论 文

对偶综合的断言推导

沈胜宇*，秦莹，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, 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 对偶综合能够自动生成特定编码器的解码器. 然而, 其用户需要在特定的配置信号上给出断言以防止编码器到达非工作状态. 为了避免这一烦琐工作, 本文提出了自动推导断言的算法.

首先, 我们提出了能够针对任意配置信号值, 确定解码器是否存在的停机算法. **其次**, 对以每一个导致解码器不存在的非法配置信号值, 我们使用cofactoring和Craig插值以推导一个覆盖另一个较大的非法配置信号值的集合. 该步骤将被重复直至所有非法配置信号值均被剔除. **最后**, 最终的断言即为所有上述推导公式的非与. 解码器存在当且仅当该公式仍然可满足.

为了展示其有效性, 我们在多个复杂编码器上运行了该算法, 包括PCI-E和以太网. 实验结果表明我们的算法始终能够推导出正确的断言.

关键词
对偶综合
推导断言
Cofactoring
Craig插值

1 引言

在通信和多媒体芯片设计中，最困难的工作是设计和验证复杂的对偶电路对(E, E^{-1}), 其中编码器E将输入数据流编码为适合传输和存储的格式, 而对应的对偶电路(或解码器)E^{-1}则恢复该数据.

为了提高该工作的效率, 对偶综合算法[1, 2]被提出以自动综合特定编码器的解码器, 通过检验特定编码器的输入能否被输出序列唯一决定. 编码器电路通常具有多种工作模式, 每种模式对应于一个非相交的状态集合:

1. 最为重要的模式是工作模式, 在该模式中编码器致力于编码输入数据. 因此编码器的输入能够被其输出唯一决定, 而这意味着编码器存在.
2. 而另一方面, 编码器仍然包哈很多其他状态, 比如测试和休眠模式, 在这些模式中编码器将致力于处理测试命令，或者不做任何事情。因此他们的输出将不能唯一决定输入，这将导致解码器不存在

因此对偶综合的用户需要手工给出配置信号上的断言以防止编码器进入非工作状态.

为了防止这一烦琐的工作，本文给出了推导断言的自动算法:

1. **首先**, 我们给出一个算法以对任意特定的配置信号值确定其解码器是否存在.

2. **其次**, 对任意导致解码器不存在的配置信号值, 我们使用cofactoring[3]和Craig插值[4]以推导一个新公式, 而该公式将覆盖一个较大的导致解码器不存在的非法配置信号值集合. 该步骤将被重复直至所有非法配置信号值均被剔除为止.

3. **最后**, 最终断言是上述所有推导公式的非与. 解码器存在当且仅当该公式仍然可满足.

我们在OCaml语言中实现了该算法, 并使用Minisat求解器[8]求解所有的SAT实例. 测试集包含了多个来自于工业界项目的复杂编码器(如PCI-E[9]以太网[10]). 实验结果表明我们的算法始终能够推导正确的断言.

本文的贡献在于: 我们提出了首个能够自动推导断言的对偶综合算法.

本文剩余部分如下组织. 节2介绍了背景资料. 节3给出了本文算法的整体结构及其正确性证明, 节4和5则分别介绍了如何从一个非法配置值中推导更大的配置信号值集合, 和如何剔除冗余的输出字符. 节6和7则分别给出了实验结果和相关工作. 最后, 节8总结全文.

2 背景

2.1 命题逻辑可满足

布尔集合记为$B = \{0, 1\}$. 对于变量集合V上的布尔公式F, 其布尔命题逻辑可满足问题(缩写为SAT)是寻找一个满足赋值$A : V \to B$, 使得F为1. 如歌A存在则F可满足; 否则不可满足. 一个求解SAT问题的计算机程序称为SAT求解其, 如Zchaff[5], Grasp[6], Berkmin[7], 和MiniSAT[8].

一般的, SAT求解器要求公式表示为CNF格式, 其中公式是一系列短句的与, 而短句是一系列文字的或, 而文字是变量或者其非. 而用CNF格式表示的公式也称为SAT实例,

2.2 Cofactoring

对于一个布尔函数$f : B^n \to B$, 我们用$supp(f)$表示其支撑集$\{v_1 \dots v_n\}$. 由[3]可知, $f(v_1 \dots v \dots v_n)$面向变量v的正和负的cofactors是$f_v = f(v_1 \dots 1 \dots v_n)$和$f_{\overline{v}} = f(v_1 \dots 0 \dots v_n)$. **Cofactoring**是将0或1赋给$v$以得到$f_v$ or $f_{\overline{v}}$的动作.

2.3 Craig插值

Craig[4]证明了下列定理:

Theorem 1 (Craig插值定理)**.** *对于两个一阶逻辑公式ϕ_A和ϕ_B, 如果$\phi_A \wedge \phi_B$不可满足, 则存在公式ϕ_I仅包含ϕ_A和ϕ_B的共同变量使得$\phi_A \to \phi_I$成立, 而$\phi_I \wedge \phi_B$不可满足. 公式ϕ_I称为ϕ_A面向ϕ_B的插值.*

在本文的剩余部分, 我们将仅讨论命题逻辑, 并使用McMillan[11]提出的算法以从不可满足证明中生成插值, 而不可满足证明由MiniSAT[8]给出.

