文章编号：1000-5013(2015)01-0029-06 **doi**：10.11830/ISSN.1000-5013.2015.01.0029

FPGA 组合逻辑程序的 Petri 网建模方法

陈珑，黄颖坤，罗继亮

（华侨大学 信息科学与工程学院，福建 厦门 361021）

摘要： 针对现场可编程门阵列(FPGA)组合逻辑程序，提出其普通 Petri 网建模方法. 首先，将状态变量描述为库所对，程序中的逻辑运算描述为变迁，从而将系统程序转换为一个普通 Petri 网结构；然后，根据 Petri 网的动态分析性能，给出系统状态可达图的计算方法，实现了状态可达图等价描述 FPGA 组合逻辑系统运行过程. 研究结果表明：该 Petri 网能够准确地描述变量间的逻辑关系，提出的方法可以为 FPGA 组合逻辑程序的形式化设计和验证提供建模依据.

关键词： 现场可编程门阵列；组合逻辑；Petri 网；建模方法；状态可达图

中图分类号： TP 273.5 **文献标志码**： A

现场可编程门阵列(field-programmable gate array，FPGA)是一种可编程使用的信号处理器件[1]. FPGA 具有灵活性强、时序控制能力强、开发周期短和产品上市速度快等优势[2]，广泛应用于通信、军事、医疗和工业控制等重要领域. 然而，FPGA 数字系统的分析和设计复杂性随系统规模指数级增长，传统的测试方法难以保证程序的正确性和可靠性，而形式化验证能够枚举验证每一个状态，因此获得了广泛关注[3-5]，形式化验证的前提是形式化建模方法. FPGA 系统可以抽象为离散事件系统，Petri 网是一种描述离散事件系统的数学模型，较自动机而言，它能够刻画系统的结构信息，具有更高的建模效率. 因此，FPGA 的 Petri 网建模方法具有重要研究价值. FPGA 的描述语言 VHDL(very-high-speed integrated circuit hardware description language)的形式化建模分析主要分为两个方面：一是利用扩展 Petri 网[6-7]和有色 Petri 网[8-10]对 FPGA 系统进行建模，这些扩展是针对某一特定的应用，适用范围比较窄，不具有一般性，并且基于一种 Petri 网模型的分析方法不能应用到另一种模型上去；二是用变迁描述 VHDL 中的执行语句，库所表示语句的执行状态，通过托肯的迁移揭示语句的执行过程[11-13]，这些模型是对程序的整体概况描述，分析能力比较差而且无法揭示变量间的逻辑关系. 为此，本文提出了一种将描述 FPGA 组合逻辑电路的 VHDL 程序转换为普通 Petri 网的算法.

1 基础知识

1.1 描述组合逻辑电路的 VHDL 程序

电路的 VHDL 描述由两大部分组成[14]：1) 以关键字 entity 引导，end entity e_name 结尾的语句部分，称为 VHDL 的实体，实体描述了电路器件的外部情况及各信号端口的基本性质，如信号流动方向、流动在其上的数据类型等；2) 以关键字 architecture 引导，end architecture a_name 结尾的语句部分，称为 VHDL 的结构体，结构体描述电路器件的内部逻辑功能和电路结构.

1.2 Petri 网[15-16]

普通 Petri 网是三元组，即 $N=(P,T,F)$，其中，P 为状态库所集合，T 为变迁集合，$F\subseteq(P\times T)\cup(T\times P)$表示库所与变迁之间有向弧的集合. Petri 网系统是$(N,\boldsymbol{m}_0)$，其中，$\boldsymbol{m}_0$ 是初始标识. 标识是一个

收稿日期： 2014-06-04

通信作者： 罗继亮(1977-)，男，副教授，博士，主要从事离散事件动态系统的研究. E-mail：jlluo@hqu.edu.cn.

基金项目： 福建省自然科学基金资助项目(2014J01241)；福建省高等学校新世纪优秀人才计划(11FJRC01)；福建省高校杰出青年科研人才培育计划项目(JA10004)

向量 $\boldsymbol{m}: P\to\{0,1,2,\cdots\}$,其中,第 i 维上的分量记作 $\boldsymbol{m}(P_i)$,表示状态库所 P_i 的标识.

