交互状态机模型模拟矢量自动生成方法*

李 暾+, 郭 阳, 李思昆

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Automatic Simulation Vector Generation Using Interacting FSM Model

LI Tun+, GUO Yang, LI Si-Kun

(School of Computer, National University of Defence Technology, Changsha 410073, China)

+ Corresponding author: Phn: 86-731-4575981, E-mail: tunlee@sina.com; chinadicom@sohu.com

http://www.nudt.edu.cn

Received 2002-05-31; Accepted 2002-09-04

Li T, Guo Y, Li SK. Automatic simulation vector generation using interacting FSM model. *Journal of Software*, 2003,14(3):628~634.

Abstract: Automatic simulation vectors generation is an efficient method to accelerate digital system's design verification process. An algorithm that generates coverage metrics and simulation vectors for state-pair of interacting FSM is presented in this paper. Compared with FSM (finite state machines) product method and the method which treats the interacting FSM as a single FSM, this algorithm can generate accurate coverage metrics and the shortest simulation vectors. Experimental results show that this algorithm is efficient in memory usage and perfectly solves the states space exploration problem.

Key words: P-ROBDD (partitioned-reduced ordered binary decision diagrams); automatic simulation vectors generation; coverage metrics; interacting finite state machines

摘 要: 模拟矢量自动生成方法是加速数字系统设计验证进程的有效手段.提出了一种针对数字系统交互状态机的状态组合、自动生成状态组合覆盖测度和状态组合覆盖模拟矢量的算法.与将交互状态机作为整体处理或构建状态机乘积的方法相比,该算法生成的模拟覆盖率测度精确,覆盖路径无回路,有效地提高了模拟验证的精度和速度.实验结果表明,该算法能高效地节省内存空间,较好地解决了状态空间爆炸问题.

关键词: 划分的化简有序二分决策图;模拟矢量自动生成;覆盖率测度;交互状态机

中图法分类号: TP391 **文献标识码**: A

现代数字系统的设计越来越复杂,验证设计的正确性也越来越困难.一般采用模拟验证和形式化验证两种方法.形式化验证可验证设计在各种输入情况下的行为,验证较为完备,但是形式化方法使用困难,还不能处理真正的大型系统.目前模拟验证仍然是验证设计功能正确性的主要方法,它能处理各种规模的设计.但是软件模拟无法穷尽设计的所有输入,经常存在未发现的设计错误.

在使用模拟验证方法时,有两个必要条件:模拟矢量和模拟覆盖率度量.目前主要采用随机生成和手工编写

* Supported by the National Natural Science Foundation of China under Grant No.69933030 (国家自然科学基金)

第一作者简介: 李暾(1974－),男,湖南郴州人,博士生,主要研究领域为并行模拟,微处理器设计验证技术,电子 CAD 技术.

相结合的方法生成模拟矢量.随机生成将产生大量冗余的模拟矢量,而无法激活真正关心的设计部分;手工编写模拟矢量是一项繁重的工作,只能测试有限的输入.现有的模拟矢量自动生成方法主要基于设计的有限状态机(finite state machines,简称 FSM)模型,以状态覆盖率或状态变迁覆盖率为测度,通过显式或隐式地遍历状态机生成模拟矢量.本文的研究主要解决交互状态机(interacting FSM)模型模拟矢量自动生成中状态空间爆炸的问题.

文献[1~4]是基于状态机模型自动生成模拟矢量的几个典型方法,此后的研究主要对这些方法进行优化.文献[1~4]在状态遍历过程中存在着状态空间爆炸问题,使得可处理的设计的规模有限.文献[1~4]主要针对单个FSM 进行模拟矢量生成.但是在数字系统中,各 FSM 存在着交互,针对单个 FSM 生成的模拟矢量,难以激活某些FSM 交互行为,而这些交互行为正是容易产生错误的地方.文献[5]通过构造 FSM 乘积的方法形式化地验证交互 FSM 模型,在求解状态机乘积的可达状态过程中进行化简.文献[6]采用随机模拟、符号模拟和限界模型检测(bounded model checking,简称 BMC)[7]搜索交互 FSM 的所有可达状态,并用化简有序二分决策图(reduced ordered binary decision diagrams,简称 ROBDD)[8]表示交互 FSM 进行非可达性分析.这两种方法都是将交互FSM 作为一个整体进行考虑,遍历其状态空间.虽然对模型进行了化简,但是没有从根本上化简状态空间.文献[6]的非可达性分析用速度换取精确度,无法为模拟程度提供可靠的度量.文献[9]对大型 FSM 模型采用分而治之的方法,利用交互 FSM 模型生成模拟矢量,但该方法本质上考虑的还是单个 FSM 的模拟矢量生成.

