，表示电路中节点的命题用BDDs表示，并且计算方式与原始的符号模型检查系统完全相同。字是命题的数组，每个命题对应一个位。表达式由应用于字的算术运算组成。可以使用算术运算的算法为字和表达式计算混合决策图。原子公式可以是表达式之间的关系，其BDD表示可以通过处理算术关系的算法计算出来。生成原子公式的BDD表示后，静态公式和时态公式的BDDs以与普通模型检查中相同的方式计算。特别是，不动点计算在两种情况下都是相同的。

通过使用字级模型检查系统，我们已成功验证了基于Pentium使用的SRT算法的除法和平方根计算电路。我们能够处理控制逻辑和数据路径。所有控制逻辑的有限状态机中的状态都已验证。此

外，我们已证明保证数据值正确性并防止溢出的不变属性。状态变量的总数超过600（远大于SMV之前检查的任何电路）。

2 字级CTL

基于二进制决策图（BDDs）的符号模型检查技术已成功用于验证控制逻辑。然而，缺乏适当的表示方法来处理将布尔向量映射到整数的函数，这阻碍了该技术用于验证算术运算。我们已经尝试了前面章节中介绍的不同表示方法。不幸的是，无论是使用多终端BDD（MTBDD）还是BDD数组表示，应用于算术电路验证都存在根本性问题。对于这类应用中出现的函数，可能的值的数量是位数的指数函数。因此，MTBDD的大小也是指数级的。另一方面，对BDD数组进行算术运算非常昂贵。特别是，由于组合乘法器中间位的BDD大小是其操作数长度的指数函数，因此BDD数组表示对于乘法来说是指数级的。

Bryant和Chen已经展示了，对于某些具有指数大小MTBDD的函数，二进制矩阵图（BMDs）可以提供一种紧凑的表示。他们已经使用这种表示来验证一些算术电路的数据路径。他们能够得出结论，如果电路和规格的BMD完全相同，则电路是正确的。然而，根据实现和控制逻辑的不同，可能存在电路正确但BMD不完全相同的情况。此外，由于他们的技术无法处理不等式，因此无法检查避免Pentium错误所需的某些属性。

我们使用混合决策图（HDDs）来表示算术电路验证中出现的整数函数。特别是，对于代表数据位的状态变量，我们使用逆Reed-Muller变换，而对于代表控制信号的状态变量，我们使用恒等变换。因此，对于数据变量，这种表示表现得像BMD，而对于控制变量，它表现得像MTBDD。通过使用这种表示，我们能够处理同时具有控制逻辑和宽数据路径的电路。由于这种表示是混合决策图的一个特例，因此可以应用前面章节中提到的所有算法。

通过使用这种表示，我们已经扩展了符号模型检查系统SMV，使其也能处理涉及数据字之间关系的属性。在原始的SMV系统中，原子公式只能包含状态变量。在扩展的系统中，我们允许原子公式包含表达式之间的等式或不等式。这些表达式表示为混合BDD。我们使用的逻辑如下：

- 原子命题：$Ap = \{p_1, \ldots, p_k\}$
- 命题公式：$Prop ::= Ap \mid Prop \wedge Prop \mid \neg Prop$
- 字：$Word ::= (Prop, Prop, \ldots, Prop)$
- 表达式：$Exp ::= Constant \mid Word \mid \text{next}(Word) \mid Exp \odot Exp \mid \text{if } SF \text{ then } Exp \text{ else } Exp$，其中$\odot$可以是$+, -, \times$。
- 原子公式：$AF ::= Ap \mid \{\mathbf{A} \mid \mathbf{E}\}(Exp \sim Exp)$，其中$\sim$可以是$=, <, \leq$。由于系统的非确定性行为，对于给定状态可能存在多个可能的下一个状态。因此，当在表达式中使用下一个状态运算符时，需要路径量词。
- 静态公式：$SF ::= AF \mid SF \wedge SF \mid \neg SF$
- 时态公式：$TF ::= SF \mid TF \wedge TF \mid \neg TF \mid \mathbf{AX}TF \mid \{\mathbf{A} \mid \mathbf{E}\}[TF\mathbf{U}TF]$

