学士論文

４ALUプロセッサによる並列処理の検討とシミュレータの設計

氏名：岩見　佑一郎
学籍番号：2260100012－5
指導教員：山崎　勝弘　教授
提出日：2014年2月20日

立命館大学　理工学部　電子情報デザイン学科

内容梗概

本論文では、Booth 乗算のアルゴリズムを、予め本研究室で設計された 1 次、2 次 Booth プログラムを参考に 3 次 Booth 乗算のアルゴリズムのプログラム設計と試作を行った。また、本研究室における研究テーマの一環である、ハード/ソフト協調システムを利用し設計されたマルチ ALU プロセッサ（MAP）システムを学習し、新たにプロセッサを 2 つ拡張した 4ALU の動作検証および並列性の評価を行った。ここでの検証用プログラムとして、1～3 次 Booth 乗算のアルゴリズムを用いて乗数と被乗数がそれぞれ正、負の同じ値で並列・連鎖演算、単一実行がどのように変化するか検証を行った。

本論文では、Booth 乗算のアルゴリズムの概要を述べ、マルチ ALU プロセッサ（MAP）についての説明、および 4ALU での並列・連鎖演算の判定を述べ、MAP の動作を確認し並列性の検証を行う。

目次

図目次

表目次

１．　はじめに

近年、組み込み機器の急速な発展によりスマートフォンやタブレットが急速に普及している。さらに最近では、メガネのように体に身に着けて持ち歩くことができるコンピュータの登場により、小さくて軽い組み込み機器が IT 市場に出始めている。これらはいわゆる「ウェアラブル」と呼ばれ、将来、スマートフォンに代わるコンピュータとして期待されている。現在のスマートフォンは、実際に指で操作するインターフェースから成り立っているがメガネのようなウェアラブルの場合、目の前にディスプレイが現れ、インターネットやアプリなど全ての操作を音声で行うことができ、操作を極めて容易にした次世代型のコンピュータとして注目を集めている。

組み込み機器とは、ソフトウェアを用いて制御される機器の総称であり、ハードウェアとそれを制御するソフトウェアから構成され、ハードウェアとソフトウェアが一体になって相互関連しながら動作するものである。現代社会において、組み込み機器は必要不可欠なものであり通信機器（携帯電話、ファックスなど)、運輸機器（自動車、電車、航空機、船など）家電機器（洗濯機、電子レンジ、エアコンなど)、AV 機器（テレビ、ビデオ、オーディオ機器など)、医療機器、他にも娯楽機器（ゲーム機、パチンコなど）などほとんどの電子機器に組み込み機器が使われている。また、機器に組み込まれその制御を行うコンピュータシステムのことを「組み込みシステム」というが、プロセッサの低コスト化や高機能化、高性能化により組み込みシステムに求められる処理能力は一段と大きくなり、そのソフトウェアも大規模化・複雑化している。これら組み込みシステムの規模が大きくなればソフトウェア規模も増大し、ハードウェアの関係も深くなる。このような背景から、大学教育にはハードウェアとソフトウェアを深く理解し、その上でシステム設計を提案できる人材を育成することが求められている。

本研究室では、プロセッサ設計を中心としたハードウェアとソフトウェアの両方の知識を習得するためのハード/ソフト協調学習システム（HSCS）が進められており、複数の ALU による並列処理が可能なマルチ ALU プロセッサ（MAP）が開発されている。[2][4]MAP システムは、HSCS を用いて設計された 32 ビットのプロセッサであり、複数の命令間の依存関係を判別し、並列演算または連鎖演算を行う。これまで、2 つの命令を１命令サイクルで同時実行する２ALU による MAP が設計されていたが、これを拡張させ４つの命令を１命令サイクルで同時実行する４ALU の設計に取り組み、さまざまな処理を行うプログラムを実行した場合でも正しい動作が行われているのかを検証している。[2]

さらに、本研究ではこれまで作成されたアセンブリプログラムのひとつである Booth（ブース）のアルゴリズムについて研究し、新たに乗算の高速化を図る３次 Booth アルゴリズムのプログラムを作成した。これから実際に Booth のアルゴリズムについて述べ、新たに作成した３次 Booth の乗算アルゴリズムについて述べる。また、以前本研究室で設計された Booth 乗算のアルゴリズムを今回検証用プログラムとして動的な並列性の検証を行う。

２．　Booth 乗算の並列化

2.1　Booth 乗算のアルゴリズム

（1）１次 Booth

Booth の乗算アルゴリズムは、m を被乗数、r を乗数と定め、m の正、負の 2 進数を P に繰り返し加算し、P を右方向に算術シフトするという形で実装できる。ここでは実際に m と r を 4 ビット、m の正の拡張ビットを M、m の負の拡張ビットを-M とし、それぞれ右端に 0 を加え 9 ビットとする。また、P も 9 ビットとする。以下にアルゴリズムの流れを説明する。

① M に被乗数 m の値を最上位（左側）に格納し、それ以外は 0 とする。
　-M には m の 2 の補数表現（-m）を最上位側のビットに格納し、それ以外は 0 とする。
　P には最上位側の 4 ビットを 0 とし、その右 4 ビットに r の値を格納する。右端のビットは 0 とする。
② P の最下位 2 ビットを調べ、01 の場合、P＋M を計算し、オーバーフローは無視する。
　10 の場合、P＋（-M）を計算し、オーバーフローは無視する。
　00 または 11 の場合、現在の P の値をそのままにする。
③ P を右に 1 ビット算術シフトする。これで P の値を更新する。
④ 上記①～③を 4 回繰り返す。
⑤ 右端のビットを取り除くと、m と r の積が算出される。

以下に例を示す。

例）m＝2、r＝（-3）の場合

・被乗数 m＝0010　　・M=0010 0000 0
・乗数 r＝1101　　・-M=1110 0000 0

① P＝0000 1101 0
最下位ビットが 10 より P+S＝1110 1101 0 その後右 1 ビット算術シフト
② P＝1111 0110 1
最下位ビットが 01 より P+A＝0001 0110 1 その後右 1 ビット算術シフト
③ P＝0000 1011 0
最下位ビットが 10 より P+S＝1110 1011 0　その後右 1 ビット算術シフト
④ P＝1111 0101 1
最下位ビットが 11 より P の値はそのまま、その後右 1 ビット算術シフト

P＝1111 1010 1 右端の 1 を除去
1111 1010　 2 の補数表現からこの値は-6

(2) 2 次 Booth

本研究室で設計された 4 ビットの 2 次 Booth の乗算アルゴリズムをフローチャートで作成し、図 1 に示す。

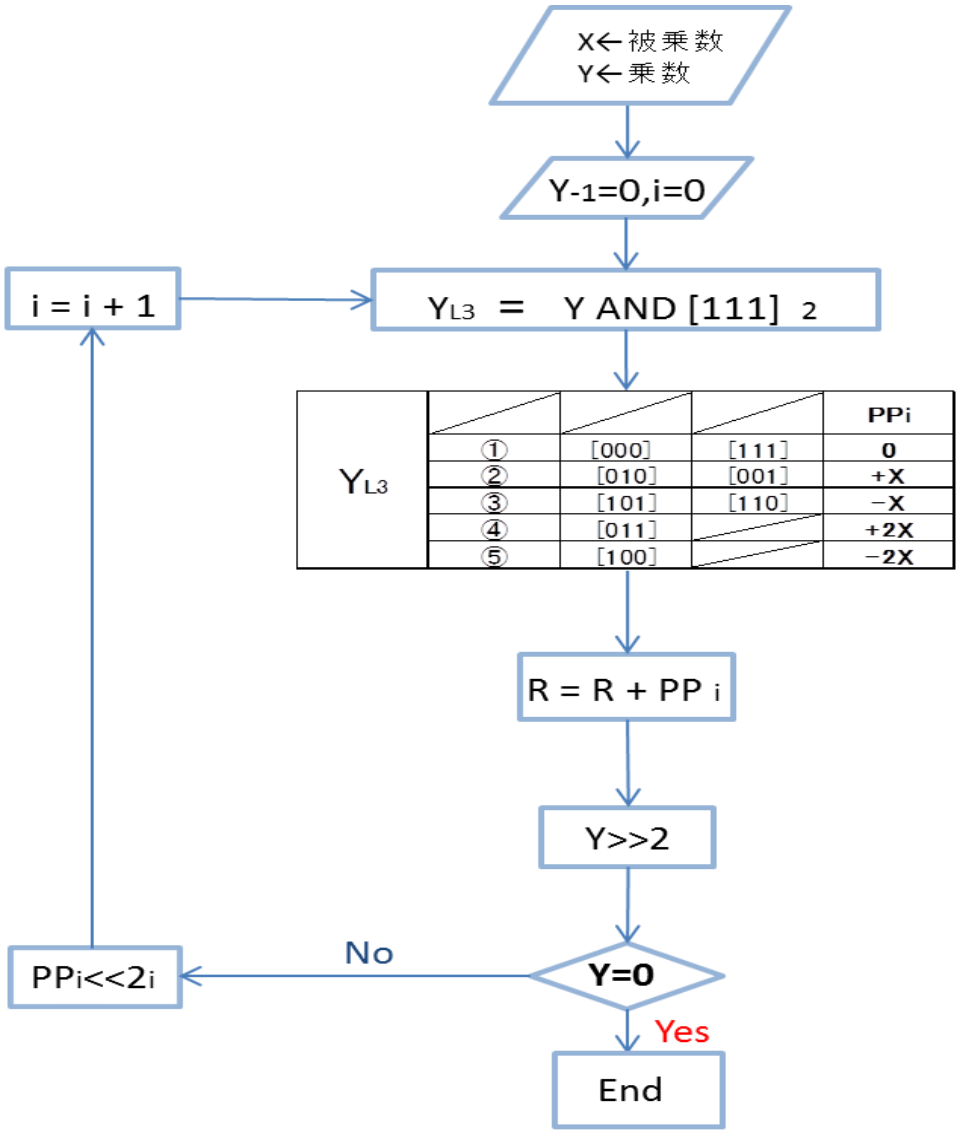

図 1. 2 次 Booth のアルゴリズム

R:乗算結果。R の初期状態は 0。

Y-1、:乗数 Y の下位 1 ビットを示す。

YL3 : Y の下位 3 ビット分を示す。

PPi:テーブルから検索された部分積。

[○○○]2 は 2 進数表記。

※本来の 2 次 Booth のアルゴリズムでは Y が偶数桁の場合、最下位ビットに 0 を一つ加え、Y が奇数桁の場合、最下位ビットに 0 を二つ加える。本研究室で設計された 2 次 Booth の乗算アルゴリズムのプログラムでは 4 ビットの乗算なので、ここでは Y を偶数桁で処理する。また、PPi の最上位ビットで符号ビット拡張を行い、8 ビットまで繰り上げる。たとえば、0011 の場合、0000 0011 とし、1101 の場合、1111 1101 とする。

