1995年6月

SFL / PARTHENONによる 教育用マイクロプロセッサの設計と試作

西田 浩一　　佐藤 政生

早稲田大学 理工学部 電子通信学科

1　はじめに

工学系の学問を履修している者は物を作れるようでなくてはその存在意義が失われる．たとえば，電気情報系の者であれば，一般社会に普及している電子情報機器がどのような素子で構成され，どのような仕組みで稼働するかを知っているべきであり、決してブラックボックスとしてとらえるべきではない．ブラックボックスとしてとらえた瞬間に技術の伝承はなくなり、進歩が中断するからである．現代の電子情報機器は通常、複数のLSI(大規模集積回路)がプリント基板上で接続された形態を取る．現代の機器にはLSIは不可欠なものとなってきた．LSIの集積度の向上にともない，LSIに搭載可能なトランジスタ数は数百万を越えるようになり，もはや手作業では設計ができない程，大規模な回路を実装できるようになった．したがって，LSIを設計する際には，論理回路，電子回路の知識はもちろんであるが，計算機上の設計支援ソフトウェア(CAD: Computer-Aided Design)の利用技術も必要である．

本稿では，大学におけるLSI設計教育の一環として試作した，教育用マイクロプロセッサの設計，および動作評価について述べる．

表 1: レジスタ構成

Register Name	Symbol	Size
General Purpose Registers	R0 - R7	16 bit
Program Counter	PC	16 bit
Instruction Registers	IR, IR2	16 bit
Memory Address Register	MAR	16 bit
Output Register	OPT	16 bit
Zero Flag	ZF	1 bit
Carry Flag	CF	1 bit
oVerflow Flag	VF	1 bit
Negative Flag	NF	1 bit

2　マイクロプロセッサの仕様

2.1　レジスタ構成

表1に，本マイクロプロセッサのレジスタ構成を示す．汎用レジスタは8本あり，R7はサブルーチン・コール命令のリターンアドレス保存に暗黙的に用いられる．プログラムカウンタ(PC)は16ビットであり，データアクセスも16ビット単位で行うため，メモリ空間は64kワード(128kバイト)となる．2段パイプライン処理を実現するため，インストラクションレジスタは2つある．メモリアドレスレジスタ(MAR)は，メモリアクセスの際のアドレス計算用のレジスタであり，アウトプットレジスタ(OPT)は，出力命令(OUT)のデータラッチ用のレジスタである．その他，演算命令で変化する4つのフラグ用レジスタ(ZF，CF，VF，NF)がある．

2.2　インストラクションセット

表2に，本マイクロプロセッサのインストラクションセットでサポートしているアドレッシングモードを示す．Rnはレジスタの値を表し，d(Rn)はアドレスRn+dのメモリの値を表す．#dは即値dを表し，Adrは絶対アドレスAdrのメモリの値を表す．アドレッシングモードは4つあるため，2ビットで表すことができる．本マイクロプロセッサではロードストア・アーキテクチャを採用したため，メモリをアクセスするアドレッシングモード(IndexとAbsolute)はロード命令とストア命令のみがサポートしている．

表3~表8に，本マイクロプロセッサのインストラクションセットを示す．

表3において，HLTはマイクロプロセッサの動作を停止する命令，NOPは何もしない命令，SCFはキャリーフラグ(CF)をセットする命令，RCFはキャリーフラグをリセットする命令である．

表 2: アドレッシングモード

Mode	Symbol	Code
Register	Rn	00
Index	d(Rn)	01
Immediate	#d	10
Absolute	Adr	11

表 3: システム命令

Symbol	Code
`HLT`	`000---00------00`
`NOP`	`000---00------01`
`SCF`	`000---00------10`
`RCF`	`000---00------11`

