KUE-CHIP 2 設計ドキュメント

神原弘之（京都高度技術研究所）
越智裕之，澤田宏，浜口清治（京都大学工学部情報工学教室）
岡田和久（京都大学工学部電子工学教室）
上嶋明（立命館大学理工学部情報工学教室）
安浦寛人（九州大学大学院総合理工学研究科）

1993 年 1 月 16 日 (土)

Version 1.10

もくじ

第 1 章

概要

1.1 はじめに

KUE-CHIP2(Kyoto University Education Chip2) は，大学などでの計算機教育のための教材として開発された 8 ビットのマイクロプロセッサである．KUE-CHIP2 は，極めて単純なアーキテクチャを持っており，計算機の基本構造と機能を学習するのに最低限の命令セットを持つ．しかも，内部のレジスタ（プログラムカウンタやアキュームレータ）の内容を外部から観測／制御でき，その機能を理解しやすい構造になっている．大学，高等工業専門学校，専修学校，企業内教育などに利用していただければ幸いである．

なお，KUE-CHIP2 に関するお問い合わせは下記へお願いします．

〒６００　京都市下京区中堂寺南町京都リサーチパーク
京都高度技術研究所
神原　弘之
電話　０７５－３１５－８６５２
電子メール　kanbara@astem.or.jp

又は，

〒８１６　福岡県春日市春日公園６－１
九州大学大学院総合理工学研究科情報システム学専攻
安浦　寛人
電話　０９２－５７３－９６１１
電子メール　yasuura@is.kyushu-u.ac.jp

1.2 ドキュメントの構成

本設計ドキュメントは，KUE-CHIP2 の開発に関する種々の設計資料をまとめたものである．全体のアーキテクチャや外部仕様に関する資料のほか，各部分モジュールの入出力仕様や内部仕様（回路設計）も含まれている．

1.3 KUE-CHIP2の構造

7ページにKUE-CHIP2のブロック図を示す．破線内が集積回路化されている部分である．3種のバス（入力データバスDBi，出力データバスDBo，アドレスバスAB）のほかに，内部状態を観測するための観測バスOBがある．観測バスには，内部のバスやレジスタの内容を出力することができる．観測バスの内容は，ボードのLEDおよび7セグメント表示用LEDで観測することができる．また，内部のレジスタやメモリの内容は，ボードのスイッチで直接書き込みができる．

演算器としては，ALU(Arithmetic Logic Unit）が1個あり，演算用のレジスタとしては，アキュームレータACCとインデックスレジスタIXを持っている．制御系は，プログラムカウンタPC，命令レジスタIR，命令デコーダIDCなどを持っている．また，メモリアクセスの際のアドレスを保持するメモリアドレスレジスタMARがある．各部の働きは以下のとおりである．

