论 文

解码器存在性的停机算法

沈胜宇*，秦莹，肖立权，王克非，张建民，李思昆

国防科技大学计算机学院, 长沙410073
*沈胜宇E-mail: syshen@nudt.edu.cn, qy123@nudt.edu.cn, lqxiao@nudt.edu.cn, kfwang@nudt.edu.cn, jmzhang@nudt.edu.cn, skli@nudt.edu.cn

国家自然科学基金(批准号: 61070132)资助项目

摘要 对偶综合算法能够自动综合特定编码器的解码器. 该算法通过检查该编码器的输入能否被其输出唯一决定，来确认其解码器是否存在. 然而，如果解码器不存在则该算法不停机.

本文提出了一个新颖的算法以解决该问题. 对于该编码器的每一条路径, 该算法首先确认其输入能否被其输出唯一决定. 若是, 则解码器存在; 否则, 该算法检查该路径中是否存在环, 通过展开这些环，解码器的不存在性可以被确认.

我们在多个benchmark上运行了该算法，以展示其有效性和有用性, 包括PCI-E and Ethernet. 实验结果表明该算法总能够区分争取的错误的编码器并停机, 且该算法比目前算法快三倍以上.

关键词
停机
对偶综合

1 引言

在设计通讯和多媒体芯片过程中，最复杂和容易出错的工作是设计对偶电路对(E, E^{-1}), 其中编码器E将特定的信息流编码为易于传输和储存的格式, 而其对偶电路(或解码器) E^{-1}则致力于回复该信息. 为了提高该工作的设计质量和效率, 对偶综合算法[1, 2]被提出以自动综合对偶电路。其基本思想是检验参数对偶条件(parameterized complementary condition，缩写PC), 即编码器的输入能否被其输出唯一决定。

然而, 如果E^{-1}不存在则该算法不停机. 另一个停机算法[3]被提出以解决该问题。该算法首先构造形如洋葱圈的一系列PC的上估计, 并检验E是否在其中. 该算法非常慢且复杂. 为此本位提出另一个简单高效的算法. 对于编码器的每一条路径, 该新算法检查两种情况:

1. 正如原始的非停机算法[1, 2], 检查输入能否被输出唯一决定. 若是则解码器存在;
2. 否则, 检查在上述路径中是否存在环. 如图1, 图1a)包含三个子路径a, b and c. 其中b是一个环。通过将该环展开为图1b)中的长路径，则在更长路径上解码器的不存在性可以被确认.

该算法已经在OCaml语言中实现. 所有产生的SAT实例均由Zchaff[4]求解. 实验对象包括多个来自于工业项目的复杂编码器，如PCI-E[5]和以太网[6]), 以及他们经过简单修改的，没有对应解码器的的错误版本. 实验结果表明本文算法始终能够停机并区分正确和不正确的版本, 并且比现有算法[3]快3倍以上. 所有实验结果和程序可以从`http://www.ssypub.org`下载.

在此我们区分两种情形：

1. 标准数据路径电路: 这些电路通常作为数字信号处理元件, 如快速傅立叶变换(FFT) 和离散余弦变换(DCT). 这些电路通常具有标准而且高度优化的实现, 如Xilinx core generator[7] and Synopsys DesignWare 库[8]. 因此这些解码器不需要我们的算法.

2. 非标准控制电路: 此类电路, 如PCI-E[5]和以太网[6], 通常用于处理通讯协议, 通常不具有标准实现. 本文算法是为他们设计的。

本文按如下方式组织. 背景信息在2节给出. 算法在3节中介绍, 而4节描述了如何移除冗余字符以减小面积. 实验结果在5节给出，相关工作在6节. 最后, 7总结全文。

2 背景知识

2.1 命题逻辑可满足性问题

布尔集合记为$B = \{0, 1\}$. 对于在变量集合V上的布尔公式F, 其命题逻辑可满足性问题(SAT)是寻找满足赋值$A : V \to B$, 以便F取值为1. 如果A存在，则F是可满足的; 否则, 其实不可满足的.

一个确认A是否存在的计算机程序称为SAT求解器, 如Zchaff[4], Grasp[9], Berkmin[10], 和MiniSAT[11]. 一个有待SAT求解器求解的公式称为SAT实例.

2.2 递归直径

编码器E可以用Mealy有限状态机[12]建模.

Definition 1. ***Mealy*有限状态机**是一个5元组$M = (S, s_0, I, O, T)$, 其中S是有限状态集合, $s_0 \in S$为初始状态, I为输入字符集合, O为输出字符集合, 迁移函数$T : S \times I \to S \times O$用于从当前状态和输入字符计算次态和输出字符.