表 4: ロード ・ストア命令

Symbol	Code1	Code2
`LD  Rd, Rs`	`001 Rd[3] 00 Rs[3] 00000`	
`LD  Rd, d(Rs)`	`001 Rd[3] 01 Rs[3]  d[5]`	
`LD  Rd, #d`	`001 Rd[3] 10    d[8]`	
`LD  Rd, Adr`	`001 Rd[3] 11  --------`	Adr[16]
`LHI Rd, #d`	`010 Rd[3] 10    d[8]`	
`STA Rs, d(Rd)`	`011 Rs[3] 01 Rd[3]  d[5]`	
`STA Rs, Adr`	`011 Rs[3] 11  --------`	Adr[16]

表 5: 演算命令

Symbol	Code
`ADD Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 000`
`ADC Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 001`
`SUB Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 010`
`SBC Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 011`
`CMP Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 100`
`AND Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 101`
`OR  Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 110`
`XOR Rd, Rs`	`100 Rd[3] 00 Rs[3] -- 111`
`ADD Rd, #d`	`100 Rd[3] 10   d[5]   000`
`ADC Rd, #d`	`100 Rd[3] 10   d[5]   001`
`SUB Rd, #d`	`100 Rd[3] 10   d[5]   010`
`SBC Rd, #d`	`100 Rd[3] 10   d[5]   011`
`CMP Rd, #d`	`100 Rd[3] 10   d[5]   100`
`AND Rd, #d`	`100 Rd[3] 10   d[5]   101`
`OR  Rd, #d`	`100 Rd[3] 10   d[5]   110`
`XOR Rd, #d`	`100 Rd[3] 10   d[5]   111`

表 4に，ロード ・ストア命令を示す．表において，LD は汎用レジスタにデータをロードする命令，LHI は 8 ビットの即値を汎用レジスタの上位 8 ビットにロードする命令，STA は汎用レジスタのデータをメモリにストアする命令である．

表 5に，演算命令を示す．表において，ADD は加算命令，ADC はキャリーフラグを含めた加算命令，SUB は減算命令，SBC はキャリーフラグを含めた減算命令，CMP はフラグレジスタのみを変化させる減算命令，AND は論理積をとる命令，OR は論理和をとる命令，XOR は排他的論理和をとる命令である．これらの演算命令は実行後，演算結果に応じてフラグレジスタ ZF，VF，CF，NF が変化する．

表 6に，シフト ・ローテート命令を示す．各シフト・ローテート命令の動作を図 1に示す．

表 7に，ブランチ ・コール命令を示す．表において，Bx (x は分岐条件）はブランチ命令，Cx はコール命令である．分岐条件には，常に分岐する A，フラグ ZF が 1 ならば分岐する Z，フラグ ZF が 0 ならば分岐する NZ，フラグ CF が 1 ならば分岐する C，フラグ CF が 0 ならば分岐する NC，フラグ VF が 1 ならば分岐する V，フラグ VF が 0 ならば分岐する NV，フラグ NF が 1 ならば分岐する N，フラグ NF が 0 ならば分岐する NN がある．Cx 命令では，現在のプログラムカウンタ (PC）の値を汎用レジスタ R7 に代入し，分岐する．RET 命令は，R7 の値を PC に代入しサブルーチンからリターンする命令である．

表 8に，入出力命令を示す．表において，IN は汎用レジスタに INP 端子の値を代入する命令，OUT は汎用レジスタの値をアウトプットレジスタ (OPT) に代入する命令である．OPT の内容はつねに OUT 端子に出力されている．

各命令の実行クロック数は，Index アドレッシングモードの命令が 2 クロック，Absolute アドレッシングモードの命令が 3 クロック，分岐命令，出力命令は 2 クロック，その他の命令は全て 1 クロックである．VerilogXL による論理シミュレーションで，クロック周波数 10MHz での動作を確認した．

