許容関数の概念を用いた表参照型FPGAの論理設計

5B－3

山下茂　上林彌彦
京都大学工学部

1　はじめに

近年，書き換え可能なゲートアレイ(FPGA:Field Programmable Gate Array)は技術的な性能が著しく発展し，柔軟な設計が可能であるため，ハードウェアの試作や再構成可能な計算機アーキテクチャーへの適用が進んできている．そのため，FPGAの論理設計手法に関する研究が盛んに行なわれるようになってきている[3]．本稿では，表参照型FPGAにマッピングされた回路を最適化する手法について述べる．本手法は設計改良による論理回路最適化手法であるトランスダクション法の許容関数の概念[2]を適用して冗長度を減らそうとするものである．

2　基本的事項

本論文では表参照型FPGAにマッピングされた回路を扱う．以下では，$f(l_i)$で論理ブロックl_iで実現されている論理関数，$IP(l_i)$でl_iの入力がつながっている論理ブロックの集合，$IS(l_i)$でl_iの出力がつながっている論理ブロックの集合を表す．$f(l_i)$を別の関数f'で置き換えても，回路の出力に変化がなければ，f'をl_iの許容関数[2]であるという．許容関数は一つとは限らないので，一般には許容関数の集合が考えられ，回路中の全論理ブロックに対して同時に変更可能な許容関数のみからなる集合をCSPF(Compatible Set of Permissible Functions)とよぶ．以下では，$G(l_i)$で論理ブロックl_iのCSPFを表し，これを用いて回路変換を行なう．$G(l_i)$は0, 1, *(don't care)の3値を要素として持つ関数である．

3　CSPFを用いた最適化手法

以下に表参照型FPGAにマッピングされた回路に対して前節で述べたCSPFを用いて冗長性を削減する手続きについて述べる．

今，ある論理ブロックl_iとそれより入力側からの段数が小さいl_jに対して，$G(l_j)=(**10)$, $G(l_i)=(*1*0)$が成り立ち，l_jが3つの入力f_1,f_2,f_3を持ち，それらで実現されている論理関数がそれぞれ，1011, 0101, 0111となっているとする．(簡単のため4ビットで考えている．) $G(l_i)\cap G(l_j)=(*110)$であるから，l_jの内部論理を変更することによって$f(l_j)$を(0011)から(*110)に含まれる論理関数に変更できれば，l_iの出力をl_jの出力で置き換え，回路の冗長度を減らすことができる．そのため，(*110)を表すf_1,f_2,f_3の積和形を図1に示される手続きSOPにより生成する．$SOP(F,f_1,f_2,\cdots,f_i)$は$f_1,f_2,\cdots,f_i$をある論理関数とし，$F$をあるCSPFとすると$F$に含まれるある論理関数を表す$f_1,f_2,\cdots,f_i$の積和形を探して，あればそれを返し，なければErrorを返すものである．図1において，True, およびFalseはそれぞれ恒真関数，恒偽関数を表し，'+'および'·'はそれぞれ論理和，論理積を表す．また，•は表1で定義されるような二項演算子である．これらの演算子による演算はオペランドのうちの1つでもErrorならErrorを返す．この例の場合には，まず(*110)を$(F_1\cdot f_1+F_0\cdot\overline{f_1})$の形に展開する．ここで，$F_1$は$(*110)\bullet f_1$によって計算され(**10)となり，$F_0$は$(*110)\bullet\overline{f_1}$によって計算され(*11*)となる．次に，同様に$F_1$を$f_2$で展開し，$(f_2\cdot(***0)+\overline{f_2}\cdot(***1))$となる．このように入力変数で以下の条件が満たされるまで展開していく．

- 展開した結果のF_1およびF_0にTrueまたはFalseが含まれる場合．(それ以上展開する必要がない．)
- 展開するための入力変数がなくなってしまった場合．(要求を満たすような関数に展開できなかったことを意味するため，Errorを返す．)

```
手続き SOP(F, f1, f2, ···, fi)
    if(i = 1){
        if(f1 ∈ F) return f1
        else if(f1' ∈ F) return f1'
        else return Error
    }
    else{
        F1 = F • f1
        if(False ∈ F1) F1' = False
        else if(True ∈ F1) F1' = True
        else {
            F1' = SOP(F1, f2, ···, fi)
            if(F1' = Error) return Error
        }
        F0 = F • f1'
        if(False ∈ F0) F0' = False
        else if(True ∈ F0) F0' =True
        else {
            F0' = SOP(F0, f2, ···, fi)
            if(F0' = Error) return Error
        }
        return (F1'·f1 + F0'·f1')
    }
```

図1: SOPのアルゴリズム

Optimization Method for Lookup-Table-Based FPGAs Using Permissible Functions
Shigeru YAMASHITA　Yahiko KAMBAYASHI
Faculty of Engineering, Kyoto University

