学生向けコンテスト「LSIデザインコンテストin沖縄2009」課題

マイクロプロセッサの設計

デバイスの記事　ビギナーズ

和田知久

学生を対象とした設計コンテスト「LSIデザインコンテストin沖縄」も早いもので，今回で12年目を迎えることになった．1998年に第1回が開催された時は，琉球大学工学部情報工学科内部の学生コンテストでしたが，回を重ねるごとに，学外，国外と参加者の枠が広がっている．ここ数年は，毎回約100名の学生の参加がある．3月に沖縄で開催される発表会では10件前後の優秀設計が発表され，みんなで大いに盛り上がっている．ただ，最近の傾向としてディジタル通信関連の課題にやや偏り，内容も高度になってきた．そこで，今回は初心に返って少し一般的な内容として，簡単なマイクロプロセッサを設計することにした．（筆者）

今回の設計コンテストの基本課題は，たった9個の命令を実行できるマイクロプロセッサです．何かプログラムが処理できないと面白くないので，数値列を大きさの順で並べ替えるソーティング（具体的にはバブル・ソート）が実行できる程度の命令を含んでいます．従って，高級プログラム言語のforループを実現できるように，無条件分岐命令と，条件付き分岐（ブランチ）命令も含んでいます．

本稿では，マイクロプロセッサをよく知らない人でも課題を設計できるように，アーキテクチャを丁寧に説明します．マイクロプロセッサという言葉を恐れずに，挑戦していただきたいと思います．

コンテストではHDL（VHDLもしくはVerilog HDL）による設計と論理合成を行います．FPGA向けに無償で提供されている設計ツールでも参加できます．HDL設計に興味のある学生はどしどし参加してください．また，余裕のある方はFPGA（Field Programmable Gate Array）などに実装して動作させれば，その努力を認めて高い評価が得られると思います．

1. Small RISC Processor（SRP）のアーキテクチャ

図1に今回の課題であるSmall RISC Processor（SRP）のアーキテクチャを示します．今回のSRPアーキテクチャは独自のものではありません．マイクロプロセッサの書籍としてはバイブル的なJohn L. HennessyとDavid A. Pattersonの“Computer Organization and Design: The Hardware/Software Interface”(1)，(2)に登場するRISCコンピュータの命令を9個に削減したものです．かなりの方が見たことのある構成だと思います．RISCマシンなので，32ワードのレジスタ・ファイルを含んでいます．

図1に示したSRPアーキテクチャはSRP本体と，二つのメモリから構成されています．今回課題として設計するのはSRPです．これをコンピュータとして動作させるには，命令とデータが必要です．ここでは命令を蓄える命令メモリと，データを蓄えるデータ・メモリを二つの独立したメモリで構成しています．その理由は，この方がメモリの使い方が単純化され，プロセッサの設計が単純になるからです．

● 命令実行の動作

命令メモリには32ビット（ここではワードと呼ぶ）の大

Keyword　コンテスト，VHDL，Verilog HDL，マイクロプロセッサ，Small RISC Processor，命令セット，メモリ・アドレッシング，バブル・ソート

きさの命令が多数蓄えられています．マイクロプロセッサは，命令メモリに蓄えられた命令を一つずつ読み出し，その32ビット命令が示す演算を実行します．その動作を順に説明します．

(1) 命令フェッチ

マイクロプロセッサにはプログラム・カウンタ(PC)というカウント機能があります．このPCの値は命令メモリのアドレス信号に対応しており，そのアドレスが示す場所の命令を読み出します．