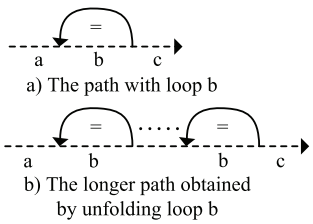

图 1 展开环以证明在更长的路径上不存在解码器

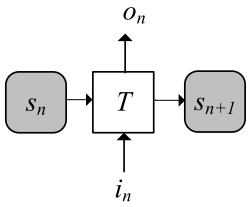

图 2 Mealy有限状态机

如图2所示, 在本文中, 状态由灰色圆角框表示, 而迁移函数T由白色方框表示. 在第n个周期的状态，输入字符和输出字符表示为s_n, i_n和o_n. 而从第n个到第m个周期的状态，输入字符和输出字符序列表示为s_n^m, i_n^m和o_n^m. 一条路径是一个状态序列s_n^m，其中对所有$n \leqslant j < m$有$\exists i_j o_j (s_{j+1}, o_j) \equiv T(s_j, i_j)$. 一个环是一个路径$s_n^m$，其中$s_n \equiv s_m$.

Kroening et al. [13]定义Mealy有限状态机M的状态变量递归半径$rrd(M)$为, 从初始状态开始，不包含环的最长路径.

$$\begin{aligned} &rrd(M) \stackrel{def}{=} \max\{i | \exists s_0 \ldots s_i i_0 \ldots i_i o_0 \ldots o_i : \\ &I(s_0) \wedge \bigwedge_{j=0}^{i-1} (s_{j+1}, o_j) \equiv T(s_j, i_j) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k\} \end{aligned} \tag{1}$$

本文中, 我们定义非初始化的状态变量递归半径$uirrd(M)$, 为最长无环路径.

$$\begin{aligned} &uirrd(M) \stackrel{def}{=} \max\{i | \exists s_0 \ldots s_i i_0 \ldots i_i o_0 \ldots o_i : \\ &\bigwedge_{j=0}^{i-1} (s_{j+1}, o_j) \equiv T(s_j, i_j) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k\} \end{aligned} \tag{2}$$

两者的唯一区别是$uirrd$不考虑初始状态. 这些定义仅用于证明下文的定理. 本文不需要计算这些半径.

2.3 检验E^{-1}存在性的原始非停机算法

原始的对偶综合算法[1]包含两步：检验E^{-1}的存在性和特征化其布尔函数。我们将仅介绍第一步。

根据我们以往的研究经历[1, 2, 3], 许多通讯协议是loseless的, 即每个输入字符均能够由其输出字符序列唯一决定.

形式化的, 如图3所示, E^{-1}的充分条件是, 存在三个参数p, d 和l, 使的编码器E的输入字符i_n能够被E的输出序列o_{n+d-l}^{n+d-1}唯一决定. 其中d是o_{n+d-l}^{n+d-1}和i_n之间的相对延迟, l是o_{n+d-l}^{n+d-1}的长度, 而p是用于剔除部分非可达状态集合的前置路径长度. 因此, PC)[1]可以形式化的定义为:

Definition 2. **参数对偶条件*(PC)*：** *对于编码器E, $E \vDash PC(p, d, l)$成立当i_n可以被o_{n+d-l}^{n+d-1}唯一决定. 这等价于公式(3)中的$F_{PC}(p, d, l)$的不可满足性. 我们另外定义$E \vDash PC$为$\exists p, d, l : E \vDash PC(p, d, l)$.*

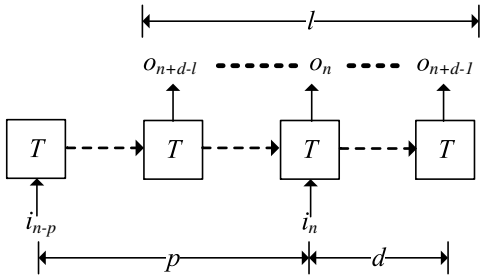

图 3 参数对偶条件

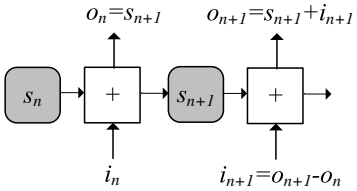

图 4 违法因果关系的电路

$$F_{PC}(p,d,l) \stackrel{def}{=} \left\{ \begin{array}{ll} & \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1},o_m) \equiv T(s_m,i_m)\} \\ \wedge & \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1},o'_m) \equiv T(s'_m,i'_m)\} \\ \wedge & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\ \wedge & i_n \neq i'_n \end{array} \right\} \tag{3}$$

公式(3)的第2和第3行分别对应于编码器的两条路径. 他们之间的唯一差别在于一个prime被附加到第三行的每一个变量上. 第4行强制上述两行的输出相等, 而最后一行强制他们的输入不相等.