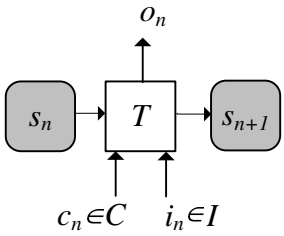

图 1 带配置的Mealy有限状态机

2.4 递归半径

一个电路可以用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为有限输入字符集合, O为有限输出字符集合, 迁移函数$T : S \times I \to S \times O$从当前状态和输入字符计算次态和输出字符.

我们记第n个周期的状态，输入字符和输出字符为s_n, i_n和o_n. 我们进一步记从第n个到第m个周期的状态，输入字符和输出字符序列为s_n^m, i_n^m和o_n^m. 一个路径是状态序列s_n^m其中$\exists i_j o_j (s_{j+1}, o_j) \equiv T(s_j, i_j)$对所有$n \leqslant j < m$成立. 一个环是一个路径$s_n^m$其中$s_n \equiv s_m$.

Kroening et al. [13]定义M的状态变量递归半径$rrd(M)$为从初始状态出发的最长无环路径.

本文中, 我们定义了一个相似的概念:非初始化的状态变量递归半径$uirrd(M)$为最长无环路径：

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

他们之间的唯一区别在于我们的$uirrd$不考虑初始状态. 这些定义仅用于证明下面的定理. 我们的算法不需要计算这些半径.

2.5 确认解码器存在性的原始算法

最早的对偶综合算法[1]包含两步: 确认解码器存在性和特征化解码器的布尔函数. 这里我们将仅仅介绍第一步.

为了建模包含配置信号的电路, 我们扩展了原始的Mealy有限状态机[12]:

Definition 2. **带配置的*Mealy*有限状态机**是一个6元组$M = (S, s_0, I, C, O, T)$, 其中S是状态集合, $s_0 \in S$为初始状态, I为有限输入字符集合, C为有限配置字符集和, O为有限输出字符集合, 迁移函数$T : S \times I \times C \to S \times O$从当前状态，输入字符和配置字符计算出次态和输出字符.

如图1所示, 在本文剩余部分中, 状态使用灰色圆角框表示, 而迁移函数T则使用白色方框表示.

我们记第n周期的配置信号字符为c_n. 并记从第n到第m个周期的配置字符序列为c_n^m.

Definition 3. 在配置信号上的断言(或公式)是一个配置字符集合R. 对于一个配置字符c, $R(c)$意味着$c \in R$. 如果$R(c)$成立, 我们也说R覆盖c.

如图2, E^{-1}存在的充分条件是, 存在参数p, d和l, 使得i_n可以被编码器的输出字符序列o_{n+d-l}^{n+d-1}唯一决定. d是o_{n+d-l}^{n+d-1}和i_n之间的相对延时, 而l是o_{n+d-l}^{n+d-1}的长度, 而p是用于剔除一部分不可达状态的前缀序列. 该条件定义如下:

Definition 4. 参数对偶条件*(PC)*: 对于编码器E, 断言R, 和三个参数p,d and l, $E \vDash PC(p,d,l,R)$成立当

1. i_n可以被o_{n+d-l}^{n+d-1}在s_{n-p}^{n+d-1}上唯一决定.
2. 断言R覆盖c_x,其中$n-p \leqslant x \leqslant n+d-1$.

这等价于公式*(2)*中的$F_{PC}(p,d,l,R)$的不可满足. 我们进一步定义$E \vDash PC(R)$为$\exists p,d,l : E \vDash PC(p,d,l,R)$.

$$
F_{PC}(p,d,l,R) \stackrel{def}{=}
\left\{
\begin{array}{ll}
 & \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1},o_m) \equiv T(s_m,i_m,c_m)\} \\
\wedge & \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1},o'_m) \equiv T(s'_m,i'_m,c'_m)\} \\
\wedge & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\
\wedge & i_n \neq i'_n \\
\wedge & \bigwedge_{x=n-p}^{n+d-1} c_x \equiv c \\
\wedge & \bigwedge_{x=n-p}^{n+d-1} c'_x \equiv c \\
\wedge & R(c)
\end{array}
\right\}
\tag{2}
$$

公式(2)的第2到5行对应于定义4的条件1. 公式(2)的第2和第3行分别对应于E的两条路径. 他们之间的唯一差别在于第3行的所有变量带有一个prime. 第4行强制这两个路径的输出序列相同, 而第5行强制他们的输入字符不同.

同时, 公式(2)的最后三行对应于定义4的条件2. 而第6he 7行强制每个周期的c均相等, 而最后一行则强制c被R覆盖.

基于检验$E \vDash PC(R)[1,2]$的算法简单的遍历p,d的l所有组合, 从小到大, 直到$F_{PC}(p,d,l,R)$不可满足, 而这意味着E^{-1}存在.