2 次 Booth 乗算のアルゴリズムの流れを、以下に例を用いて説明する。

例）X=（-6)、Y=3 の場合

① 被乗数 X=1010、乗数 Y=0011 より Y の最下位ビットとして 0 を加え Y＝0011 0 とする。

② Y の下位 3 ビットを調べ、図 1 中のデコード表により-2X～2X の範囲で部分積 PPi を求める。この場合 Y の下位 3 ビットは 110 なので-X を PP_0 として R に格納する。

R=0000 0110　Y=0011 0

③ Y を右に 2 ビット算術シフトする。その後、Y=0000 であれば終了、そうでなければ部分積 PPi を左に 2i ビット算術シフトする。この場合 i=0 よりそのまま R を格納する。

R=0000 0110　Y=0000 1

④ Y の下位 3 ビットは 001 なので+X を PP_1 として R に加え、Y を右に 2 ビット算術シフトする。

R=1110 1110　Y=0000 0

⑤ Y=0（0000）となったので終了し、R の値は 2 の補数表現で-18 となる。

(3) 3 次 Booth

先の 2.2 で示した 2 次 Booth の乗算アルゴリズムを参考に、表 2. の Decode table を拡張し、新たに 3 次 Booth の乗算アルゴリズムのアセンブリプログラムを設計し Cygwin 上で実行した。以下に、表 1 の 3 次 Booth デコード表を示す。(被乗数 X、乗数 Y のビット数はそれぞれ 6 ビットずつとする。)

表 1. 3 次 Booth デコード表

Y_{i+1}	Y_i	Y_{i-1}	Y_{i-2}	PPi
0	0	0	0	0
0	0	0	1	X
0	0	1	0	X
0	0	1	1	2X
0	1	0	0	2X
0	1	0	1	3X
0	1	1	0	3X
0	1	1	1	4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-X
1	1	1	0	-X
1	1	1	1	0

3 次 Booth の乗算アルゴリズムの利点としては、部分積の数が被乗数 X の桁数の 1/3 個となり、部分積の加算が高速で行えるようになる。
欠点として、実際にデコーダなどで設計する際、複雑になるので回路の規模が大きくなることや、遅延が起こるなどの問題がある。

2.2 Booth の並列性評価

4ALU の MAP が、2ALU の MAP とどのように演算の回数が変わり性能向上が見込めるかを調べるために、実際に 2 次 Booth の乗算アルゴリズムのアセンブリプログラムを用いてハンドシミュレーションを行った。2 次 Booth アセンブリプログラムを図 2 に、X=2、Y=-3 の数値で 2、4ALU の単一・連鎖・並列回数をそれぞれ図 3 と図 4 で示す。

アドレス					
1		LD	$1	0[$0]	
2		LD	$2	4[$0]	
3		ANDI	$2	$2	15
4		SLL	$1	$1	5
5		SLL	$2	$2	1
6	LOOP:	ADDI	$15	$15	1
7		SEQI	$16	$15	3
8		BNEZ	$16	LAST	
9		ANDI	$3	$2	7
10		SEQI	$5	$3	1
11		BNEZ	$5	SKIP1	
12		SEQI	$4	$3	2
13		BNEZ	$4	SKIP1	
14		SEQI	$5	$3	3
15		BNEZ	$5	SKIP3	
16		SEQI	$5	$3	4
17		BNEZ	$5	SKIP4	
18		SEQI	$5	$3	5
19		BNEZ	$5	SKIP2	
20		SEQI	$5	$3	6
21		BNEZ	$5	SKIP2	
22		SRL	$2	$2	2
23		JUMP	LOOP		
24	SKIP1:	ADD	$2	$2	$1
25		SRL	$2	$2	2
26		JUMP	LOOP		
27	SKIP2:	SUB	$2	$2	$1
28		SRL	$2	$2	2
29		JUMP	LOOP		
30	SKIP3:	ADD	$2	$2	$1
31		ADD	$2	$2	$1
32		SRL	$2	$2	2
33		JUMP	LOOP		
34	SKIP4:	SUB	$2	$2	$1
35		SUB	$2	$2	$1
36		SRL	$2	$2	2
37		JUMP	LOOP		
38	LAST:	SRL	$2	$2	1
39		ST	$2	12[$0]	
40		HALT			

図 2. 2 次 Booth アセンブリプログラム

step	番号	命令						2並列	2連鎖	単一
1	1		LD	$1	0[$0]			1 2		
	2		LD	$2	4[$0]					
2	3		ANDI	$2	$2	15		3 4		
	4		SLL	$1	$1	5				
3	5		SLL	$2	$2	1		5 6		
	6	LOOP:	ADDI	$15	$15	1				
4	7		SEQI	$16	$15	3			7 8	
	8		BNEZ	$16	LAST					
5	9		ANDI	$3	$2	7			9 10	
	10		SEQI	$5	$3	1				
6	11		BNEZ	$5	SKIP1					11
7	12		SEQI	$4	$3	2			12 13	
	13		BNEZ	$4	SKIP1					
8	24	SKIP1:	ADD	$2	$2	$1			24 25	
	25		SRL	$2	$2	2				
9	26		JUMP	LOOP						26
10	6	LOOP:	ADDI	$15	$15	1			6 7	
	7		SEQI	$16	$15	3				
11	8		BNEZ	$16	LAST					8
12	9		ANDI	$3	$2	7			9 10	
	10		SEQI	$5	$3	1				
13	11		BNEZ	$5	SKIP1					11
14	12		SEQI	$4	$3	2			12 13	
	13		BNEZ	$4	SKIP1					
15	14		SEQI	$5	$3	3			14 15	
	15		BNEZ	$5	SKIP3					
16	16		SEQI	$5	$3	4			16 17	
	17		BNEZ	$5	SKIP4					
17	18		SEQI	$5	$3	5			18 19	
	19		BNEZ	$5	SKIP2					
18	20		SEQI	$5	$3	6			20 21	
	21		BNEZ	$5	SKIP2					
19	27	SKIP2:	SUB	$2	$2	$1			27 28	
	28		SRL	$2	$2	2				
20	29		JUMP	LOOP						29
21	6	LOOP:	ADDI	$15	$15	1			6 7	
	7		SEQI	$16	$15	3				
22	8		BNEZ	$16	LAST					8
23	38	LAST:	SRL	$2	$2	1			38 39	
	39		ST	$2	12[$0]					
24	40		HALT							40
							合計	3	14	7

図 3. アセンブリプログラムを用いた 2ALU の演算の流れ(X=2, Y=-3)

step	番号	命令						2並列	2連鎖	3連鎖	4連鎖	単一
1	1		LD	$1	0[$0]				(1,4)(2,3)			
	2		LD	$2	4[$0]							
	3		ANDI	$2	$2	15						
	4		SLL	$1	$1	5						
2	5		SLL	$2	$2	1		5 6				
	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	3						
	8		BNEZ	$16	LAST							
3	9		ANDI	$3	$2	7				9 10 11		
	10		SEQI	$5	$3	1						
	11		BNEZ	$5	SKIP1							
4	12		SEQI	$4	$3	2			12 13			
	13		BNEZ	$4	SKIP1							
5	24	SKIP1:	ADD	$2	$2	$1			24 25			
	25		SRL	$2	$2	2						
	26		JUMP	LOOP				25 26				
6	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	3						
	8		BNEZ	$16	LAST							
7	9		ANDI	$3	$2	7				9 10 11		
	10		SEQI	$5	$3	1						
	11		BNEZ	$5	SKIP1							
8	12		SEQI	$4	$3	2			12 13			
	13		BNEZ	$4	SKIP1							
9	14		SEQI	$5	$3	3			14 15			
	15		BNEZ	$5	SKIP3							
10	16		SEQI	$5	$3	4			16 17			
	17		BNEZ	$5	SKIP4							
11	18		SEQI	$5	$3	5			18 19			
	19		BNEZ	$5	SKIP2							
12	20		SEQI	$5	$3	6			20 21			
	21		BNEZ	$5	SKIP2							
13	27	SKIP2:	SUB	$2	$2	$1			27 28			
	28		SRL	$2	$2	2						
	29		JUMP	LOOP				28 29				
14	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	3						
	8		BNEZ	$16	LAST							
15	38	LAST:	SRL	$2	$2	1			38 39			
	39		ST	$2	12[$0]							
16	40		HALT									40
							合計	3	11	5	0	1