该不停机算饭[1]简单的遍历所有p,d 和l的组合, 从小到大, 直到找到p,d和l使的公式(3)不可满足. 如此即可证明解码器存在. 然而, 如果解码器不存在, 该算法将永不停机. 该问题将在下一节中解决.

如图3所示, 如果$l > d$, 则, 为了计算i_n, 我们需要知道o_m其中$m < n$. 这看起来似乎违反了因果关系. 非形式化的, 在不同的状态中, 编码器会为相同的输入i_n产生不同的输出. 旖旎次对于$m < n$的o_m的知识有助于标定处理i_n的状态. 该问题能够用图4的电路解释. 假设其迁移函数T为:

$$\begin{aligned} s_{n+1} &= i_n + s_n \\ o_n &= i_n + s_n \end{aligned} \tag{4}$$

直观的, 该电路将输入和当前状态求和, 并将结果送到次态和输出. 因此, 为了恢复i_{n+1}, 输出字符o_n需要被从o_{n+1}中减去. 在这种情况下, l是1, 而d是0. 这就解释了为什么l可以比d打.

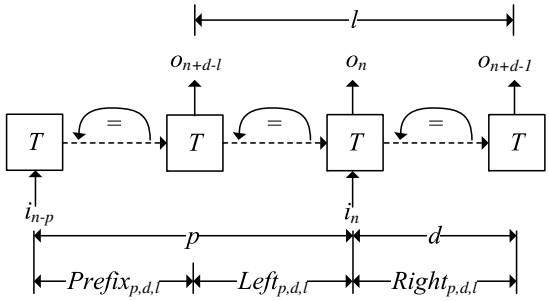

图 5 环形非对偶条件

3 确认解码器存在性的停机算法

3.1 确认解码器不存在

定义2中的PC只定义了如何确认E^{-1}存在. 而如何确认E^{-1}不存在则没有定义. 因此, 得到停机算法的关键在于如何确认E^{-1}不存在.

根据定义2和图3, E^{-1}存在当存在$< p,d,l >$, 使的$E \vDash PC(p,d,l)$成立. 因此, 直观的, E^{-1}不存在如果对于任意$< p,d,l >$, 总能找到$< p',d',l' >$满足$p' > p$,$l' > l$和$d' > d$, 使的$E \vDash PC(p',d',l')$不成立.

这种情况可以被图5中的SAT实例检测, 该图类似于图3, 出了三个约束被插入以检测状态序列s_{n-p}^{n+d-l}, $s_{n+d-l+1}^{n}$ 和s_{n+1}^{n+d} 上的环. 如果对于$< p,d,l >$该SAT实例是可满足的, 则这三个环可以如下展开: 假设s_{n-p}^{n+d-l}, $s_{n+d-l+1}^{n}$ 和s_{n+1}^{n+d} 上的环的长度分别为l_1, l_2和l_3, 且这些环被展开q次. 则, 从这次展开中得到的SAT实例如图6所示. 该SAT实例对应于$F_{LN}(p",d",l")$, 其中:

$$\begin{aligned} p" &= l_1 * q + (d - l + p - l_1) + l_2 * q + (l - d - l_2) \\ d" &= l_3 * q + (d - l_3) \\ l" &= l_2 * q + (l - d - l_2) + l_3 * q + (d - l_3) \end{aligned} \tag{5}$$

很显然对于任意$< p,d,l >$和$< p',d',l' >$, 总存在q, 使的展开得到的$s_{n-p"}^{n+d"-l"}$,$s_{n+d"-l"+1}^{n}$和$s_{n+1}^{n+d"}$ 不短于$s_{n-p'}^{n+d'-l'}$,$s_{n+d'-l'+1}^{n}$和$s_{n+1}^{n+d'}$. 该SAT实例的可满足性将在引理1中证明. 这意味着对任意$< p',d',l' >$, 我们总能找到$< p",d",l" >$, 使的$E \vDash PC(p",d",l")$不成立. 因此解码器存在.