2 组合逻辑程序的 Petri 网设计

针对组合逻辑电路的 VHDL 程序,程序实体中是一系列逻辑表达式,输入量和输出量抽象为不同的系统状态.控制变量值的变化抽象为一个事件,以变量间的逻辑关系为研究对象,考虑电路零延迟情况下,FPGA 组合逻辑程序的 Petri 网建模方法.

算法 1 从 FPGA 组合逻辑程序到普通 Petri 网的转换算法

输入:组合逻辑电路 VHDL 程序

输出:Petri 网$(N,\boldsymbol{m}_0)$

步骤 1 在程序实体中找出输入量 $X_1,X_2,\cdots,X_n(n\in N^+)$和输出量 $Y_1,Y_2,\cdots,Y_m(m\in N^+)$,从结构体的描述语句中确定变量间的逻辑函数表达式为

$$Y_i = f_i(X_1,X_2,\cdots,X_n),\qquad 1\leqslant i\leqslant m. \tag{1}$$

为了叙述简便,以下只以一个逻辑输出表达式进行说明,即

$$Y_1 = f_1(X_1,X_2,\cdots,X_n). \tag{2}$$

步骤 2 通过公式法或卡诺图法对式(2)进行化简得到

$$Y_1 = \bigvee_{g=1}^{G}\Phi_g(X_1,X_2,\cdots,X_n). \tag{3}$$

步骤 3 对式(3)进行逻辑运算得到

$$Y_1 = Y_1\Big[\bigvee_{g=1}^{G}\Phi_g(X_1,X_2,\cdots,X_n)\Big]+Y'_1\Big[\bigvee_{g=1}^{G}\Phi_g(X_1,X_2,\cdots,X_n)\Big]. \tag{4}$$

步骤 4 对式(3)两边同时取非得到

$$Y'_1 = \Big[\bigvee_{g=1}^{G}\Phi_g(X_1,X_2,\cdots,X_n)\Big]'. \tag{5}$$

步骤 5 对式(5)进行化简得到

$$Y'_1 = \bigvee_{l=1}^{L}\psi_l(X_1,X_2,\cdots,X_n). \tag{6}$$

步骤 6 对式(6)进行逻辑运算得到

$$Y'_1 = Y_1\Big[\bigvee_{l=1}^{L}\psi_l(X_1,X_2,\cdots,X_n)\Big]+Y'_1\Big[\bigvee_{l=1}^{L}\psi_l(X_1,X_2,\cdots,X_n)\Big]. \tag{7}$$

步骤 7 分别用一对库所$(P_{X_j^0},P_{X_j^1})(1\leqslant j\leqslant n)$表示每个输入量 $X_j(1\leqslant j\leqslant n)$的“0”和“1”两种状态;并在每对库所$(P_{X_j^0},P_{X_j^1})(1\leqslant j\leqslant n)$之间分别加上两个变迁 t_s^+ 和 $t_s^-(1\leqslant s\leqslant n)$,有向弧集合 $F=\{(P_{X_j^0},t_{\text{in},s}^+),(t_{\text{in},s}^+,P_{X_j^1}),(P_{X_j^1},t_{\text{in},s}^-),(t_{\text{in},s}^-,P_{X_j^0})\}$.

步骤 8 用一对库所$(P_{Y_1^0},P_{Y_1^1})$表示输出量 Y_1 的“0”和“1”两种状态,并在库所$(P_{Y_1^0},P_{Y_1^1})$之间加入两个变迁 $t_{\text{out},1}^+$和 $t_{\text{out},1}^-$,有向弧集合 $F=\{(P_{Y_1^0},t_{\text{out},1}^+),(t_{\text{out},1}^+,P_{Y_1^1}),(P_{Y_1^1},t_{\text{out},1}^-),(t_{\text{out},1}^-,P_{Y_1^0})\}$.

步骤 9 根据式(4)得出:Y_1 从当前状态值“0”变为下一个状态值“1”(即当前托肯在状态库所 $P_{Y_1^0}$ 中转移到库所 $P_{Y_1^1}$ 中时)需要项 $Y'_1\Big[\bigvee_{g=1}^{G}\Phi_g(X_1,X_2,\cdots,X_n)\Big]=1$,又存在 G 个变迁 $t_{\text{out},1,g}^+(1\leqslant g\leqslant G)$.用双向弧把每项 $\Phi_g(X_1,X_2,\cdots,X_n)(1\leqslant g\leqslant G)$中所涉及的输入量的状态库所与对应的变迁 $t_{\text{out},1,g}^+$相连.