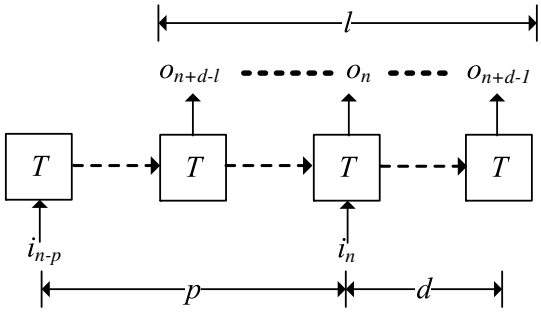

图 2 参数对偶条件

3　推导断言的停机算法

定义4中的PC只定义了如何确认E^{-1}的存在. 因此当其不存在时, 该算法$E \vDash PC(R)$[1, 2]将不停机.

为了找到一个停机算法, 我们必须定义如何确认E^{-1}不存在. 这将在小节3.1中讨论, 而其正确性证明将在小节3.2中给出. 基于这些结果, 本文算法的整体框架将在小节3.3中给出,

3.1　确认解码器不存在

从定义4和图2可知, E^{-1}存在当存在$< p, d, l >$使得$E \vDash PC(p, d, l, R)$成立.

因此直观的, E^{-1}不存在当对任意$< p, d, l >$, 我们总能找到另一个$< p', d', l' >$其中$p' > p$,$l' > l$且$d' > d$, 使得$E \vDash PC(p', d', l', R)$不成立.

这种情形可以被图3中的SAT实例检验, 改图类似于2, 除了三个新的约束被用来检验s^{n+d-l}_{n-p},$s^{n}_{n+d-l+1}$和s^{n+d}_{n+1}上的环.

如果该SAT实例可满足, 对于任意$< p, d, l >$, 我们可以展开这些环直至我们找到大于$< p, d, l >$的$< p', d', l' >$. 在下一小节中我们将证明该展开的SAT实例是可满足的, 这意味着$E \vDash PC(p', d', l', R)$不成立. 因此解码器不存在.

由公式(2)的行2和行3, 实际上存在两条路径, 因此我们需要在这两条路径上检测环, 即在如下定义的M^2上检测环:

Definition 5. **乘积机*:*** 对于*Mealy*有限状态机$M = (S, s_0, I, C, O, T)$, 其乘积机为$M^2 = (S^2, s_0^2, I^2, C^2, O^2, T^2)$, 其中$T^2$定义为$(< s_{m+1}, s'_{m+1} >, < o_m, o'_m >) = T^2(< s_m, s'_m >, < i_m, i'_m >, < c_m, c'_m >)$其中$(s_{m+1}, o_m) = T(s_m, i_m, c_m)$且$(s'_{m+1}, o'_m) = T(s'_m, i'_m, c'_m)$.

由此我们定义如下的环形非对偶条件以检验E^{-1}的不存在:

Definition 6. **环形非对偶条件*(LN):*** 对于编码器E及其*Mealy*有限状态机$M = (S, s_0, I, C, O, T)$, 假设其乘积机为$M^2 = (S^2, s_0^2, I^2, C^2, O^2, T^2)$, 则$E \vDash LN(p, d, l, R)$成立当$i_n$不能够在路径$s^{n+d-1}_{n-p}$上被$o^{n+d-1}_{n+d-l}$唯一决定, 且在$(s^2)^{n+d-l}_{n-p}$, $(s^2)^{n}_{n+d-l+1}$和$(s^2)^{n+d}_{n+1}$上存在三个环. 这等价于公式*(3)*中的$F_{LN}(p, d, l, R)$的可满足. 我们进一步定义$E \vDash LN(R)$为$\exists p, d, l : E \vDash LN(p, d, l, R)$.