从公式(3)的第二和第三行可知, 实际上存在两条路径, 因此这些环需要在这两条路径上同时检测, 也就是说, 需要在如下定义的乘积状态机M^2上检测环:

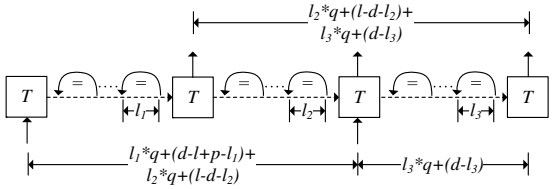

图 6 展开q次的环形非对偶条件

Definition 3. 乘积状态机: 对于*Mealy*状态机$M = (S, s_0, I, O, T)$, 其乘积状态机定义为$M^2 = (S^2, s_0^2, I^2, O^2, T^2)$, 其中

1. $S^2 = S \times S$
2. $s_0^2 = s_0 \times s_0$
3. $I^2 = I \times I$
4. $O^2 = O \times O$
5. T^2定义为$(< s_{m+1}, s'_{m+1} >, < o_m, o'_m >) = T^2(< s_m, s'_m >, < i_m, i'_m >)$, 其中$(s_{m+1}, o_m) = T(s_m, i_m)$且$(s'_{m+1}, o'_m) = T(s'_m, i'_m)$.

因此, 环形非对偶条件可以定义如下以确认解码器E^{-1}不存在:

Definition 4. 环形非对偶条件*(LN)*: 对于编码器E及其*Mealy*状态机$M = (S, s_0, I, O, T)$, 假设其乘积状态机为$M^2 = (S^2, s_0^2, I^2, O^2, T^2)$, 则$E \vDash LN(p, d, l)$成立当且仅当$i_n$不能被$o_{n+d-l}^{n+d-1}$唯一决定, 而且在状态序列$(s^2)_{n-p}^{n+d-l}$,$(s^2)_{n+d-l+1}^{n}$和$(s^2)_{n+1}^{n+d}$上存在环. 这等价于公式*(6)*中的$F_{LN}(p, d, l)$的不可满足性. $E \vDash LN$被进一步的定义为$\exists p, d, l : E \vDash LN(p, d, l)$.

$$
F_{LN}(p,d,l) \overset{def}{=}
\left\{
\begin{array}{ll}
 & \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1}, o_m) \equiv T(s_m, i_m)\} \\
\wedge & \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\} \\
\wedge & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\
\wedge & i_n \neq i'_n \\
\wedge & \bigvee_{x=n-p}^{n+d-l-1} \bigvee_{y=x+1}^{n+d-l}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\
\wedge & \bigvee_{x=n+d-l+1}^{n-1} \bigvee_{y=x+1}^{n}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\
\wedge & \bigvee_{x=n+1}^{n+d-1} \bigvee_{y=x+1}^{n+d}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\}
\end{array}
\right\} \tag{6}
$$

通过比较公式(3) 和(6), 很明显他们之间的唯一差别在于公式(6)中最后三行新插入的约束, 他们将被用于检测下面三个路径上的环:

$$
\begin{aligned}
Prefix_{p,d,l} &= (s^2)_{n-p}^{n+d-l} \\
Left_{p,d,l} &= (s^2)_{n+d-l+1}^{n} \\
Right_{p,d,l} &= (s^2)_{n+1}^{n+d}
\end{aligned} \tag{7}
$$

该方法的正确性将在下一小节证明.

3.2 正确性证明

在证明该方法的正确性之前, 我们需要一些引理.

Lemma 1. 对于图6中的$F_{LN}(p", d", l")$, $E \vDash LN(p, d, l)$导致$E \vDash LN(p", d", l")$.

Proof. 公式$F_{LN}(p", d", l")$为:

$$F_{LN}(p", d", l") \overset{def}{=}$$
$$\left\{\begin{array}{ll} & \bigwedge_{m=n-p"}^{n+d"-1}\{(s_{m+1}, o_m) \equiv T(s_m, i_m)\} \\ \wedge & \bigwedge_{m=n-p"}^{n+d"-1}\{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m)\} \\ \wedge & \bigwedge_{m=n+d"-l"}^{n+d"-1} o_m \equiv o'_m \\ \wedge & i_n \neq i'_n \\ \wedge & \bigvee_{x=n-p"}^{n+d"-l"-1} \bigvee_{y=x+1}^{n+d"-l"}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+d"-l"+1}^{n-1} \bigvee_{y=x+1}^{n}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+1}^{n+d"-1} \bigvee_{y=x+1}^{n+d"}\{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \end{array}\right\} \tag{8}$$

$E \vDash LN(p, d, l)$意味着$F_{LN}(p, d, l)$可满足. 假设其满足赋值为A. 图7中的编号为1至12的有向边给出了$F_{LN}(p, d, l)$和$F_{LN}(p", d", l")$的对应关系.