本文提出的算法采用划分的化简有序二分决策图(partitioned-ROBDD,简称 P-ROBDD)[10]表示交互 FSM 的状态变迁函数和输出函数,为交互 FSM 自动生成所有可能的状态组合作为模拟覆盖率度量,同时为每个可达状态组合生成一条从初始状态开始的路径、激活该路径的模拟矢量以及对应的预期状态和输出.本文的算法避免了通过建立 FSM 乘积或将交互 FSM 看做是单个 FSM 来遍历所有状态组合所存在的状态空间爆炸的问题,采用基于 P-ROBDD 的方法节省了大量内存空间,较好地解决了状态空间爆炸问题.生成的覆盖路径最短,不存在自回路和回路.

1 相关定义

在介绍算法之前,先给出与算法相关的一些定义.

定义 1. 有限状态机(FSM):有限状态机是一个六元组 $FSM=(I,O,S,S_0,\delta,\lambda)$.$I$ 是输入符号集合;O 是输出符号集合,$O=B_p$;S 是所有状态的集合,$S=B_m$;S_0 是初始状态;δ 是状态变迁函数,即 $\delta:S\times I\to S$;λ 是输出函数,即 $\lambda:S\times I\to O$.其中,$B=\{0,1\}$为取值域,对 $s\in S,s=v_0v_1\ldots v_{m-1},v_i\in B,0\le i<m$,称为状态 s 的编码.s'_i 为 s_i 的后续状态,则 $s'_i=\delta(i,s_i)$.某个状态 s_n 称为可达状态是指存在一条从 $init\in S_0$ 开始的并以 s_n 为终点的路径 $\Pi=init,s_1,\ldots,s_n$,其中 $s_{i+1}=\delta(i,s_i)$.如果路径存在一个片断 $s_k,s_{k+1},s_{k+2},\ldots,s_{l-1},s_l$,其中 $s_k=s_{k+1}=\ldots=s_{l-1}$,并且 $s_k\ne s_l$,则称路径中存在自回路.如果 $s_k=s_1$ 且 $s_k\ne s_{k+1}\ne\ldots\ne s_{l-1}$,则称路径中存在回路.这两种情况在模拟时都应尽量避免,因为这段模拟不会进入新状态,将浪费宝贵的模拟资源.

对于给定的 FSM 模型,其变迁关系是函数 $T:S\times I\times S\to\{0,1\}$,$T(s,i,s')=1\Leftrightarrow s'=\delta(i,s)$,$\Leftrightarrow$为等价符号.状态集 $A(A\subset S)$可被关联到其特征函数 $A:S\to\{0,1\}$;$A(s)=1\Leftrightarrow s\in S$.符号模型检测(symbolic model checking,简称 SMC)[11]用 ROBDD 来表示 FSM 的状态变迁关系和输出关系,是一种高效、紧凑的方法,能处理的状态可达 10^{20}.对给定的状态机,某一状态的前续状态和后续状态的计算分别称为 pre-image 和 image 运算.当用 ROBDD 表示状态机时,$Image(s)=(\exists s,i)[T(s,i,s')\bullet A(s)]$,$Pre\text{-}Image(s)=(\exists s',i')[T(s,i,s')\bullet A(s')]$.如图 1 所示,交互 FSM 是指,存在共同输入和相互间存在交互信号的一组 FSM.对交互 FSM 的形式化由定义 2 给出.