步骤 10 根据式(7)得出:Y'_1 从当前状态值“0”变为下一个状态值“1”(即当前托肯在状态库所 $P_{Y_1^1}$ 中转移到库所 $P_{Y_1^0}$ 中时)需要项 $Y'_1\Big[\bigvee_{l=1}^{L}\psi_l(X_1,X_2,\cdots,X_n)\Big]=1$,又存在 L 个变迁 $t_{\text{out},1,l}^-(1\leqslant l\leqslant L)$.用双向弧把每项 $\psi_l(X_1,X_2,\cdots,X_n)(1\leqslant l\leqslant L)$中所涉及的输入量的库所与变迁 $t_{\text{out},1,l}^-(1\leqslant l\leqslant L)$相连接.

在算法 1 中,$\bigvee_{g=1}^{G}\Phi_g(X_1,X_2,\cdots,X_n)$表示有 G 个不同的逻辑表达式 $\Phi_g(X_1,X_2,\cdots,X_n)$相或,而每个 $\Phi(X_1,X_2,\cdots,X_n)$式表示某些输入量之间的与、逆与运算,同理式子 $\bigvee_{l=1}^{L}\psi_l(X_1,X_2,\cdots,X_n)$.若托肯在状态库所 $P_{X_j^0}(P_{Y_1^0})$中,表示当前该输入量(输出量)取值为“0”;相反,若托肯在状态库所 $P_{X_j^1}(P_{Y_1^1})$中,则表示当前该输入量(输出量)取值为“1”.激发任意一个输出变迁 $t_{\text{out},1,g}^+(1\leqslant g\leqslant G)$,会使得托肯从库所 $P_{Y_1^0}$ 中移到库所 $P_{Y_1^1}$ 中,在逻辑上实现输出量的值从“0”到“1”的转换;然而激发任意一个输出变迁 $t_{\text{out},1,l}^-$

($1\leqslant l\leqslant L$)，会使得托肯从库所 $P_{Y_1^1}$ 中移到库所 $P_{Y_1^0}$ 中，在逻辑上实现输出量的值从“1”到“0”的转换.

3　FPGA 组合逻辑系统的状态可达图

系统程序的 Petri 网模型已经建立，Petri 网的动态行为有效模拟了 FPGA 系统行为，揭示了变量间的逻辑关系. 因此，利用 Petri 网的可达图分析法可以进一步分析程序的运行，便于计算机枚举验证每个状态. 但是，Petri 网描述的是一个比 FPGA 系统更复杂的并发系统，理论上只要变迁满足使能条件就能被激发，这样就会生成很多无关状态. 为了避免这样的问题，算法 2 提出了一种可以等价描述 FPGA 组合逻辑系统运行过程的状态可达图的计算方法.

定义 1　假设 Petri 网系统 $(N,\boldsymbol{m}_0)$ 是一个 FPGA 组合逻辑程序的 Petri 模型，其中，$T=T_{\text{in}}\cup T_{\text{out}}$，$T_{\text{e,in}}\subset T_{\text{in}}$，$T_{\text{e,out}}\subset T_{\text{out}}$，$T_{\text{in}}$ 和 T_{out} 分别是输入和输出变迁集合，$T_{\text{e,in}}$ 和 $T_{\text{e,out}}$ 分别是可使能的输入和输出变迁集合.

定义 2　假设三元组 $G_{\text{FPGA}}=\langle M,E,W\rangle$ 是 FPGA 组合逻辑系统状态可达图，其中，$M=M_{\text{in}}\cup M_{\text{out}}$，$E=E_{\text{in}}\cup E_{\text{out}}$.

集合 M 中的每个节点对应系统的一个状态. 其中：M_{in} 是以实线圈表示节点的输入状态(由激发输入变迁得到的状态)集合；M_{out} 是以虚线圈表示节点的门级输出状态(由激发输出变迁得到的状态)集合；E 是状态节点间的有向边集合；E_{in} 是标有输入变迁的实线有向边集合；E_{out} 是标有输出变迁的虚线有向边集合；W 是集合 E 到 T 的一个映射，即每条有向边上的变迁标记的集合.

算法 2　FPGA 组合逻辑系统状态可达图生成算法如下.