アキュームレータ ACC
演算に使用する8ビットのレジスタ．演算のオペランドや演算結果を保持する．

インデックスレジスタ IX
演算に使用する8ビットのレジスタ．演算のオペランドや演算結果を保持する．また，修飾アドレス指定のときのアドレス修飾にも利用される．

フラグ FLAG
桁上げフラグCF，桁あふれフラグVF，ネガティブフラグNF，零フラグZFの4ビットからなる．演算結果に従ってフラグがセット／リセットされる．

演算器 ALU
論理演算，算術演算を行なう．データの演算のほか，アドレスの計算にも利用される．

命令レジスタ IR
実行される命令が保持される．

命令デコーダ IDC
命令レジスタの内容を解読する回路．

制御回路 Cont.
クロック発生回路とともに順序回路を構成し，命令デコーダの出力をもとに計算機の各部へ制御信号を送る．

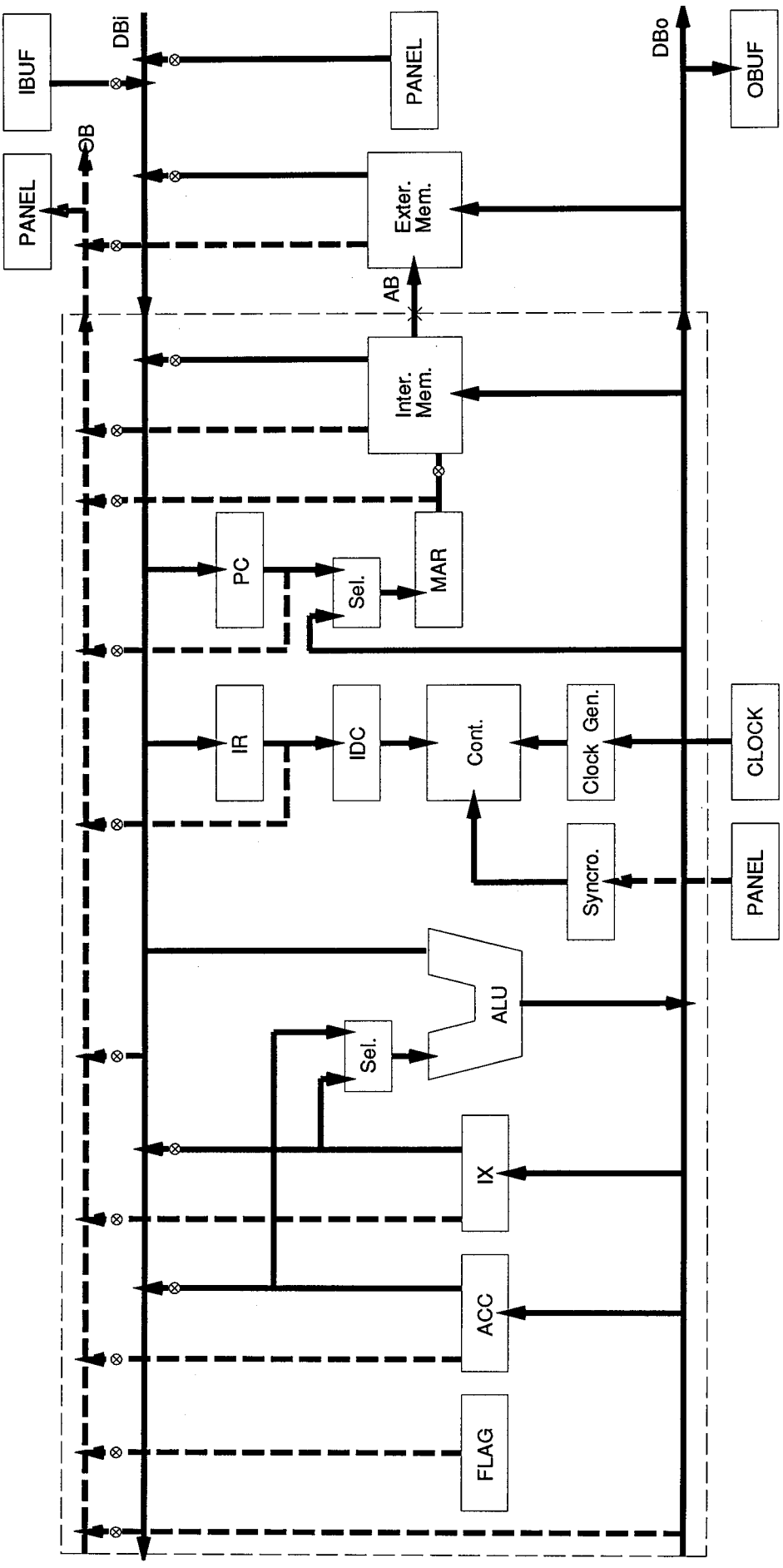
クロック発生回路 Clock Gen.
外部から与えられるクロックからフェーズ信号P0, P1, P2, P3, P4を生成する回路．