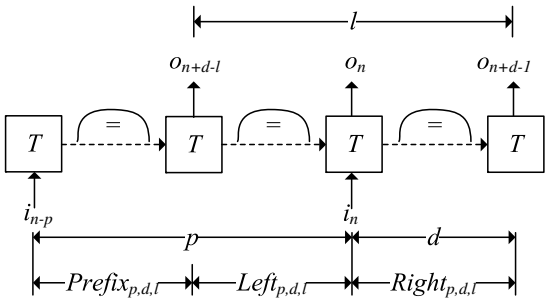

图 3 环形非对偶条件

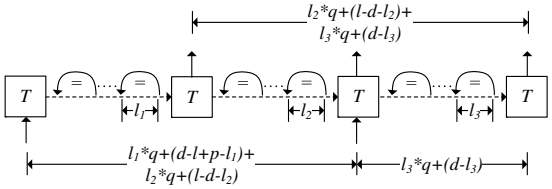

图 4 展开q次的环形非对偶条件

$$
F_{LN}(p,d,l,R) \stackrel{def}{=}
\left\{
\begin{array}{ll}
 & \bigwedge_{m=n-p}^{n+d-1}\{(s_{m+1},o_m)\equiv T(s_m,i_m,c_m)\} \\
\wedge & \bigwedge_{m=n-p}^{n+d-1}\{(s'_{m+1},o'_m)\equiv T(s'_m,i'_m,c'_m)\} \\
\wedge & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\
\wedge & i_n \neq i'_n \\
\wedge & \bigwedge_{x=n-p}^{n+d-1} c_x \equiv c \\
\wedge & \bigwedge_{x=n-p}^{n+d-1} c'_x \equiv c \\
\wedge & R(c) \\
\wedge & \bigvee_{x=n-p}^{n+d-l-1}\bigvee_{y=x+1}^{n+d-l}\{s_x\equiv s_y \wedge s'_x \equiv s'_y\} \\
\wedge & \bigvee_{x=n+d-l+1}^{n-1}\bigvee_{y=x+1}^{n}\{s_x\equiv s_y \wedge s'_x \equiv s'_y\} \\
\wedge & \bigvee_{x=n+1}^{n+d-1}\bigvee_{y=x+1}^{n+d}\{s_x\equiv s_y \wedge s'_x \equiv s'_y\}
\end{array}
\right\} \tag{3}
$$

通过比较公式(2)和(3), 很明显他们之间的差异在于公式(3)的最后三行, 他们将被用于在图3中检测一下三个路径上的环:

$$
\begin{aligned}
Prefix_{p,d,l} &= (s^2)_{n-p}^{n+d-l} \\
Left_{p,d,l} &= (s^2)_{n+d-l+1}^{n} \\
Right_{p,d,l} &= (s^2)_{n+1}^{n+d}
\end{aligned} \tag{4}
$$

在下一小节中, 我们将证明$E \vDash LN(R) \leftrightarrow \neg\{E \vDash PC(R)\}$, 即$LN$和$PC$是互斥的.

在进入下一小节之前, 我们需要定义如何展开上述的三个环. 假设$Prefix_{p,d,l}$, $Left_{p,d,l}$和$Right_{p,d,l}$中的环的长度分别为l_1, l_2和l_3. 我们进一步假设他们被展开q次. 则从该展开中的到的SAT实例如图4所示, 很明显, 该展开的SAT实例对应于$F_{LN}(p", d", l", R)$, 其中:

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

因此对于任意$< p,d,l >$ and $< p',d',l' >$, 我们总能找到q, 使得$Prefix_{p",d",l"}$, $Left_{p",d",l"}$ 和$Right_{p",d",l"}$不短于$Prefix_{p',d',l'}$,$Left_{p',d',l'}$和$Right_{p',d',l'}$.

3.2　正确性证明

在证明$E \vDash LN(R) \leftrightarrow \neg\{E \vDash PC(R)\}$之前, 我们需要以下引理.

Lemma 1. 对于图4中的$F_{LN}(p", d", l", R)$, 我们有$F_{LN}(p, d, l, R) \rightarrow F_{LN}(p", d", l", R)$

Proof. $F_{LN}(p", d", l", R)$是:

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

假设$F_{LN}(p, d, l, R)$可满足, 且其满足赋值为A. 我们需要证明$F_{LN}(p", d", l", R)$也可以被A满足.

图5中编号从1到12的有向边给出了$F_{LN}(p, d, l, R)$和$F_{LN}(p", d", l", R)$之间的对应关系.

边2和3意味着我们可以将$Right_{p,d,l}$中环的赋值赋予$Right_{p",d",l"}$中展开的环. 而边1和4意味着我们可以将$Right_{p,d,l}$非环部分的赋值赋予$Right_{p",d",l"}$. 由这四条边, 我们可以让$Right_{p",d",l"}$可满足.

类似的, 我们也可以让$Prefix_{p",d",l"}$和$Left_{p",d",l"}$可满足.

则公式(6)的第2行可以被A满足.

类似的, 公式(6)的第3到8行可以被A满足.

同时, $Prefix_{p",d",l"}$, $Left_{p",d",l"}$和$Right_{p",d",l"}$中存在q个环, 这将使得公式(6)的最后三行满足.

$F_{LN}(p, d, l, R)$的满足赋值A可以使$F_{LN}(p", d", l", R)$满足.

得证. □

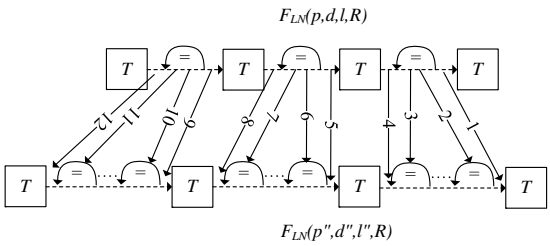

图 5　$F_{LN}(p, d, l, R)$和$F_{LN}(p", d", l", R)$之间的对应关系

Lemma 2. 对于两个三元组$< p,d,l >$和$< p',d',l' >$, 如果$Prefix_{p',d',l'}$,$Left_{p',d',l'}$ 和$Right_{p',d',l'}$分别不短于$Prefix_{p,d,l}$,$Left_{p,d,l}$ 和$Right_{p,d,l}$, 则$E \vDash PC(p,d,l,R) \rightarrow E \vDash PC(p',d',l',R)$.

