

## 順序回路からの算術演算機能の抽出

大村昌彦† 安浦寛人‡ 田丸啓吉†

†京都大学 工学部

‡九州大学大学院 総合理工学研究科

我々は、組合せ回路からその機能情報を抽出する技術について、研究を行なって来た。この技術は、論理設計の検証の非常に有力な手段になると考えられる。我々が提案した手法の大きな利点の一つとして、人間の目で見てできるだけわかりやすい記述を抽出することができる、ということが挙げられる。そのために、論理設計の検証が非常に容易に行なえるようになる。本稿では、データレジスタを持つ同期式順序回路から、できるだけわかりやすい機能記述を抽出する手法を紹介する。

## Extraction of Functional Information from Sequential Circuits

Masahiko OHMURA†, Hiroto YASUURA‡, and Keikichi TAMARU†

†Faculty of Engineering, Kyoto University  
Kyoto 606, Japan

‡Department of Information Systems, Kyushu University  
Kasuga-shi, Fukuoka 816, Japan

We have been studying on the technique of functional information extraction from combinational circuits. This technique will be very useful for logic verification. One of the strong points of our technique is that we can represent extracted functions as understandably as possible for human, so verification can be easily done. In this report, we present how to extract understandable functional descriptions from synchronous sequential circuits with data registers.

## 1 まえがき

LSI の設計工程において、設計された回路が与えられた仕様を満足するかどうかを確認する設計検証の問題は、非常に重要である。現在、設計検証の一手法として、シミュレーションが広く用いられているが、これは設計された回路が全く誤りを含んでいないということは保証できない。なぜなら、大規模な回路では、全ての入力パターンに対してシミュレーションを行なうことができず、いくつかの代表的な入力パターンに対して、その動作を確かめているに過ぎないからである。近年、シミュレーションを用いない検証方式として、形式的検証に関する研究が盛んに行なわれている。また、設計された記述から設計前の記述を抽出する技術も研究されており、レイアウト設計のレベルにおいては、既に回路抽出という形で実用化されている。さらにここ 2、3 年の間には、スイッチレベルあるいはゲートレベルの回路記述から機能記述を抽出するさまざまな手法が提案されている [1, 2, 3]。

我々は、このような機能抽出の技術が今後ますます重要になると考え、その研究を行なってきた [4]。我々の手法は、組合せ回路のゲートレベルの記述から、論理演算機能および算術演算機能を抽出するものであるが、その特徴として、抽出された機能を人間の目で見てもわかりやすい記述で表す、ということが挙げられる。例えば、加算機能のような算術演算機能は、複雑な論理式ではなく簡潔な算術式で表現されるので、設計検証を行なう場合に、人間の目で見ても容易に行なうことができる。

本稿では、新たにこの手法を順序回路に適用することを考える。基本的な考え方は、与えられた順序回路からクロック信号とフリップフロップを取り除き、組合せ回路の問題に帰着する、ということである。こうして得られた組合せ回路において、もとの順序回路の状態変数に相当する入力に適切な値を与えながら機能表を作成し、それらを連結することによって状態遷移表を作成することができる。もし、与えられた回路がデータレジスタとして用いられているフリップフロップを含んでいる算術演算回路であれば、状態の遷移の様子を記述するよりも、演算の結果としてレジスタに蓄えられていたデータがどのように移動するかを記述した方が人間の目で見ても理解しやすい。本稿では、このような順序回路からも算術演算機能を抽出する手法を説明する。

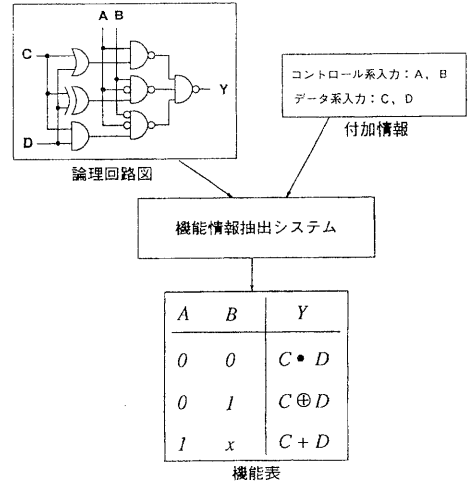


図 1: 機能情報抽出の概要

まず 2 節においては、同期式順序回路からの機能情報抽出の基本手法について述べる。次に 3 節において、データレジスタを持つ順序回路から算術演算機能を抽出する手法について説明する。4 節においては、試作システムを用いた実例を紹介する。