输入：程序的 Petri 网系统

输出：$G_{\text{FPGA}}=\langle M,E,W\rangle$，$M=M_{\text{in}}\cup M_{\text{out}}$，$E=E_{\text{in}}\cup E_{\text{out}}$

步骤 1　令 $M_{\text{new}}=\varnothing$，$M_{\text{old}}=\varnothing$，$E=\varnothing$，$W=\varnothing$.

步骤 2　将初始状态 $\boldsymbol{m}_0$ 标记为“new”，并将 $\{\boldsymbol{m}_0\}\rightarrow M_{\text{new}}$.

步骤 3　若未计算的系统状态集合 $M_{\text{new}}\neq\varnothing$，则继续以下操作；否则算法结束，输出 $G_{\text{FPGA}}=\langle M,E,W\rangle$.

步骤 4　从集合 M_{new} 中任取一个标记为“new”的状态 $\boldsymbol{m}$.

步骤 4.1　若状态 $\boldsymbol{m}$ 与可达图已有的其他状态相同，将其标记为“old”，则已计算获得的系统状态集合 $M_{\text{old}}=M_{\text{old}}\cup\{\boldsymbol{m}\}$，然后转向步骤 4；若状态 $\boldsymbol{m}$ 与可达图已有的其他状态不相同，则进行以下操作.

步骤 4.2　如果在状态 $\boldsymbol{m}$ 下，没有使能的输入变迁和输出变迁，则将 $\boldsymbol{m}$ 标记为“dead end”，然后转向步骤 4. 如果在状态 $\boldsymbol{m}$ 下存在使能变迁，此时会有两种情况：一种是存在使能的输入变迁且有使能的输出变迁，则跳转到步骤 5；另一种是只存在使能的输入变迁，则跳转到步骤 6.

步骤 5　只要可使能的输出变迁集合 $T_{\text{e,out}}=\{t_{\text{out}}\mid\boldsymbol{m}[t_{\text{out}}\rangle\}\neq\varnothing$，$t_{\text{out}}\in T_{\text{out}}$，就要优先激发所有可使能的输出变迁，生成门级输出状态.

步骤 5.1　从集合 $T_{\text{e,out}}$ 中任取一个输出变迁 t_{out}，激发该变迁，生成输出状态 $\boldsymbol{m}'_{\text{out}}$.

步骤 5.2　将 $\{\boldsymbol{m}'_{\text{out}}\}\rightarrow M_{\text{out}}$，如果输出状态 $\boldsymbol{m}'_{\text{out}}$ 与可达图中已有的状态相同，则 $M_{\text{out}}=M_{\text{out}}\cup\{\boldsymbol{m}'_{\text{out}}\}$；否则，从状态 $\boldsymbol{m}$ 到输出状态 $\boldsymbol{m}'_{\text{out}}$ 之间画一条虚线有向边，则集合 $E_{\text{out}}=E_{\text{out}}+\{\langle\boldsymbol{m},\boldsymbol{m}'_{\text{out}}\rangle\}$；并在该虚线上标记输出变迁 t_{out}，则有向边上的变迁集合为 $\{W(\langle\boldsymbol{m},\boldsymbol{m}'_{\text{out}}\rangle)=t_{\text{out}}\}\rightarrow W$，说明在状态 $\boldsymbol{m}$ 下通过激发输出变迁 t_{out} 会生成输出状态 $\boldsymbol{m}'_{\text{out}}$.

步骤 5.3　因为从集合 $T_{\text{e,out}}$ 中取走了一个使能变迁 t_{out}，所以 $T_{\text{e,out}}=T_{\text{e,out}}-\{t_{\text{out}}\}$. 判断集合 $T_{\text{e,out}}$ 是否为空集，如果 $T_{\text{e,out}}\neq\varnothing$，即存在使能输出变迁，则返回步骤 5.1；如果 $T_{\text{e,out}}=\varnothing$，即不存在使能输出变迁，则继续以下操作.

步骤 5.4　因为标记为“new”的状态 $\boldsymbol{m}$ 是从集合 M_{new} 中取出的，所以集合 $M_{\text{new}}=M_{\text{new}}-\{\boldsymbol{m}\}$，并返回步骤 3.