2.3 アーキテクチャ

図 2 にアーキテクチャを示す．本マイクロプロセッサでは近年の RISC プロセッサの思想を反映し，ロードストア ・アーキテクチャ[1]，レジスタファイルを採用し，命令のフェッチ，実行を並列に行いレイテンシを小さくするために，簡単な 2 段パイプラインを採用している．本アーキテクチャではプログラムカウンタ専用のインクリメンタ (+1) を用意したため，プログラムカウンタのインクリメント，IR への命令の読み込み，ALU や SHIFTER による汎用レジスタ間演算を並列に行うことができる．マイクロプロセッサの

表 6: シフト・ローテート命令

Symbol	Code
SRA Rd	101 Rd[3] 00 ----- 000
SLA Rd	101 Rd[3] 00 ----- 001
SRL Rd	101 Rd[3] 00 ----- 010
SLL Rd	101 Rd[3] 00 ----- 011
RRA Rd	101 Rd[3] 00 ----- 100
RLA Rd	101 Rd[3] 00 ----- 101
RRL Rd	101 Rd[3] 00 ----- 110
RLL Rd	101 Rd[3] 00 ----- 111

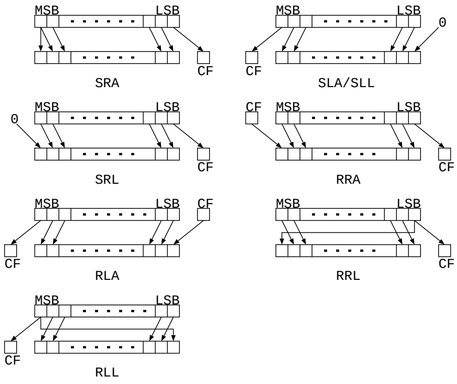

図 1: シフト・ローテート命令の動作

プログラムではループが多用されるため，ほとんどの条件分岐命令は条件が成立する．そこでインクリメンタ(+1)の前にマルチプレクサ(Mux)を追加し，条件が成立する場合の条件分岐命令の実行クロック数を3から2に縮めている．条件が成立する場合，1クロック目でメモリからMARに分岐アドレスを読み込み，2クロック目でインクリメンタによりMAR+1をPCに代入すると同時にMARの示すアドレスからIRに次の命令を読み込む．バスの代わりにマルチプレクサ(S1，S3)を用いているが，これはNTTエレクトロニクス・テクノロジー（株）(以下，NEL)がPARTHENONでバスを合成した実績がないためであり，安全のためマルチプレクサで代用した．今回のマイクロプロセッサではピン数に余裕があったため，バスクロックが必要になる双方向端子を用いずに，全て単方向端子を用いた．

2.4 ハードウェア

本マイクロプロセッサのSFLによる記述は約500行である．PARTHENONによる論理合成の結果，2入力NAND換算で5391ゲートとなった．

表 7: ブランチ・コール命令

Symbol	Code1	Code2
BA ADR	110 --- 11 --- 0 0000	Adr[16]
BZ ADR	110 --- 11 --- 0 0001	Adr[16]
BNZ ADR	110 --- 11 --- 0 0010	Adr[16]
BC ADR	110 --- 11 --- 0 0011	Adr[16]
BNC ADR	110 --- 11 --- 0 0100	Adr[16]
BV ADR	110 --- 11 --- 0 0101	Adr[16]
BNV ADR	110 --- 11 --- 0 0110	Adr[16]
BN ADR	110 --- 11 --- 0 0111	Adr[16]
BNN ADR	110 --- 11 --- 0 1000	Adr[16]
CA ADR	110 --- 11 --- 1 0000	Adr[16]
CZ ADR	110 --- 11 --- 1 0001	Adr[16]
CNZ ADR	110 --- 11 --- 1 0010	Adr[16]
CC ADR	110 --- 11 --- 1 0011	Adr[16]
CNC ADR	110 --- 11 --- 1 0100	Adr[16]
CV ADR	110 --- 11 --- 1 0101	Adr[16]
CNV ADR	110 --- 11 --- 1 0110	Adr[16]
CN ADR	110 --- 11 --- 1 0111	Adr[16]
CNN ADR	110 --- 11 --- 1 1000	Adr[16]
RET	110 --- 00 --------	