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

下面的两个定理将证明$E \vDash LN(R) \leftrightarrow \neg\{E \vDash PC(R)\}$.

Theorem 2. $E \vDash LN(R) \rightarrow \neg\{E \vDash PC(R)\}$

Proof. 我们使用反证法. 假设$E \vDash LN(R)$和$E \vDash PC(R)$同时成立. 这意味着存在$< p,d,l >$和$< p',d',l' >$, 使得$E \vDash PC(p,d,l,R)$ 和$E \vDash LN(p',d',l',R)$.

一方面, $E \vDash LN(p',d',l',R)$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, 我们能够得到另一个三元组$< p",d",l" >$, 使得:

1. $Prefix_{p",d",l"}$,$Left_{p",d",l"}$ 和$Right_{p",d",l"}$分别长于$Prefix_{p,d,l}$,$Left_{p,d,l}$和$Right_{p,d,l}$。
2. 由引理1可知, $F_{LN}(p",d",l",R)$可满足.

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

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

该矛盾导致定理得证. □

Theorem 3. $E \vDash LN(R) \leftarrow \neg\{E \vDash PC(R)\}$

Proof. 仍然使用反证法. 假设$E \vDash LN(R)$和$E \vDash PC(R)$均不成立. 则对于任意$< p,d,l >$和$< p',d',l' >$, $F_{PC}(p,d,l,R)$可满足, 而$F_{LN}(p',d',l',R)$不可满足.

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

$$\begin{aligned} p &= uirrd(M^2) * 2 + 2 \\ d &= uirrd(M^2) + 1 \\ l &= uirrd(M^2) * 2 + 2 \end{aligned} \tag{7}$$

基于以上定义, 很明显$Prefix_{p,d,l}$,$Left_{p,d,l}$和$Right_{p,d,l}$全部长于$uirrd(M^2)$. 这意味着在这三个路径上均存在环, 这将使得$F_{LN}(p,d,l,R)$可满足. 这与$F_{LN}(p',d',l',R)$对任意$< p',d',l' >$均不可满足的实施冲突.

该矛盾导致定理得证. □

3.3 算法实现

定理2和3指出, 我们可以从小到大遍历$< p,d,l >$的所有组合, 并在每个循环中检验$E \vDash PC(p,d,l,R)$和$E \vDash LN(p,d,l,R)$. 该过程最终总会停机，并在$E \vDash PC(R)$和$E \vDash LN(R)$之间给出唯一答案. 该算法的实现在算法1中给出.

在算法1的行1中, NA将被用于记录所有导致解码器不存在的配置字符. 他们都是由行13中的$InferCoveringFormula$函数给出的, 该函数的作用在于找到一个公式不仅覆盖c, 而且覆盖大量其他导致解码器不存在的配置字符. 该函数的更多细节将在节4中给出.

Algorithm 1 InferAssertion

1: $NA = \{\}$
2: **for** $x = 0 \rightarrow \infty$ **do**
3: $< p, d, l >=< 2x, x, 2x >$
4: **if** $F_{PC}(p, d, l, \bigwedge_{na \in NA} \neg na)$不可满足 **then**
5: **if** $\bigwedge_{na \in NA} \neg na$可满足 **then**
6: 解码器存在且最终断言为$\bigwedge_{na \in NA} \neg na$
7: **else**
8: 解码器不存在
9: **end if**
10: 停机
11: **else if** $F_{LN}(p, d, l, \bigwedge_{na \in NA} \neg na)$可满足 **then**
12: 令c为导致解码器不存在的配置字符
13: $na \leftarrow InferCoveringFormula(c)$
14: $NA \leftarrow NA \cup \{na\}$
15: **end if**
16: **end for**

行3确保$Prefix_{p,d,l}$,$Left_{p,d,l}$和$Right_{p,d,l}$的长度全部被设置为x, 而其值将在行2中被遍历. 如此, 大量p,d和l的冗余组合将不再被遍历. 因此算法1的性能将得到巨大的提升.

行4意味着在断言$\bigwedge_{na \in NA} \neg na$下，输入字符能够被输出序列唯一决定. 行5意味着至少存在一个配置字符导致解码器存在, 而最终的断言可以通过将行6中的NA中的所有断言取反并与在一起得到.

行7意味着$\bigwedge_{na \in NA} \neg na$已经剔除了所有的配置字符, 即不存在能够导致解码器存在的配置字符. 这意味着在编码器中必然存在某些Bug.

行11意味着针对行12的c解码器不存在. 我们需要剔除c以便算法1可以继续搜索其他导致解码器不存在的配置字符. 行13中的函数$InferCoveringFormula$将被用于推导一个公式na使其不仅能够覆盖c, 而且能够覆盖大量其他的非法配置字符. 而所有这些配置字符将在行14中被剔除.

我们可以证明算法1是可停机的.

Theorem 4. *算法1是可停机的.*

Proof. 由定理2和3, 算法1将最终到达行4或11.