有向边2和3意味着将$Right_{p,d,l}$中的满足赋值应用于$Right_{p",d",l"}$中的展开环. 而有限变1和4意味着将非环部分的赋值应用于$Right_{p",d",l"}$. 基于有向边1至4, 路径$Right_{p",d",l"}$是可满足的.

类似的, 路径$Prefix_{p",d",l"}$和$Left_{p",d",l"}$也能够被A满足. 因此公式(8)的第二行可以被A满足.

类似的, 公式(8)的第三至第五行也能够被A满足.

同时, 在$Prefix_{p",d",l"}$, $Left_{p",d",l"}$和$Right_{p",d",l"}$中有q个环, 这将使得公式(8)的最后三行被满足.

因此,$F_{LN}(p, d, l)$的满足赋值A可以使的$F_{LN}(p", d", l")$被满足. 得证. □

Lemma 2. 对于$< p, d, l >$和$< p', d', l' >$, 如果$Prefix_{p',d',l'}$,$Left_{p',d',l'}$和$Right_{p',d',l'}$不短于$Prefix_{p,d,l}$,$Left_{p,d,l}$和$Right_{p,d,l}$, 则$E \vDash PC(p, d, l) \rightarrow E \vDash PC(p', d', l')$.

Proof. 很明显$F_{PC}(p, d, l)$是$F_{PC}(p', d', l')$的子公式, 因此前者的不可满足导致了后者的不可满足。因此, $E \vDash PC(p, d, l) \rightarrow E \vDash PC(p', d', l')$成立. □

下列的两个定理证明了$E \vDash LN \leftrightarrow \neg\{E \vDash PC\}$.

Theorem 1. $E \vDash LN \rightarrow \neg\{E \vDash PC\}$

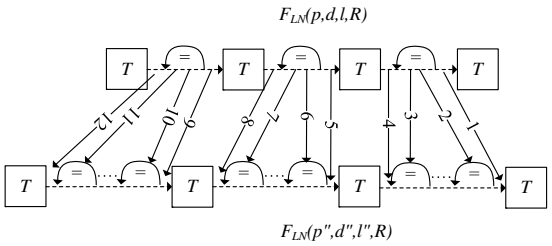

图 7 $F_{LN}(p, d, l)$和$F_{LN}(p", d", l")$之间的对应关系

Proof. 基于反证法. 假设$E \vDash LN$和$E \vDash PC$都成立. 这意味着存在$< p,d,l >$和$< p',d',l' >$, 使的$E \vDash PC(p,d,l)$和$E \vDash LN(p',d',l')$成立.

一方面, $E \vDash LN(p',d',l')$成立意味着在$Prefix_{p',d',l'}$,$Left_{p',d',l'}$和$Right_{p',d',l'}$上存在环. 通过展开这些环, 可以得到另一个$< p",d",l" >$, 使的:

1. $Prefix_{p",d",l"}$,$Left_{p",d",l"}$ 和$Right_{p",d",l"}$不短于$Prefix_{p,d,l}$,$Left_{p,d,l}$ 和$Right_{p,d,l}$;
2. 有引理1可知, $F_{LN}(p",d",l")$是可满足的.

$F_{PC}(p",d",l")$是$F_{LN}(p",d",l")$的子公式, 因此$F_{PC}(p",d",l")$也是可满足的, 这意味着$E \vDash PC(p",d",l")$不成立.

另一方面, 由引理2可知, $E \vDash PC(p",d",l")$成立.

该矛盾导致定理得到证明. □

Theorem 2. $E \vDash LN \leftarrow \neg\{E \vDash PC\}$

Proof. 基于反证法. 假设$E \vDash LN$和$E \vDash PC$都不成立. 则对于任意$< p,d,l >$和$< p',d',l' >$, $F_{PC}(p,d,l)$是可满足的, 而$F_{LN}(p',d',l')$是不可满足的.

因此, 假设$uirrd(M^2)$是E的乘积状态机的非初始化状态递归半径. 定义$< p,d,l >$为:

$$\begin{aligned} p &= uirrd(M^2) * 2 + 2 \\ d &= uirrd(M^2) + 1 \\ l &= uirrd(M^2) * 2 + 2 \end{aligned} \tag{9}$$

基于该定理, 很明显$Prefix_{p,d,l}$,$Left_{p,d,l}$ 和$Right_{p,d,l}$都长于$uirrd(M^2)$. 这意味着在这三个路径上存在环, 这意味着$F_{LN}(p,d,l)$可满足. 这与$F_{LN}(p',d',l')$对任意$< p',d',l' >$不可满足的事实矛盾.