步骤 6　当状态 $\boldsymbol{m}$ 下只存在使能的输入变迁，即 $T_{\text{e,in}}=\{t_{\text{in}}\mid\boldsymbol{m}[t_{\text{in}}\rangle\}\neq\varnothing$，$t_{\text{in}}\in T_{\text{in}}$，则继续激发一个使能的输入变迁，来改变输入状态.

步骤 6.1 从集合 $T_{e,in}$ 中任取一个输入变迁 t_{in}，激发该变迁，生成输入状态 $\boldsymbol{m}'_{in}$.

步骤 6.2 将 $\{\boldsymbol{m}'_{in}\}\rightarrow M_{in}$，如果 $\boldsymbol{m}'_{in}$ 与可达图中已有的状态相同，则集合 $M_{old}=M_{old}\cup\{\boldsymbol{m}'_{in}\}$；否则从状态 $\boldsymbol{m}$ 到 $\boldsymbol{m}'_{in}$ 之间画一条实线有向边，则集合为 $E_{in}=E_{in}+\{\langle\boldsymbol{m},\boldsymbol{m}'_{in}\rangle\}$；在该实线上标记输入变迁 t_{in}，则有向边上的变迁集合为 $\{W(\langle\boldsymbol{m},\boldsymbol{m}'_{in}\rangle)=t_{in}\}\rightarrow W$，说明在状态 $\boldsymbol{m}$ 下，通过激发输入变迁 t_{in} 会生成输入状态 $\boldsymbol{m}'_{in}$.

步骤 6.2.1 判断输入状态 $\boldsymbol{m}'_{in}$ 下是否存在可使能的输出变迁，如果 $\boldsymbol{m}'_{in}$ 下存在可使能的输出变迁，则跳转到步骤 6.2.2；否则，跳转到步骤 6.2.5.

步骤 6.2.2 在集合 $T_{e,out}=\{t_{out}\mid\boldsymbol{m}'_{out}[t_{out}\rangle\}\neq\varnothing$，$t_{out}\in T_{out}$ 中任取一个输出变迁 t_{out}，激发该变迁，生成输出状态 $\boldsymbol{m}''_{out}$.

步骤 6.2.3 将 $\{\boldsymbol{m}''_{out}\}\rightarrow M_{out}$，如果 $\boldsymbol{m}''_{out}$ 与可达图中已有的状态相同，则集合 $M_{old}=M_{old}\cup\{\boldsymbol{m}''_{out}\}$；否则，从状态 $\boldsymbol{m}'_{in}$ 到 $\boldsymbol{m}''_{out}$ 之间画一条虚线有向边，则集合为 $E_{out}=E_{out}+\{\langle\boldsymbol{m}'_{in},\boldsymbol{m}''_{out}\rangle\}$；在该虚线上标记输出变迁 t_{out}，则集合为 $\{W(\langle\boldsymbol{m}'_{in},\boldsymbol{m}''_{out}\rangle)=t_{out}\}\rightarrow W$，说明在输入状态 $\boldsymbol{m}'_{in}$ 下，通过激发输出变迁 t_{out} 会生成输出状态 $\boldsymbol{m}''_{out}$.

步骤 6.2.4 集合 $T_{e,out}=T_{e,out}-\{t_{out}\}$. 判断输入状态 $\boldsymbol{m}'_{in}$ 下的集合 $T_{e,out}$ 是否为空集，如果可集合 $T_{e,out}\neq\varnothing$，即存在使能的输出变迁，那么返回步骤 6.2.2；如果集合 $T_{e,out}=\varnothing$，即不存在使能的输出变迁，则继续以下操作.

步骤 6.2.5 因为在状态 $\boldsymbol{m}$ 下从集合 $T_{e,in}$ 中取走了一个使能输入变迁 t_{in}，所以 $T_{e,in}=T_{e,in}-\{t_{in}\}$. 判断集合 $T_{e,in}$ 是否为空集，如果集合 $T_{e,in}\neq\varnothing$，即存在使能的输入变迁，那么返回步骤 6.1；如果集合 $T_{e,in}=\varnothing$，即不存在使能的输入变迁，则继续以下操作.

步骤 6.3 未计算的系统状态集合 $M_{new}=M_{new}-\{\boldsymbol{m}\}$，并返回步骤 3.