表 8: 入出力命令

Symbol	Code
IN Rd	111 Rd[3] 00 ------- 0
OUT Rs	111 Rs[3] 00 ------- 1

本マイクロプロセッサのパッケージは，177ピンPGAのセラミックパッケージである．図3に，マイクロプロセッサの外観を示す．プロセステクノロジは$0.5\mu m$のBiCMOSゲートアレイである．

表9に，入出力ピンを示す．P_RESETピンをアクティブにすると内部のプログラムカウンタがクリアされ，待機状態になる．次にSTARTをアクティブにすると，動作を開始する．本チップでは全ての出力ピンにTTL3ステートバッファが接続されており，$\overline{\mathrm{CT1}}$ 〜 $\overline{\mathrm{CT3}}$で出力／ハイ・インピーダンスの制御を行う．3ステートバッファの出力は，$50k\Omega$の抵抗でプルダウンされている．全ての入力ピンにはTTL入力バッファが接続されており，$50k\Omega$の抵抗でプルアップされている．

本マイクロプロセッサを動作させるための基本周辺回路を図4に示す．待機状態で，STARTを1クロックの間だけアクティブにするため，Dフリップフロップによるシフトレジスタを用いる．PARTHENONで論理合成した順序回路は，SFLのfinish命令によりタスクを終了すると待機状態に戻り，待機状態でSTARTをアクティブにすると，現在の状態から動作を再開す

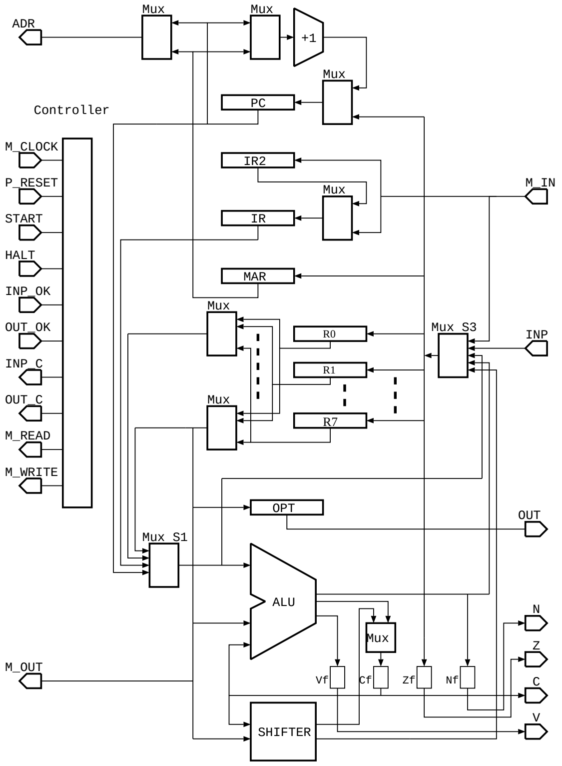

図 2: アーキテクチャ

る．本マイクロプロセッサでは，HLT 命令を実行すると，finishi し，待機状態になる．したがって，プログラムの実行時間より長い間 START をアクティブにすると，HLT 命令の次のアドレスの命令から実行を再開してしまう．そのため，START に直接スイッチを接続すると，誤動作の原因となる．本マイクロプロセッサでは，メモリデータ入力ピンとメモリデータ出力ピンが分かれているが，多くの SRAM ではデータの入出力ピンが共通になっている．そのため 3 ステートバッファを用いて，M_WRITE 信号がアクティブになっている時だけメモリデータ出力ピンから SRAM にデータが送られるようにする．