(2) 命令デコード

命令フェッチで読みだされた命令の内容に従って，必要な動作をするために命令の解読を実施し，必要な制御を行います．**図1**では制御回路の仕事となります．

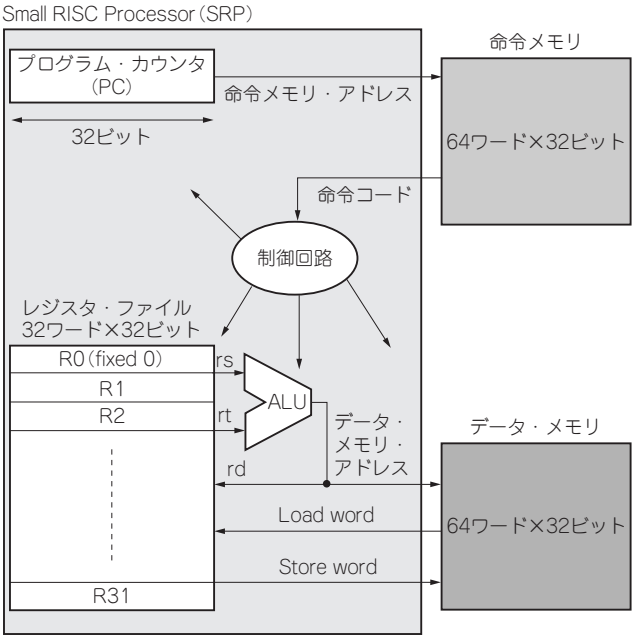

図1　Small RISC Processor SRP アーキテクチャ

John L. Hennessy と David A. Patterson の“Computer Organization and Design：The Hardware/Software Interface”に登場するRISCコンピュータの命令を9個に削減したものである．命令メモリとデータ・メモリは独立した構成にしている．

(3) 演算とデータ・メモリ・アクセス

命令の解読結果に従って，レジスタ・ファイルから必要な値を読み出し，ALU(Arithmetic Logical Unit)にて演算します．必要があれば，データ・メモリに対してアクセスを行い，データを書き込み，もしくは読み出しします．

(4) ライト・バック

(3)の演算結果や，データ・メモリから読み出した値をレジスタ・ファイルに書き込みます．そして，プログラム・カウンタ(PC)の次のサイクルの値を用意します．

(1)～(4)の動作で一つの命令の実行になります．これを繰り返すことで命令を順番に実行することができます．

● 命令セット

表1にSRPがサポートする九つの命令を示します．算術演算，論理演算，データ転送，条件分岐，無条件分岐の5種類に区分されています．

R1，R2のような記号はレジスタ・ファイルのアドレスを示しています．Rnではアドレスはnです．SRPでは32ビット・レジスタを32個搭載しており，R0～R31が存在します[注1]．

●算術・論理演算(add，sub，and，or)命令

表1の中に，R1< = R2 + R3 というような表記があります．これは，レジスタ・ファイルのアドレス2番地に保持されている値とアドレス3番地に保持されている値をALUで加算し，アドレス1番に書き戻すということを意味しています．算術・論理演算として，+，−，and，orをサポートしています．

●lw命令

lw(Load Word)命令はデータ・メモリ内の32ビットのデータをレジスタ・ファイルに転送する命令です．

注1：SRPでは，通常のRISCコンピュータに従って，R0を特別なレジスタとしている．書き込みはできず，常にALL0の値を保持している．ALL0，すなわち0は，プログラムでよく使われる数値なので，レジスタ・ファイルの0番地をその特別な数値として使用している．

表1
SRPがサポートする命令

区　分	命　令	アセンブラ例	例の意味	備　考
算術演算	add	add R1, R2, R3	R1 <= R2 + R3	加算
	subtract	sub R1, R2, R3	R1 <= R2 - R3	減算
論理演算	and	and R1, R2, R3	R1 <= R2 and R3	各ビットごとにAND
	or	or R1, R2, R3	R1 <= R2 or R3	各ビットごとにOR
データ転送	load word	lw R1, 100 (R2)	R1 <= メモリ[R2+100]	メモリからレジスタへの転送
	store word	sw R1, 100 (R2)	メモリ[R2+100] <= R1	レジスタからメモリへの転送
条件分岐	branch on equal	beq R1, R2, 25	if (R1=R2) go to PC+4+25*4	等しいときにPC相対分岐
	set on less than	slt R1, R2, R3	if (R2<R3) R1<=1 else R1<=0	
無条件分岐	jump	j 2500	go to 2500*4	絶対アドレス・ジャンプ

```
lw R1, 100(R2)
```

は，レジスタ・ファイルのアドレス2番に保持されている値と命令中に指定された100という整数値の加算をALUで行い，その結果をアドレス値としてデータ・メモリから値を読み出し，その値をレジスタ・ファイルのアドレス1番に書き戻すという意味です．

●sw命令

```
sw R1, 100(R2)
```

で示されるsw(Store Word)命令は，レジスタ・ファイルのアドレス1番から値を読み出し，レジスタ・ファイルのアドレス2番に保持されている値と命令中に指定された100という整数値の加算をALUで行い，その結果をアドレス値としてデータ・メモリに値を書き込みます．lw命令との違いは最初のswかlwかだけですが，データの転送方向は逆方向です．

