7E−2

同期式順序回路からの動作記述の抽出

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

1 はじめに

我々は従来より、論理回路からの機能情報の抽出に関する研究を行なってきている[1, 2]。組合せ回路の場合には、与えられた回路記述から、その入出力変数間の関係を抽出し、論理式・算術式などを用いた機能記述で表した。また、同期式順序回路の場合には、クロックパルスを1回加えた時に、レジスタの値がどのように変化するか、あるいはどのレジスタからどのレジスタへデータが移動するか、というように機能を記述した。しかし、マイクロプロセッサのように複数回クロック入力を加えて一つの動作(命令)を実行するような回路の場合、クロック入力1回ごとの機能を抽出することも重要であるが、全体としてどのような命令が実行されるかということを抽出する方が、回路の動作を理解する上でより効果的であると考えられる。本稿では、まず同期式順序回路の機能を抽出して機能記述言語で表す手法について述べる。さらにその記述から、マイクロプロセッサの命令コード表のように、その回路の全体的な動作を表す記述を抽出する手法について検討する。

2 準備

本稿で取り扱う回路は、組合せ回路部分と、単一のクロックパルスが同時に与えられる幾つかのD-FF(レジスタ)から構成される、同期式順序回路である。各レジスタは、0又は1の値を内部状態として保存しており、その値は出力線を通じて外部に現れる。これら現状態変数の値)と外部入力の値から、組合せ回路部において、外部出力と次状態変数の値が一意に決定される。次状態変数の値は各レジスタの入力信号線を通じ、クロックパルスが加えられた時に、新たな状態として各レジスタに保存される。従って、組合せ回路部分のみに注目して、従来より我々が提案している組合せ回路の機能情報抽出の手法[1]を適用すれば、現状態変数と次状態変数の関係を知ることが出来る。

レジスタには、制御状態を表すものと、データを蓄えておくものの2通りが考えられる。制御レジスタは、そのレジスタに蓄えられている値によって回路の機能が異なる、というものである。(例えばCPUにおける)命令レジスタ(IR)などがそうである。一方データレジスタは、そこに蓄えられている値が0か1かで機能が異なるというものではなく、その値がデータとしての意味をもっているものである。アキュムレータ(ACC)などはその例である。また本稿では、RAMやROMなどのメモリも、データレジスタの一種とみなす。一般に、与えられた回路記述から制御レジスタとデータレジスタを区別するのは困難である。そこで、機能情報抽出を行なうにあたって、それらの区別は付加情報として外部からあらかじめ与えられるものとする。

3 同期式順序回路からの動作記述の抽出

3.1 概要

前節で触れた、現状態変数と次状態変数の関係というのは、ある状態においてどのような機能が実行され、次にどの状態に遷移するか、ということを記述したものに他ならない。ここでは、記述言語としてUDL/Iを用いて、その機能を記述する。UDL/Iは、オートマトン記述によって状態の遷移を記述する。また機能記述には、論理式だけでなく算術式なども含めることが出来る。3.2では、簡単なマイクロプロセッサを例にとり、UDL/Iにより機能記述を作成する手法について述べる。

3.2 実例を用いた説明

図1(a)にマイクロプロセッサのブロック図を示す。PC, MAR, MBR, Aはそれぞれ8ビットのレジスタであり、メモリは8ビット、256語である。実行できる命令は4種類で、レジスタIRは2ビットである。これは、MBRの下位2ビットからロードされる。また、タイミングレジスタTは3ビットからなる。レジスタは全て、ビット幅分のD-FFから構成され、それぞれPC<7:0>, MAR<7:0>, MBR<7:0>, A<7:0>, IR<1:0>, T<2:0>と表す。また、メモリは8×256個のD-FFからなり、M<255:0, 7:0>と表す。但し本手法では、取り扱いの煩雑さを避けるため、メモリはブラックボックスとみなす。即ち、MARの値がnである場合には、M<n, 7:0>を出力するものとし、これを省略してM<7:0>と記述する。

さて、このブロック図の論理回路記述から、その機能を抽出する。PC, MAR, MBR, Aはデータレジスタであり、IRとTは制御レジスタである。従って、$2^{2+3}=32$個の制御状態が存在し得る。ここで初期状態がIR=00, T=000と与えられているものとし、ここから遷移し得る状態のみを追ってゆく。機能表においては、データレジスタの現在の値PC, MAR, MBR, Aと次の値PC', MAR', MBR', A'との関係に着目した。例えば、

```
MAR'<7>=PC<7> MAR'<6>=PC<6> ...  MAR'<0>=PC<0>
```