该矛盾导致本定理得到证明. □

定理1 和2表明, 一个停机算法可以按照下述方法实现从小到大便利所有$< p,d,l >$的组合, 在每个循环中检验$E \vDash PC(p,d,l)$和$E \vDash LN(p,d,l)$. 该过程最终总会给出$E \vDash PC$和$E \vDash LN$之间的一个且仅有一个答案. 下一小节将给出该算法的实现.

3.3 算法实现

不想原始算法便利p, d和l的所有组合[1, 2], 算法1的行2,3 和4确保$Prefix_{p,d,l}$,$Left_{p,d,l}$和$Right_{p,d,l}$的长度不短于第一行中的x. 因此, 通过不便利这些冗余组合, 算法速度得以大大提高.

有定理1和2, 算法1最终总能够在行6或9结束.

4 剔除冗余输出字符

虽然算法1足以确认E^{-1}是否存在, 但是由算法1第6行找到的参数存在冗余, 这将造成电路面积和特征化上较大的开销.

例如, 如图8所示, 假设l是导致$E \vDash PC(p,d,l)$成立的最小值, 而且$l < d$, 这意味着i_n能够被o_k唯一确定, 其中$k > n$. 进一步假设算法1的第6行证明了$E \vDash PC(p,d,l')$. 很明显$l' > d$, 这将导致i_n依赖于某些o_k,

Algorithm 1 *check_PCLN*

for $x = 1 \to \infty$ **do**
  $p = 2x$
  $d = x$
  $l = 2x$
  if $F_{PC}(p, d, l)$ is unsatisfiable **then**
    print "E^{-1} exists with $< p, d, l >$"
    halt;
  else if $F_{LN}(p, d, l)$ is satisfiable **then**
    print "E^{-1} does not exist"
    halt;
  end if
end for

Algorithm 2 *RemoveRedundancy*(p, d, l)

for $p' = p \to 0$ **do**
  if $F_{PC}(p' - 1, d, l)$可满足 **then**
    break
  end if
end for
for $d' = d \to 0$ **do**
  if $F_{PC}(p', d' - 1, l)$可满足 **then**
    break
  end if
end for
for $l' = 1 \to l - (d - d')$ **do**
  if $F_{PC}(p', d', l')$不可满足 **then**
    break
  end if
end for
print "最终结果为$< p', d', l' >$"

表 1 实验对象信息

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
Verilog代码行数	214	466	24	1139	1073
寄存器个数	15	135	58	22	48
数据路径宽度	8	64	66	10	10

其中$k \leqslant n$. 因此$o_{n+d-l'}^{n+d-l-1}$是冗余输出字符, 他们应当被移除以免在E^{-1}中被当作寄存器实例化. 同时, 从图8中可知, 较大的p和d将导致在特征化中较大的时间开销.

因此, 算法2将被用于在将$< p, d, l >$传送给特征化算法之前缩减他们.

5 实验结果

本算法在OCaml语言中实现. 产生的SAT实例由Zchaff求解器[4]求解. 所有实验结果可以从`http://www.ssypub.org`下载.

5.1 实验对象

表1给出了实验对象的信息.

1. XGXS是符合IEEE-802.3ae 2002标准[6]短句48的以太网编码器.
2. XFI 是符合同一个IEEE标准的短句49的以太网编码器.
3. 66位scrambler用于确保特定数据流中包含足够多的0-1翻转，以确保其能够穿过高噪声通讯信道.
4. 一个PCI-E[5]物理层编码模块.

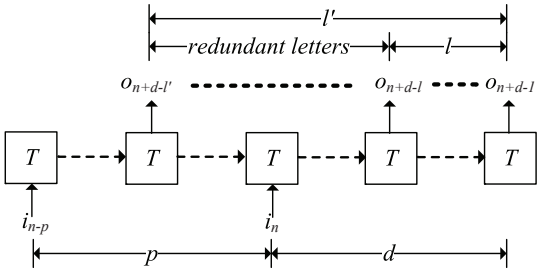

图 8 冗余字符

5. Sun’s OpenSparc T2 处理器的以太网编码器.

5.2　对正确设计的编码器确认其解码器的存在

表2的第1和3行在[3]和本文算法之间比较了检查$E \models PC$的运行时间. 第4行给出了本文算法在[3]基础上的改进百分比. 很明显本文算法比[3]有明显加速.

第2和5行比较了发掘的参数值, 在p上有一些差别. 这是由检查不同组合的次序差异造成的.

5.3　比较解码器面积

表3比较了手工构造的解码器和本文算法构造的解码器的面积. 这些解码器是使用Synopsys Design-Compiler提供的LSI10K单元库进行综合的.