図 4. アセンブリプログラムを用いた 4ALU の演算の流れ(X=2, Y=-3)

2, 4ALU の演算結果の比較を表 2 に示す。

表 2. 2,4ALU の演算結果の比較

	命令個数					並列性(%)				
ALU数	2並列	2連鎖	3連鎖	4連鎖	単一	2並列	2連鎖	3連鎖	4連鎖	単一
2ALU	3	14			7	13	58			29
4ALU	0	11	5	0	4	0	55	25	0	20

考察

2 次 Booth の乗算アルゴリズムのアセンブリプログラムを用いて、2ALU の演算を図 3 のようにハンドシミュレーションしてみたところ、単一演算の割合は約 29%、2 連鎖演算の割合は約 58%、並列演算は約 13%を占めており、単一演算の割合が約 3 割を占めていることが分かった。一方、このプログラムを図 4 のように同じく 4ALU でハンドシミュレーションしてみたところ、単一演算の割合は約 20%、2 連鎖演算の割合は約 55%、3 連鎖演算の割合は約 25%という結果が得られた。4ALU 演算の場合、このように見ると 2ALU 演算に比べて並列演算がなくなっているが、新たに 3 つの ALU が連鎖演算を行っていることが分かった。booth 乗算のアセンブリプログラムは、上位命令と依存する分岐命令が多いため、4ALU の動的演算では連鎖演算が増え、並列演算の回数がなくなったと考えられる。また、全体として単一演算の割合が約 9%減少していることが分かった。プログラム内における並列性の可能性としては、2ALU 演算が全体で約 71%であったのに対し、4ALU 演算が全体で約 80%であった。この結果から、全体で演算の並列性が約 9%増加していることが分かり、単一実行演算の減少した割合分並列性の割合が増加していたことから、複数の ALU を用いた並列化の有用性が見込まれるといえる。

３．　4ALU による並列化の検討

3.1　MAP の構成　[2][4][5][6][7]

MAP は複数の ALU を有する 32 ビットのプロセッサである。図 5 に MAP の構成(データパス)を示す。MAP の主な特徴は、以下の通りである。

① 複数の（2、4、8）ALU による並列処理が可能である。

② ALU を用いて並列演算、連鎖演算を行う。

③ １ALU を 32 ビットで制御し、命令長は 32 ビットである。

また、構成している各部の役割としては、以下の通りである。

・PC（プログラムカウンタ）：次に実行するメモリのアドレスを保持するレジスタ。

・IM（命令メモリ）：実行する命令を保持するメモリ。

・RF（レジスタファイル）：演算結果を保持し、オペランドに用いるレジスタ。

・ALU（Arithmetic Logic Unit）：算術論理演算を行う回路。

・DM（データメモリ）：プロセッサで演算するための入力データを保持するメモリ。

・PPU（Parallel Processing Unit）：並列実行を判断する並列・連鎖・単一処理判定部。

・CU(コントロールユニット)：命令ごとによって違うデータパスを管理し、各種レジスタ、メモリ及びすべての動作をコントロールする制御部。

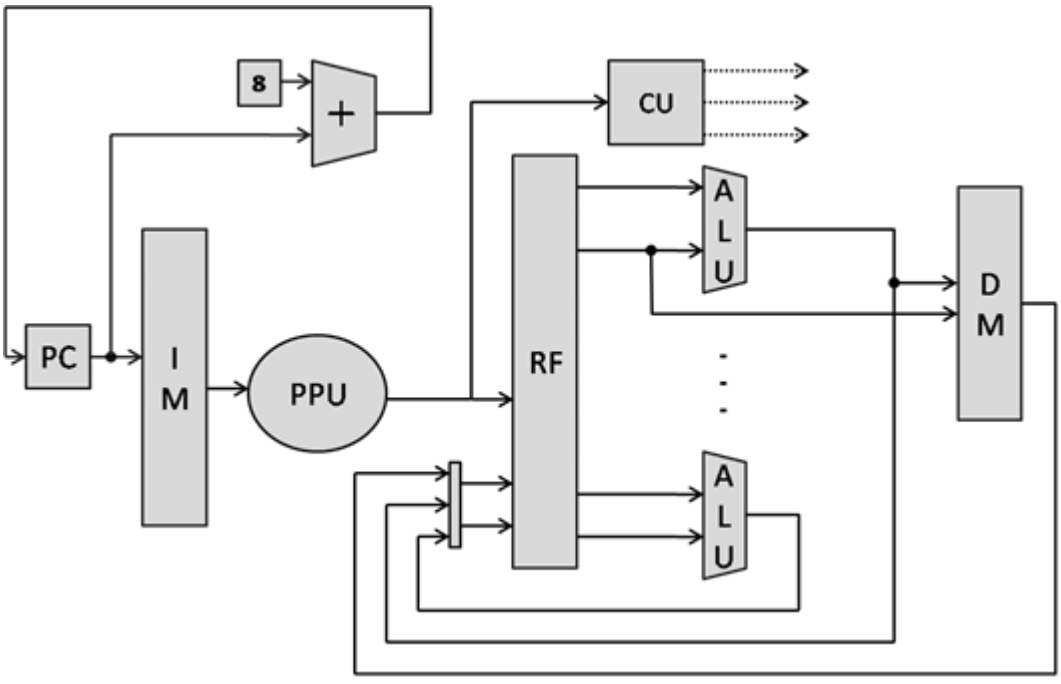

図 5. MAP の構成（データパス）

3.2 MAP の命令セットアーキテクチャ [2][4][5][6][7]

MAP には、Jump 形式（J 形式)、Register 形式（R 形式)、Immediate 形式（I 形式)、そして Long 形式（L 形式）の４つの命令形式がある。J 形式には無条件分岐命令の JUMP、プログラム終了命令となる HALT を定義している。R 形式にはレジスタ間の演算を行う命令を定義している。I 形式にはレジスタ値と即値演算を行う命令を定義している。L 形式には条件分岐命令とメモリ・レジスタへのデータ転送命令を定義している。L 形式、J 形式では DM へのアクセスと PC に JUMP する範囲を多く取ることにより、従来のプロセッサにはできなかった大規模な計算を行うことができる。表 3 に MAP の命令形式を示す。

表 3. MAP の命令形式

命令語長	32					
命令形式	6	5	5	5	5	6
R 形式	Op	Rs	Rt	Rd	Shamt	Fn
I 形式	Op	Rs	Rd	Imm		
L 形式	Op	Rs	Rd	Address/immediate		
J 形式	Op	address				

命令フィールドの意味は以下の通りである。

・Op（6bit)：命令を識別
・Rs（5bit)：ソースレジスタ
・Rt（5bit)：ソースレジスタ
・Rd（5bit)：演算結果を格納するレジスタ
・Shamt（5bit)：シフト量
・Fn（6bit)：ファンクション、R 形式命令を詳細に識別
・Imm（11～58bit)：アドレスと即値

また、命令セットを表 4 に示す。
R 形式は、加算命令 ADD や減算命令の SUB、レジスタ同士の比較を行う SLT、SNE などの命令が含まれる。
I 形式は、レジスタと即値との演算の際に用いられる。
L 形式は、分岐命令、メモリアクセスを行う命令を扱う命令形式である。
J 形式は JUMP 命令と終了命令である HALT を用いる。

表 4. 命令セット内容

R 形式	ADD, SUB, AND, OR, XOR, NOT, NOP, SLT, SGT, SLE, SGE, SEQ, SNE, SLL, SRL, SRA, JR
I 形式	ADDI, SUBI, ANDI, ORI, XORI, SLTI, SGTI, SLEI, SGEI, SEQI, SNEI, LDHI, LDLI
L 形式	BEQZ, BNEZ, LD, ST, JAL
J 形式	JUMP, HALT

3.3 4ALU の並列・連鎖・単一演算

MAP には、上記の命令セット内容の、37 種類の命令が存在する。MAP は並列処理が可能であり、２、４、８のプロセッサごとに並列・連鎖演算、単一実行など命令に合わせて処理を変更していく。ここでは 4 命令を取り出し判定する、4ALU での三種類の演算について記述する。

(1) 4ALU による並列演算

並列演算は、4 命令に関して依存関係がないときに行う演算であり、4ALU ではその中でも並列のパターンは多くの種類に分かれる。以下に 3 つの例を示す。

(a) 4 並列演算

命令 1　ADD　$1 $2 $3
命令 2　AND　$4 $3 $2
命令 3　SUB　$5 $6 $3
命令 4　AND　$2 $3 $3

上記の命令 1 ～4 の Rd にあたる$1、$4、$5、$2 とそれぞれの Rs/Rt にあたる各々のレジスタ番号を比較してみると、この 4 命令に関して全て依存関係が無く、メモリアクセス命令（LD, ST）や終了命令なども含まれていないので、この 4 命令は全て並列演算として処理することができる。(4 並列)

(b) 3 並列演算

命令 1　LD　$2 0[$0]
命令 2　AND　$5 $1 $6
命令 3　SUB　$4 $3 $2
命令 4　AND　$7 $4 $9

命令 1 の LD 命令は他の下位命令と依存関係がなく、命令 2 もその他の命令と依存関係がない。命令 3 と 4 は依存関係があるので、2 連鎖演算となるが、それらは命令 1 と命令 2 との並列演算で処理することができる。(3 並列、2 連鎖)

（c）2 並列演算

命令 1　ADD　$5 $1 $6
命令 2　AND　$3 $2 $3
命令 3　SUB　$4 $3 $2
命令 4　AND　$7 $8 $9

上記の命令 2、命令 3 を見てみると、依存関係が生じておりこれらは 2 連鎖演算として処理されるが、命令１、命令 4 は依存関係がなく、メモリアクセス命令や終了命令も存在していないので、これらは並列演算として処理することができる。(2 並列、2 連鎖)