となっている場合には、PCの内容がMARに移されているので、MAR'=PCと表す。また、

```
PC'<7>=PC<7>@(PC<6>&PC<5>&PC<4>& ...  &PC<0>)
PC'<6>=PC<6>@(PC<5>&PC<4>& ...  &PC<0>)
...
PC'<1>=PC<1>@PC<0>
PC'<0>=^PC<0>
```

となっている場合には、既に考案した算術演算機能抽出の手法[3]を用いることにより、PCがインクリメントされていることがわかる。従って、PC'=PC+1と表せる。(但し、@は排他的論理和、&は論理積、^は否定、+は算術加算を表す。またここには現れていないが、論理和は!で表す。)

次状態はIR',T'の値から決まる。これらが0, 1で表せない時は、全ての場合に展開して考える。例えば、

```
IR'<1>=MBR<1> IR'<0>=MBR<0>
T'<2>=0 T'<1>=1 T'<0>=1
```

となった場合、次状態は00011, 01011, 10011, 11011の4通りであるとする。

こうして、初期状態から遷移し得る状態について、その状態における各データレジスタの値を表したところ、図1(b)のようになった。ここで制御レジスタIR, Tの表す値をそれぞれi, jとし、その状態名を$Q_{i,j}$で表すと、同図(c)のような状態遷移図が得られる。またUDL/I記述は同図(d)のようになる。

4 考察

前節の例で得られたUDL/I記述から動作記述を抽出する手法について検討する。ここで、PCはプログラムカウンタ、MARはメモリアドレスレジスタ、MBRはメモリバッファレジスタ、Aは汎用レジスタであるという情報は、あらかじめ与えられているものとする。状態がQ00 -> Q01 -> Q02と遷移する時に、MAR:=PC; PC:=INC(PC); MBR:=M<MAR>; という動作が実行されているが、これはPCの示すメモリのアドレスからMBRに1語読み出していることを表している。次にMBRの下位2ビットの値によって、4

Extraction of Behavioral Descriptions from Synchronous Sequential Circuits

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

†Kyoto University, ‡Kyushu University

つの状態に分岐している。Q03 では、何もせずに初期状態に戻っているので、この命令は NOP (No operation) である。Q13 では、メモリから1語読み出して、レジスタ A に転送しているので、この命令は即値ロードである。Q23 では、メモリから1語読み出し、その値を MAR に移して、その指示するメモリのアドレスからもう1語読み出し、レジスタ A に転送しているので、この命令は直接アドレス指定のロードである。Q33 では、メモリから1語読み出し、レジスタ A の値と加算し、レジスタ A に保存しているので、レジスタ A とオペランドの加算を表している。こうして、図1(e) のように命令コードを表すことが出来る。

ここでは、人手の援助により命令コード表を作成したが、メモリからの1語読み出し、汎用レジスタへのデータのロード、などという基本操作を知識として蓄えておけば、幾らかの付加情報を与えるだけで、自動的に命令コード表を抽出することができると考えられる。

5 おわりに

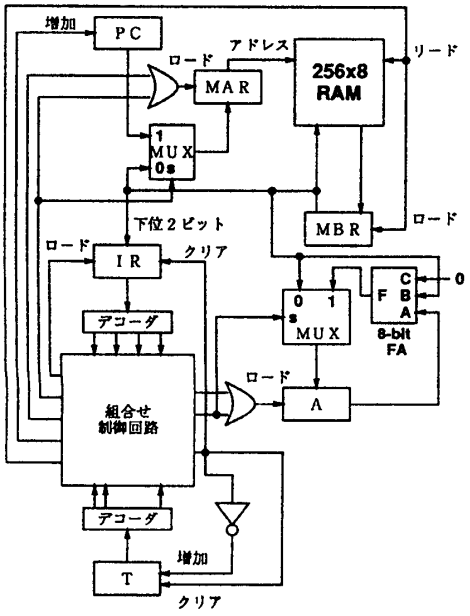
本稿では、同期式順序回路からその機能情報を抽出し、UDL/I を用いて記述する手法を紹介した。また、マイクロプロセッサの命令コードなどの、さらに高位レベルの動作記述を抽出する手法について検討した。本手法はまだ構想の段階であり、今後、既存の機能情報抽出システムを利用して動作記述の自動作成システムを実現していく予定である。

参考文献

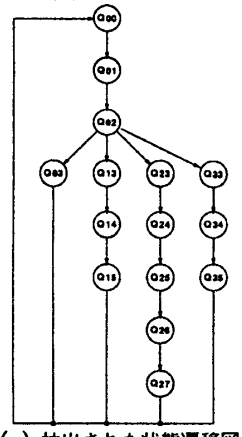
[1] 大村, 安浦, 田丸, "組合せ回路の機能情報抽出", 信学論 A (Feb. 1991) .