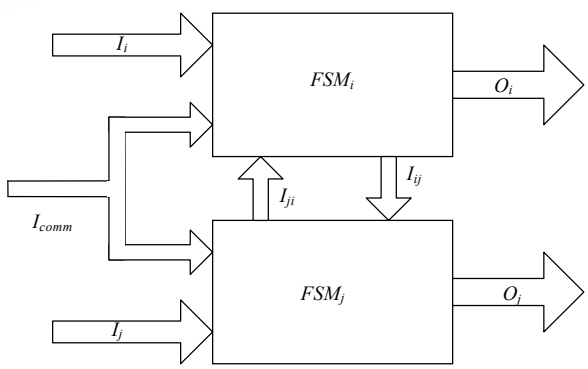

Fig.1 Model of interacting FSM
图 1 交互状态机模型

定义 2. 交互状态机.交互 FSM 定义为 3 元组 $IFSM=\{F,I,O\}$,F 是状态机集合,I 是所有状态机输入集合,O 是所有输出集合.当考虑两个状态机

FSM_i 和 FSM_j 的交互时,I_i 和 I_j 分别为两个状态机的输入集合,O_i,O_j 为输出集合,$I_{comm}=I_i\wedge I_j$ 称为 FSM 的共同输入,$I_{ij}=O_i\wedge I_j$ 为 FSM_i 到 FSM_j 的输出信号,并作为 FSM_j 的一部分输入.$I_{ji}=O_j\wedge I_i$ 为 FSM_j 到 FSM_i 的输出,同时也是 FSM_i 输入的一部分.(s_i,s_j)称为一个状态对或状态组合,其中 $s_i\in S_i,s_j\in S_j,S_i$ 和 S_j 分别是 FSM_i 和 FSM_j 的状态集.

状态对$(s'_{i,k},s'_{j,l})$是状态对$(s_{i,k-1},s_{j,l-1})$的后续状态对,当且仅当:(1) $s'_{i,k}=s_{i,k-1}$ 且 $s'_{j,l}=\delta_j(I_j,s_{j,l-1})$或 $s_{j,l}=s_{j,l-1}$;(2) $s'_{i,k}=\delta_i(I_i,s_{i,k-1})$并且 $s'_{j,l}=\delta_j(I_j,s_{j,l-1})$或 $s_{j,l}=s_{j,l-1}$.状态对$(s_{i,k},s_{j,l})$是可达状态对,当且仅当存在一条从初始状态对$(init_i,init_j)$开始,以$(s_{i,k},s_{j,l})$结束的路径 $\Pi=(init_i,init_j),(s_{i,1},s_{j,1}),\ldots,(s_{i,k},s_{j,l})$.路径中的回路和自回路与单个 FSM 的定义类似.

定义 3. Partitioned-ROBDD(P-ROBDD)[10].对给定的定义于 n 输入 $X_n=\{x_1,x_2,\ldots,x_n\}$的函数 $f:B^n\to B$,P-ROBDD 用 k 个函数对 X_f 来表示 f,$X_f=\{(w_1,f_1),\ldots,(w_k,f_k)\}$,其中 $w_i:B^n\to B$,$f_i:B^n\to B$,也是定义于 X_n 上的函数,并满足下述条件:

- w_i 和 f_i 都用 ROBDD 表示,变量顺序是 Π_i,$1\le i\le k$;
- $w_1+w_2+\ldots+w_k=1$;
- $w_i\wedge w_j=0$,$i\ne j$;
- $f_i=w_1\wedge f$,$1\le i\le k$,

其中的+和∧是布尔或和与运算.

从定义 3 可以看出,P-ROBDD 最适合于表示一组 FSM,这些 FSM 之间的依赖关系最小.

定义 4. FSM 乘积.对给定的 n 个状态机 $M_1,M_2,\ldots,M_n$,$M_i=(X_i,S_i,O_i,\delta_i,\lambda_i,Init_i)$,其中 S_i 为状态集,δ_i 为状态转换函数,λ_i 为输出函数,X_i 为输入集,O_i 为输出集,$Init_i$ 为初始状态集.状态机乘积 $M=M_1\times M_2\ldots\times M_n$ 的状态集为 $S=S_1\times S_2\ldots\times S_n$,输出 $O=O_1\times O_2\ldots\times O_n$.其中"×"为笛卡儿乘积.其输出函数和状态变迁函数分别为

$$\lambda(x,o)=\prod_{k=1}^{n}\lambda_k(x_k,o_k),$$

$$\delta(s,s')=\prod_{k=1}^{n}[\delta_k(s,x_k,s')\cdot\lambda_k(x_k,o_k).$$

2 模拟覆盖率测度生成

为了判断对设计的模拟验证程度,需要定义覆盖率测度,对于交互 FSM 模型,状态对覆盖率测度(state-pair coverage metrics)定义如下:

$$SCM=\frac{\text{已覆盖的状态对}}{\text{所有可达状态对}}.$$

通过遍历 FSM 组合的状态空间,可以求得所有可达状态对.已有的研究对交互 FSM 是通过建立 FSM 乘积来求所有的状态对的.当用 ROBDD 表示 FSM 进行可达状态求解时,ROBDD 的节点数和内存消耗随变量数的增大而成倍数地增加,所以当 FSM 乘积的变量数目很大时,能处理的设计规模不会很大.造成这种情况的主要原因是,由于变量顺序对 ROBDD 规模影响很大,当把 FSM 组合在一起时,此时的变量顺序相比于表示单个 FSM 时的变量顺序并不是最优的.而 P-ROBDD 对 FSM 组合中的各 FSM 采用单独表示法,每个 FSM 中的变量顺序都是不同的,但却是最优的.因此可大大地减少 ROBDD 的规模和内存开销[12].使用 P-ROBDD 表示 FSM 的一个问题是要将 FSM 划分成尽量独立的多个 FSM.

在交互 FSM 组合成的系统中,各 FSM 之间本来就是相互独立的,相比于划分一个 FSM 为多个 FSM,将多个 FSM 组合成一个大 FSM 要简单得多,因此用 P-ROBDD 来表示交互 FSM 系统是很自然的.当用 P-ROBDD 来表示多个 FSM 时,需要修改传统的 FSM 可达状态计算算法.文献[12]给出了基于 P-ROBDD 的可达性计算算法,但是该算法是针对单个状态,而不是针对状态对的.

传统的 FSM 可达状态计算算法如图 2 所示.

```
FSM_TRAVERSAL(Init,T(s′,s,i)){
  Reached(s)=New(s)=Init(s)
  While (New(s)≠0){
      N(s′)=IMAGE[New(s),T(s′,s,i)]
      New(s)=N(s′←s)–Reached(s)
      Reached(s)=Reached(s)+New(s)
  }
}
```

Fig.2 Traditional state traversal algorithm

图 2 传统的状态遍历算法

$Reached(s)$表示可达状态集合,$New(s)$是每一次迭代生成的新可达状态,$Init(s)$是初始状态.$IMAGE[New(s),T(s',s,i)]=(\exists x,i)(New(s)\wedge T(s',s,i))$.

两个状态机 FSM_i 和 FSM_j 的可达状态对是 FSM_i 和 FSM_j 的所有可达状态的组合,即在$(s_{i,k},s_{j,l})$中,$s_{i,k}\in Reached(FSM_i),s_{j,l}\in Reached(FSM_j)$.所有的状态对数目是$|Reached(FSM_i)|\times|Reached(FSM_j)|$,$|Set|$表示集合 Set 中的元素个数.因此,最直接的方法是同时求得 FSM_i 和 FSM_j 的可达状态数,依次从 $Reached(FSM_i)$中取元素与 $Reached(FSM_j)$中的状态组成可达状态对.但是考虑交互 FSM 间的交互信号和共同输入在某些状态组合下的冲突,上述可达状态对的计算是真正可达状态对数的上界,对判断模拟覆盖程度没有帮助.

我们的方法采用传统的可达状态计算算法,但是在每一次迭代时生成可达状态对,并判断输入冲突.在第 m 次迭代计算出 FSM_i 和 FSM_j 的 $New_m(FSM_i)$和 $New_m(FSM_j)$后,在计算新的 $Reached$ 集合之前,此时有 4 个状态集合,即第 m–1 次迭代的 $Reached_{m-1}(FSM_i)$和 $Reached_{m-1}(FSM_j)$以及 $New_m(FSM_i)$和 $New_m(FSM_j)$,可以形成 4 个状态组合对集合:

- S_{ri-rj}:$(s_{i,r},s_{j,r}),s_{i,r}\in Reached_{m-1}(FSM_i),s_{j,r}\in Reached_{m-1}(FSM_j)$;
- S_{ri-nj}:$(s_{i,r},s_{j,n}),s_{i,r}\in Reached_{m-1}(FSM_i),s_{j,n}\in New_m(FSM_j)$;
- S_{ni-rj}:$(s_{i,n},s_{j,r}),s_{i,n}\in New_m(FSM_i),s_{j,r}\in Reached_{m-1}(FSM_j)$;
- S_{ni-nj}:$(s_{i,n},s_{j,n}),s_{i,n}\in New_m(FSM_i),s_{j,n}\in New_m(FSM_j)$.