(2)4ALU による連鎖演算

連鎖演算では、4 命令に関して依存関係があるときに行う演算であり、以下に例を示す。

命令 1　ADD　$1 $2 $3
命令 2　AND　$4 $3 $1
命令 3　SUB　$5 $6 $4
命令 4　AND　$2 $5 $3

上記の命令 1 の Rd にあたる$1 は下位命令の命令 2 で扱われており、命令 2 で扱われている Rd にあたる$4 は、命令 3 で扱われているので、この 4 命令は全て依存関係を持つ。従って、この 4 命令は 4 連鎖の演算として処理される。(4 連鎖)
また、4 命令の中にその他に依存関係が生じている場合、その数に応じて連鎖として処理することができる。(3 連鎖) (2 連鎖)

(3)単一演算

4ALU による単一演算の実行は、以下のとおりである。

（a）分岐命令が 1 クロック内の最上位命令で存在しているとき

4 命令中、1 クロック内で分岐命令が最上位命令である場合、そこで 1 クロックを終わらせ次のクロックに移る。

※分岐命令が最上位命令でなく、かつ上位命令と依存関係にある場合、並列または連鎖の演算として処理する。

（b）終了命令のとき

終了命令を実行するとき、他の演算と同時に終了命令を実行することはできないため単一演算として処理する。

図 6 に 4ALU のブロック構成を、図 7 に 4ALU の命令判定を示す。

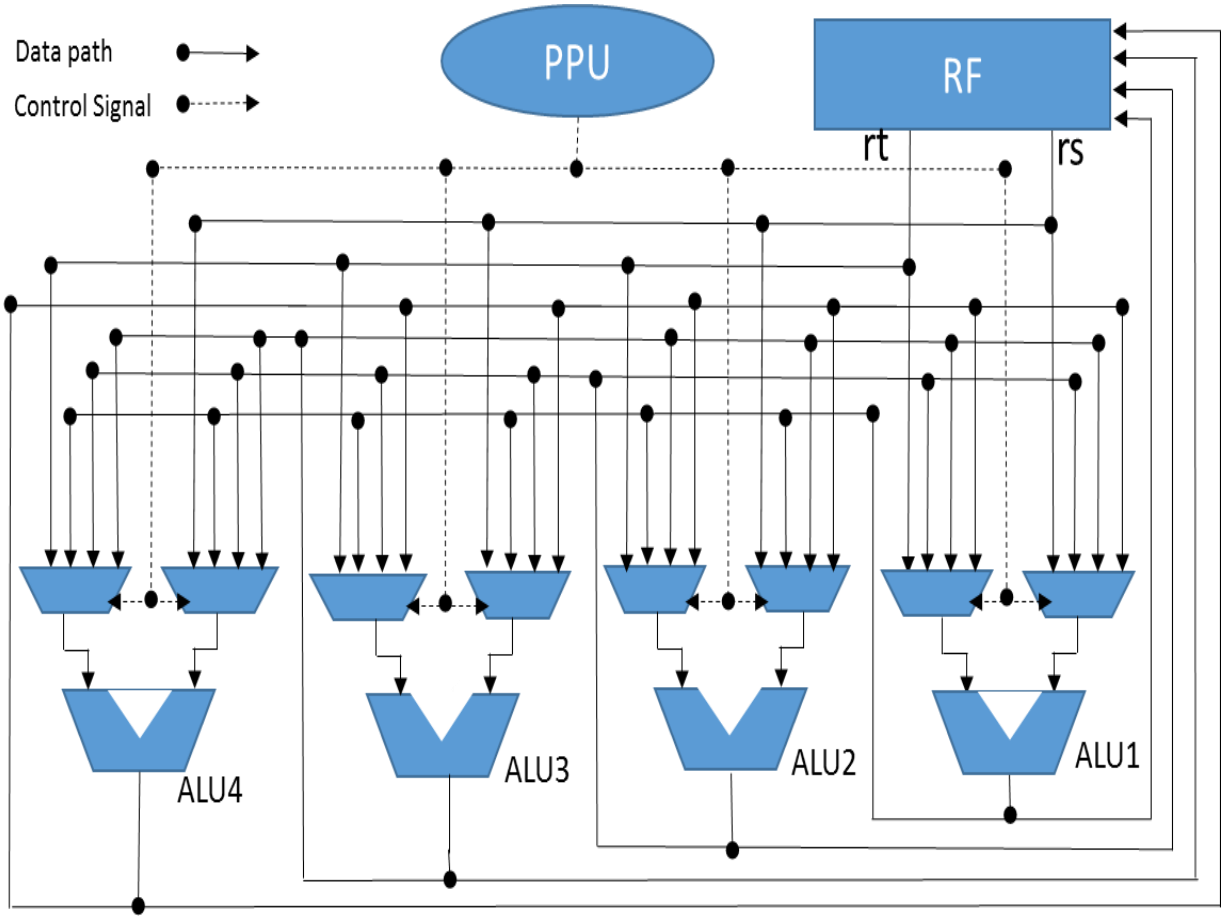

図 6. 4ALU のブロック構成

各機能の役割は 3.1MAP の構成で説明しているので、ここでは省略する。

命令数/1clock	1	2	3	4
命令の種類	単一	2並列	3並列	4並列
		2連鎖	2並列・2連鎖	4連鎖
			3連鎖	3並列・2連鎖
				2並列・3連鎖
				2連鎖・2連鎖

図 7. 4ALU の命令判定

４．　MAP シミュレータの設計

4.1　4ALU による MAP シミュレータのアルゴリズム

これまでは 4ALU での動作検証を行い、2ALU の演算を行ってきたが、実際に 4ALU で動作検証を行うための MAP シミュレータアルゴリズムを考える。具体的な例として、以下に 4 連鎖の依存関係があるアセンブリプログラムの 4 命令を挙げる。

命令 1：ADD $1 $5 $4
命令 2：SUB $2 $1 $3
命令 3：ADD $3 $2 $4
命令 4：ADD $4 $3 $1

MAP シミュレータのアルゴリズムを以下に記述する。まず、アセンブリプログラムの 4 命令をフェッチし、依存表に保存する。表 5 はレジスタ格納を、表 6 は依存表を示す。RS,RT は該当命令の第一ソースオペランドと第二ソースオペランドの依存先を保存している。さらに、該当する命令の依存先が存在しているかどうかを確認する flag を設定する。次に、1ALU を使う命令の Flag を更新し、実行が完了したことを示す 0 を記す。このような動作の流れを 1Step とし、flag が全て 0 になるまで行う。

表 5. レジスタ格納

I	Dop	S1	S2
i1	$1	$5	$4
i2	$2	$1	$3
i3	$3	$2	$4
i4	$4	$3	$1

表 5 の名称は以下のとおりである。

・I:命令番地
・Dop:デスティネーションレジスタ
・S1:ソースレジスタ 1
・S2:ソースレジスタ 2

以下に並列・連鎖演算の判定方法を記す。

・並列演算の回数…flag を 1 から 0 にするタイミングが同じ個数を並列演算としてカウントする。一つだけ flag の値を 0 にする場合はカウントせず、同時に 2 個 flag の値を 0 にした場合は 2 並列、3 個の場合は 3 並列、4 個の場合は 4 並列となる。

・連鎖演算の回数…RS、RT の中の値を 0 にした回数+1 でカウントする。RS、RT が同時に 0 となった場合のカウントは 1 回とする。

以下に、上記の 4 命令を全 4Step で、アルゴリズムを用いて説明する。

Step:1 依存表を作成、最上位命令の flag を 0 にする

初期設定としてまず、4ALU 依存表を完成させる。まず全ての flag の値を 1 とする。次に命令 2 の$1 (RS) が命令 1 の Dop と依存しているため、2 行目の RS を 1 と記す。その他も同様にして、表 6 を完成させる。

Step1-1:flag の更新

1 行目の RS、RT を調べると、双方の値が 0 なので 1 行目の flag を 0 にする。これを表 7 に示す。

表 6. 4ALU 依存表

flag	RS	RT
1	0	0
1	1	0
1	2	0
1	3	1

表 7. 4ALU 依存 Step1-1

flag	RS	RT
0	0	0
1	1	0
1	2	0
1	3	1

Step2-1:2 行目の RS、RT の依存関係を調べる

Step1-1 に続き、2 行目の RS、RT の値を調べると、2 行目の RS と 4 行目の RT は、1 行目の flag の値が 0 となったため依存関係が生じていない。よって、それらの値を表 8 のように 0 にしこれを Step2-1 とする。これにより連鎖演算の回数を 1 カウントする。

表 8. 4ALU 依存 Step2-1

flag	RS	RT
0	0	0
1	0	0
1	2	0
1	3	0

Step2-2:flag の更新

Step1-1 と同様、2 行目の RS、RT の値も 0 となったため 2 行目の flag も 0 とする。これを表 9 に示す。

表 9. 4ALU 依存 Step2-2

flag	RS	RT
0	0	0
0	0	0
1	2	0
1	3	0

Step3:3 行目の RS、RT の依存関係を調べ、その後 flag を更新する

Step1、Step2 同様、3 行目も引き続き RS、RT の依存関係を調べ、その数値を 0 とする。このとき連鎖演算の回数を 1 カウントし、これにより 3 行目の flag も 0 にする。これらを以下の表 10、表 11 に示す。