3　マイクロプロセッサの設計

3.1　設計の流れと設計方法

図 5に，本マイクロプロセッサ設計の流れを示す．論理合成から先の工程は NEL にて行った．

インストラクションセットおよびアーキテクチャの設計

インストラクションセット設計では，マイクロプロセッサで実行される命令および命令コードを決定する．アーキテクチャ設計では，マイクロプロセッサのレジスタ構成，ALU 等の演算器の仕様，およびそれら各機能素子の接続関係，入出力等を決定する．インストラクションセット設計とアーキテクチャ設計は互いに密に関連しているので同時に行う必要がある．

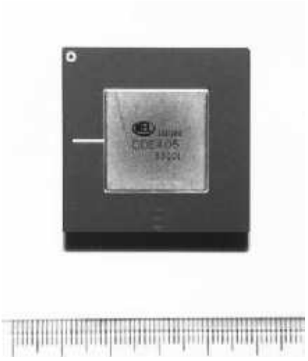

図 3: マイクロプロセッサの外観

表 9: マイクロプロセッサの入出力ピン

ピン名	I/O	機能
VDD	I	+5V 電源
GND	I	グランド
M_CLOCK	I	クロック入力
$\overline{\mathrm{P\_RESET}}$	I	パワーオン・リセット
START	I	動作開始
HALT	I	動作停止
ADR[16]	O	メモリアドレス出力
M_IN[16]	I	メモリデータ入力
M_OUT[16]	O	メモリデータ出力
M_READ	O	メモリリード発生
M_WRITE	O	メモリライト発生
INP[16]	I	入力ポート
OUT[16]	O	出力ポート
INP_C	O	ポート入力発生
OUT_C	O	ポート出力発生
INP_OK	I	ポート入力データセット完了
OUT_OK	I	ポート出力データ読み込み完了
N	O	N フラグ出力（テスト用）
Z	O	Z フラグ出力（テスト用）
C	O	C フラグ出力（テスト用）
V	O	V フラグ出力（テスト用）
$\overline{\mathrm{CT1}}$	I	ADR/M_OUT/M_READ /M_WRITE 3 ステート制御
$\overline{\mathrm{CT2}}$	I	OUT/INP_C/OUT_C 3 ステート制御
$\overline{\mathrm{CT3}}$	I	N/Z/C/V 3 ステート制御

動作記述

今回は LSI 設計支援システム PARTHENON を使用することができたため，ハードウェア記述言語 (SFL) により動作の記述を行った．SFL の記述例を図 6に示す．この記述は Index アドレッシングモード時の 1 クロック目の記述であり，以下の動作を並列に行っている．

- 端子 s1 に，インストラクションレジスタのインデックス値を接続する．
- 端子 s2 に，レジスタファイルから読み込んだアドレス値を接続する．
- 端子 s3 に，alu で計算した端子 s1 と端子 s2 の和を接続する．
- メモリアドレスレジスタ (MAR) に端子 s3 の値を代入する．
- プログラムカウンタ (PC) をインクリメンタによりインクリメントする．
- メモリからインストラクションレジスタ 2(IR2) に次命令を読み込む．
- exec2 へ状態遷移する．

言語レベル・シミュレーション

PARTHENON は，動作記述言語レベルのシミュレータ (SECONDS) をサポートしている．そのためマイクロプロセッサの各命令の動作確認を SECONDS により行った．今回は，SECONDS の対話処理により，命令の動作を確認しながら SFL の記述を行った．言語レベルでのシミュレーションにより致命的なバグ（設計ミス）を回避することが容易になる．

論理合成

SFL による動作記述は，PARTHENON の論理合成によってゲートレベルのネットリスト (nld) に自動変換される．セルライブラリは，NEL 提供の MSC5000 シリーズ用 PARTHENON セルライブラリを使用した．NEL 提供のネットリスト変換/設計ルール検証ツール STC により，設計検証を行うと同時に，nld ファイルを MSC5000 シリーズ用セルライブラリ対応の Verilog-HDL ネットリストに変換した．