其中 S_{ri-rj} 中的状态对已经出现过,而其他 3 个状态对组合集合中的状态对可能有新的组合出现,因此,每次迭代时只需考虑 $S_{ri-nj},S_{ni-rj},S_{ni-nj}$ 中的状态对.

为了考虑共同输入和交互信号的冲突影响,对 IMAGE 的计算要进行修改,以 FSM_i 为依据,当计算出 $N(FSM_i)=IMAGE[New_i(s_i),\delta_i(s'_i,s_i,i_i)]$之后,$N(FSM_i)$中同时包含了 s'_I——后续状态,s_i——当前状态,i_i——引起状态变迁的输入.在计算 $N(FSM_j)$时,需要考虑 I_{comm} 和 I_{ij} 的制约因素.因此,$N(FSM_j)=IMAGE_EXTEND=IMAGE[New_j(s_j),\delta_j(s'_j,s_j,i_j)]\wedge I_{comm}(N(FSM_i))\wedge(I_{ij}),I_{comm}(N(FSM_i))$为从 $N(FSM_i)$中提取 I_{comm} 的操作.I_{ij} 的计算需要计算 FSM_i 在当前状态下的输出,即 $I_{ij}=O_{ij}(\lambda_i(o'_i,s_i,i_i))$.这将去除 FSM_j 中与 $N(FSM_i)$和 $O(FSM_i)$中存在输入冲突的可达状态.同样,可以计算 I_{ji},即 FSM_j 到 FSM_i 的输出.

算法如图 3 所示,以 FSM_1 和 FSM_2 的初始状态集和各自的状态变迁函数为输入,计算所有可达的状态对.算法中 $CONSTRUCT\_PAIR$ 操作用于构造来自两个状态集合的状态对.在生成新的 $Reached\_Pair$ 时,要检查新生成的状态对是否已存在.可知该算法有下述性质.

定理 1. 基于 P-ROBDD 的可达状态对算法是完备的.

证明:算法是完备的,即不存在某个状态对(s_1,s_2)是可达的,而算法没有计算出来.假设存在着某个状态(s_1,s_2)实际是可达的,但是我们的算法无法计算出该状态对,则根据可达的定义,存在一条开始于$(init_1,init_2)$,以(s_1,s_2)结束的路径 $\Pi=(init_1,init_2),(s_{1,1},s_{2,1}),\ldots,(s_1,s_2)$,其中一定存在这样两个状态对序列$(s_{1,i},s_{2,j})$和$(s_{1,i+1},s_{2,j+1})$,其中$(s_{1,i},s_{2,j})$存在于某次迭代的 $Reached\_Pair$ 中,而$(s_{1,i+1},s_{2,j+1})$不存在于下一次迭代的 $Reached\_Pair$ 中.

这种情况只有当 $s_{1,i+1}\neq s_{1,i}$ 且 $s_{1,i+1}\neq IMAGE[s_{1,i},T_1(s'_1,s_1,i_1)]$以及 $s_{2,j+1}\neq s_{2,j}$ 且 $s_{2,j+1}\neq IMAGE[s_{2,j},T_2(s'_2,s_2,i_2)]$时,才会出现.而根据算法,$s_{1,i+1}(s_{2,j+1})$只能等于 $s_{1,i}(s_{2,j})$,或 $s_{1,i+1}=IMAGE[s_{1,i},T_1(s'_1,s_1,i_1)](s_{2,j+1}=IMAGE[s_{2,j},T_2(s'_2,s_2,i_2)])$.因此不会出现$(s_{1,i},s_{2,j})$和$(s_{1,i+1},s_{2,j+1})$这样的状态对序列.由此可证算法能遍历所有可达状态对. □

```
STATE_PAIR_TRAVERSAL(Init1,BDD(FSM1),Init2,BDD(FSM2)){
    Reached_Pair=CONSTRUCT_PAIR[Init1,Init2]
    I12=O1(λ1(o'1,Init1,i1))
    I21=O2(λ2(o'2,Init2,i2))
    Reached (FSM1)=New(FSM1)=Init1
    Reached (FSM2)=New(FSM2)=Init2
    While (New(FSM1)<>0||New(FSM2)<>0){
        N(FSM1)=IMAGE[New(FSM1),δ1(s'1,s1,i1)]
        N(FSM1)=N(FSM1)∧I21
        New(FSM1)=N(FSM1)−Reached(FSM1)
        N(FSM2)=IMAGE_EXTEND[New(FSM2),δ2(s'2,s2,i2),N(FSM1)]
        New(FSM2)=N(FSM2)−Reached(FSM2)
        S1=CONSTRUCT_PAIR[Reached(FSM1),New(FSM2)]
        S2=CONSTRUCT_PAIR[New(FSM1),Reached(FSM2)]
        S3=CONSTRUCT_PAIR[New(FSM1),New(FSM2)]
        Reached_Pair=Reached_Pair+S1+S2+S3
        I12=O1(λ1(o'1,New(FSM1),i1))
        I21=O2(λ2(o'2,New(FSM2),i2))
        Reached(FSM1)=Reached(FSM1)+New(FSM1)
        Reached(FSM2)=Reached(FSM2)+New(FSM2)
```