表 10. 4ALU 依存 Step3-1

Flag	RS	RT
0	0	0
0	0	0
1	0	0
1	3	0

表 11. 4ALU 依存 Step3-2

flag	RS	RT
0	0	0
0	0	0
0	0	0
1	3	0

Step4:4 行目の RS の値を 0 にし、flag を更新する

これまでの Step と同様、3 行目までの flag が更新されたので 4 行目の RS には依存関係が生じておらず、その数値を 0 とする。これで連鎖演算の回数を 1 カウントし、最後の flag を 0 にしたところですべての処理が完了したこととなる。これらを表 12、表 13 に示す。

表 12. 4ALU 依存 Step4-1

flag	RS	RT
0	0	0
0	0	0
0	0	0
1	0	0

表 13. 4ALU 依存 Step4-2

flag	RS	RT
0	0	0
0	0	0
0	0	0
0	0	0

以上より、並列演算の回数は、flag を同時に 1 から 0 にする個数がないため、0 回である。連鎖演算の回数は Step2-1、Step3-1、Step4-1 の 3 回に 1 を加え 4 回とし、4 連鎖が確認できる。

※1 クロック内の命令の、最上位にある分岐命令（BEQ,BNEZ など）や、終了命令（HALT）などは予め flag の値を 0 としておけば連鎖の数とみなされないので単一の判定をすることができる。

4.2　2,4ALU による命令間の連鎖、並列動作比較

これまで学習してきたBoothのアルゴリズムについて､1次Boothから3次Boothまで2、4ALUの動作検証を行い演算の回数を比較する。表14に2ALUのX=2、Y=(-3)を用いた演算の回数と演算の占める割合を、同様に表15に、4ALUのX=2、Y=(-3)を用いた演算の回数と演算の占める割合を示す。

表14. 2ALUによるX=2、Y=(-3)の演算比較

	連鎖あり						連鎖なし			
	命令個数			並列性(%)			命令個数		並列性(%)	
演算の種類	並列	連鎖	単一	並列	連鎖	単一	並列	単一	並列	単一
1次booth	4	15	12	13	48	39	7	36	16	84
2次booth	3	14	7	13	58	29	5	31	14	86
3次booth	4	30	6	10	75	15	5	64	7	93
平均	4	20	8	12	60	28	6	44	12	88

表15. 4ALUによるX=2、Y=(-3)の演算比較

	連鎖あり										連鎖なし			
	命令個数					並列性(%)					命令個数		並列性(%)	
演算の種類	2並列	2連鎖	3連鎖	4連鎖	単一	2並列	2連鎖	3連鎖	4連鎖	単一	2並列	単一	2並列	単一
1次	4	7	10	0	1	18	32	45	0	5	7	36	16	84
2次	3	11	5	0	1	15	55	25	0	5	5	31	14	86
3次	2	25	5	1	2	6	71	14	3	6	5	64	7	93
平均	3	14	7	0	1	13	53	28	1	5	6	44	12	88

4.3　考察

1次Boothの乗算プログラムでは、2ALUから4ALUへと変更した場合、単純なプログラムの繰り返しであったため、あまり演算の変化が見られなかったが、2次、3次へと次数をあげていくとプログラムも長くなるので、2次Boothの乗算プログラムは2ALUに比べて4ALUの並列演算が減少し新たに3つのALUを用いて3連鎖の数が増えていた。これにより並列演算の数が減少したと思われるが、全体の並列性としては単一演算の数が減少し、並列性が上がったと見込まれる。

5. おわりに

本論文では、本研究室で設計された 1 次・2 次 Booth 乗算のアルゴリズムによるアセンブリプログラムに加え、新たに 3 次 Booth 乗算のアルゴリズムによるアセンブリプログラムを設計し、それらの並列性の検証を行った。Booth 乗算のプログラムに関しては、連鎖なしでの演算が、80%単一演算を占め、連鎖演算が演算の高速化に必要であると確認をすることができた。さらに、2ALU と 4ALU を比較することで、並列性の性能向上が見込まれ高速に演算を処理する有用性を見出すことができた。また、今後 4ALU シミュレータを設計する上での 4ALU シミュレータのアルゴリズムを考え、4 命令分を読みだすことで並列・連鎖の演算を同時に行うことができると認識できた。実際にこのアルゴリズムをプログラムで作成し、実装できるかが今後の課題である。

謝辞

本研究の機会を与えて下さり、ご指導を頂きました山崎勝弘教授に深く感謝致します。また、本研究に関して様々な相談から貴重なご意見、ご指導して頂いた孟林助教、石川陽章氏、杵川大智氏をはじめ、様々な面で貴重な助言や励ましを下さった研究室の皆様に深く感謝いたします。

参考文献

[1]泉知論：マイクロプロセッサデザイン,演算器 1,講義ノート,2012.
[2]境直樹：演算レベル並列処理用マルチ ALU プロセッサの設計と実現,立命館大学大学院理工学研究科創造理工学専攻,修士論文,2013.
[3]境直樹：MAP 仕様書,2011.
[4]杵川大智：マルチ ALU プロセッサの並列プログラミングと動作検証,立命館大学理工学部電子情報デザイン学科,卒業論文,2013.
[5]石川陽章：マルチ ALU プロセッサのデバッガ用 GUI の設計と実現,立命館大学理工学部電子情報デザイン学科,卒業論文,2013.
[6]田中亮佑：マルチ ALU プロセッサにおけるアセンブラの設計と試作（Ⅰ）,立命館大学理工学部電子情報デザイン学科,卒業論文,2012.
[7]高松良太：マルチ ALU プロセッサにおけるシミュレータの設計と試作,立命館大学理工学部電子情報デザイン学科,卒業論文,2012.
[8]尾形幸亮、姚駿、嶋田創、三輪忍、富田眞治：ALU Cascading を行う動的命令スケジューラ,情報処理学会,研究報告,2007-ARC-173(16).
[9]石川陽章、杵川大智、境直樹、孟林、山崎勝弘：演算レベル並列処理用マルチ ALU プロセッサの設計と実現, 第 12 回情報科学技術フォーラム,FIT2013,C-005,2013.

付録A　Boothの乗算アセンブリプログラムによる2ALU演算結果

（1）1次Booth

step	番号	命令						2並列	2連鎖	単一
1	1		LD	$1	0[$0]			1 2		
	2		LD	$2	4[$0]					
2	3		ANDI	$2	$2	15		3 4		
	4		SLL	$1	$1	5				
3	5		SLL	$2	$2	1		5 6		
	6	LOOP:	ADDI	$15	$15	1				
4	7		SEQI	$16	$15	5			7 8	
	8		BNEZ	$16	LAST					
5	9		ANDI	$3	$2	3			9 10	
	10		SEQI	$4	$3	2				
6	11		BNEZ	$4	SKIP1					11
7	16	SKIP1:	SUB	$2	$2	$1			16 17	
	17		SRL	$2	$2	1				
8	18		JUMP	LOOP						18
9	6	LOOP:	ADDI	$15	$15	1			6 7	
	7		SEQI	$16	$15	5				
10	8		BNEZ	$16	LAST					8
11	9		ANDI	$3	$2	3			9 10	
	10		SEQI	$4	$3	2				
12	11		BNEZ	$4	SKIP1					11
13	12		SEQI	$5	$3	1			12 13	
	13		BNEZ	$5	SKIP2					
14	19	SKIP2:	ADD	$2	$2	$1			19 20	
	20		SRL	$2	$2	1				
15	21		JUMP	LOOP						21
16	6	LOOP:	ADDI	$15	$15	1			6 7	
	7		SEQI	$16	$15	5				
17	8		BNEZ	$16	LAST					8
18	9		ANDI	$3	$2	3			9 10	
	10		SEQI	$4	$3	2				
19	11		BNEZ	$4	SKIP1					11
20	16	SKIP1:	SUB	$2	$2	$1			16 17	
	17		SRL	$2	$2	1				
21	18		JUMP	LOOP						18
22	6	LOOP:	ADDI	$15	$15	1			6 7	
	7		SEQI	$16	$15	5				
23	8		BNEZ	$16	LAST					8
24	9		ANDI	$3	$2	3			9 10	
	10		SEQI	$4	$3	2				
25	11		BNEZ	$4	SKIP1					11
26	12		SEQI	$5	$3	1			12 13	
	13		BNEZ	$5	SKIP2					
27	14		SRL	$2	$2	1		14 15		
	15		JUMP	LOOP						
28	6	LOOP:	ADDI	$15	$15	1			6 7	
	7		SEQI	$16	$15	5				
29	8		BNEZ	$16	LAST					8
30	22	LAST:	SRL	$2	$2	1			22 23	
	23		ST	$2	12[$0]					
31	24		HALT							24
							合計	4	15	12

