FPGAブロックによる複数関数実現を利用したFPGA回路最適化

1R－3

幸田武範　上林彌彦
京都大学工学研究科

1　はじめに

FPGA(Field Programable Gate Arrays)を構成する論理ブロックは、1つのブロックで1つのk変数関数を実現するだけでなく、複数の$h(<k)$変数関数を実現することが可能であることが多い。これまでにも後者の特徴を生かした手法として、1つのブロックにh変数以下の関数を複数詰め込むことでブロック数を減少させる手法[2][3]などが提案されている。しかし、これらの手法も論理ブロックのもつ特徴を完全に生かしているとはいえない。

本稿で提案する手法は、一部の内部セルを他のブロック出力又はそれを構成する内部関数と置換するものである。その結果、置換後のセルは従来の出力とともに取り込まれたブロックの出力関数を構成する一部としても利用されることになる。このようなブロック統合を行うことで、回路最小化などの効果を得ることができる。高水準なブロック統合を実現するためには、置換対象となる内部セルの満すべき条件を限りなく少なくすることが重要となる。ブロックやセルの出力の満すべき条件を文献[1]でNTTの山下らが提案したDistinguished Pairs(DPs)の概念を用いて表現した。DPsを用いたFPGAブロックの冗長性の表現法として、同文献にSPFDs (Sets of Pairs of Functions to be Desitinguished)が提案されている。しかし、SPFDsは現在の内部セルの関数に依存するため、今回の用途においては必ずしもすべての冗長度を表現できるわけではない。そこで我々は、そのブロックの満すべき条件を特定の内部論理セルに集中させ、置換対象となる論理セルの満すべき条件を緩和するという方針により、ブロックの内部関数に依存せずに常にSPFDsと同等かそれ以上の冗長度を求められる手法を考案した。本稿では、前述のDPsによる内部セルの条件の計算法を中心に、FPGA回路のブロック統合手法について述べる。

2　FPGAブロックの構造とその利用法

ここで扱うXILINX社のXC4000シリーズは、現在最もよく利用されているFPGAで、その論理ブロックの構造は図1に示す通りである。

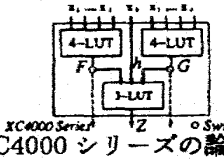

図1: XC4000シリーズの論理ブロック

仮にブロック入力を$x_0,\cdots,x_4$とした場合、論理セル（内部関数）に相当するLUT(Look Up Table)の出力F,Gとブロック出力Zは以下の式で表現できる。尚、$f_*()$は括弧内の変数を入力と持つ任意関数を意味する。

$$F = f_f(x_1,x_2,x_3,x_4)$$
$$G = f_g(x_1,x_2,x_3,x_4)$$
$$Z = f_h(h,F,G) \quad \text{ただし } h = x_0$$

これまでの手法は図1に示すように、1ブロックで任意の5変数関数Zを1つだけ実現し利用していた。近年では、ZのかわりにF,Gの出力を直接出力することにより、2つの4変数以下の関数を実現する手法[2](図2(a))が提案されるようになってきた。さらに文献[3]では、5変数関数Zを特定形（例：$Z = h \cdot F$）に変形することで5変数関数Zと他の4変数以下の関数G'を1つのブロックで実現する手法（図2(b)）も提案されている。しかし、この手法においても、Zを特定形に変形できない場合は、従来のようにZは単独で1つのブロックを利用するしかなかった。そこで本稿で述べる設計手法では、これらの利用法に加えて図2(c)に示すように5変数関数の内部関数をそのまま利用することで、ブロックの効率的な利用を実現している。

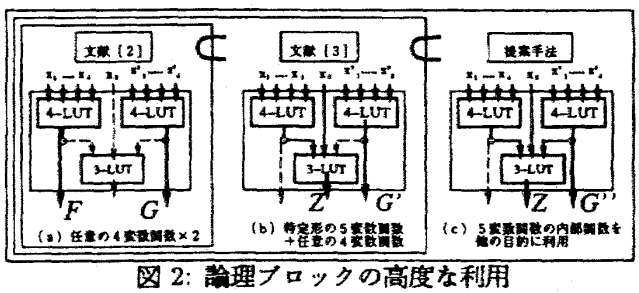

図2: 論理ブロックの高度な利用

FPGAの場合Fが任意関数を実現できることから、図3に示すようにZの満すべき条件を可能な限りFに集め、Gの制約を減らすことが可能である。その結果、計算されたGの条件さえ満す関数であれば元来のGのかわりに利用でき、他ブロックとの統合を実現できる。