## 2 同期式順序回路の機能情報抽出

### 2.1 概要

我々が提案してきている機能情報抽出とは、与えられた論理ゲートレベルの回路記述およびその回路に関する付加情報から、回路の機能を抽出し、できるだけ簡潔に記述するというものである。図 1 にその概要を示す。付加情報は、機能情報抽出を行なう際に役に立つ何らかの情報で、ネットリストなどの回路記述には陽には記述されていないものである。例えば図 1 においては、信号線 A と B がコントロール系の入力であり、信号線 C と D がデータ系の入力である、という付加情報がユーザによって与えられている。このような付加情報が与えられると、同図のように、コントロール系入力 A と B の値によって回路の機能を 3 通りに分割した簡潔な機能表が得られる。

組合せ回路の出力は、入力変数の論理関数で一意に記述できる。従って、以下の手順によって容易にその機能を記述することができる。

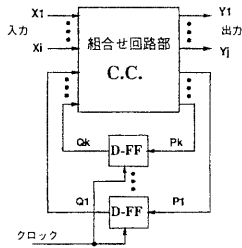


図 2: 同期式順序回路

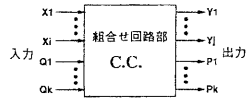


図 3: 図 2 の組合せ回路部

1. 各出力変数を入力変数の論理関数で表す。
2. 各論理関数を、コントロール系入力変数の値によって幾つかに分割する。
3. 分割された論理関数を、データ系入力変数の論理式によって表す。

順序回路の場合は、フリップフロップなどの記憶素子を含むため、入力信号の値が定まっても、出力変数の値を一意に定めることはできない。これらの値は、フリップフロップに保存されている値に依存する。ここで、我々が本稿で取り扱う回路を、図 2 に示したような同期式順序回路に限定する。この回路は、組合せ回路部分と記憶部分から構成され、すべてのフリップフロップが単一のクロック信号によって同時に駆動される、というものである。この回路において、入力信号は  $X_1, \dots, X_i$ 、出力信号は  $Y_1, \dots, Y_j$  である。また、各 D-FF の出力  $Q_1, \dots, Q_k$  は現状態を表す状態変数であり、入力  $P_1, \dots, P_k$  は次状態を表す状態変数である。また、本稿においては状態（即ち、状態変数を要素とするベクトル）を文字  $S$  で表す。

図 2 の回路の動作は、以下のように記述できる。

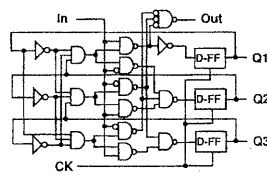
- (1) 組合せ回路部分において、入力変数と現状態変数の値から、出力変数と次状態変数の値が決定される。
- (2) クロック信号が与えられると、次状態変数にセットされた値が、現状態変数に移され、D-FF に保存される。

機能情報抽出の観点から考えると、(1) の動作が (2) の動作よりも重要である。そこで (1) の動作に注目するため、図 3 に示したように、図 2 の回路から D-FF とクロック信号を取り除いた組合せ回路を考える。この組合せ回路の入力は  $X_1, \dots, X_i, Q_1, \dots, Q_k$  であり、出力は  $Y_1, \dots, Y_j, P_1, \dots, P_k$

である。次に、この回路に従来の手法を適用して状態遷移表を作成する手法を、実例を示しながら説明する。

## 2.2 状態遷移表の作成例

ここでは、図 4(a) の回路から状態遷移表を作成する手法を説明する。まず、3 つの D-FF の出力をそれぞれ  $Q_1, Q_2, Q_3$ 、入力を  $P_1, P_2, P_3$  とおく。クロック信号と D-FF を取り除くことによって、4 入力 4 出力の組合せ回路が得られる。ここで、回路本来の入力変数  $In$  および新たに付加された入力変数  $Q_1, Q_2, Q_3$  は、全てコントロール系入力



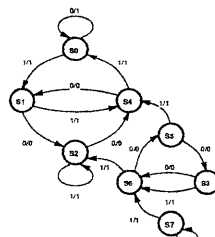
(a) 回路図

Q1	Q2	Q3	In	P1	P2	P3	Out
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	1	0	0	1
0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	1
0	1	1	0	1	1	0	0
0	1	1	1	1	1	0	1
1	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	0	1	0	1	0
1	1	0	1	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	1	1	0	1