図8．1次Boothのプログラムを用いた2×（-3）の2ALU演算結果（連鎖あり）

step	番号	命令						2並列	単一
1	1		LD	$1	0[$0]			1 2	
	2		LD	$2	4[$0]				
2	3		ANDI	$2	$2	15		3 4	
	4		SLL	$1	$1	5			
3	5		SLL	$2	$2	1		5 6	
	6	LOOP:	ADDI	$15	$15	1			
4	7		SEQI	$16	$15	5			7
5	8		BNEZ	$16	LAST				8
6	9		ANDI	$3	$2	3			9
7	10		SEQI	$4	$3	2			10
8	11		BNEZ	$4	SKIP1				11
9	16	SKIP1:	SUB	$2	$2	$1			16
10	17		SRL	$2	$2	1		17 18	
	18		JUMP	LOOP					
11	6	LOOP:	ADDI	$15	$15	1			6
12	7		SEQI	$16	$15	5			7
13	8		BNEZ	$16	LAST				8
14	9		ANDI	$3	$2	3			9
15	10		SEQI	$4	$3	2			10
16	11		BNEZ	$4	SKIP1				11
17	12		SEQI	$5	$3	1			12
18	13		BNEZ	$5	SKIP2				13
19	19	SKIP2:	ADD	$2	$2	$1			19
20	20		SRL	$2	$2	1		20 21	
	21		JUMP	LOOP					
21	6	LOOP:	ADDI	$15	$15	1			6
22	7		SEQI	$16	$15	5			7
23	8		BNEZ	$16	LAST				8
24	9		ANDI	$3	$2	3			9
25	10		SEQI	$4	$3	2			10
26	11		BNEZ	$4	SKIP1				11
27	16	SKIP1:	SUB	$2	$2	$1			16
28	17		SRL	$2	$2	1		17 18	
	18		JUMP	LOOP					
29	6	LOOP:	ADDI	$15	$15	1			6
30	7		SEQI	$16	$15	5			7
31	8		BNEZ	$16	LAST				8
32	9		ANDI	$3	$2	3			9
33	10		SEQI	$4	$3	2			10
34	11		BNEZ	$4	SKIP1				11
35	12		SEQI	$5	$3	1			12
36	13		BNEZ	$5	SKIP2				13
37	14		SRL	$2	$2	1		14 15	
	15		JUMP	LOOP					
38	6	LOOP:	ADDI	$15	$15	1			6
39	7		SEQI	$16	$15	5			7
40	8		BNEZ	$16	LAST				8
41	22	LAST:	SRL	$2	$2	1			22
42	23		ST	$2	12[$0]				23
43	24		HALT						24
							合計	7	36

図 9. 1 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖なし）

(2) 2次 Booth

step	番号	命令					並列	2連鎖	単一
1	1		LD	$1	0[$0]		1 2		
	2		LD	$2	4[$0]				
2	3		ANDI	$2	$2	15	3 4		
	4		SLL	$1	$1	5			
3	5		SLL	$2	$2	1	5 6		
	6	LOOP:	ADDI	$15	$15	1			
4	7		SEQI	$16	$15	3		7 8	
	8		BNEZ	$16	LAST				
5	9		ANDI	$3	$2	7		9 10	
	10		SEQI	$5	$3	1			
6	11		BNEZ	$5	SKIP1				11
7	12		SEQI	$4	$3	2		12 13	
	13		BNEZ	$4	SKIP1				
8	24	SKIP1:	ADD	$2	$2	$1		24 25	
	25		SRL	$2	$2	2			
9	26		JUMP	LOOP					26
10	6	LOOP:	ADDI	$15	$15	1		6 7	
	7		SEQI	$16	$15	3			
11	8		BNEZ	$16	LAST				8
12	9		ANDI	$3	$2	7		9 10	
	10		SEQI	$5	$3	1			
13	11		BNEZ	$5	SKIP1				11
14	12		SEQI	$4	$3	2		12 13	
	13		BNEZ	$4	SKIP1				
15	14		SEQI	$5	$3	3		14 15	
	15		BNEZ	$5	SKIP3				
16	16		SEQI	$5	$3	4		16 17	
	17		BNEZ	$5	SKIP4				
17	18		SEQI	$5	$3	5		18 19	
	19		BNEZ	$5	SKIP2				
18	20		SEQI	$5	$3	6		20 21	
	21		BNEZ	$5	SKIP2				
19	27	SKIP2:	SUB	$2	$2	$1		27 28	
	28		SRL	$2	$2	2			
20	29		JUMP	LOOP					29
21	6	LOOP:	ADDI	$15	$15	1		6 7	
	7		SEQI	$16	$15	3			
22	8		BNEZ	$16	LAST				8
23	38	LAST:	SRL	$2	$2	1		38 39	
	39		ST	$2	12[$0]				
24	40		HALT						40
							合計	3	14	7

図 10. 2次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖あり）

step	番号	命令						並列	単一
1	1		LD	$1	0[$0]			1 2	
	2		LD	$2	4[$0]				
2	3		ANDI	$2	$2	15		3 4	
	4		SLL	$1	$1	5			
3	5		SLL	$2	$2	1		5 6	
	6	LOOP:	ADDI	$15	$15	1			
4	7		SEQI	$16	$15	3			7
5	8		BNEZ	$16	LAST				8
6	9		ANDI	$3	$2	7			9
7	10		SEQI	$5	$3	1			10
8	11		BNEZ	$5	SKIP1				11
9	12		SEQI	$4	$3	2			12
10	13		BNEZ	$4	SKIP1				13
11	24	SKIP1:	ADD	$2	$2	$1			24
12	25		SRL	$2	$2	2		25 26	
	26		JUMP	LOOP					
13	6	LOOP:	ADDI	$15	$15	1			6
14	7		SEQI	$16	$15	3			7
15	8		BNEZ	$16	LAST				8
16	9		ANDI	$3	$2	7			9
17	10		SEQI	$5	$3	1			10
18	11		BNEZ	$5	SKIP1				11
19	12		SEQI	$4	$3	2			12
20	13		BNEZ	$4	SKIP1				13
21	14		SEQI	$5	$3	3			14
22	15		BNEZ	$5	SKIP3				15
23	16		SEQI	$5	$3	4			16
24	17		BNEZ	$5	SKIP4				17
25	18		SEQI	$5	$3	5			18
26	19		BNEZ	$5	SKIP2				19
27	20		SEQI	$5	$3	6			20
28	21		BNEZ	$5	SKIP2				21
29	27	SKIP2:	SUB	$2	$2	$1			27
30	28		SRL	$2	$2	2		28 29	
	29		JUMP	LOOP					
31	6	LOOP:	ADDI	$15	$15	1			6
32	7		SEQI	$16	$15	3			7
33	8		BNEZ	$16	LAST				8
34	38	LAST:	SRL	$2	$2	1			38
35	39		ST	$2	12[$0]				39
36	40		HALT						40
							合計	5	31

図 11. 2 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖なし）

(3) 3 次 Booth

番号	命令						2並列	2連鎖	単一
1		LD	$1	0[$0]			1 2		
2		LD	$2	4[$0]					
3		ANDI	$2	$2	63		3 4		
4		SLL	$1	$1	7				
5		SLL	$2	$2	1		5 6		
6	LOOP:	ADDI	$15	$15	1				
7		SEQI	$16	$15	3			7 8	
8		BNEZ	$16	LAST					
9		ANDI	$3	$2	15			9 10	
10		SEQI	$5	$3	1				
11		BNEZ	$5	SKIP1					11
12		SEQI	$4	$3	2			12,13	
13		BNEZ	$4	SKIP1					
14		SEQI	$5	$3	3			14,15	
15		BNEZ	$5	SKIP3					
16		SEQI	$5	$3	4			16,17	
17		BNEZ	$5	SKIP3					
18		SEQI	$5	$3	5			18,19	
19		BNEZ	$5	SKIP5					
20		SEQI	$5	$3	6			20,21	
21		BNEZ	$5	SKIP5					
22		SEQI	$5	$3	7			22,23	
23		BNEZ	$5	SKIP7					
24		SEQI	$5	$3	8			24,25	
25		BNEZ	$5	SKIP8					
26		SEQI	$5	$3	9			26,27	
27		BNEZ	$5	SKIP6					
28		SEQI	$5	$3	10			28,29	
29		BNEZ	$5	SKIP6					
59	SKIP6:	SUB	$2	$2	$1			59,60	
60		SUB	$2	$2	$1				
61		SUB	$2	$2	$1			61,62	
62		SRL	$2	$2	3				
63		JUMP	LOOP						63
6	LOOP:	ADDI	$15	$15	1			6,7	
7		SEQI	$16	$15	3				
8		BNEZ	$16	LAST					8
9		ANDI	$3	$2	15			9,10	
10		SEQI	$5	$3	1				
11		BNEZ	$5	SKIP1					11
12		SEQI	$4	$3	2			12,13	
13		BNEZ	$4	SKIP1					
14		SEQI	$5	$3	3			14,15	
15		BNEZ	$5	SKIP3					
16		SEQI	$5	$3	4			16,17	
17		BNEZ	$5	SKIP3					
18		SEQI	$5	$3	5			18,19	
19		BNEZ	$5	SKIP5					
20		SEQI	$5	$3	6			20,21	
21		BNEZ	$5	SKIP5					
22		SEQI	$5	$3	7			22,23	
23		BNEZ	$5	SKIP7					
24		SEQI	$5	$3	8			24,25	
25		BNEZ	$5	SKIP8					
26		SEQI	$5	$3	9			26,27	
27		BNEZ	$5	SKIP6					
28		SEQI	$5	$3	10			28,29	
29		BNEZ	$5	SKIP6					
30		SEQI	$5	$3	11			30,31	
31		BNEZ	$5	SKIP4					
32		SEQI	$5	$3	12			32,33	
33		BNEZ	$5	SKIP4					
34		SEQI	$5	$3	13			34,35	
35		BNEZ	$5	SKIP2					
36		SEQI	$5	$3	14			36,37	
37		BNEZ	$5	SKIP2					
38		SRL	$2	$2	3		38,39		
39		JUMP	LOOP						
6	LOOP:	ADDI	$15	$15	1			6,7	
7		SEQI	$16	$15	3				
8		BNEZ	$16	LAST					8
76	LAST:	SRL	$2	$2	1			76,77	
77		ST	$2	12[$0]					
78		HALT							78
						合計	4	30	6