シンクロナイザ Syncro.
ボードの制御スイッチ（SI, SP, S/Sなど）からの信号を受理する回路．

プログラムカウンタ PC
次に実行する命令のメモリ上でのアドレスを保持する．

メモリアドレスレジスタ MAR
メモリアクセスのためのアドレスを保持する．ここに保持されたアドレスの内容が読み出しや書き込みの対象となる．

メモリ空間は５１２バイトでバイト単位に指定される．０番地から２５５番地はプログラム領域(program region)とよばれ，プログラムはこの範囲におかれなければならない．これは，プログラムカウンタが8ビットであるため，２５６バイトしか指定できないことによる．２５６番地から５１１番地まではデータ領域(data region)と呼ばれ，プログラムは格納できないが，データを格納する領域としては利用できる．アドレス（9ビット）の最上位ビットは命令デコードの後，コントローラにおいて直接生成される．このビットで，プログラム領域とデータ領域のどちらにアクセスするかが決められる．アドレスの下位8ビットはメモリアドレスレジスタの内容によって指定される．


IBUF
DBi
PANEL
DBo
OBUF
PANEL
OB
Exter. Mem.
AB
Inter. Mem.
PC
Sel.
MAR
IR
IDC
Cont.
Clock Gen.
CLOCK
Syncro.
PANEL
ALU
Sel.
IX
ACC
FLAG

KUE-CHIP2では，チップ内に内部メモリ（Internal Memory）５１２バイトを持っている．チップの外の外部メモリ（External Memory）５１２バイトを利用することもできる．この内部メモリと外部メモリはまったく同等で，ボード上のスイッチでマニュアルで切り替えてどちらでも利用できる．しかし，プログラム中で両者を切り替えて使うことはできない．

KUE-CHIP2の命令語は１バイトあるいは２バイトで構成される（2.2節参照）．命令語の１バイト目は，上位４ビットあるいは５ビット（ビット７からビット４あるいは３）で命令の種類を表す．分岐命令では下位４ビットが分岐条件，シフト命令では下位３ビットがシフトモードを示す．演算命令やロード／ストア命令では，ビット３が演算用レジスタの指定（第１オペランド），下位３ビット（ビット２からビット０）で第２オペランドのアドレスモードを指定する．２語長命令の２語目は，アドレスまたはデータである．データ語は，最上位ビット（第７ビット）が符号に対応し，２の補数表示で負の数を表す．

2.2節に，KUE-CHIP2の命令セットを示す．命令は１９種で，入出力命令２，シフト命令２，算術論理演算命令８，フラグセット命令２，ロード命令，ストア命令，ブランチ命令，NOP命令，停止命令からなる．アドレス指定は４つのモードがあり，レジスタ，即値アドレス，絶対アドレス（プログラム領域Pとデータ領域D），インデックスレジスタによる修飾アドレス（プログラム領域Pとデータ領域D）の指定ができる．即値アドレスでは２語目がそのままデータとなる．絶対アドレスでは２語目がデータが格納されているアドレスを示す（プログラム領域，データ領域のいずれでもよい）．修飾アドレスでは２語目の内容とインデックスレジスタIXの内容を加算して決まる値が第２オペランドのアドレスとなる（この場合もプログラム領域，データ領域のいずれでもよい）．

各命令の簡単な動作は以下のとおりである．

NOP(No OPeration)	何もしない．
HLT(HaLT)	停止命令．
OUT(OUTput)	出力命令．出力ポートへACCの内容を出力する．
IN(INput)	入力命令．入力ポートからACCへデータを取り込む．
RCF(Reset CF)	桁上げフラグCFをリセットする．
SCF(Set CF)	桁上げフラグCFをセットする．
Bcc(Branch cc)	分岐命令．条件コードccによって分岐．
Ssm(Shift sm)	シフト命令．smはシフトモードの指定．
Rsm(Rotate sm)	巡回シフト命令．smはシフトモードの指定．
LD(LoaD)	ロード命令．メモリからデータをレジスタへ移す．
ST(STore)	ストア命令．レジスタの内容をメモリへ格納する．
SBC(SuBtract with Carry)	減算命令．桁上げフラグを考慮する．
ADC(ADd with Carry)	加算命令．桁上げフラグを考慮する．
SUB(SUBtract)	減算命令．桁上げフラグを考慮しない．
ADD(ADD)	加算命令．桁上げフラグを考慮しない．
EOR(Exclusive OR)	ビット毎の排他的論理和．
OR(OR)	ビット毎の論理和．
AND(AND)	ビット毎の論理積．
CMP(CoMPare)	比較命令．