●beq命令

beq(Branch on Equal)命令は，レジスタ・ファイル内の二つの値が同じかどうか比較して(減算してその答えが0かどうか)，同じであれば分岐します．

プログラム・カウンタ(PC)は，命令メモリ内の実行する命令のアドレス番地を示します．通常分岐がない場合には，命令メモリ内の次番地にある命令が実行されます．コンピュータでは一般的に，8ビット単位(バイト単位)のアドレスが付けられています．ここでは，命令もデータも32ビットを想定しているので，4アドレス分で一つの命令もしくはデータが記憶されています．従って，次の命令の番地は現在実行中の命令のアドレス＋4となります．

```
beq R1,R2,25
```

ではR1レジスタとR2レジスタの値を比較します．同じであれば，次の命令のアドレス(現状のPC＋4)値を変更します．この例では命令中に25という整数がありますが，これは25先の命令に分岐するという意味です(この数値が負の場合は戻ることになる)．これはアドレスを計算すると，PC＋4＋25×4となります．1命令の大きさが4アドレス分なので，4倍が必要なわけです．

図2
バイト・アドレッシング
右の四角は8ビットの記憶ブロックを意味しており，いわゆるバイト・データに対応する．

●slt命令

```
slt R1,R2,R3
```

はSet on Less thanという命令です．R2の値＜R3の値であれば，R1の値を‘1’にセットし，そうでなければ‘0’にリセットします．sltとbeqをうまく使えば多彩な条件分岐が実現できます．

●j命令

条件分岐では，PC＋4の値に対して値を加算(負の数の加算も想定)していました．

```
j 2500
```

はメモリの絶対的なアドレスで0番地から見て，2500個目の命令への分岐を示しています．メモリのアドレスはバイト単位(バイト・アドレッシング)を想定しているので，実際のメモリのアドレス値では4倍の2500×4＝10000番地への無条件の分岐ということになります．

● メモリ・アドレッシング

コンピュータでは一般的なバイト単位のアドレッシングを**図2**に示します．図の右の四角は8ビットの記憶ブロックを意味しており，いわゆるバイト・データに対応します．SRPではバイト・アドレスを用いているので，1バイト進むごとに，アドレス値が1上昇します．

本課題で扱う命令のサイズは4バイト(32ビット)を想定しています．**図2**に示すように横方向の四つのブロックが一つの命令の記憶単位となります．従って命令を数えるにはワード・アドレッシングが便利です(**図3**)．このワード・アドレスの値を4倍すると，4バイト・ブロックの左端バイトのアドレスとなります．

図1に示したSRPアーキテクチャでは二つの外部メモリがあります．これらのメモリは32ビット単位でアドレスが割り当てられており，メモリをアクセスする場合にはワード・アドレスへの変更が必要です．

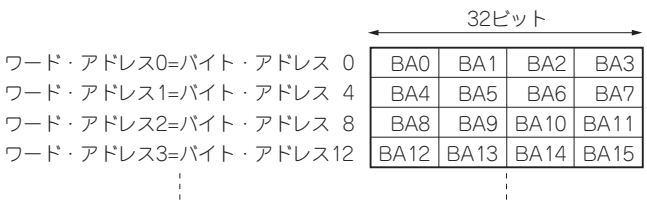

図3　ワード・アドレッシング
命令のサイズは4バイト(32ビット)を想定しているので，横方向の四つのブロックが一つの命令の記憶単位となる．命令を数えるにはワード・アドレッシングが便利．

● 命令フォーマット

これまでの説明で一つの命令が32ビットであることは理解いただけていると思います．命令には算術演算，論理演算，分岐などいろいろな種類があることも説明しました．命令にはレジスタ・ファイルのアドレスや，メモリのアドレスの一部を示す必要もあります．

通常はこのような多くの意味を示すために，32ビットを分割して使用します．その分割された部分をフィールドといいます．このようなビットでの命令の表現をアセンブラ表現，もしくは機械語表現といいます．