表 1: 二項演算 •

First element \ Second element		
•	0	1
0	*	0
1	*	1
*	*	*

この例では, $((f_2 \cdot (***0) + \overline{f_2} \cdot (***1)) \cdot f_1 + (*11*) \cdot \overline{f_1})$ と展開に成功し, l_j の内部論理を $(f_1 \cdot \overline{f_2} + \overline{f_1})$ と変更した後で, l_i の出力を l_j で置き換えて回路より l_i を取り除くことができる. このように回路を最適化する手順を以下に示す.

step1 回路の出力側から順に論理ブロックを選び, l_i としてstep2へ. 論理ブロックがなくなれば終了.

step2 回路の中で l_i より回路の入力側からの段数が小さく, $G(l_i) \cap G(l_j)$ が空集合でないような論理ブロック l_j を選びstep3へ. そのような論理ブロックがなければstep1へ.

step3 手続きSOPにより l_j の内部論理を変更し, l_j で実現される関数を $(G(l_i) \cap G(l_j))$ に含まれる関数に変更してstep4へ. 手続きSOPによってそのような変更ができない場合にはstep1へ.

step4 l_i が回路全体の出力なら l_i を l_j で置き換えてstep8へ. そうでなければstep5へ.

step5 $IS(l_i)$ の中から論理ブロックを順に選び l_k としてstep6へ. 論理ブロックがなくなればstep8へ.

step6 $IP(l_k)$ に l_j が含まれていればstep7へ. そうでなければ, l_k の l_i との入力を l_j につけかえてstep5へ.

step7 l_i と l_k の間の結線を切り, l_k の内部論理の中で l_i に相当する論理を削除してstep5へ.

step8 l_i とそのすべての入力との間の結線を取り除きstep1へ.

step2において, l_j の段数が l_i より小さい理由は, l_j の方が入力側からの段数が大きいと, l_i の代わりに l_j をつなぐことによって回路の最長パスの段数が増える可能性があるためである. また, l_j の方が l_i より入力側からの段数が小さいためstep4で選ばれる l_k が必ず l_j より入力側からの段数が大きくなるので, step5において l_j から l_k に結線をつないでもループを作らないことが保証される. 次にstep7では, l_j が $IP(l_k)$ に含まれる場合 l_k の入力は l_j と既につながっているので, step5で l_k の l_i との入力を l_j につけかえる必要はなく, 単に l_i と l_k の間の結線を切れば良い. ただし, l_k の内部論理が, 例えば l_k の入力の積和形で表現されているとすれば, l_i に相当するリテラルをその積和形から削除する必要がある.

4 実験結果および考察

前節で述べた手法を実装し, MCNCベンチマーク回路に対して実験を行なった. その実験結果を表2に示す. 内部での関数表現には, 現NTT湊氏によるSBDDパッケージを用いている. 論理ブロックは5入力のものを仮定し, 初期回路はMIS[1]のコマンドにより生成されたものを用いた. 表中の'ブロック数'は回路中の論理ブロックの数を表している. また, 表中では良くなった結果を太字で示している. 表2より, MISのコマンドにより生成された初期回路に対して本手法は平均して9%論理ブロック数を削減していることがわかる. これにより, 本手法によりある程度の冗長度を削減することが可能であることが示された.

表 2: 実験結果

回路名	初期回路			本手法適用後		
	ブロック数	結線数	段数	ブロック数	結線数	段数
C432	122	317	13	**109**	**277**	13
C499	243	408	11	**217**	**370**	11
alu2	141	540	22	**137**	**526**	22
alu4	264	891	25	**258**	**870**	**24**
apex7	129	318	7	**128**	**313**	7
b9	90	197	3	**87**	**188**	3
c8	87	235	4	**85**	**228**	4
cordic	55	107	7	**51**	**96**	**6**
example2	195	445	5	**192**	**433**	**4**
i9	478	1490	9	**420**	**1262**	9
lal	83	213	5	**75**	**183**	5
sct	65	169	4	**62**	**162**	4
term1	173	553	8	**141**	**424**	8
too_large	352	1250	13	**280**	**960**	**11**
vda	400	1683	5	**367**	**1518**	5

謝辞

有益な御助言, 御指導下さるイリノイ大学計算機科学科室賀三郎教授, 九州大学工学部澤田助手, ならびにSBDDパッケージを使用させていただいた矢島研究室の皆様に深謝する. なお本研究は文部省科学研究費 (国際学術研究) によるものである.

参考文献

[1] R. Brayton, E. Detjens, S. Krishna, T. Ma, P. McGeer, L. Pei, N. Phillips, R. Rudell, R. Seagal, A. Wang, R. Yung and A. Sangiovanni-Vincentelli, "Multiple-Level Logic Optimization System", Proc. IEEE Internatinal Conference on Computer Aided Design, pp.356-359, Nov. 1986.

[2] S. Muroga, Y. Kambayashi, H. C. Lai, J. N. Culliney, " The Transduction Method-Design of Logic Networks Based on Permissible Functions", *IEEE Trans. Comput.*, Vol.38, No.10, October 1989.

[3] Stephen D.Brown, Robert J.Francis, Jonathan Rose, Zvonko G.Vranesic, "FIELD-PROGRAMMABLE GATE ARRAYS", Kluwer Academic Publishers.