模型由以下部分给出：

- 状态：$S = 2^{Ap}$。
- 转换关系：$R \subseteq S \times S$
- 初始状态：$S_0 \subseteq S$
- 原子命题的赋值映射：$V : Ap \times S \to \{0, 1\}$

逻辑的语义由以下给出：

- 命题公式：$P : Prop \times S \to \{0, 1\}$

$$
\begin{aligned}
&P(p_i, s) = V(p_i, s) \\
&P(f_1 \wedge f_2) = P(f_1, s) \wedge P(f_2, s) \\
&P(\neg f, s) = \neg P(f, s)
\end{aligned}
$$

- 字：$W : Word \times S \to N$

$$
W((f_0, f_1, \ldots, f_n), s) = \sum_{i=0}^{n} P(f_i, s) 2^i
$$

- 表达式：$E : Exp \times S \times S \to N$。状态$s'$用于处理表达式中出现的下一个状态运算符。

$$
\begin{aligned}
&E(e_1 \odot e_2, s, s') = E(e_1, s, s') \odot E(e_2, s, s') \\
&E(\text{if } f \text{ then } e_1 \text{ else } e_2, s, s') = \\
&\quad \text{if } (s \vDash f) \text{ then } E(e_1, s, s') \text{ else } E(e_2, s, s') \\
&E(w, s, s') = W(w, s) \\
&E(\text{next}(w), s, s') = W(w, s')
\end{aligned}
$$

- 原子公式：

$$
\begin{aligned}
&s \vDash p_i \Leftrightarrow V(p_i, s) = 1 \\
&s \vDash \mathbf{A}(e_1 \sim e_2) \Leftrightarrow \\
&\quad \forall s'.\, R(s, s') \to (E(e_1, s, s') \sim E(e_2, s, s')) \\
&s \vDash \mathbf{E}(e_1 \sim e_2) \Leftrightarrow \\
&\quad \exists s'.\, R(s, s') \wedge (E(e_1, s, s') \sim E(e_2, s, s'))
\end{aligned}
$$

公式$\mathbf{A}(e_1 \sim e_2)$在状态s为真，当且仅当$(e_1 \sim e_2)$对所有后继状态都成立。同样地，$E(e_1 \sim e_2)$在状态s为真，当且仅当$(e_1 \sim e_2)$对某些后继状态成立。

- SF和TF的语义与CTL中的相同。

这种逻辑自然可以分为三层。顶层包含原子公式、静态公式和时态公式。第二层包含字和表达式。第三层包含原子命题和命题公式。顶层和底层的所有对象都是布尔函数，而第二层的对象是将布尔向量映射到整数的函数。因此，在字级模型检查系统中，所有的原子命题、命题、原子公式、静态公式和时态公式都表示为BDDs；而字和表达式则表示为混合决策图。

3 字级模型检查

模型检查是一种技术，用于找到状态转换图中满足给定CTL公式的状态集。有一个名为EMC的模型检查器，使用高效的图遍历技术解决这个问题。如果模型表示为状态转换图，算法的复杂度与图的大小和公式的长度成线性关系。这个算法在实践中相当快速。然而，当从具有许多进程或组件的有限状态并发系统中提取状态转换图时，模型的大小可能会爆炸。在符号模型检查系统中，使用二进制决策图（BDDs）来表示转换关系和状态集。模型检查过程是通过对这些BDD进行不动点操作来执行的。通过使用符号模型检查技术，可以验证的转换系统的大小已经大幅增加。尽管这些技术在验证控制逻辑方面取得了成功，但它们不能直接用于验证算术电路。这是因为涉及整数值的单词的表达式无法得到适当处理。

为了能够对前一节讨论的逻辑进行模型检查，需要实现对混合决策图的各种操作。我们考虑标量乘法、两个函数的加法和乘法，以及if-then-else操作。尽管这些算法的最坏情况复杂度可能是指数级的，但在实践中，这个算法表现得相当好。对字级属性进行模型检查还需要计算满足$f_1 \sim f_2$的赋值集，其中$\sim$可以是$=, \neq, <, \leq, >$或$\geq$。我们已经开发了一个高效的算法，可以计算满足算术关系的BDD。特别是，这个算法对线性表达式具有线性复杂度。