在算法 2 中，$\langle\boldsymbol{m},\boldsymbol{m}'_{out}\rangle$ 表示从状态 $\boldsymbol{m}$ 指向 $\boldsymbol{m}'_{out}$ 的一条有向边，$W(\langle\boldsymbol{m},\boldsymbol{m}'_{out}\rangle)=t_{out}$ 表示在状态 $\boldsymbol{m}$ 下通过激发变迁 t_{out} 得到 $\boldsymbol{m}'_{out}$. M_{new} 是未计算的状态集合，M_{old} 是已计算获得的状态集合. 当 M_{new} 中某个可达状态被计算获得，则将其从 M_{new} 中剔除并添加到 M_{old} 中，直至 M_{new} 为空集，算法结束.

根据算法 2 可知：在某个电路状态下，如果同时存在使能的输入变迁和输出变迁，应当优先激发该电路状态下所有的输出变迁，得到一个稳定的门级输出状态；如果在某个电路状态下，只存在使能的输入变迁，则激发一个输入变迁，得到一个稳定的输入状态，通过改变输入量的取值，再判断该输入状态下是否存在使能的输出变迁.

4 实例分析

某化工原料生产反应釜，如图 1 所示. 系统启动后，当液位低于 S_1，V_1 打开，注入原料 A；当液位到达 S_1，V_1 关闭，同时打开 V_2 阀，注入原料 B；当液位到达 S_2，V_2 关闭，启动 M 加热；当温度值到达 S_3，M 停止加热，同时打开 V_3 阀，并且 L 启动计时；一段时间后，定时器 L 关闭，V_3 关闭，系统回到最初状态. 根据系统要求，某程序员给出图 1 所示反应釜控制系统的部分 VHDL 程序，如图 2 所示.

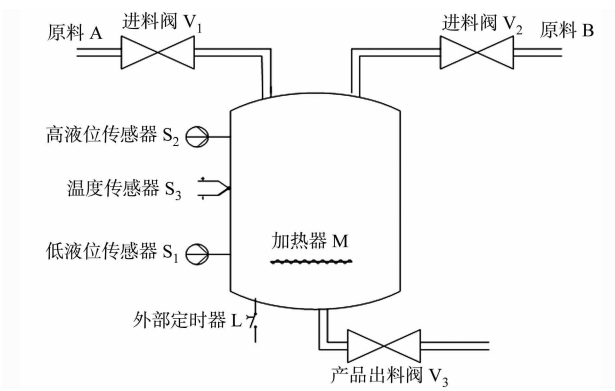

图 1 某化工原料生产反应釜

Fig. 1 A chemical raw materials production reactor

```
library ieee;
use ieee.std_logic_1164.all;
entity e_name is
port (S1, S2, S3: in bit; V1, V2, M: out bit; L, V3: buffer bit);
end entity e_name;
architecture a_name of e_name is
begin
V1 <= not S1 and not V3;
V2 <= S1 and not S2 and not V3;
M <= S2 and not S3;
L <= S3;
V3 <= S3 and L;
end architecture a_name;
```

图 2 反应釜控制系统的部分 VHDL 程序

Fig. 2 A part of VHDL program of the reactor

根据算法 1，针对图 2 的反应釜控制系统程序，首先将程序中的每个变量分别用一对库所表示，抽象为运行(on)和休息(off)状态，即逻辑上表示“1”和“0”两个状态；其次在输入变量的库所间加上输入变迁，在输出变量的库所间加上输出变迁；然后根据逻辑表达式所描述的输出量与输入量间的逻辑关系，描述出输入量库所与输出变迁间的控制关系；最后由算法 1，将图 2 的系统程序转换为图 3 的 Petri 网模型. 根据算法 2，由反应釜控制系统的 Petri 网模型，从某个初始状态开始，分别计算出每个输入状态下所对应的稳定的输出状态，得到如图 4 所示的系统状态可达图. 由图 4 分析可知：系统具有可逆性和活性，其中每个状态的表现形式为