对于前一种情况, 该算法将在行10停机.

而对于后一种情况, 一个新公式na将被推导, 其将覆盖c. 由于此类c的数目是有限的, 他们最终将完全被$\bigwedge_{na \in NA} \neg na$剔除. 则算法1最终还是将到达行4, 并在行10中停机.

□

4　推导能够覆盖c的公式

本节将介绍算法1行13中的函数$InferCoveringFormula$. 该函数将被用来推导公式na 而该公式不仅覆盖c, 而且覆盖大量其他导致解码器不存在的配置字符. 该工作将由以下三步完成:

1. 转换F_{LN}为带有目标变量的等价形式, 以便可以用其定义函数f.

2. 使用cofactoring[3]削减f的支撑集, 直到只剩下c.

3. 使用Craig插值特征化f. 该f同时也是算法1行13中的na.

这三步将在以下的三个小节中分别描述.

4.1 F_{LN}的等价形式

如上面所述, 我们需要将F_{LN}转换为带有目标变量的等价形式.
首先, 我们需要将公式(3)的第4行和最后三行用于构造下面的新公式:

$$G(p,d,l) \stackrel{def}{=} \left\{ \begin{array}{ll} & \bigwedge_{m=n+d-l}^{n+d-1} o_m \equiv o'_m \\ \wedge & \bigvee_{x=n-p}^{n+d-l-1} \bigvee_{y=x+1}^{n+d-l} \{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+d-l+1}^{n-1} \bigvee_{y=x+1}^{n} \{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \\ \wedge & \bigvee_{x=n+1}^{n+d-1} \bigvee_{y=x+1}^{n+d} \{s_x \equiv s_y \wedge s'_x \equiv s'_y\} \end{array} \right\} \tag{8}$$

因此, F_{LN}可以被转变为:

$$F'_{LN}(p,d,l,R) \stackrel{def}{=} \left\{ \begin{array}{ll} & \bigwedge_{m=n-p}^{n+d-1} \{(s_{m+1}, o_m) \equiv T(s_m, i_m, c_m)\} \\ \wedge & \bigwedge_{m=n-p}^{n+d-1} \{(s'_{m+1}, o'_m) \equiv T(s'_m, i'_m, c'_m)\} \\ \wedge & i_n \neq i'_n \\ \wedge & \bigwedge_{x=n-p}^{n+d-1} c_x \equiv c \\ \wedge & \bigwedge_{x=n-p}^{n+d-1} c'_x \equiv c \\ \wedge & R(c) \\ \wedge & t \equiv G(p,d,l) \end{array} \right\} \tag{9}$$

很明显F_{LN}和$F'_{LN} \wedge t \equiv 1$是等价的.

同时, 由图3可知, F'_{LN}实际上定义了一个函数$f' : S^2 \times I^{(d+p)*2} \times C \to B$, 该函数的支撑集为$supp(f')$ is $\{s_{n-p}, s'_{n-p}, i_{n-p}^{n+d-1}, (i')_{n-p}^{n+d-1}, c\}$, 而其输出为公式(9)的最后一行的目标变量$t$.

4.2 Cofactoring

由算法1的行11, F_{LN}是可满足的. 我们进一步假设A是F_{LN}的满足赋值. 我们可以直接将$i_{n-p}^{n+d-1}, (i')_{n-p}^{n+d-1}$, s_{n-p}和$(s')_{n-p}$的满足赋值赋予F'_{LN}, 从而得到:

$$F"_{LN}(c,t) \stackrel{def}{=} \left\{ \begin{array}{rc} & F'_{LN} \\ \wedge & i_{n-p}^{n+d-1} \equiv A(i_{n-p}^{n+d-1}) \\ \wedge & (i')_{n-p}^{n+d-1} \equiv A((i')_{n-p}^{n+d-1}) \\ \wedge & s_{n-p} \equiv A(s_{n-p}) \\ \wedge & (s')_{n-p} \equiv A((s')_{n-p}) \end{array} \right\} \tag{10}$$

现在, $F"_{LN}$定义了另一个函数$f"$, 其支撑集为c. 很明显$F"_{LN}(c,t) \wedge t \equiv 1$ 已经是一个覆盖多个非法配置字符的公式. 然而他仍然是一个复杂的CNF短句集合. 为了减小其尺寸, 我们需要下一小节的特征化算法.

4.3 使用Craig插值特征化 $f"$

我们将$F"_{LN}(c,t)$编码为CNF格式, 并记为$CNF(F"_{LN}(c,t))$. 假设$CNF'(F"_{LN}(c,t'))$是$CNF(F"_{LN}(c,t))$的一个拷贝. 他们之间只共享c, 而所有其他变量都独立编码. 如此, 我们可以构造连个如下公式ϕ_A和ϕ_B:

$$\phi_A \stackrel{def}{=} CNF(F"_{LN}(c,t)) \wedge t \equiv 1 \tag{11}$$

$$\phi_B \stackrel{def}{=} CNF'(F"_{LN}(c,t')) \wedge t' \equiv 0 \tag{12}$$

很明显$\phi_A \wedge \phi_B$不可满足. 通过使用McMillan[11]提出的插值算法, 我们可以产生一个插值电路ITP : $C \rightarrow B$.