まず，32ビットを**表2**のように分割します．九つの命令は三つの形式に分類します．

R形式はレジスタ・アドレスを三つ示す必要のある命令用の形式です．add，sub，and，or，sltがR形式を用います．レジスタは0番～31番の32種類があるので，5ビットを用いてレジスタのアドレスを示しています．また，R形式では左端の6ビットがすべて0であり，これがR形式であることを示しています．add，sub，and，or，sltの区別は右端の6ビットの値で区別されています．また，右から二つめの5ビット・フィールドは用いられていません．

I形式は，二つのレジスタ・アドレスと数値（2の補数表現を用いて正，0，もしくは負の数を表す）を示しています．なるべく大きな数値を示せる方がよいのですが，32ビットの制限で，数値は16ビットです．2の補数表現を用いているので，**表3**が表現可能な数値の範囲となります．この16ビット・フィールドは，lw，swではデータ・メモリ・アドレスの計算（バイト単位）に用いられており，beqでは命令ワードの分岐数を示しています（ワード単位）．

J形式ではレジスタ・アドレスを示す必要がないので，26ビットの数値が用いられています．この数値を4倍すると絶対メモリ・アドレスとなるので，負の数を表現する必要はなく，符号なし数で正または0が表現範囲です．

実際のフィールドの値（10進数表示）と機械語のフィールドの順序が異なることに注意が必要です．各フィールドの名前を**表4**に示します．

opフィールドとfuncフィールドが操作を指定しています．rsはソース・レジスタ，rtも通常はソース・レジスタ，rdはデスティネーション・レジスタと呼びます．

2. SRPの動作

図1で二つのメモリを示しましたが，今回の課題では与えられたプログラムを実行することを想定しています．

● 命令ROMとデータRAMのサイズ

今回は**図4**に示すメモリ構成を想定します．

命令ROMは64ワード×32ビット構成で，256バイトの容量を持ちます．0番地より255番地のアドレスを命令ROMに割り当てます．

データRAMも同じく64ワード×32ビット構成で，256バイトの容量を持ちます．256番地より511番地のアドレスを割り当てます．

512番地以上のアドレスは使用しません．

● 検証用VHDLコードとバブル・ソートのプログラム

SRPをHDL（VHDLまたはVerilog HDL）で実現し，8ワード（ワード＝4バイト）のソーティングを実行します．

バブル・ソートのプログラムが入った命令ROMと初期データが格納されたデータRAMのVHDLモデルは，コンテストのホームページ（http://www.lsi-contest.com/）からダウンロード可能です．

表2　命令フォーマット
各フィールドの値は10進数で示している．

命　令	フォーマット名	例						備　考
		6ビット	5ビット	5ビット	5ビット	5ビット	6ビット	
add	R形式	0	2	3	1	0	32	add R1, R2, R3
sub		0	2	3	1	0	34	sub R1, R2, R3
and		0	2	3	1	0	36	and R1, R2, R3
or		0	2	3	1	0	37	or R1, R2, R3
slt		0	2	3	1	0	42	slt R1, R2, R3
lw	I形式	35	2	1	100			lw R1, 100（R2）
sw		43	2	1	100			sw R1, 100（R2）
beq		4	1	2	25			beq R1, R2, 25
j	J形式	2	2500					j 2500

表3　16ビット（2の補数）の表現範囲

	2進数（2の補数表現）	10進数
最大値	0111 1111 1111 1111	32767
0	0000 0000 0000 0000	0
最小値	1000 0000 0000 0000	− 32768

Reset信号が‘0’に解除された後，プログラム・カウンタPC＝0番地として，実行を開始します．データRAMのデータをモニタし，**図5**のようなシミュレーション波形を確認できれば，バブル・ソートができています．

表5に示すように，256番地から始まる8ワードの配列に対して値の大きい数値を下位のアドレスからソートします．配列の先頭番地は384番地のワードに記憶されています．配列の大きさはバイト単位で388番地に記憶されています．ここでは8×4＝32です．ワードの大きさをバイト単位で表せば4であり，これが392番地に記憶されています．

バブル・ソートのプログラムが納められている命令ROMの内容を**表6**に示します．

● データRAMの動作

データRAMはデータの読み出しに関しては，アドレス入力信号6ビットに対して，32ビット・データを出力する組み合せ回路です．書き込みClock入力の立ち上りエッジでWE信号が‘1’の時に行われます．詳細の動作波形を**図6**に示します．