(b) 機能表 (1)

現	In=0		In=1	
	次	Out	次	Out
S0	S0	1	S1	1
S1	S2	0	S4	1
S2	S4	0	S2	1
S3	S6	0	S6	1
S4	S1	0	S0	1
S5	S3	0	S4	1
S6	S5	0	S2	1
S7	S7	0	S6	1

(c) 状態遷移表 (1)



(d) 状態遷移図 (1)

(e-1)

Q1	Q2	Q3	In	P1	P2	P3	Out
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1

(e-2)

Q1	Q2	Q3	In	P1	P2	P3	Out
0	0	1	0	0	1	0	0
0	0	1	1	1	0	0	1

(e-3)

Q1	Q2	Q3	In	P1	P2	P3	Out
0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	1

(e-4)

Q1	Q2	Q3	In	P1	P2	P3	Out
1	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1

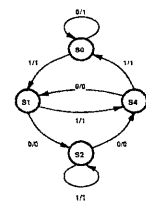
(e-5)

Q1	Q2	Q3	In	P1	P2	P3	Out
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	1	0	0	1
0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	1
0	1	1	0	1	1	0	0
1	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	1

(e) 機能表 (2)

現	In=0		In=1	
	次	Out	次	Out
S0	S0	1	S1	1
S1	S2	0	S4	1
S2	S4	0	S2	1
S4	S1	0	S0	1

(f) 状態遷移表 (2)



(g) 状態遷移図 (2)

図 4: 状態遷移表作成の例

場合、時刻 4 において以下のように記述できる。

$$\begin{aligned}
 q_{F_0}[4] &= q_C[0] \oplus q_{A_0}[0] \oplus q_{B_0}[0] \\
 q_{F_1}[4] &= (q_C[0] \cdot (q_{A_0}[0] + q_{B_0}[0]) + q_{A_0}[0] \cdot q_{B_0}[0]) \oplus \\
 &\quad q_{A_1}[0] \oplus q_{B_1}[0] \\
 q_{F_2}[4] &= ((q_C[0] \cdot (q_{A_0}[0] + q_{B_0}[0]) + q_{A_0}[0] \cdot q_{B_0}[0]) \cdot \\
 &\quad (q_{A_1}[0] + q_{B_1}[0]) + q_{A_1}[0] \cdot q_{B_1}[0]) \oplus q_{A_2}[0] \oplus q_{B_2}[0] \\
 q_{F_3}[4] &= (((q_C[0] \cdot (q_{A_0}[0] + q_{B_0}[0]) + q_{A_0}[0] \cdot q_{B_0}[0]) \cdot \\
 &\quad (q_{A_1}[0] + q_{B_1}[0]) + q_{A_1}[0] \cdot q_{B_1}[0])) \cdot (q_{A_2}[0] + \\
 &\quad q_{B_2}[0]) + q_{A_2}[0] \cdot q_{B_2}[0] \oplus q_{A_3}[0] \oplus q_{B_3}[0] \\
 q_C[4] &= (((((q_C[0] \cdot (q_{A_0}[0] + q_{B_0}[0]) + q_{A_0}[0] \cdot q_{B_0}[0]) \cdot \\
 &\quad (q_{A_1}[0] + q_{B_1}[0]) + q_{A_1}[0] \cdot q_{B_1}[0])) \cdot (q_{A_2}[0] + \\
 &\quad q_{B_2}[0]) + q_{A_2}[0] \cdot q_{B_2}[0]) \cdot (q_{A_3}[0] + q_{B_3}[0]) + \\
 &\quad q_{A_3}[0] \cdot q_{B_3}[0] \\
 q_{A_0}[4] &= IA[0], & q_{B_0}[4] &= IB[0] \\
 q_{A_1}[4] &= IA[1], & q_{B_1}[4] &= IB[1] \\
 q_{A_2}[4] &= IA[2], & q_{B_2}[4] &= IB[2] \\
 q_{A_3}[4] &= IA[3], & q_{B_3}[4] &= IB[3]
 \end{aligned}$$