有定理1可知, ITP与公式(12)中的ϕ_B不兼容. 因此其特征化了一个集合$C' \subseteq C$能够是公式(11)中的ϕ_A可满足. 由公式(9)和(10)可知, 很明显:

1. 算法1的行12的c被包含于C'中. 由定理4可知, 这将确保算法1停机.

2. 所有$c' \in C'$也能够导致解码器不存在. 这将极大的加速算法1.

5 移除冗余

算法1得到的$< p, d, l >$包含一些冗余, 浙江导致芯片面积和特征化运行时间上的不必要开销. 因此, 算法2被用于在将$< p, d, l >$传给特征化算法前将其削减.

该算法逐步削减p, d和l, 并测试他们是否仍然能够使$E \vDash PC(R)$成立. 我们在这里将不再进一步深入细节.

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

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

6 试验结果

我们在OCaml语言中实现了该算法, 并使用Minisat求解器[8]求解生成的SAT实例. 所有的实验在一台PC上进行，该PC具有一个2.4GHz的Intel Core 2 Q6600处理器, 8GB内存和Ubuntu 10.04 linux操作系统是.

6.1 Benchmarks

表1给出了以下Benchmark的相关信息.

表 1 Benchmarks的相关信息

	XGXS	XFI	scrambler	PCIE	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

表 2 试验结果

		XG-XS	XFI	scra-mbler	PCI-E	T2 e-ther
[14]	检查 *PC* 时间(sec)	0.07	17.84	2.70	0.47	30.59
	d, p, l	1,2,1	0,3,2	0,2,2	2,2,1	4,2,1
本文算法	配置信号数量	3	120	1	16	26
	推到断言时间	3.69	372.15	3.14	30.53	188.56
	d, p, l	1,3,1	0,4,2	0,2,2	2,2,1	4,4,1

1. 一个遵从IEEE-802.3ae 2002标准[10]短句48的XGXS编码器.
2. 一个遵从同一个IEEE标准的短句49的XFI编码器.
3. 一个66位加扰器用于确保输出数据中包含足够多的0-1迁移, 以便其能够可靠的穿越高噪声通讯信道.
4. 一个PCI-E物理编码子层模块[9].
5. Sun公司的OpenSparc T2处理器的以太网模块.

6.2 Results

表2的第2行给出了Shen et.al[14]的算法检验解码器存在性的运行时间. 而行3给出了该算法发掘的d, p和l的值.

同时, 行4给出了配置信号的数量, 而行5给出了本文算法推到断言的运行时间。而最后一行给出了本文算法发掘的d, p和l的值.

通过比较第2和5行, 很明显本文算法比[14]慢很多, 这是由$InferCoveringFormula$造成的.

通过比较第3和6行, 狠命向他们在发掘的参数值上有一些差别. 这是有检验不同参数组合的循序不同造成的.

6.3 推导得到的断言

下面将给出推导得到的断言.

对于XGXS:

!((bad_disp & !rst & bad_code)) & !((rst & bad_code)) & !((!rst & !bad_disp & bad_code))

对于**XFI**:

!((!RESET & !TEST_MODE & !DATA_VALID & DATA _PAT_SEL)) & !((!RESET & !TEST_MODE & !DATA_VA LID & !DATA_PAT_SEL))

对于**scrambler**:

True

对于**PCI-E**:

!((CNTL_RESETN_P0 & !TXELECIDLE & CNTL_TXE nable_P0 & CNTL_Loopback_P0) — (CNTL_RESETN_P0 & !TXELECIDLE & !CNTL_TXEnable_P0)) & !((CNTL_RE SETN_P0 & TXELECIDLE) — (CNTL_RESETN_P0 & !TXELECIDLE & !CNTL_TXEnable_P0) — (!CNTL_RESET N_P0))

对于**T2 以太网**:

!((reset_tx & !txd_sel[0] & !txd_sel[1] & link_up_loc & jitter_study_pci[0] & !jitter_study_pci[1])) & !((reset_tx & !txd_sel[0] & !txd_sel[1] & link_up_loc & jitter_study_pci[0] & jitter_study_pci[1])) & !((reset_tx & !txd_sel[0] & !txd_sel[1] & link_up_loc & !jitter_study_pci[0] & jitter_study_pci[1])) & !((jitter_study_pci[1] & jitter_study_pci[0] & !txd_sel[0] & !txd_sel[1] & !reset_tx & link_up_loc)) & !((jitter_study_pci[1] & !jitter_study_pci[0] & !reset_tx & !txd_sel[0] & !txd_sel[1] & link_up_loc)) & !((jitter_study_pci[0] & !jitter_study_pci[1] & !txd_sel[0] & !txd_sel[1] & !reset_tx & link_up_loc)) & !((reset_tx & !jitter_study_pci[0] & !txd_sel[0] & !txd_sel[1] & link_up_loc & !jitter_study_pci[1]))

7 相关工作

7.1 对偶综合和程序取反

对偶综合的概念首先由Shen et.al[1, 2]提出. 其主要问题在于该算法不停机. Shen et.al[14]通过构造一系列类似于洋葱圈的上估计解决了该问题.