Fig.3 Coverage metric computation algorithm

图 3 覆盖测度计算算法

3 模拟矢量自动生成

为了使所有的可达状态对生成输入序列,以便产生一条从初始状态对开始的覆盖目标状态对的路径,一种方法是在计算出可达状态对集合之后,依次选择每个可达状态对,利用 Pre-Image 计算该状态对的前续状态对,直至到达初始状态对.该方法没有充分利用可达状态对计算的中间信息,存在冗余计算;而且在 Pre-Image 计算中,可能会在某个状态对上生成自回路,使得模拟时在某个状态对上浪费大量的模拟周期,而不能很快地到达新状态对.

在计算可达状态对的算法中,稍加修改就能在计算出可达状态对的同时生成覆盖路径,而且由于算法中能保证每一次迭代中只会加入新的状态对,避免了状态对自回路,生成的覆盖路径是最短的.修改后的算法如图 4 所示.

在 $IMAGE[New(s),\delta(s',s,i)]$结果中,包含了当前状态、下一状态以及对应的输入.除了生成新的可达状态对外,还将为该状态对生成从前一状态对到该状态对的输入.具体做法是定义 3 个 Trace 表,一个用于保存 FSM_1 的 Trace 信息,即可达状态及从前一状态到达该状态时对应的输入,一个用于保存 FSM_2 的 Trace 信息,第 3 个用于保存状态对及其对应的输入,分别用 $Trace\_FSM_1$,$Trace\_FSM_2$ 和 $Trace\_Pair$ 表示.在计算出 $N(FSM_1)$和 $N(FSM_2)$之后,将在 $Trace\_FSM_1$ 和 $Trace\_FSM_2$ 表中加入新的状态及其输入.当生成新的状态对(s'_1,s'_2)时,在 $Trace\_Pair$ 中插入该状态对,首先判断该状态对是否已经存在,否则处理下一状态对.如果该状态对不存在,则在 $Trace\_Pair$ 中根据定义 2 查找满足成为其前一状态对的状态对,假设找到的前一状态对为(s_1,s_2),则分别在 $Trace\_FSM_1$ 和 $Trace\_FSM_2$ 中查找满足 $s_1 \rightarrow s'_1$ 和 $s_2 \rightarrow s'_2$ 所对应的输入,将该输入作为状态对的输入与新状态对一起加入 $Trace\_Pair$ 中,并保存指向其前一状态对的信息.