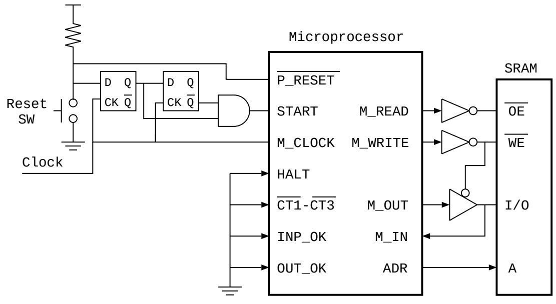

図 4: マイクロプロセッサの基本周辺回路

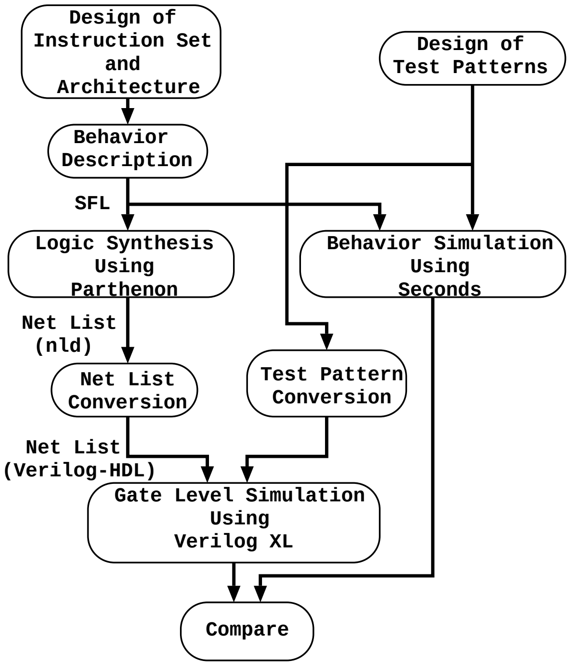

図 5: 設計の流れ

```
state exec1 any {
  ir<9:8> == IDX_A : par {
    s1 = 0b00000000000 || ir<4:0>;
    s2 = reg_file.do(0b1, 0b0,
      0x0000, 0b000, ir<7:5>, 0b000).b;
    s3 = alu.do(s1, s2, 0b0, ADD).s;
    mar := s3;
    pc := inc.do(pc).out;
    ir2 := m_read(pc).m_in;
    goto exec2;
  }
```

図 6: SFL 記述（一部）

TTL I/O バッファの取り付け

試作したチップは内部は 3.3V で動作するため，入出力端子に TTL I/O バッファを取り付ける必要がある．Verilog-HDL ネットリストに対し，全ての入出力端子に TTLI/O バッファを手作業で取り付けた．

ゲートレベル・シミュレーション

設計した LSI を所望のクロック周波数で確実に動作させるために，VerilogXL によりゲートレベルのシミュレーションを行った．全ての命令，アドレッシン

グモード，機能を網羅し，レジスタ，データの可能な限りの組み合わせを考慮した約 31000 パタンの SECONDS 用入力パタンを C 言語により自動生成し，NEL 提供のツールで VerilogXL の入力パタンに変換した．SECONDS によるシミュレーション結果と VerilogXL によるゲートレベル・シミュレーション結果を自作のツールで比較することによりゲートレベルの動作確認を行った．VerilogXL の結果が SECONDS の結果と一致しなかった場合は，クロック周波数や出力値のサンプリング位置を変更した．最終的に，クロック周波数を 10MHz，出力値のサンプリングをクロックサイクルの最初から 60%の位置とし，最小遅延，標準遅延，最大遅延のそれぞれの場合について，動作を確認した．基本的には各命令を実行し，結果を OUT 命令で OUT 端子に出力することにより命令動作の確認を行っている．チップ完成後のリーク電流テストのために，数パタンのテストパタンを作成した．