3．SRPの実現法

SRPの設計アーキテクチャの典型的な例を**図7**に示します．1命令を1クロック・サイクルごとに実現する例です．

命令ROMは，ワード・アドレスを入力すると命令が読み出されるROMを想定しています．SRP内部のレジスタ・ファイルでra1とra2の二つは読み出し用アドレス信号です．このアドレスに従ってdo1とdo2にレジスタの値がそれぞれ読み出されます．waは書き込みアドレス信号であり，書き込むデータはdinに与えられます．write RF＝‘1’のとき，Clockの立ち上がりエッジでデータがレジスタ・ファイルに書き込まれます．

レジスタ・ファイルへのデータの書き込みが必要な命令はadd，sub，and，or，lw，sltです(**表7**)．

ALUはadd，sub，and，orの演算をサポートします．また，演算結果(result)が0であるかどうかを調べるフラグ(zero)を計算します．slt命令では，ALUは減算

表4　フィールド名

フォーマット名	6ビット	5ビット	5ビット	5ビット	5ビット	6ビット
R形式	op	rs	rt	rd	−	func
I形式	op	rs	rt	offset		
J形式	op	address				

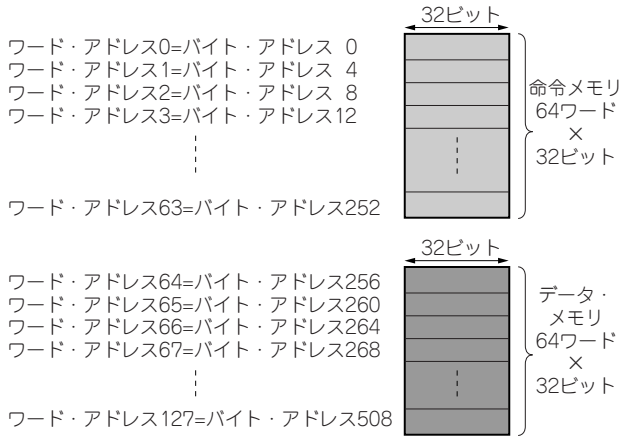

図4　メモリ・アーキテクチャ
命令ROMとデータRAMはそれぞれ64ワード×32ビット構成である．

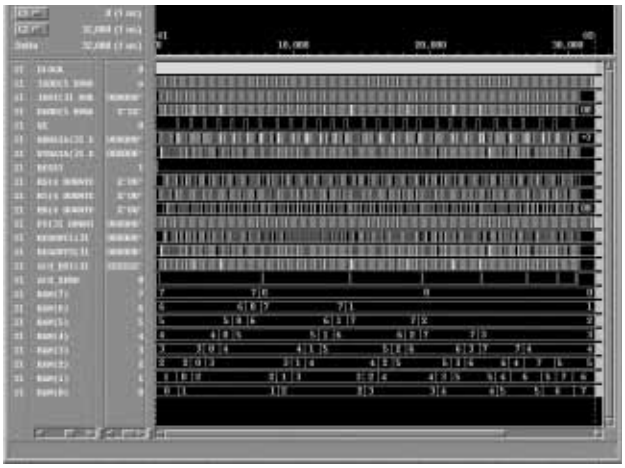

図5　シミュレーション波形
バブル・ソートを実行している様子を示す．

表5　データRAMの内容

		データRAM				ソート後の内容			
バイト・アドレス	データRAMのアドレス入力	00	01	10	11	00	01	10	11
256	0	0				7			
260	1	1				6			
264	2	2				5			
268	3	3				4			
272	4	4				3			
276	5	5				2			
280	6	6				1			
284	7	7				0			
		…							
384	32	256				256			
388	33	32				32			
392	34	4				4			
		…							