[2] 大村, 安浦, 田丸, "順序回路からの算術演算機能の抽出", 情処研報 DA 60-12(Dec. 1991).

[3] M.Ohmura, H.Yasuura, K.Tamaru, "Extraction of Arithmetic Functions from Combinational Circuits", 信学技報 VLD 90-104(Feb. 1991).



(a) ブロック図



(c) 抽出された状態遷移図

コード	ニモニック	オペランド	動作
00	NOP	なし	何もしない
01	LDI	OPRD	A<-OPRD
10	LDA	ADDR	A<-M<ADDR>
11	ADD	OPRD	A<-A+OPRD

(e) 命令コード表

IR <1><0>	IR <2><1><0>	T <1>	T <0>	IR' <2><1><0>	PC'	MAR'	MBR'	A'
0	0	0	0	0	PC	PC	MBR	A
0	0	0	0	1	PC+1	MAR	M	A
0	0	0	1	0	MBR<1>	MBR<0>	0	1
0	0	0	1	1	PC	MAR	MBR	A
0	1	0	1	1	PC	PC	MBR	A
0	1	1	0	0	PC+1	MAR	M	A
0	1	1	0	1	PC	MAR	MBR	MBR
1	0	0	1	1	PC	PC	MBR	A
1	0	1	0	0	PC+1	MAR	M	A
1	0	1	0	1	PC	MBR	MBR	A
1	0	1	1	0	PC	MAR	M	A
1	0	1	1	1	PC	MAR	MBR	MBR
1	1	0	1	1	PC	PC	MBR	A
1	1	1	0	0	PC+1	MAR	M	A
1	1	1	0	1	PC	MAR	MBR	A+MBR

(b) 抽出された機能表

```
IDENT:      EXAMPLE;
DATE:       01/01/92;
COMMENT:    This description was automatically generated.

NAME:       EXAMPLE;
CLOCK:      CLK;
RESET:      RST;

BEHAVIOR_SECTION
TERMINAL:   tmp;
REGISTER:   PC<7:0>,MAR<7:0>,MBR<7:0>,A<7:0>,IR<1:0>,T<2:0>;
RAM:        M<255:0,7:0>

BEGIN
tmp:=1B0;
AT RST DO
    PC:=8H00; MAR:=8H00; A:=8H00; IR:=2B00; T:=3B000;
END_DO;
END;
AUTOMATON:  TEST: RST: CLK;
    Q00:    WAIT(tmp): MAR:=PC; -> Q01;
    Q01:    WAIT(tmp): PC:=INC(PC); MBR:=M; -> Q02;
    Q02:    WAIT(tmp):
                CASE MBR<1:0> OF
                    #2B00 -> Q03; #2B01 -> Q13;
                    #2B10 -> Q23; #2B11 -> Q33;
                END_CASE;
    Q03:    WAIT(tmp): -> Q00;
    Q13:    WAIT(tmp): MAR:=PC; -> Q14;
    Q14:    WAIT(tmp): PC:=INC(PC); MBR:=M; -> Q15;
    Q15:    WAIT(tmp): A:=MBR; -> Q00;
    Q23:    WAIT(tmp): MAR:=PC; -> Q24;
    Q24:    WAIT(tmp): PC:=INC(PC); MBR:=M; -> Q25;
    Q25:    WAIT(tmp): MAR:=MBR; -> Q26;
    Q26:    WAIT(tmp): MBR:=M; -> Q27;
    Q27:    WAIT(tmp): A:=MBR; -> Q00;
    Q33:    WAIT(tmp): MAR:=PC; -> Q34;
    Q34:    WAIT(tmp): PC:=INC(PC); MBR:=M; -> Q35;
    Q35:    WAIT(tmp): A:=ADD(A,MBR,1B0); -> Q00;
END_AUTO;
END_SECTION;
END;
CEND;
```

(d) 抽出された UDL/I 記述

図 1: マイクロプロセッサから機能記述を抽出する例