$$\boldsymbol{m}=\left\{\begin{matrix}\boldsymbol{m}(p_{\mathrm{S}_1,\mathrm{off}}),\boldsymbol{m}(p_{\mathrm{S}_1,\mathrm{on}}),\boldsymbol{m}(p_{\mathrm{S}_2,\mathrm{off}}),\boldsymbol{m}(p_{\mathrm{S}_2,\mathrm{on}}),\boldsymbol{m}(p_{\mathrm{S}_3,\mathrm{off}}),\boldsymbol{m}(p_{\mathrm{S}_3,\mathrm{on}}),\boldsymbol{m}(p_{\mathrm{V}_1,\mathrm{off}}),\boldsymbol{m}(p_{\mathrm{V}_1,\mathrm{on}}),\\ \boldsymbol{m}(p_{\mathrm{V}_2,\mathrm{off}}),\boldsymbol{m}(p_{\mathrm{V}_2,\mathrm{on}}),\boldsymbol{m}(p_{\mathrm{M,off}}),\boldsymbol{m}(p_{\mathrm{M,on}}),\boldsymbol{m}(p_{\mathrm{V}_3,\mathrm{off}}),\boldsymbol{m}(p_{\mathrm{V}_3,\mathrm{on}}),\boldsymbol{m}(p_{\mathrm{L,off}}),\boldsymbol{m}(p_{\mathrm{L,on}})\end{matrix}\right\}^{\mathrm{T}},$$

部分状态标识为

$$\boldsymbol{m}_0=(1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0)^{\mathrm{T}},$$
$$\boldsymbol{m}_2=(0,1,1,0,1,0,0,1,1,0,1,0,1,0,1,0)^{\mathrm{T}},$$
$$\boldsymbol{m}_{10}=(0,1,0,1,0,1,1,0,1,0,0,1,1,0,1,0)^{\mathrm{T}},$$
$$\boldsymbol{m}_{18}=(0,1,0,1,0,1,1,0,1,0,1,0,0,1,1,0)^{\mathrm{T}}.$$

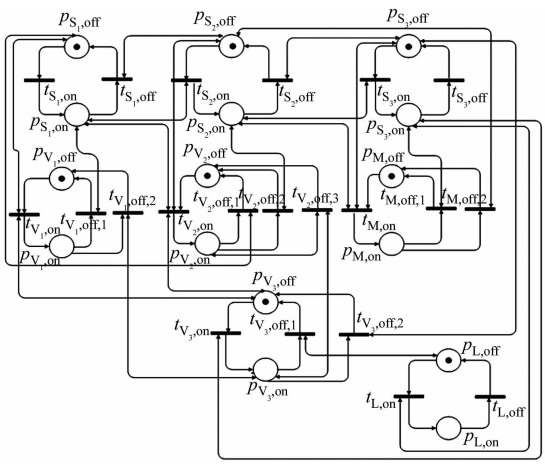

图 3　反应釜控制系统的 Petri 模型

Fig. 3　Petri net model of the reactor control system

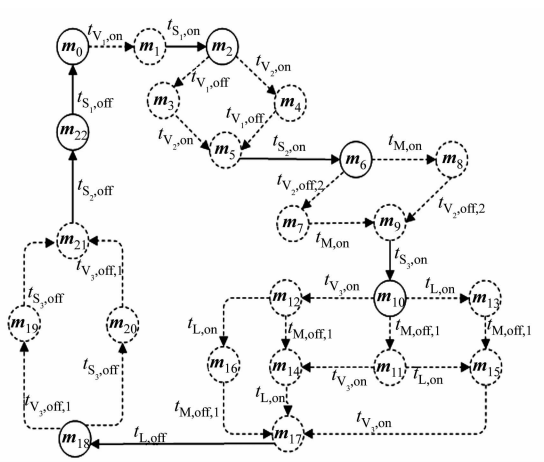

图 4　反应釜控制系统的 Petri 网模型的状态可达图

Fig. 4　State reachable graph of Petri net model for the reactor control system

5　结束语

提出了一个完整的将 FPGA 组合逻辑程序自动转换为普通 Petri 网的方法，与现有基于扩展 Petri 网的建模方法相比，文中基于普通 Petri 网的建模方法更具一般性，应用范围更广，对于系统变量间的逻辑功能关系有更强的分析能力. 另外，在考虑电路零延迟的情况下，根据已建好的程序 Petri 网模型，通过定义新的变迁激发规则，建立一个可以等价描述 FPGA 组合逻辑系统运行过程的状态空间，去掉一些无关的中间状态，指数级地压缩状态空间，为后续的形式化验证[17-18]提高效率. 后续工作将利用计算机通过系统状态可达图对程序进行形式化验证，检测存在逻辑错误的系统程序.