现在我们能够处理算术操作和算术关系，就有可能扩展符号模型检查算法，使其能够验证字级属性。转换关系和所有命题的BDD与原始符号模型检查系统中的生成方式完全相同。字的混合决策图表示可以使用上述操作计算为

$$\sum_{i=1}^{n}(\text{if } f_i \text{ then } 2^i \text{ else } 0)$$

大多数表达式的混合决策图表示可以使用类似的操作计算。唯一的例外是next操作，可以通过变量替换来执行。替换将混合决策图中的当前状态变量替换为相应的下一个状态变量。用于计算使代数关系成立的变量赋值集的BDD的算法可以用来计算原子公式的BDD。生成原子公式的BDD表示后，静态公式和时态公式的BDD以与普通模型检查中相同的方式计算。特别是，不动点计算在两种情况下都是相同的。

由于我们使用与普通模型检查算法中相同的算法来计算转换关系，字级模型检查算法在转换关系没有简洁表示时表现不佳。例如，考虑一个乘法器。设x和y为输入寄存器，z为输出寄存器。假设转换关系可以表示为：

$$\mathrm{Tr}(x, y, z) = \mathrm{Tr}'(x, y) \wedge (\mathrm{next}(z) = x \times y)$$

显然，转换关系的BDD表示具有指数大小，因为乘法器中间位的BDD表示是指数级的。有时可以通过转换关系的合取分解来避免这个问题。设$\bar{x}, \bar{y}$和$\bar{z}$为分别编码x, y和z的当前状态值的状态变量。设$\bar{x}', \bar{y}'$和$\bar{z}'$为分别编码x, y和z的下一个状态值的状态变量。假设我们想验证形式为$f(x, y, z)$的字级属性。如果出现$\mathrm{next}(z)$，我们可以在字级将其替换为$x \times y$，得到新的公式。希望得到的公式与z无关，公式的BDD表示可以表示为$f'(\bar{x}, \bar{y})$。在这种情况下，我们可以使用Tr'作为转换关系来执行不动点操作。即使f'依赖于z的某些位，我们通常也可以通过消除不需要的位的值的合取来获得更简单的转换关系。

4 验证SRT除法电路

通过使用字级模型检查系统，我们已成功验证了基于Pentium使用的SRT算法的除法和平方根计算电路。我们能够同时处理控制逻辑和数据路径。我们研究的除法电路有5个状态：空闲（idle）、初始化（init）、循环（loop）、最后（last）和余数（rem）。这个电路可以执行两种操作：除法和求余。当操作是除法时，计算的步骤是：

$$\text{idle} \rightarrow \text{init} \rightarrow \text{loop}^* \rightarrow \text{last} \rightarrow \text{idle}$$

当操作是求余时，步骤是：

$$\text{idle} \rightarrow \text{init} \rightarrow \text{loop}^* \rightarrow \text{last} \rightarrow \text{rem} \rightarrow \text{idle}$$

图1展示了循环状态时电路的数据路径。所有的字都有70位。然而，只有部分余数和除数的倍数的前导位被用来计算下一个周期的商位。

我们已经验证了包含控制逻辑和数据路径的电路。有限状态机中的所有状态都已经被检查。设r为部分余数，q为商，d为除数。我们检查了以下属性：

- 表达式$r + q \cdot d$始终等于左移后的被除数，即$r + q \cdot d = 2^{2k}$被除数。
- 计算不会溢出。这通过$-\frac{8}{3}d \leq r \leq \frac{8}{3}d$保证。