レイアウトおよび LSI の製造

レイアウトおよび LSI の製造は NEL が行った．

3.2 設計者の身分，人数，役割

設計者は，理工学部電子通信学科・修士課程 1 年（設計当時）であり，1 人で全ての設計を行った．

3.3 設計工程の工数

インストラクションセットおよびアーキテクチャの設計に約 50 時間，SFL 記述および論理合成に約 75 時間，テストパタン生成とシミュレーションに約 100 時間を要した．

4 マイクロプロセッサの動作確認

試作したマイクロプロセッサの動作を確認するために，ホストコンピュータと RS232C で接続する評価用ボードを製作した．簡単なテストプログラムを作成し，ボード上での動作確認を行った．

4.1 評価用ボードの設計・製作

評価用ボードではメインメモリに対し，外部からプログラムやデータを読み書きできることが望ましい．本評価用ボードでは，ホストコンピュータと RS232C シリアルインターフェイスで接続することにより，ホストコンピュータからボード上のメモリに対し，プログラムやデータを読み書きすることができる．RS232C による通信の制御には市販の Z80 ワンチップ・マイコンキットを使用し，マイコンキットのパラレル入出力ポートに，ボード上のメインメモリを接続する．図 7 に，製作した評価用ボードの外観を示す．

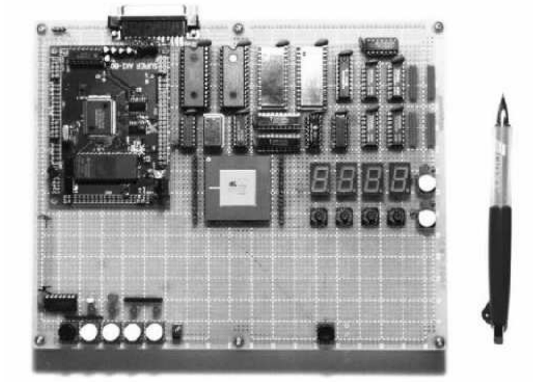

図 7: マイクロプロセッサの評価用ボード

```
0010001000000000B       ;START: LD  R0,#0
0010011000000001B       ;       LD  R1,#1
1110100000000000B       ;       IN  R2
1100001100010000B,0008H ;       CA  COUNT
1110000000000001B       ;       OUT R0
1100001100000000B,0000H ;       BA  START
1000000000100000B       ;COUNT: ADD R0,R1
1000011000001000B       ;       ADD R1,#1
1000101000001010B       ;       SUB R2,#1
1100001100000010B,0008H ;       BNZ COUNT
1100000000000000B       ;       RET
```

図 8: 1 から n までの和を求めるプログラム

4.2 マイクロプロセッサの動作報告

本マイクロプロセッサの動作を確認するため，2 つの簡単なテストプログラムを作成した．図 8は 1 から n（n は INP 端子から入力）までの和を求めるプログラムである．n=100 を入力した結果，13bah(5050) が出力された．図 9は 100 個の素数を求めるプログラムである．実行結果を図 10に示す．