図 12. 3 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖あり）

番号	命令						2並列	単一
1		LD	$1	0[$0]			1,2	
2		LD	$2	4[$0]				
3		ANDI	$2	$2	63		3,4	
4		SLL	$1	$1	7			
5		SLL	$2	$2	1		5,6	
6	LOOP:	ADDI	$15	$15	1			
7		SEQI	$16	$15	3			7
8		BNEZ	$16	LAST				8
9		ANDI	$3	$2	15			9
10		SEQI	$5	$3	1			10
11		BNEZ	$5	SKIP1				11
12		SEQI	$4	$3	2			12
13		BNEZ	$4	SKIP1				13
14		SEQI	$5	$3	3			14
15		BNEZ	$5	SKIP3				15
16		SEQI	$5	$3	4			16
17		BNEZ	$5	SKIP3				17
18		SEQI	$5	$3	5			18
19		BNEZ	$5	SKIP5				19
20		SEQI	$5	$3	6			20
21		BNEZ	$5	SKIP5				21
22		SEQI	$5	$3	7			22
23		BNEZ	$5	SKIP7				23
24		SEQI	$5	$3	8			24
25		BNEZ	$5	SKIP8				25
26		SEQI	$5	$3	9			26
27		BNEZ	$5	SKIP6				27
28		SEQI	$5	$3	10			28
29		BNEZ	$5	SKIP6				29
59	SKIP6:	SUB	$2	$2	$1			59
60		SUB	$2	$2	$1			60
61		SUB	$2	$2	$1			61
62		SRL	$2	$2	3			
63		JUMP	LOOP				62,63	
6	LOOP:	ADDI	$15	$15	1			6
7		SEQI	$16	$15	3			7
8		BNEZ	$16	LAST				8
9		ANDI	$3	$2	15			9
10		SEQI	$5	$3	1			10
11		BNEZ	$5	SKIP1				11
12		SEQI	$4	$3	2			12
13		BNEZ	$4	SKIP1				13
14		SEQI	$5	$3	3			14
15		BNEZ	$5	SKIP3				15
16		SEQI	$5	$3	4			16
17		BNEZ	$5	SKIP3				17
18		SEQI	$5	$3	5			18
19		BNEZ	$5	SKIP5				19
20		SEQI	$5	$3	6			20
21		BNEZ	$5	SKIP5				21
22		SEQI	$5	$3	7			22
23		BNEZ	$5	SKIP7				23
24		SEQI	$5	$3	8			24
25		BNEZ	$5	SKIP8				25
26		SEQI	$5	$3	9			26
27		BNEZ	$5	SKIP6				27
28		SEQI	$5	$3	10			28
29		BNEZ	$5	SKIP6				29
30		SEQI	$5	$3	11			30
31		BNEZ	$5	SKIP4				31
32		SEQI	$5	$3	12			32
33		BNEZ	$5	SKIP4				33
34		SEQI	$5	$3	13			34
35		BNEZ	$5	SKIP2				35
36		SEQI	$5	$3	14			36
37		BNEZ	$5	SKIP2				37
38		SRL	$2	$2	3		38,39	
39		JUMP	LOOP					
6	LOOP:	ADDI	$15	$15	1			6
7		SEQI	$16	$15	3			7
8		BNEZ	$16	LAST				8
76	LAST:	SRL	$2	$2	1			76
77		ST	$2	12[$0]				77
78		HALT						78
						合計	5	64

図 13. 3 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖なし）

付録 B　Booth のアセンブリプログラムによる 4ALU 演算結果

(1) 1 次 Booth

step	番号	命令						2並列	2連鎖	3連鎖	4連鎖	単一
1	1		LD	$1	0[$0]				(1,4) (2,3)			
	2		LD	$2	4[$0]							
	3		ANDI	$2	$2	15						
	4		SLL	$1	$1	5						
2	5		SLL	$2	$2	1		5 6		6 7 8		
	6	LOOP:	ADDI	$15	$15	1						
	7		SEQI	$16	$15	5						
	8		BNEZ	$16	LAST							
3	9		ANDI	$3	$2	3				9 10 11		
	10		SEQI	$4	$3	2						
	11		BNEZ	$4	SKIP1							
4	16	SKIP1:	SUB	$2	$2	$1			16 17			
	17		SRL	$2	$2	1						
	18		JUMP	LOOP				17 18				
5	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	5						
	8		BNEZ	$16	LAST							
6	9		ANDI	$3	$2	3				9 10 11		
	10		SEQI	$4	$3	2						
	11		BNEZ	$4	SKIP1							
7	12		SEQI	$5	$3	1			12 13			
	13		BNEZ	$5	SKIP2							
8	19	SKIP2:	ADD	$2	$2	$1			19 20			
	20		SRL	$2	$2	1						
	21		JUMP	LOOP				20 21				
9	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	5						
	8		BNEZ	$16	LAST							
10	9		ANDI	$3	$2	3				9 10 11		
	10		SEQI	$4	$3	2						
	11		BNEZ	$4	SKIP1							
11	16	SKIP1:	SUB	$2	$2	$1			16 17			
	17		SRL	$2	$2	1						
	18		JUMP	LOOP				17 18				
12	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	5						
	8		BNEZ	$16	LAST							
13	9		ANDI	$3	$2	3				9 10 11		
	10		SEQI	$4	$3	2						
	11		BNEZ	$4	SKIP1							
14	12		SEQI	$5	$3	1			12 13			
	13		BNEZ	$5	SKIP2							
15	14		SRL	$2	$2	1		14 15				
	15		JUMP	LOOP								
16	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	5						
	8		BNEZ	$16	LAST							
17	22	LAST:	SRL	$2	$2	1			22 23			
	23		ST	$2	12[$0]							
	24		HALT									24
							合計	4	7	10	0	1

図 14. 1 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖あり）

step	番号	命令						2並列	単一
1	1		LD	$1	0[$0]			1 2	
	2		LD	$2	4[$0]				
2	3		ANDI	$2	$2	15		3 4	
	4		SLL	$1	$1	5			
3	5		SLL	$2	$2	1		5 6	
	6	LOOP:	ADDI	$15	$15	1			
4	7		SEQI	$16	$15	5			7
5	8		BNEZ	$16	LAST				8
6	9		ANDI	$3	$2	3			9
7	10		SEQI	$4	$3	2			10
8	11		BNEZ	$4	SKIP1				11
9	16	SKIP1:	SUB	$2	$2	$1			16
10	17		SRL	$2	$2	1		17 18	
	18		JUMP	LOOP					
11	6	LOOP:	ADDI	$15	$15	1			6
12	7		SEQI	$16	$15	5			7
13	8		BNEZ	$16	LAST				8
14	9		ANDI	$3	$2	3			9
15	10		SEQI	$4	$3	2			10
16	11		BNEZ	$4	SKIP1				11
17	12		SEQI	$5	$3	1			12
18	13		BNEZ	$5	SKIP2				13
19	19	SKIP2:	ADD	$2	$2	$1			19
20	20		SRL	$2	$2	1		20 21	
	21		JUMP	LOOP					
21	6	LOOP:	ADDI	$15	$15	1			6
22	7		SEQI	$16	$15	5			7
23	8		BNEZ	$16	LAST				8
24	9		ANDI	$3	$2	3			9
25	10		SEQI	$4	$3	2			10
26	11		BNEZ	$4	SKIP1				11
27	16	SKIP1:	SUB	$2	$2	$1			16
28	17		SRL	$2	$2	1		17 18	
	18		JUMP	LOOP					
29	6	LOOP:	ADDI	$15	$15	1			6
30	7		SEQI	$16	$15	5			7
31	8		BNEZ	$16	LAST				8
32	9		ANDI	$3	$2	3			9
33	10		SEQI	$4	$3	2			10
34	11		BNEZ	$4	SKIP1				11
35	12		SEQI	$5	$3	1			12
36	13		BNEZ	$5	SKIP2				13
37	14		SRL	$2	$2	1		14 15	
	15		JUMP	LOOP					
38	6	LOOP:	ADDI	$15	$15	1			6
39	7		SEQI	$16	$15	5			7
40	8		BNEZ	$16	LAST				8
41	22	LAST:	SRL	$2	$2	1			22
42	23		ST	$2	12[$0]				23
43	24		HALT						24
							合計	7	36

図 15. 1 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖なし）

(2) 2次 Booth

step	番号	命令						2並列	2連鎖	3連鎖	4連鎖	単一
1	1		LD	$1	0[$0]				(1,4)(2,3)			
	2		LD	$2	4[$0]							
	3		ANDI	$2	$2	15						
	4		SLL	$1	$1	5						
2	5		SLL	$2	$2	1		5 6				
	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	3						
	8		BNEZ	$16	LAST							
3	9		ANDI	$3	$2	7				9 10 11		
	10		SEQI	$5	$3	1						
	11		BNEZ	$5	SKIP1							
4	12		SEQI	$4	$3	2			12 13			
	13		BNEZ	$4	SKIP1							
5	24	SKIP1:	ADD	$2	$2	$1			24 25			
	25		SRL	$2	$2	2						
	26		JUMP	LOOP				25 26				
6	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	3						
	8		BNEZ	$16	LAST							
7	9		ANDI	$3	$2	7				9 10 11		
	10		SEQI	$5	$3	1						
	11		BNEZ	$5	SKIP1							
8	12		SEQI	$4	$3	2			12 13			
	13		BNEZ	$4	SKIP1							
9	14		SEQI	$5	$3	3			14 15			
	15		BNEZ	$5	SKIP3							
10	16		SEQI	$5	$3	4			16 17			
	17		BNEZ	$5	SKIP4							
11	18		SEQI	$5	$3	5			18 19			
	19		BNEZ	$5	SKIP2							
12	20		SEQI	$5	$3	6			20 21			
	21		BNEZ	$5	SKIP2							
13	27	SKIP2:	SUB	$2	$2	$1			27 28			
	28		SRL	$2	$2	2						
	29		JUMP	LOOP				28 29				
14	6	LOOP:	ADDI	$15	$15	1				6 7 8		
	7		SEQI	$16	$15	3						
	8		BNEZ	$16	LAST							
15	38	LAST:	SRL	$2	$2	1			38 39			
	39		ST	$2	12[$0]							
16	40		HALT									40
							合計	3	11	5	0	1