这样,在可达状态对计算结束时,遍历 $Trace\_Pair$,就能为所有的可达状态对生成一条从初始状态对开始的覆盖路径,并且该路径不会存在自回路和回路.因为在 $Trace\_Pair$ 表中,不会出现两个一样的状态对,因此能保证每条覆盖路径中都不会有两个相同的状态对.这样的覆盖路径可大大节省模拟时间,因为每一步模拟到达的都是新状态对.

```
STATE_PAIR_TEST_GEN(Init1,BDD(FSM1),Init2,BDD(FSM2)){
  Add(Pair_Trace_Table,[Init1,Init2])
  Add(Trace_FSM1,Init1)
  Add(Trace_FSM2,Init2)
  Reached_Pair=CONSTRUCT_PAIR[Init1,Init2]]
  I12=O1(λ1(o'1,Init1,i1))
  I21=O2(λ2(o'2,Init2,i2))
  Reached (FSM1)=New(FSM1)=Init1
  Reached (FSM2)=New(FSM2)=Init2
  While (New(FSM1)<>0||New(FSM2)<>0){
    N(FSM1)=IMAGE[New(FSM1),δ1(s'1,s1,i1)]
    N(FSM1)=N(FSM1)∧I21
    New(FSM1)=N(FSM1)−Reached(FSM1)
    N(FSM2)=IMAGE_EXTEND[New(FSM2),δ2(s'2,s2,i2),N(FSM1)]
    New(FSM2)=N(FSM2)−Reached(FSM2)
    Add(Trace_FSM1,New(FSM1))
    Add(Trace_FSM2,New(FSM2))
    S1=CONSTRUCT_PAIR[Reached(FSM1),New(FSM2),Trace_FSM1,Trace_FSM2,Pair_Trace_Table]
    S2=CONSTRUCT_PAIR[New(FSM1),Reached(FSM2),Trace_FSM1,Trace_FSM2,Pair_Trace_Table]
    S3=CONSTRUCT_PAIR[New(FSM1),New(FSM2),Trace_FSM1,Trace_FSM2,Pair_Trace_Table]
    Reached_Pair=Reached_Pair+S1+S2+S3
    I12=O1(λ1(o'1,New(FSM1),i1))
    I21=O2(λ2(o'2,New(FSM2),i2))
    Reached(FSM1)=Reached(FSM1)+New(FSM1)
    Reached(FSM2)=Reached(FSM2)+New(FSM2)
  }
For each reached state-pair in Pair_Trace_Table do
  Gen_Test(state_pair)
}
```

Fig.4 Automatic simulation vector generation algorithm

图 4 模拟矢量自动生成算法

4 实验结果

基于科罗拉多大学的 ROBDD 操作库 CUDD 库[13],在配置为 P4 1.4G,256MB 的 PC 机上实现了可达状态对计算算法和模拟矢量自动生成算法,并将该算法用于 PicoJava II[14]芯片的数据 Cache 单元(data cache unit,简称 DCU)的验证.该 DCU 是一个有 385 个 Latch 的电路,包含 4 个交互状态机,分别为 cache-fill(5 个 latch),cache-miss (6 个 latch),write-back(8 个 latch)和 zero-out(6 个 latch),状态机都采用 one-hot 编码,这样,当考虑由两个状态机组成的状态机时,理论上可达状态对为 6×5+6×6+6×8+6×5+5×8+6×8=232 个.但是,由于受到相互间交互信号和共同输入的制约,实际的可达状态对没有这么多.表 1 是用 ROBDD 计算可达状态时,与构造状态机乘积的方法的比较.表 2 是与文献[6]中将交互状态机作为一个状态机进行处理时的结果.

表中本文的算法的实验结果是两个状态机 ROBDD 最大结点数和最大活跃结点数之和,因为在算法中,这两个状态机的 ROBDD 是同时活跃的.从实验结果可以看出,我们的算法对可达状态对数的计算更加精确.在节省内存使用和减少 ROBDD 结点数方面,与 FSM 乘积的方法相比,BDD 最大结点数节省倍数最小为 9 倍,最大倍数为 65 倍,而最大活跃结点数节省倍数最小为 20 倍,最大为 80 倍.与文献[6]中的方法比较,BDD 最大结点数节省倍数最小为 4 倍,最大倍数为 50 倍,而最大活跃结点数节省倍数最小为 11 倍,最大为 65 倍.特别是对大电路,如 Miss-WB,Zero-WB 和 Fill-WB 这 3 个状态机组合,效果更加明显.

Table 1 Experimental results comparing with the FSM product method

表 1 与状态机乘积方法的比较

Interacting FSM	Reachable state-pairs in theory	Reachable state-pairs in fact	The method of this paper		FSM product based method		Improvement	
			Max BDD nodes	Max active nodes	Max BDD nodes	Max active nodes	Max nodes	Active nodes
Zero-Miss	36	30	7 154	2 766	113 442	72 478	15	26
Zero-Fill	30	10	4 088	1 677	38 836	34 893	9	20
Zero-WB	48	48	13 286	10 271	312 732	306 460	23	29
Miss-Fill	30	30	7 154	3 161	76 650	69 236	10	21
Miss-WB	48	27	16 352	11 755	385 294	376 493	23	32
Fill-WB	40	40	13 286	10 666	863 590	855 260	65	80

Table 2 Experimental results comparing with the method that treats interacting FSM as a single FSM