各プログラムは，クロック周波数 625kHz, 1.25MHz, 2.5MHz, 4MHz で安定に動作したが，5MHz で誤動作した．SRAM，TTL による 3 ステートバッファおよび，NAND ゲートの遅延を考慮して計算すると最高 5.7MHz 程度で動作するはずであるが，ラッピング配線による容量やマイコンキットの I/O ポートの容量，クロックの波形の乱れなどが影響したものと考えられる．チップ単体では，NEL により 10MHz の動作が確認されている．

```
0011001001100100B       ;       LD  R4,#100
0010001000000010B       ;       LD  R0,#2
0110001100000000B,1000H ;       STA R0,1000H
0010111000000001B       ;       LD  R3,#1
0010101000000001B       ;       LD  R2,#1
1000100010000100B       ;LOOP1: CMP R2,R4
1100001100000100B,0024H ;       BNC EXIT2
1000111000010000B       ;       ADD R3,#2
0010011000000000B       ;       LD  R1,#0
1000010001000100B       ;LOOP2: CMP R1,R2
1100001100000100B,001AH ;       BNC EXIT1
0011010001100000B       ;       LD  R5,R3
0010001000000000B       ;       LD  R0,#00H
0100001000010000B       ;       LH  R0,#10H
1000000000100000B       ;       ADD R0,R1
0011100100000000B       ;       LD  R6,0(R0)
1100001100010000B,0025H ;       CA  WARI
1100001100000011B,001AH ;       BC  EXIT1
1000011000001000B       ;       ADD R1,#1
1100001100000000B,000BH ;       BA  LOOP2
1000010001000100B       ;EXIT1: CMP R1,R2
1100001100000010B,0006H ;       BNZ LOOP1
0010001000000000B       ;       LD  R0,#00H
0100001000010000B       ;       LH  R0,#10H
1000000001000000B       ;       ADD R0,R2
0110110100000000B       ;       STA R3,0(R0)
1000101000001000B       ;       ADD R2,#1
1100001100000000B,0006H ;       BA  LOOP1
0000000000000000B       ;EXIT2: HLT
1001010011000010B       ;WARI:  SUB R5,R6
1100001100000001B,002CH ;       BZ  EXIT3
1100001100000011B,002EH ;       BC  EXIT4
1100001100000000B,0025H ;       BA  WARI
0000000000000010B       ;EXIT3: SCF
1100000000000000B       ;       RET
0000000000000011B       ;EXIT4: RCF
1100000000000000B       ;       RET
```

図 **9: 100** 個の素数を求めるプログラム

```
0002 0003 0005 0007 000b 000d 0011 0013
0017 001d 001f 0025 0029 002b 002f 0035
003b 003d 0043 0047 0049 004f 0053 0059
0061 0065 0067 006b 006d 0071 007f 0083
0089 008b 0095 0097 009d 00a3 00a7 00ad
00b3 00b5 00bf 00c1 00c5 00c7 00d3 00df
00e3 00e5 00e9 00ef 00f1 00fb 0101 0107
010d 010f 0115 0119 011b 0125 0133 0137
0139 013d 014b 0151 015b 015d 0161 0167
016f 0175 017b 017f 0185 018d 0191 0199
01a3 01a5 01af 01b1 01b7 01bb 01c1 01c9
01cd 01cf 01d3 01df 01e7 01eb 01f3 01f7
01fd 0209 020b 021d
```

図 **10: 100** 個の素数を求めるプログラムの実行結果

5　おわりに

本稿では，大学における LSI 設計教育の一環として試作した，教育用マイクロプロセッサの設計，および動作評価について述べた．

今回は，研究室に設計環境を整備するための十分な時間がなかったため，論理合成以降は NEL に通う必要があったが，余裕のある設計を行うためには，あらかじめ環境の整備を行っておくことが不可欠であると思われる．特に PARTHENON で設計を行う場合，ネットリスト変換，VerilogXL による論理シミュレーションのための環境は準備しておいた方が良い．論理シミュレーションを行う場合，細かいタイミング設定が必要になるため，SFL 記述と PARTHENON により合成される回路の対応を良く調べておくことが必要である．その際，大きな回路ではわかりにくいので，数ビット程度の入出力，レジスタを用いる簡単な有限状態遷移マシン等をいくつか記述し，論理合成して回路を確認してみるのが良いと思われる．PARTHENON で設計を行う場合でも，I/O バッファの取付時に Verilog-HDL のネットリストを変更する必要があるため，Verilog-HDL のネットリスト記述法は知っておいた方が良い．

参考文献

[1] Jhon L. Hennessy, David A. Patterson, コンピュータ ·アーキテクチャ 設計 ·実現 ·評価の定量的アプローチ, 日経 BP 社, 1992.