次にこれらの論理式から、算術演算機能を抽出する。これは、我々がすでに提案している組合せ回路の算術演算機能抽出の手法を用いて行なうことができる。この手法を用いれば、レジスタ F および C の値を表す上記の論理式から、以下に示す繰り返しパターンを抽出できる。

$$\begin{cases}
 q_{F_i}[4] = C_i \oplus q_{A_i}[0] \oplus q_{B_i}[0] \\
 C_{i+1} = C_i \cdot (q_{A_i}[0] + q_{B_i}[0]) + q_{A_i}[0] \cdot q_{B_i}[0]
 \end{cases} \quad (i = 0, 1, 2, 3)$$

ここに、 $C_i$  は中間変数であり、 $C_0 = q_C[0]$ 、 $C_4 = q_C[4]$  である。この繰り返しパターンは、データベースにあらかじめ登録されてある加算機能の繰り返しパターン、

$$\begin{cases}
 F_i = C_i \oplus A_i \oplus B_i \\
 C_{i+1} = C_i \cdot (A_i + B_i) + A_i \cdot B_i
 \end{cases} \quad (i = 0, 1, \dots, n-1)$$

に一致するので、この回路が加算機能を持っていることがわかる。結局、図 6 に与えられた回路の機能は、「時刻 0 において 4 ビットレジスタ A, B に蓄えられていた 2 進数データに対して、加算を施し、時刻 4 においてその結果をレジスタ F に、桁上げ出力をレジスタ C に保存する」と記述することができる。

### 3.3 手法のまとめ

これまでに幾つかの例を示したが、ここで一般的な手法についてまとめておく。まず、入力として順序回路の回路

記述を与える。これからクロック信号とフリップフロップを取り除いてフィードバックループを切り、組合せ回路に変換する。この際、状態を表すレジスタの出力であった信号線は、コントロール系の入力変数として、またデータレジスタの出力であった信号線は、データ系入力変数として新たに加えられる。さらに、各レジスタの入力変数であった信号線は、出力変数となる。ここで、コントロール系状態変数に初期状態の値を代入し、2.2 で示した手法にしたがって、初期状態から遷移し得る状態において機能表を作成する。

データレジスタを持つ回路においては、ここで得られた機能表の出力状態変数に論理式が現れるので、これらから何らかの機能を抽出する。まず得られた機能表から漸化式を作成し、次にこれらを順次適用していくことによって、各出力を初期状態におけるデータ変数の論理式で表す。算術演算回路の場合、これらの論理式が繰り返し構造を持っているので、あらかじめデータベースに登録されてある各算術演算の繰り返しパターンと比較することによって、算術演算機能を抽出することができる。ただし現状では、上述の加算機能の繰り返しパターンのように中間変数がせいぜい 1 つだけ存在するものしか抽出することはできない。例えば乗算機能などは抽出できない、という限界がある。

## 4 試作システム

我々は既に、組合せ回路の機能情報抽出システム FINES (Functional Information Extraction System) を試作した [4]。今回は、前節までで述べた手法が実現できるように、この試作システムを改良した。なお、プログラムは Sparc-Station 2 (主メモリ 16MB) 上で C 言語を用いて作成した。

システムの入力は、論理ゲートレベルのネットリストであり、幾つかの付加情報がこのネットリストに記述されている。図 7(a) に示したのは、図 5(a) の回路のネットリストである。ネットリストは、幾つかのリスト構造からなり、各リストの最初の要素がキーワードとなる。キーワード“IN”は、その後続く変数名が入力変数であることを意味する。この行は、論理関数の内部表現である二分決定グラフ [5] を作成する際の変数の順序、という付加情報も含んでいる。キーワード“OUT”は、その後ろに続く変数名が出力変数であることを示す。入力変数、出力変数ともに、2 進数を構成する場合は、最下位ビットから順に並べられてあるも

```

(IN CK In J)
(OUT Qa Qb Qc Qd)
(DATA 1)
(STATE 1)
(REGISTER 4)
(xor1 XOR 5 In Qz n1)
(inv INV 3 Qz n2)
(nand1 NAND 4 Qz J n3)
(nand2 NAND 4 n1 Qd n4)
(nand3 NAND 4 n3 n4 n5)
(dff1 DFF 10 CK n1 Qz)
(dff2 DFF 10 CK n5 Qa)
(dff3 DFF 10 CK Qa Qb)
(dff4 DFF 10 CK Qb Qc)
(dff5 DFF 10 CK Qc Qd)

(IN Qz In Qa Qb Qc Qd J)
(OUT Qz+ Qa+ Qb+ Qc+ Qd+)
(DATA 5)
(xor1 XOR 5 In Qz n1)
(inv INV 3 Qz n2)
(nand1 NAND 4 Qz J n3)
(nand2 NAND 4 n1 Qd n4)
(nand3 NAND 4 n3 n4 n5)
(dff1_buf BUF 10 n1 Qz+)
(dff2_buf BUF 10 n5 Qa+)
(dff3_buf BUF 10 Qa Qb+)
(dff4_buf BUF 10 Qb Qc+)
(dff5_buf BUF 10 Qc Qd+)