表3表明, 除了最复杂的XFI, 本文算法的结果均好于手工构造的解码器. 然而, 这种比较是不公正的，因为手工构造的解码器中包含额外功能, 如测试逻辑.

另一方面, 对于XFI, 本文算法的结果是手工解码器的两倍. 这意味着在未来的工作中我们需要致力于改善面积.

5.4　比较解码器时序

表4比较了手工构造的解码器和本文算法构造的解码器的时序性能. 他们的综合设置与小节5.3相同. 对于所有的电路, 本文算法构造的解码器均具有更好的时序性能.

5.5　对非正确的编码器确认其解码器不存在

为了进一步展示本文算法的有用性, 我们需要一些没有对应解码器的非正确编码器. 我们通过修改正确编码器的输出语句, 使的他们对相同的不同的输入字符产生相同的输出序列. 如此, 即可放着输入字符i_n能够从输出序列中恢复出来.

表5的第一行给出了[3]的算法在这些非正确编码器上的运行时间, 而第二行给出了本文算法的运行时间. 第三行给出了本文算法相对于[3]的改善. 这表明本文算法总是能够在非正确的编码器上停机, 而且比[3]的算法有明显的速度提升.

6　相关工作

6.1　C对偶综合

对偶综合的概念首先在[1]中被提出. 其主要缺点在于其不停机性, 且其特征化算法的运行开销过大.

停机问题在[3]中通过构造类似于洋葱圈的嵌套上估计被首次解决, 而在[2]中，我们通过抽取不可满足核降低时间开销.

6.2 程序取反

从Gulwani[16]可知, 程序取反是针对特定程序P, 推倒其反运算P^{-1}. 因此程序取反非常类似于对偶综合.

程序取反的早期工作使用基于证明的方法[17], 但是该算法只能支持非常简单的语法类型和非常小的程序.

Glück et.al [18]使用基于LR的语法分析方法，去除一阶函数程序中的不确定性，从而完成取反. 然而使用一节函数语言的要求与本文的应用背景不兼容.

Srivastava et.al [19]假设反程序和原始程序具有相似的程序结构, 因此反程序的可能结构可以通过自动发掘原始程序中的表达式，谓词和控制流得到. 该算法递归的移除那些不能满足反程序要求的路径，直至留下唯一一个可能的路径. 因此该算法只能保证存在解, 而不能保证该解的正确性.

6.3 限界模型检验的完备性

限界模型检验(BMC) [14]是一种只考虑有限长度路径的模型检验算法. 因此它是非完备的. 许多研究人员致力于为BMC寻找完备算法.

其中一个研究方向[14, 13]致力于寻找一个限界b, 使的当特定属性在该限界内成立时, 其总能在所有长度的路径上成立. 算法1的行8寻找能够保证解码器不存在的p,d和l的值, 这非常类似于[14, 13].

另一个研究方向[15]致力于找到一个归纳模式, 使得特定属性能够成为符合归纳的正确属性. 本文算法通过展开环证明解码器的不存在. 这非常类似于**[15]**的归纳模式.

6.4 协议转换器综合

协议转换器综合是在不同的通讯协议之间综合一个接口逻辑的问题. 这与本文的关系在于两者皆致力于综合通讯电路.

Avnit et.al [20, 21]首先提出了描述通讯协议的通用模型. 然后提出了一个算法用于判决某个协议中是否有某一功能无法在另一个协议中表达. 最后, 转换器即为接口buffer的控制函数的不动点. Avnit et.al[22]引进了更为高效的解空间遍历算法.

7 结论

本文提出了一个更为简单和快速的停机算法，以确认特定编码器的解码器是否存在. 理论分析表明本文算法总能够停机. 实验表明本文算法比现有算法[3]有明显的速度提升.

Acknowledgment

本文作者感谢编辑和未具名的审稿人的辛勤工作.

本文的工作受到中国国家自然科学基金会项目60603088和61070132的资助.

参考文献

1 S. Shen, J. Zhang, Y. Qin, and S. Li, ”Synthesizing complementary circuits automatically,” in Proc. IEEE ICCAD, Nov. 2009, pp. 381-388.

2 S. Shen, Y. Qin, K. Wang, L. Xiao, J. Zhang, and S. Li, ”Synthesizing Complementary Circuits Automatically,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 29, no. 8, pp. 1191-1202, Aug. 2010.

3 S. Shen, Y. Qin, J. Zhang, and S. Li, ”A Halting Algorithm to Determine the Existence of Decoder,” in Proc. IEEE FMCAD, Oct. 2010, pp. 91-100.