表 2 与将交互状态机当作一个状态机方法的比较实验结果

Interacting FSM	Reachable state-pairs in theory	Reachable state-pairs in fact	The method of this paper		Single FSM method		Improvement	
			Max BDD nodes	Max active nodes	Max BDD nodes	Max active nodes	Max nodes	Active nodes
Zero-Miss	36	30	3 066	1 107	30 662	17 712	10	16
Zero-Fill	30	10	2 044	612	12 265	7 957	6	13
Zero-WB	48	48	4 088	1 684	61 323	30 646	15	18
Miss-Fill	30	30	4 088	1 418	16 355	15 601	4	11
Miss-WB	48	27	6 132	2 475	85 849	61 879	14	25
Fill-WB	40	40	4 088	1 980	204 403	128 700	50	65

5 总 结

本文提出了一种针对交互状态机模型状态组合的覆盖率测度和模拟矢量自动生成算法,并以 PicoJava II 的DCU实验了该算法,获得了很显著的内存空间节省.今后我们将在下面几个方向继续研究:研究利用ATPG技术来处理设计的组合逻辑部分,将生成的状态机模拟矢量传播到初始输入上;研究以状态对变迁关系为覆盖率测度时的模拟矢量自动生成方法;研究多个状态机状态组合时的模拟矢量生成技术.

References:

[1] Ho R, Yang C, Horowitz M, Dill D. Architecture validation for processors. In: Patterson DA, ed. Proceedings of the International Symposium on Computer Architecture. Santa Margerita Ligure: ACM Press, 1995. 404~413.

[2] Shen J, Abraham JA. An RTL abstraction technique for processor microarchitecture validation and test generation. Journal of Electronic Testing: Theory and Application, 1999,16(1,2):67~81.

[3] Moundanos D, Abraham JA, Hoskote YV. Abstraction techniques for validation coverage analysis and test generation. IEEE Transactions on Computers, 1998,47(1):2~13.

[4] Geist D, Farkas M, Landver A, Lichtenstein Y, Ur S, Wolfsthal Y. Coverage-Directed test generation using symbolic techniques. In: Srivas MK, Camilleri AJ, eds. Proceedings of the Conference on Formal Methods in Computer-Aided Design. Palo Alto, CA: Springer-Verlag, 1996. 143~158.

[5] Aziz A, Kukula J, Shiple T. Hybrid verification using saturated simulation. In: Irwin MJ, ed. Proceedings of the 35th Design Automation Conference. San Francisco, CA: ACM Press, 1998. 615~618.

[6] Ho P-H, Shiple T, Harer K, Kukula JH, Damiano R, Bertacco VM, Taylor J, Long J. Smart simulation using collaborative formal and simulation engines. In: Proceedings of the International Conference on Computer Aided Design. San Jose, CA: IEEE Press, 2000. 120~126.

[7] Biere A, Cimatti A, Clarke E, Fujita M, Zhu Y. Symbolic model checking using SAT procedures instead of BDDs. In: Irwin MJ, ed. Proceedings of the 36th Design Automation Conference. New Orleans, LA: ACM Press, 1999. 317~320.

[8] Bryant RE. Graph-Based algorithms for Boolean function manipulation. IEEE Transactions on Computers, 1986,35(8):677~691.

[9] Liu CNJ, Yen C-C, Jou J-Y. Automatic functional vector generation using the interacting FSM model. In: Proceedings of the International Symposium on Quality Electronic Design. San Jose, CA: ACM Press, 2001. 372~377.

[10] Narayan A, Jain J, Fujita M, Sangiovanni-Vincentelli AL. Partitioned-ROBDDs——a compact, canonical and efficiently manipulable representation for Boolean functions. In: Rutenbar RA, Otten RHJM, eds. Proceedings of the International Conference on Computer Aided Design. San Jose, CA: ACM and IEEE Computer Society, 1996. 547~554.

[11] McMillan KL. Symbolic Model Checking. Boston: Kluwer Academic Publishers, 1994.

[12] Narayan A, Isles AJ, Jain J, Brayton RK. Reachability analysis using partitioned-ROBDDs. In: Rutenbar RA, ed. Proceedings of the International Conference on Computer Aided Design. San Jose, CA: ACM and IEEE Computer Society, 1997. 388~393.

[13] Somenzi F. CUDD: CU Decision Diagram Package. 2001. ftp://vlsi.colorado.edu/.

[14] Sun Microsystems. PicoJava technology. 1999. http://www.sun.com/microelectronics/communitysource/picojava/.