```

(a) 順序回路のネットリストの例

(b) (a) から得られる組合せ回路のネットリスト

図7: ネットリストの例

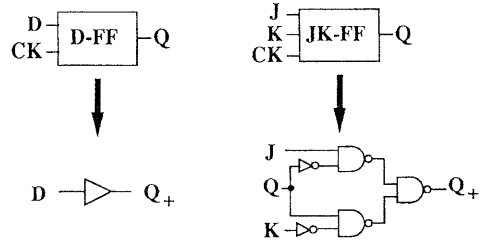
のと仮定される。キーワード“DATA”は、データ系入力の個数を表す。入力変数は、コントロール系入力、データ系入力の順に並べられてあるので、データ系入力の個数が1ということは、入力変数のうち最後の1つだけがデータ系入力であるとみなされる。“STATE”は、状態を表す変数の個数、“REGISTER”は、データレジスタの出力を表す変数の個数を表す。両者の和は、フリップフロップの総数に等しくなる。それら以外のキーワードは、ゲートの名前を表している。この行は、最初の要素がゲートの名前、2番目がゲートの種類、3番目がそのゲートの遅延時間、そして4番目以降がそのゲートの入力信号名および出力信号名を表している。

まず、与えられた順序回路のネットリストを組合せ回路のネットリストに変換する。前節では、D-FFで説明したが、本システムではJK-FF等も扱うことができる。すなわち、図8に示したように、フリップフロップを等価な組合せ回路で置き換え、クロック信号を取り除き、新たに生じた入出力変数を付け加える。このようにして、図7(b)のように、組合せ回路のネットリストが得られる。

我々のシステムは、論理関数の内部表現として共有二分決定グラフ(SBDD) [6]を用いており、省メモリで高速な処理が行なわれている。図6に示した4ビット直列加算器の例においては、組合せ回路のネットリストから算術演算機能情報を抽出するのに、約0.13秒かかった。

## 5 あとがき

本稿では、同期式順序回路からその機能情報を抽出する手法を紹介した。本手法を用いることにより、データレジスタを持つ順序回路から加算機能などの算術演算機能情報を抽出し、機能を簡潔に記述することができる。したがって、



(a) D-FFの場合 (b) JK-FFの場合

図8: フリップフロップの組合せ回路への置き換え

論理設計の検証を人間の目で行なうことができ、非常に有用であると考えられる。現状では、加算や減算のような単純な算術演算しか取り扱うことができないが、乗算のようなより複雑な算術演算を抽出することが、今後の課題の一つである。

## 参考文献

- [1] J.Madre, and J.P. Billon, "Proving Circuit Correctness using Formal Comparison Between Expected and Extracted Behavior", Proc. 25th DAC, pp.205-210, (June 1988).
- [2] A.P.Kosteljik, "VERA, a Rule-Based Verification Assistant for VLSI Circuit Design", Proc. VLSI89, pp.89-98, (Aug. 1989).
- [3] D.T.Blaauw, *et al.*, "Automatic Generation of Behavioral Models from Switch-Level Descriptions", Proc. 26th DAC, pp.179-184, (June 1989).
- [4] 大村昌彦, 安浦寛人, 田丸啓吉, "組合せ回路の機能情報抽出", 電子情報通信学会論文誌 A, Vol. J74-A, No.2, pp.247-255, (Feb. 1991).
- [5] R.E.Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Trans. Comput., Vol. C35, No.8, pp.677-691, (Aug. 1986).
- [6] 湊真一, 石浦葉岐佐, 矢島脩三, "論理関数の共有二分決定グラフによる表現とその効率的処理手法", 情報処理学会論文誌, Vol.32 No.1, pp.77-85, (Jan. 1989).