図 16. 2次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖あり）

step	番号	命令						2並列	単一
1	1		LD	$1	0[$0]			1 2	
	2		LD	$2	4[$0]				
2	3		ANDI	$2	$2	15			
	4		SLL	$1	$1	5		3 4	
3	5		SLL	$2	$2	1		5 6	
	6	LOOP:	ADDI	$15	$15	1			
4	7		SEQI	$16	$15	3			7
5	8		BNEZ	$16	LAST			8 9	
	9		ANDI	$3	$2	7			
6	10		SEQI	$5	$3	1			10
7	11		BNEZ	$5	SKIP1				11
8	12		SEQI	$4	$3	2			12
9	13		BNEZ	$4	SKIP1				13
10	24	SKIP1:	ADD	$2	$2	$1			24
11	25		SRL	$2	$2	2		25 26	
	26		JUMP	LOOP					
12	6	LOOP:	ADDI	$15	$15	1			6
13	7		SEQI	$16	$15	3			7
14	8		BNEZ	$16	LAST				8
15	9		ANDI	$3	$2	7			9
16	10		SEQI	$5	$3	1			10
17	11		BNEZ	$5	SKIP1				11
18	12		SEQI	$4	$3	2			12
19	13		BNEZ	$4	SKIP1				13
20	14		SEQI	$5	$3	3			14
21	15		BNEZ	$5	SKIP3				15
22	16		SEQI	$5	$3	4			16
23	17		BNEZ	$5	SKIP4				17
24	18		SEQI	$5	$3	5			18
25	19		BNEZ	$5	SKIP2				19
26	20		SEQI	$5	$3	6			20
27	21		BNEZ	$5	SKIP2				21
28	27	SKIP2:	SUB	$2	$2	$1			27
29	28		SRL	$2	$2	2		28 29	
	29		JUMP	LOOP					
30	6	LOOP:	ADDI	$15	$15	1			6
31	7		SEQI	$16	$15	3			7
32	8		BNEZ	$16	LAST				8
33	38	LAST:	SRL	$2	$2	1			38
34	39		ST	$2	12[$0]				39
35	40		HALT						40
							合計	5	31

図 17. 2 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖なし）

(3) 3 次 Booth

番号	命令						2並列	2連鎖	3連鎖	4連鎖	単一
1		LD	$1	0[$0]				(1,4)(2,3)			
2		LD	$2	4[$0]							
3		ANDI	$2	$2	63						
4		SLL	$1	$1	7						
5		SLL	$2	$2	1		5,6		6,7,8		
6	LOOP:	ADDI	$15	$15	1						
7		SEQI	$16	$15	3						
8		BNEZ	$16	LAST							
9		ANDI	$3	$2	15				9,10,11		
10		SEQI	$5	$3	1						
11		BNEZ	$5	SKIP1							
12		SEQI	$4	$3	2			12,13			
13		BNEZ	$4	SKIP1							
14		SEQI	$5	$3	3			14,15			
15		BNEZ	$5	SKIP3							
16		SEQI	$5	$3	4			16,17			
17		BNEZ	$5	SKIP3							
18		SEQI	$5	$3	5			18,19			
19		BNEZ	$5	SKIP5							
20		SEQI	$5	$3	6			20,21			
21		BNEZ	$5	SKIP5							
22		SEQI	$5	$3	7			22,23			
23		BNEZ	$5	SKIP7							
24		SEQI	$5	$3	8			24,25			
25		BNEZ	$5	SKIP8							
26		SEQI	$5	$3	9			26,27			
27		BNEZ	$5	SKIP6							
28		SEQI	$5	$3	10			28,29			
29		BNEZ	$5	SKIP6							
59	SKIP6:	SUB	$2	$2	$1					59,60,61,62	
60		SUB	$2	$2	$1						
61		SUB	$2	$2	$1						
62		SRL	$2	$2	3						
63		JUMP	LOOP								63
6	LOOP:	ADDI	$15	$15	1				6,7,8		
7		SEQI	$16	$15	3						
8		BNEZ	$16	LAST							
9		ANDI	$3	$2	15				9,10,11		
10		SEQI	$5	$3	1						
11		BNEZ	$5	SKIP1							
12		SEQI	$4	$3	2			12,13			
13		BNEZ	$4	SKIP1							
14		SEQI	$5	$3	3			14,15			
15		BNEZ	$5	SKIP3							
16		SEQI	$5	$3	4			16,17			
17		BNEZ	$5	SKIP3							
18		SEQI	$5	$3	5			18,19			
19		BNEZ	$5	SKIP5							
20		SEQI	$5	$3	6			20,21			
21		BNEZ	$5	SKIP5							
22		SEQI	$5	$3	7			22,23			
23		BNEZ	$5	SKIP7							
24		SEQI	$5	$3	8			24,25			
25		BNEZ	$5	SKIP8							
26		SEQI	$5	$3	9			26,27			
27		BNEZ	$5	SKIP6							
28		SEQI	$5	$3	10			28,29			
29		BNEZ	$5	SKIP6							
30		SEQI	$5	$3	11			30,31			
31		BNEZ	$5	SKIP4							
32		SEQI	$5	$3	12			32,33			
33		BNEZ	$5	SKIP4							
34		SEQI	$5	$3	13			34,35			
35		BNEZ	$5	SKIP2							
36		SEQI	$5	$3	14			36,37			
37		BNEZ	$5	SKIP2							
38		SRL	$2	$2	3		38,39				
39		JUMP	LOOP								
6	LOOP:	ADDI	$15	$15	1				6,7,8		
7		SEQI	$16	$15	3						
8		BNEZ	$16	LAST							
76	LAST:	SRL	$2	$2	1			76,77			
77		ST	$2	12[$0]							
78		HALT									78
						合計	2	25	5	1	2

図 18. 3 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖あり）

番号	命令						並列	単一
1		LD	$1	0[$0]			1,2	
2		LD	$2	4[$0]				
3		ANDI	$2	$2	63		3,4	
4		SLL	$1	$1	7			
5		SLL	$2	$2	1		5,6	
6	LOOP:	ADDI	$15	$15	1			
7		SEQI	$16	$15	3			7
8		BNEZ	$16	LAST				8
9		ANDI	$3	$2	15			9
10		SEQI	$5	$3	1			10
11		BNEZ	$5	SKIP1				11
12		SEQI	$4	$3	2			12
13		BNEZ	$4	SKIP1				13
14		SEQI	$5	$3	3			14
15		BNEZ	$5	SKIP3				15
16		SEQI	$5	$3	4			16
17		BNEZ	$5	SKIP3				17
18		SEQI	$5	$3	5			18
19		BNEZ	$5	SKIP5				19
20		SEQI	$5	$3	6			20
21		BNEZ	$5	SKIP5				21
22		SEQI	$5	$3	7			22
23		BNEZ	$5	SKIP7				23
24		SEQI	$5	$3	8			24
25		BNEZ	$5	SKIP8				25
26		SEQI	$5	$3	9			26
27		BNEZ	$5	SKIP6				27
28		SEQI	$5	$3	10			28
29		BNEZ	$5	SKIP6				29
59	SKIP6:	SUB	$2	$2	$1			59
60		SUB	$2	$2	$1			60
61		SUB	$2	$2	$1			61
62		SRL	$2	$2	3			
63		JUMP	LOOP				62,63	
6	LOOP:	ADDI	$15	$15	1			6
7		SEQI	$16	$15	3			7
8		BNEZ	$16	LAST				8
9		ANDI	$3	$2	15			9
10		SEQI	$5	$3	1			10
11		BNEZ	$5	SKIP1				11
12		SEQI	$4	$3	2			12
13		BNEZ	$4	SKIP1				13
14		SEQI	$5	$3	3			14
15		BNEZ	$5	SKIP3				15
16		SEQI	$5	$3	4			16
17		BNEZ	$5	SKIP3				17
18		SEQI	$5	$3	5			18
19		BNEZ	$5	SKIP5				19
20		SEQI	$5	$3	6			20
21		BNEZ	$5	SKIP5				21
22		SEQI	$5	$3	7			22
23		BNEZ	$5	SKIP7				23
24		SEQI	$5	$3	8			24
25		BNEZ	$5	SKIP8				25
26		SEQI	$5	$3	9			26
27		BNEZ	$5	SKIP6				27
28		SEQI	$5	$3	10			28
29		BNEZ	$5	SKIP6				29
30		SEQI	$5	$3	11			30
31		BNEZ	$5	SKIP4				31
32		SEQI	$5	$3	12			32
33		BNEZ	$5	SKIP4				33
34		SEQI	$5	$3	13			34
35		BNEZ	$5	SKIP2				35
36		SEQI	$5	$3	14			36
37		BNEZ	$5	SKIP2				37
38		SRL	$2	$2	3		38,39	
39		JUMP	LOOP					
6	LOOP:	ADDI	$15	$15	1			6
7		SEQI	$16	$15	3			7
8		BNEZ	$16	LAST				8
76	LAST:	SRL	$2	$2	1			76
77		ST	$2	12[$0]				77
78		HALT						78
						合計	5	64

図 19. 3 次 Booth のプログラムを用いた 2×（-3）の 2ALU 演算結果（連鎖なし）