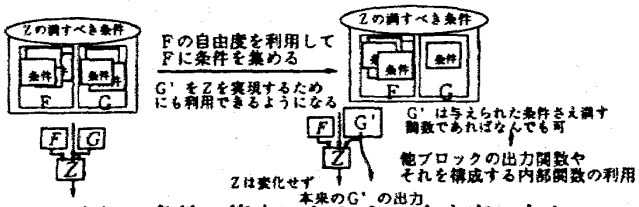

図3: 条件の集中によるGの自由度の向上

3　内部関数の満すべき条件の計算法

ここでは、内部関数の満すべき条件の計算法を説明する。例として、回路入力X_0,X_1,X_2に対する真理値表を考えた場合、処理対象ブロックの入力関数$x_0,\cdots,x_4$及びに出力関数Z, 内部関数F,Gが表1のようになる場合を考える。Zは$h(=x_0)$と$x_1,\cdots,x_4$を入力に持つ4変数関数であるF,Gを入力とする3LUTの出力関数である。表内の識別子a_{ijk}は、回路入力の値が$X_0=i, X_1=j, X_2=k$である入力組に対応する行を示す。

表1: 内部関数の満すべき条件の計算例

識別子	x_1	x_2	x_3	x_4	$h(=x_0)$	F	G	Z
a_{000}	0	0	0	0	1	1	1	1
a_{100}	0	1	0	1	0	0	1	0
a_{010}	0	0	0	0	0	1	1	1
a_{110}	1	1	0	1	1	0	0	0
a_{001}	1	1	1	1	0	1	1	1
a_{101}	1	1	0	1	0	0	0	1
a_{011}	0	1	0	0	1	1	0	0
a_{111}	1	1	1	0	1	0	1	0

ブロック出力Zの満すべき条件とは、Zの値が1となる入力組(ON-Set)と0となる組(OFF-Set)を区別できることである。そこで、まず区別すべき入力組の対を識別子a_{ijk}を用いて表現する。

(1) 区別すべき対(当該ブロックが満すべき条件)：
(ON)×(OFF)=

FPGA Circuit Optimization Utilizing Multiple-Output Capability of FPGA Blocks, Takenori KOUDA, Yahiko KAMBAYASHI, Department of Information Science, Kyoto University

$(a_{000}, a_{010}, a_{001}, a_{101}) \times (a_{100}, a_{110}, a_{011}, a_{111})$ より
(a_{000}, a_{100}),(a_{000}, a_{110}), (a_{000}, a_{011}),(a_{000}, a_{111}),
(a_{010}, a_{100}),(a_{010}, a_{110}), (a_{010}, a_{011}),(a_{010}, a_{111}),
(a_{001}, a_{100}),(a_{001}, a_{110}), (a_{001}, a_{011}),(a_{001}, a_{111}),
(a_{101}, a_{100}),(a_{101}, a_{110}), (a_{101}, a_{011}),(a_{101}, a_{111}),

これらの対がブロック内の情報によりすべて区別できれば、ブロック入力や内部論理がどのような関数であっても、ブロック出力は Z となる。

次に、3LUT の入力である h, F, G について調べる。今、回路入力の組み合わせに対する 1 変数 h の値を (1) と同様の方法で調べた場合、以下の (2) に示す対が区別可能である。

(2) h により区別可能な対：(ON)×(OFF)=
$(a_{000}, a_{110}, a_{011}, a_{111}) \times (a_{100}, a_{010}, a_{001}, a_{101})$ より
(a_{000}, a_{100}),(a_{000}, a_{010}), (a_{000}, a_{001}),(a_{000}, a_{101}),
(a_{110}, a_{100}),(a_{110}, a_{010}), (a_{110}, a_{001}),(a_{110}, a_{101}),
(a_{011}, a_{100}),(a_{011}, a_{010}), (a_{011}, a_{001}),(a_{011}, a_{101}),
(a_{111}, a_{100}),(a_{111}, a_{010}), (a_{111}, a_{001}),(a_{111}, a_{101}),

(1) の中から、(2) の情報により区別可能となった対を除いたものが (3) である。

(3) F, G により区別すべき対：
(a_{000}, a_{110}),(a_{000}, a_{011}),(a_{000}, a_{111}),
(a_{010}, a_{100}),(a_{001}, a_{100}),(a_{101}, a_{100})

これらを 3LUT の残りの入力である F,G で区別できれば、(1) に示された条件のすべてを満せる。

ここで、ブロック入力 $x_1, \cdots, x_4$ を考えた場合、これらの組合わせでは例えば a_{000} と a_{010} に対応するブロック入力の値はどちらも $\{x_1, x_2, x_3, x_4\}=\{0000\}$ となるため、$x_1, \cdots, x_4$ の組合わせでは a_{000} と a_{010} を区別することはできない。そのため、$x_1, \cdots, x_4$ の組合わせにより生成される F では、a_{000} と a_{010} に対応する値が同じ値となるという制約が発生する。同様に、a_{110} と a_{101} に対応する値も同じ値となる。尚、G についても、$x_1, \cdots, x_4$ の組合わせにより実現する場合には同様の制約がかかるが、本手法においては元の G のかわりに他ブロックの出力等を用いることを目標とするため、現時点では G にはブロック入力組による制約は考慮する必要がない。