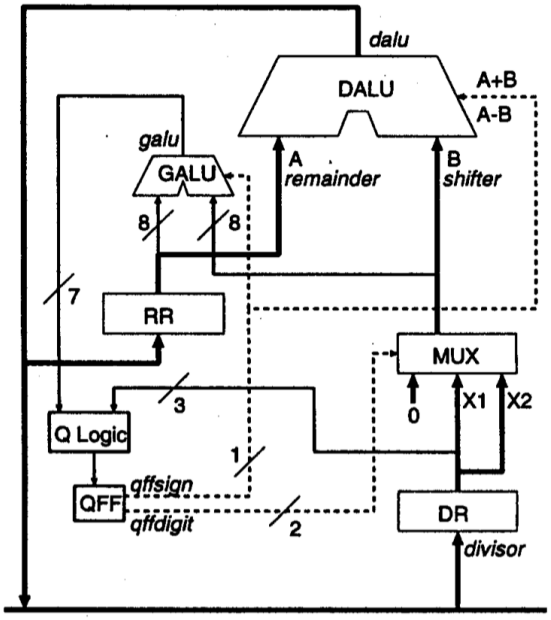

Figure 1: The data path for the division circuit

例如，我们已经证明在初始化状态，余数是被除数，商是零。因此，$r + q \cdot d$的初始值等于被除数。此外，上述不等式在初始化状态也成立。

SPEC $AG(\text{state} = \text{init} \rightarrow r = \text{dividend} \& q = 0)$

SPEC $AG(\text{state} = \text{init} \rightarrow (-8) \cdot d \leq 3 \cdot r \leq 8 \cdot d)$

我们还证明了在循环状态，不等式始终成立，并且$r + q \cdot d$对于左移是不变的。

上述属性足以保证在循环状态，$r + q \cdot d$始终等于左移后的被除数。类似的属性也被证明适用于最后和余数状态。此外，我们还验证了一个计算平方根的电路。我们验证的电路中状态变量的总数超过600（远大于SMV之前检查的任何电路）。

5 未来研究方向

我们已经使用字级符号模型检查来复现Pentium FDIV错误，并成功验证了修正后的电路。在本文中，我们描述了使用我们的字级模型检查器对基于SRT算法的浮点除法电路进行形式验证。我们计划在更多电路上进行实验。可能的应用包括浮点乘法器、浮点加法器等。

我们解决算术关系的算法对于线性方程和不等式表现非常好。尽管当前算法也可以处理一些非线性方程和不等式，但可能有可能扩展这个算法或找到一个新的算法来处理更复杂的非线性方程和不等式。

这种技术还有一个问题。它只能用于保持数据精确值的电路。当发生四舍五入时，函数变得不够规则，混合BDD表示的大小可能会爆炸。在这些情况下，四舍五入后得到的新值可以通过一组不等式来描述，验证过程就简化为解决这些系统。在另一个研究项目中，我们基于符号计算系统Mathematica构建了一个定理证明器，名为Analytica。Analytica在处理方程和不等式方面表现很好。我们相信，在一些修改后，Analytica将有助于解决计算机算术中由于四舍五入产生的不等式问题。

Reference

以下是论文中提到的参考文献及其对应的超链接：

1. R. E. Bryant and Y. A. Chen. Verification of arithmetic functions with Binary Moment Diagrams. In Proceedings of the 32nd ACM/IEEE Design Automation Conference, pages 535-541. IEEE Computer Society Press, June 1995.
   - 链接
2. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. Information and Computation, 98(2):142-170, June 1992.
   - 链接
3. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In Logic of Programs: Workshop, Yorktown Heights, NY, May 1981, volume 131 of Lecture Notes in Computer Science. Springer-Verlag, 1981.
   - 链接
4. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems, 8(2):244-263, 1986.
   - 链接

5. E. M. Clarke, M. Fujita, and X. Zhao. Hybrid Decision Diagrams - overcoming the limitations of MTBDDs and BMDs. In Proceedings of the 1995 Proceedings of the IEEE International Conference on Computer Aided Design, pages 159-163. IEEE Computer Society Press, November 1995.
    - 链接
6. E. M. Clarke and X. Zhao. Analytica: A theorem prover for Mathematica. The Journal of Mathematica, 3(1), 1993.
    - 链接
7. K. L. McMillan. Symbolic Model Checking. Kluwer Academic Publishers, 1993.
    - 链接
8. G. S. Taylor. Compatible hardware for division and square root. In Proceedings of the Fifth IEEE Symposium on Computer Arithmetic, 1993.
    - 链接