表6　命令ROMの内容

バイト・アドレス	命令アドレスのアドレス入力	命令ROMの内容	説　明
0	0	NOP	
4	1	NOP	
8	2	LW R1, 388(R0)	R1 <= 32
12	3	LW R2, 384(R0)	R2 <= 256，先頭番地
16	4	LW R3, 392(R0)	R3 <= 4
20	5	ADD R5, R1, R2	R5 <= 288，最終番地の次
24	6	SUB R5, R5, R3	R5 <= R5 − 4 = 284，最終番地
28	7	ADD R6, R2, R0	R6 <= R2 = 256　転送，R0は0固定
32	8	ADD R7, R6, R0	R7 <= R6，比較するデータ1のアドレス
36	9	ADD R8, R7, R3	R8 <= R7 + 4，比較するデータ2のアドレス
40	10	LW R10, 0(R7)	R10 <= 比較するデータ1
44	11	LW R11, 0(R8)	R11 <= 比較するデータ2
48	12	SLT R9, R10, R11	R10 < R11ならR9 <= 1，違うならR9 <=0
52	13	BEQ R9, R0, +2	R9が0ならばPC=PC+4+2×4=64へ分岐
56	14	SW R10, 0(R8)	比較データ1をスワップしてメモリへ
60	15	SW R11, 0(R7)	比較データ2をスワップしてメモリへ
64	16	ADD R7, R7, R3	R7 <= R7 +4，アレイインデックスを進める
68	17	BEQ R7, R5, +1	インデックスが最終アドレスならLOOP 1を抜ける
72	18	J 9	9番地へジャンプ，LOOP1
76	19	SUB R5, R5, R3	R5 <= R5 − 4，最終番地を下げる．ここに最小値がある．
80	20	BEQ R5, R2, +1	最終番地が先頭番地と一致するとLOOP2を抜ける．
84	21	J 8	8番地へジャンプ，LOOP2
88	22	NOP	

を実行し，zeroフラグを調べるように実装します．

データRAMへの書き込みが必要な命令はsw命令であり，データRAMの読み出しが必要な命令はlwです．

図7において，ブロックに付いている小さな三角の印はクロック入力を意味しています．クロックの立ち上がりエッジで書き込みが発生する順序回路です．三角の印がないほかのブロックはすべて組み合わせ回路で構成できます．

“&”はビット連結，“+”は加算器，“MUX”はマルチプレクサ，“SE”は符号を保存したビット幅の16ビットから32ビットへの拡張回路です．**表8**に拡張の例を示します．16ビット表現でのMSBを16回左側に延ばせば実現できます．

図7の右上の3入力マルチプレクサ（MUX）は分岐に関するものです．3入力の一番上が選択される場合は通常の分岐がない場合であり，PCの値は4ずつインクリメントしています．3入力の中央は無条件分岐に対応しています．

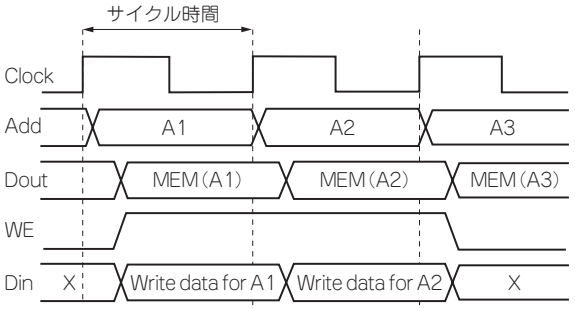

図6　データRAMの動作波形

表7　レジスタ・ファイルへのデータの書き込みが必要な命令と読み出しデータ数

区　分	命　令	レジスタ・ファイルへのデータの書き戻し	レジスタ・ファイルより読み出す値
算術演算	add	1	2
	subtract	1	2
論理演算	and	1	2
	or	1	2
データ転送	load word	1	1
	store word	0	2
条件分岐	branch on equal	0	2
	set on less than	1	2
無条件分岐	jump	0	0

表8　符号を保存したビット幅の16ビットから32ビットへの拡張

項　目	16ビット（2の補数表現）	32ビット（2の補数表現）
最大値	0111 1111 1111 1111	0000 0000 0000 0000 0111 1111 1111 1111
0	0000 0000 0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
最小値	1000 0000 0000 0000	1111 1111 1111 1111 1000 0000 0000 0000

その前方の“&”はビットの接続であり，下記に“00”を連接することにより，4倍を実現しています．また，上位のビットの不足分4ビットは元のPCの上位ビットをコピーしています．3入力の一番下は，条件付き分岐が成立した（TAKEN）場合です．PC + 4 + offset × 4の計算になっています．

図8に，32番地，36番地，40番地を実行中の動作波形を示します．PCの値によって，Iaddが生成され，命令Instが命令ROMより取り出されています．命令の各フィールドの値より，ra1，ra2，waが生成されていることが分かります．