4 M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, ”Chaff: Engineering an efficient SAT solver,” in Proc. IEEE Int. DAC, Jun. 2001, pp. 530-535.

5 ”PCI Express Base Specification Revision 1.0”. [Online]. Available: http://www.pcisig.com

6 ”IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation”, IEEE Std. 802.3, 2002.

7 Xilinx. (2008) Xilinx core generator system. [Online]. Available: http://www.xilinx.com/tools/coregen.htm

8 Synopsys. (2008) Designware library. [Online]. Available: http://www.synopsys.com/IP/DESIGNWARE/Pages/default.aspx

9 J. P. M. Silva and K. A. Sakallah, ”GRASP - a new search algorithm for satisfiability,” in Proc. IEEE ICCAD, Nov. 1996, pp. 220-227.

10 E. Goldberg and Y. Novikov, ”BerkMin: A fast and robust Sat-solver,” Discrete Applied Mathematics, vol. 155, no. 12, pp. 1549-1561, Dec. 2007.

11 N. Eén and N. Sörensson., ”Extensible SAT-solver,” in Proc. Int. Conf. SAT, May 2003, pp. 502-518.

12 G. H. Mealy , ”A method for synthesizing sequential circuits,” Bell Syst. Tech. J., vol. 34, pp. 1045-1079, Jan. 1955.

13 D. Kroening and O. Strichman, ”Efficient Computation of Recurrence Diameters,” in Proc. Int. Conf. VMCAI, Jan. 2003, pp. 298-309.

14 E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu, ”Bounded Model Checking Using Satisfiability Solving,” Formal Methods in System Design, vol. 19, no. 1, pp. 7-34, Jan. 2001.

15 M. Sheeran, S. Singh, and G. Stalmarck, ”Checking Safety Properties Using Induction and a SAT-Solver,” in Proc. IEEE FMCAD, Oct. 2000, pp. 108-125.

16 S. Gulwani, ”Dimensions in program synthesis,” in Proc. ACM PPDP, Jul. 2010, pp. 13-24.

17 E. W. Dijkstra, ”Program Inversion,” in Program Construction, 1978, pp. 54 - 57.

18 R. Glück and M. Kawabe, ”A method for automatic program inversion based on LR(0) parsing,” Fundam. Inf., vol. 66, no. 4, pp. 367-395, Dec. 2005.

19 S. Srivastava, S. Gulwani, S. Chaudhuri, and J. Foster, ”Program inversion revisited,” Technical Report MSR-TR-2010-34, Microsoft Research, 2010.

20 K. Avnit, V. D’Silva, A. Sowmya, S. Ramesh, and S. Parameswaran, ”A Formal Approach To The Protocol Converter Problem,” in Proc. IEEE DATE Conf. Exposit., Mar. 2008, pp. 294-299.

21 K. Avnit, V. D’Silva, A. Sowmya, S. Ramesh, and S. Parameswaran, ”Provably correct on-chip communication: A formal approach to automatic protocol converter synthesis,” ACM Trans. Design Autom. Electr. Syst., vol. 14, no. 2, pp. 1-41, Apr. 2009.

22 K. Avnit, and A. Sowmya, ”A formal approach to design space exploration of protocol converters,” in Proc. IEEE DATE Conf. Exposit., Mar. 2009, pp. 129-134.

表 2 对正确设计的编码器的实验

		XGXS	XFI	scrambler	PCI-E	T2 ethernet
[3]	Time to check *PC*(sec)	1.06	70.52	5.74	2.40	66.37
	d, p, l	1,1,1	0,3,2	0,2,2	2,1,1	4,1,1
This paper	Time to check *PC*(sec)	0.29	17.86	2.67	0.47	29.64
	improve %	72.64	74.67	53.48	80.42	55.34
	d, p, l	1,2,1	0,3,2	0,2,2	2,2,1	4,4,1

表 3 比较解码器面积

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
手工构造的解码器	921	6002	1629	852	1446
本文算法构造的解码器	700	12754	1455	455	552

表 4 比较关键路径长度

	XGXS	XFI	scrambler	PCI-E	T2 ethernet
手工构造的解码器	12.33	46.65	6.54	19.03	23.36
本文算法构造的解码器	11.96	28.13	6.54	9.09	12.69

表 5 比较在非正确的编码器上的运行时间

	XGXS	XFI	scra-mbler	PCI-E	T2 et-hernet
[3]的算法(秒)	0.98	35.08	2.54	1.36	17.39
本文算法(秒)	0.16	7.59	1.17	0.33	2.19
改善%	83.67	78.36	53.94	75.74	87.41