基于Gulwani[15], 程序取反是针对特定程序P, 生成具有相反功能的程序P^{-1}. 因此程序取反的定义和对偶综合非常类似.

早期的程序取反算法使用基于证明的算法[16], 然而该算法仅能处理简单的语法结构和很小的程序.

Glück et.al [17]使用基于LR的分析算法去除程序中的不确定性, 从而取反一阶函数程序. 然而使用函数式语言的要求使得该算法不能适用于我们的应用环境.

Srivastava et.al [18]假设一个反程序和原始程序是紧密相关的, 因此潜在的正确反程序可以通过挖掘原始程序的表达式, 谓词和控制流而得到. 该算法逐步的剔除不能满足反程序要求的运行路径, 以缩减可能的解空间直至剩下唯一的一个解. 因此该算法只能保证解的存在性, 而不能保证其正确性.

7.2 协议转换器综合

协议转换器综合是在两个不兼容的通讯协议之间综合一个接口电路.

Avnit et.al [19]首先定义了一份描述不同通讯协议的通用模型. 而后提出了一个能够确认两个协议是否兼容的算法, 即是否存在一个协议中的功能不能在另一个协议中实现. 最后, 该算法通过为缓冲

区的管理函数计算一个不动点而得到最终的接口电路. Avnit et.al[20]通过提出一个更高效的解空间遍历算法从而提高算法求解效率.

8 结论

本文提出了一个自动推导断言的对偶综合算法. 试验结果表明本文算法能够为许多复杂的编码器推导出正确的断言, 如PCI-E[9]和以太网[10],.

参考文献

1 S. Shen, J. Zhang, Y. Qin, and S. Li. Synthesizing complementary circuits automatically. In *ICCAD09 Conference Proceedings*, pages 381–388. IEEE, November 2009.

2 S. Shen, Y. Qin, K. Wang, L. Xiao, J. Zhang, and S. Li. Synthesizing complementary circuits automatically. *IEEE transaction on CAD of Integrated Circuits and Systems*, 29(8):1191–1202, August 2010.

3 M. K. Ganai, A. Gupta, and P. Ashar. Efficient sat-based unbounded symbolic model checking using circuit cofactoring. In *ICCAD04 Conference Proceedings*, pages 510–517. IEEE, November 2004.

4 W. Craig. Linear reasoning: A new form of the herbrand-gentzen theorem. *J. Symbolic Logic*, 22(3):250–268, 1957.

5 M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC01 Conference Proceedings*, pages 530–535. IEEE, June 2001.

6 J. P. M. Silva and K. A. Sakallah. GRASP - a new search algorithm for satisfiability. In *ICCAD96 Conference Proceedings*, pages 220–227. IEEE, November 1996.

7 E. Goldberg and Y. Novikov. BerkMin: a fast and robust SAT-solver. In *DATE02 Conference Proceedings*, pages 142–149. IEEE, March 2002.

8 N. Eén and N. Sörensson. An extensible sat-solver. In *SAT03 Conference Proceedings*, pages 502–518. Springer, May 2003.

9 *PCI Express Base Specification Revision 1.0. Download from http://www.pcisig.com*.

10 *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.*

11 K. L. McMillan. Interpolation and sat-based model checking. In *CAV03 Conference Proceedings*, pages 1–13. Springer, July 2003.

12 G. H. Mealy. A method for synthesizing sequential circuits. *Bell Systems Technical Journal*, 34(5):1045–1079, 1955.

13 D. Kroening and O. Strichman. Efficient computation of recurrence diameters. In *VMCAI03 Conference Proceedings*, pages 298–309. Springer, January 2003.

14 S. Shen, Y. Qin, J. Zhang, and S. Li. A halting algorithm to determine the existence of decoder. In *FMCAD10 Conference Proceedings*, pages 91–100. IEEE, October 2010.

15 S. Gulwani. Dimensions in program synthesis. In *PPDP10 Conference Proceedings*, pages 13–24. ACM, July 2010.

16 E. W. Dijkstra. Program inversion. In *Program Construction 1978 Conference Proceedings*, pages 54–57, 1978.

17 R. Glück and M. Kawabe. A method for automatic program inversion based on lr(0) parsing. *Fundam. Inf.*, 66(4):367–395, November 2005.

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

19 K. Avnit, V. D'Silva, A. Sowmya, S. Ramesh, and S. Parameswaran. A formal approach to the protocol converter problem. In *DATE08 Conference Proceedings*, pages 294–299. IEEE, March 2008.

20 K. Avnit and A. Sowmya. A formal approach to design space exploration of protocol converters. In *DATE09 Conference Proceedings*, pages 129–134. IEEE, March 2009.