4. 課題

● LEVEL1：基本課題

基本課題では，**図7**に示すような構成で設計します．バブル・ソートを実行し，動作することを確認してください．

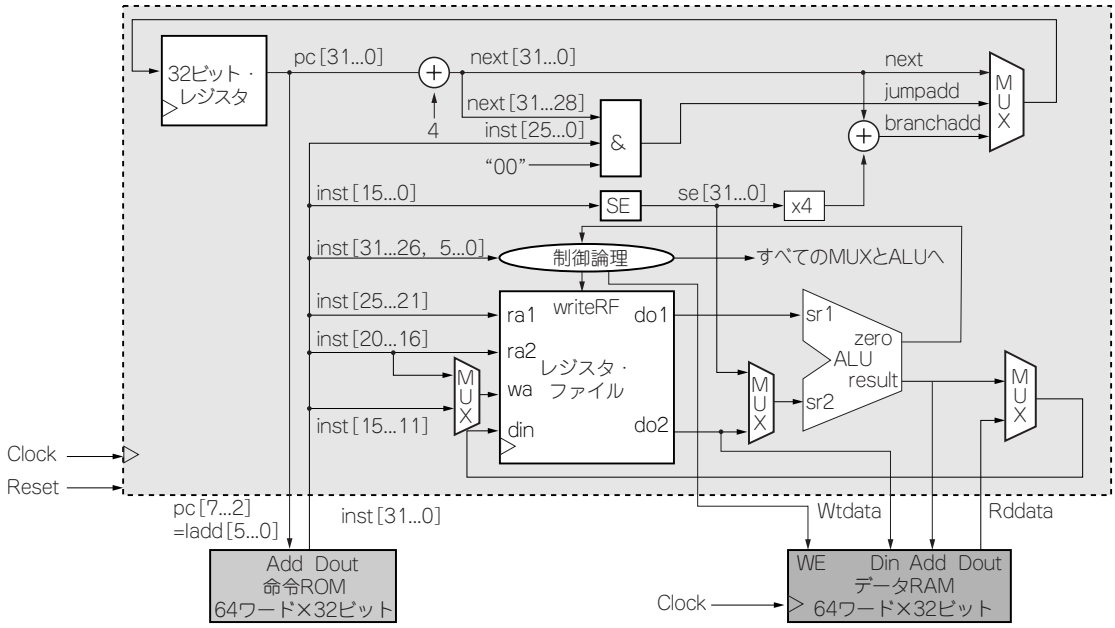

図7
SRPの構成例

SRPの入出力ピンを**表9**に示します.

● LEVEL2：自由課題

自由課題では，SRPを自由に変更・拡張してください．以下に拡張案を示します.

- パイプライン化：パイプライン動作にすることで，サイクル時間を削減し高性能化が可能と思われます.
- 命令の追加など自由な拡張

5. 速度と規模の単位

筆者の勤務する大学では，論理合成用のツールとして米国Synopsys社の「Design Compiler」を使用していますが，このツールは誰もが使えるわけではありません．そこで，

- 2入力XORゲート一つの遅延時間を1UNIT_DELAY
- 2入力XOR ゲート一つの面積を1UNIT_AREA

とします.

具体的には，50入力XOR 回路を合成し，その1段当たりの遅延時間を「単位時間」として，速度の単位とします．50入力XOR 回路のVHDL ソース・コードを**リスト1**に示します(このファイルも本コンテストのWeb サイトからダウンロード可能).

● 速度

Design Compilerでは，6段，49個のXOR回路が合成

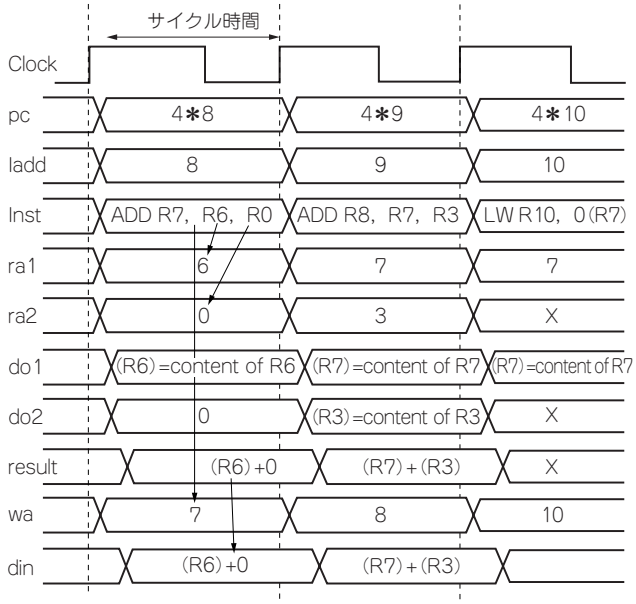

図8　SRPの動作波形図
32番地，36番地，40番地を実行中の動作波形を示す.