参考文献：

[1]　王芯，孙富明，李磊，等. FPGA 设计安全性综述[J]. 小型微型计算机系统，2010，31(7)：1333-1335.

[2]　杨海钢，孙嘉斌，王慰. FPGA 器件设计技术发展综述[J]. 电子与信息学报，2010，32(3)：716-718.

[3]　王彦本. 集成电路形式化验证方法研究[J]. 电子科技，2008，21(8)：4-7.

[4]　GHARBI A，KHALGUI M，BEN A S，et al. Optimal model checking of safe control embedded software components [C]// 15th Conference on Emerging Technologies and Factory Automation. Bilbao：IEEE Press，2010：1-8.

[5]　PATIL S，VYATKIN V，SOROURI M，et al. Formal verification of intelligent mechatronic systems with decentral-

ized control logic[C]//17th Conference on ETFA. Krakow:IEEE Press,2012:1-7.
[6] 古天龙.组合逻辑电路的Petri网仿真分析[J].系统仿真学报,1994,6(2):32-36.
[7] TSAI J I,TENG C C,LEE C H. Test generation and site of fault for combinational circuits using logic Petri-nets [C]//International Conference on Systems Man and Cybernetics. Taipei:IEEE Press,2006:8-11.
[8] 欧阳星明,胡青海.基于有色Petri网的逻辑电路仿真模型设计[J].华中科技大学学报:自然科学版,2006,34(3):18-20.
[9] BUKOWIEC A,ADAMSKI M. Synthesis of Petri nets into FPGA with operation flexible memories[C]//15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems. Tallinn:IEEE Press,2012:16-21.
[10] KOKASH N,ARBAB F. Formal design and verification of long-running transactions with extensible coordination tools[J]. IEEE Transactions on Services Computing,2013,6(2):186-200.
[11] OLCOZ S,COLOM J M. A Petri net approach for the analysis of VHDL descriptions[M]. Berlin Heidelberg: Springer,1993:15-26.
[12] WALTER D,LITTLE S,SEEGMILLER N,et al. Symbolic model checking of analog/mixed-signal circuits[C]// Asia and South Pacific Design Automation Conference. Yokohama:IEEE Press,2007:316-323.
[13] MOUTINHO F,GOMES L. State space generation algorithm for gals systems modeled by IOPT Petri nets[C]// 37th Annual Conference on Industrial Electronics Society. Melbourne,VIC:IEEE Press,2011:7-10.
[14] 藩松,黄继业.EDA技术与VHDL[M].北京:清华大学出版社,2009:42-47.
[15] LUO Ji-liang,NONAMI K. Approach for transforming linear constraints on Petri nets[J]. IEEE Transactions on Automatic Control,2011,56(11):2751-2765.
[16] DAVID R,ALIA H. Discrete continuous and hybrid Petri nets[M]. Berlin Heidelberg:Springer,2005:24-40.
[17] SCHWARICK M,ROHR C,HEINER M. MARCIE-model checking and reachability analysis done efficiently[C]// 8th Conference on Quantitative Evaluation of Systems. Aachen:IEEE Press,2011:91-100.
[18] KHALGUI M,MOSBAHI O,LI Z W,et al. Reconfigurable multiagent embedded control systems: From modeling to implementation[J]. IEEE Transactions on Computers,2011,60(4):538-551.

Modeling Method for FPGA Combinational Logic Program Based on Petri Net

CHEN Long, HUANG Ying-kun, LUO Ji-liang

(College of Information Science and Engineering, Huaqiao University, Xiamen 361021, China)

Abstract: In the view of field-programmable gate array (FPGA) combinational logic program, this research proposed an ordinary Petri net modeling method. First, a state variable is represented by a pair of places, while a logical operation is described by a transition. Consequently, a system program can be modeled by a common Petri net. Then, based on the dynamic analysis capability of Petri net, the computing method of system state reachable graph is given, and achieved the equivalence description of operational process between state reachable graph and FPGA combinational logic system. The results show that the Petri net can accurately describe the logic of the relationship between variables. The proposed method in this paper can be used for the formal design and verification of FPGA combinational logic program.

Keywords: field-programmable gate array; combinational logic; Petri net; modeling method; state reachable graph

(**责任编辑**:黄晓楠 **英文审校**:吴逢铁)