(4) ブロック入力組による F の満すべき制約
$a_{000} = a_{010}$, $a_{110} = a_{101}$

次に条件グラフ (図 4) を作成する。条件グラフは、(3) で示した対を順に登録したものである。ブロック入力組により与えられた F の制約 (4) も同時に登録する。その結果、最大 2 つのグラフを得ることができる

(5) 条件グラフ生成

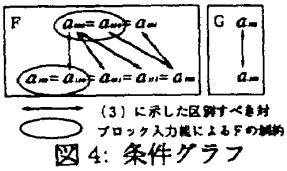

図 4: 条件グラフ

条件を多く含む方 (ブロック入力組の制約を与えた方) を F の満す条件、残りを G の満すべき条件とする。

次に計算された条件より、実現すべき関数を求める。F については、それぞれ識別子 $a_{000}, a_{010}, a_{001}$ に対応する値を 1、それ以外を 0 とおく。条件に含まれていない識別子に対応する値は任意値 $*$ となる。その結果得られる関数を表 2 に示す。G については、G で実現すべき関数の自由度を減らさないために値を 0 か 1 のどちらかに決定せず、NTT の山下らが提案した DPs[1] の概念を用いて、対の関数により「区別する」という情報のみを表現する。その例を表 2 の「G の条件」に示すように表現する。尚、A と $\overline{A}$ は DFs で相反する値である。

今、他のブロックの出力関数又はそれを構成する内部関数が例えば表 2 の G_1,G_2 のいずれかであった場合、これらの関数は G の満すべき条件を満しているため、現在の G と交換することが可能である。当該論理ブロックの内部論理は、Z が ON-Set となる h, F, G の組合わせより、以下の様に求めることができる。

表 2: F と G の自由度

識別子	Z	F	G の条件	$SPFDs(G)$	G_1	G_2
a_{000}	1	1	*	B	1	1
a_{100}	0	0	$\overline{A}$	$\overline{A}$	1	0
a_{010}	1	1	*	*	0	0
a_{110}	0	0	*	*	0	1
a_{001}	1	1	*	*	0	0
a_{101}	1	0	A	A	0	1
a_{011}	0	0	*	$\overline{B}$	0	1
a_{111}	0	0	*	*	1	1

A,$\overline{A}$,B,$\overline{B}$:Distinguished Pairs(DPs)

$$Z = h \cdot F + \overline{h} \cdot \overline{F} \cdot G_i + \overline{h} \cdot F = F + \overline{h} \cdot \overline{F} \cdot G_i$$
(G_i は G の条件を満す関数なら何でもよい)

ここで比較のため、本手法と同様に DPs の概念を用いたもので、FPGA ブロックの実現関数の自由度を効率よく表現できる SPFDs を用いて G の自由度を表現した例を示す。SPFDs は現在の h,F,G を元に計算されるものであるため、これらの関数の初期値によって表現される G の自由度が変化する。しかし本手法は、F.G の初期値によらず条件グラフの生成法のみに依存するため、常に最大限の自由度を表現できることが特徴である。尚、本手法は G の自由度を最大にするという戦略に特化した SPFDs の一種ともいえる。

仮に図 4 の条件グラフが 1 つになった場合には、G の条件が全て $*$ 要素のみになる。この場合、G は Z の実現のためには不要であることを示し、図 2(b) に示すブロック利用法が可能となる。

4 多出力特性を利用したブロック統合

前述の手法を用いて、G の部分に他ブロック又は内部のセルを取り込む。その結果、内包されたブロック (セル) は元々の出力関数を出力するとともに、Z を実現するための内部要素としても利用されることになる。

現在は、本手法を C 言語にて実装中である。発表の際には実験結果も示し、本手法の有用性を検証したい。

5 おわりに

本稿では、FPGA の論理ブロックが複数の関数を実現できることに注目した設計手法の提案を行った。本手法は、出力関数を構成する内部関数の満すべき条件を意図的に一部の内部関数に集中させることで、残りの内部関数の自由度を広げている。その結果、元々の内部関数のかわりに他ブロックの出力関数やそれを構成する内部関数による置換を可能とし、ブロック数の減少を実現できるものと思われる。現在本手法の実装を行っているところである。実装終了後、ベンチマーク回路に対する実験を行い有用性を検証する予定である。

謝辞

有益なご助言をしていただいた NTT コミュニケーション科学研究所の山下氏、名古屋氏及びに当学上林研究室の皆様に深謝します。

参考文献

[1] S.Yamashita, H.Sawada, A.Nagoya, "A New Method tp Express Functional Permissibilities for LUT based FPGAs and its Applications", Proc. of IEEE/ACM ICCAD'96, pp.254-pp.261, Nov.1996

[2] J.Cong, J. Peck, and Y. Ding, "RASP: A General Logic Synthesis System for SRAM-based FPGAs", Proc. of ACM/SIGDA FPGA'96, Monterey, California, Feb. 1996.

[3] 幸田武範、上林彌彦、"ブロック統合を用いた FPGA 面積最小化", 情報処理学会研究報告, 97-DA-85, pp.105-112, Oct. 1997