表9　基本課題のピン配置

信号名	入出力	ビット幅	説　明
Clock	IN	1	クロック入力
Reset	IN	1	'1' でリセット
Iadd	OUT	6	命令ROMのアドレス，ワード・アドレッシング
Inst	IN	32	32ビット命令
Dadd	OUT	6	データRAMのアドレス，ワード・アドレッシング
WE	OUT	1	'1' でデータRAMの書き込み
Wtdata	OUT	32	データRAMへ書き込みデータ
Rddata	IN	32	データRAMからの読み出しデータ

されます．クリティカル・パス遅延はreport_timingコマンドにより7.17であることが分かります．そこで7.17/6 = 1.195を単位（1UNIT_DELAY）とします．例えば，ある遅延が20ならば，20/1.195 = 17.74UNIT_DELAYということになります．

● 規模

面積はreport_areaコマンドのtotal cell areaにより147.0であることが分かります．XORゲート数は49個なので，147.0/49 = 3.0を単位（1UNIT_AREA）とします．例えば，ある回路面積が200ならば，200/3.0 = 66.67 UNIT_AREAとします．

● RAMやROMを用いる場合の注意

多めのデータを取り扱う回路では，メモリを使用する場合が想定されます．メモリを使う場合も，回路規模に含まれるように合成させてください．具体的には，メモリは合成可能なHDLコードで記述し，フリップフロップなどを用いる回路で実現してください．ROMテーブルを使用する場合も，マクロなどは使用せず，組み合わせ回路で実現してください．

ただし，FPGAのみをターゲットとする場合は，内蔵のメモリ・ブロックを使用しても構いません．その場合は，レポートに分かりやすく明示してください．

6．提出レポートについて

今回の応募レポートは，IEEEの論文スタイルに統一します．http://www.ieee.org/portal/cms_docs_iportals/iportals/publications/journmag/transactions/TRANSJOUR.docを参考にしてください．特に，ユニークな点や技術的に優れた点の説明を，十分かつ簡潔明瞭に記述してください．枚数は，最大で6ページまでとします．

海外からの審査員が理解できるように，できるだけ英語での執筆およびコンテスト当日の発表を勧めます．日本語でも構いませんが，国際化ということで多少の文法誤りは気にせずに，できるだけ英語にチャレンジしてみてください．

レポートはPDF化して，LSI-contest@dsp.cse.kyutech.ac.jpまで提出してください．締め切りは，2009年1月15日午後5時とします．

リスト1　50入力XOR回路のVHDLソース・コード

```
library  IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity PARITY is
    port ( A : in  unsigned(49 downto 0);
           Y : out std_logic );
end PARITY;

architecture RTL of PARITY is
begin

    process(A)
    variable TMP : std_logic;
    begin
        TMP := '0';
        for i in 0 to 49 loop
            TMP := TMP xor A(i);
        end loop;

      Y <= TMP;
    end process;
end RTL;
```

7．審査のポイント

速度や回路規模だけでなく，アーキテクチャのユニークさ，アイデアを十分に考慮して審査します．

LSIデザインコンテスト実行委員会で1次審査を行い，上位10チーム程度を2009年3月に沖縄で開催予定のコンテスト発表会に招待します．そこで，プレゼンテーションによる最終審査を実施し，上位3チームに対しアワードを贈呈します．大学院生，学部学生，高専生のレベルに応じて審査します．

本コンテストは，主催：LSIコンテスト実行委員会，共催：琉球大学工学部情報工学科，株式会社沖縄産業振興センター，九州半導体イノベーション協議会，協賛：ソニーLSIデザインにより実施されています．

参考・引用*文献

(1) David A. Patterson, John L. Hennessy ; Computer Organization and Design: The Hardware/Software Interface, 2nd Edition, Morgan Kaufmann Publisher.

(2) David A. Patterson, John L. Hennessy ; パターソン＆ヘネシー コンピュータの構成と設計―ハードウエアとソフトウエアのインタフェース，第2版，日経BP社.

わだ・ともひさ
琉球大学工学部情報工学科 教授