2.3節に，シフト命令SsmとRsmの動作を示す．

フラグは４種類あり，演算の結果により，零フラグZF，ネガティブフラグNF，桁あふれフラグVF，桁上げフラグCFのそれぞれが設定される．このフラグの状態によってブランチ命令の分岐条件が判定される．フラグの設定条件は2.4節に示す．

1.4 KUE-CHIP2 の動作

KUE-CHIP2 は，最大 5 フェーズ（P0 から P4）で 1 命令を実行する．KUE-CHIP2 には，通常の動作モードのほかに，1 命令ごとに動作をとめるシングルインストラクションモードと 1 フェーズごとに動作をとめるシングルフェーズモードがある．これらのモードを利用して，各フェーズでどのように回路が動いているかが理解できる．

2.6節に，KUE-CHIP2 の各命令の実行フェーズ表を示す．各フェーズの動作の内最終的なレジスタの値の変更は，そのフェーズが終了して次のフェーズに入る時点で行なわれることに注意されたい．

フェーズ P0 では，命令をフェッチするために，命令の入っているアドレス（プログラムカウンタ PC に記憶されている）をメモリアドレスレジスタへ送る．この後，プログラムカウンタの内容は自動的に 1 増加する．

フェーズ P1 では，メモリに対して READ がかかり，メモリアドレスレジスタの内容を番地とするメモリの語がバス DBi を通して命令レジスタに転送される．この結果，次に実行される命令が解読可能な状態になる．ここまではすべての命令において共通である．

フェーズ P2 以降は命令によって動作が異なる．2 語長の命令では，2 語目のアドレス（プログラムカウンタが示している）をメモリアドレスレジスタへセットし，READ または WRITE をかける．この後，プログラムカウンタは再び 1 増加する．

例として，次のような場合を考えてみよう．2 つのレジスタ ACC および IX の内容はそれぞれ 01101100，00000011 であるとする．プログラムカウンタ PC は 00001100（１２番地）を指しているとする．メモリの１２番地には減算命令 10100111 が，１３番地には 00000110 が，２６５番地には 00111011 が格納されていると仮定する．各フェーズでの動作は以下のようになる．

P0 プログラムカウンタの内容をメモリアドレスレジスタ MAR に転送し，プログラムカウンタは 1 カウントアップ（１３となる）する．

P1 メモリの読みだしが行なわれる．メモリアドレスレジスタの内容で指定された１２番地の内容（10100111）が読みだされ，入力バス DBi を通って命令レジスタ IR に転送される．

P2 命令デコーダ IDC で命令レジスタの内容が解読され，修飾アドレスの減算命令であることがわかる．第 1 オペランドは，ACC である．第 2 オペランドのアドレスを生成するために，2 語目を読みだす．このために，再び，プログラムカウンタの内容（１３）をメモリアドレスレジスタ MAR に転送し，プログラムカウンタは 1 カウントアップ（１４となる）する．

P3 メモリの読みだしが行なわれる．メモリアドレスレジスタの内容で指定された１３番地の内容（00000110）が読みだされ，入力バス DBi を通って ALU へ送られる．第 2 オペランドのアドレスは，メモリから読みだされた値（6）にインデックスレジスタの内容（3）を加えて決定されるので，IX の内容も ALU へ送られ，加算が行なわれる．加算結果（9）は出力バス DBo を経由してメモリアドレスレジスタに送られる．

P4 第 2 オペランドの読みだしが行なわれる．命令のアドレスモードがデータ領域を指しているので，アドレスの最上位ビットは 1 となりアクセスされるデータは２６５番地（２５６＋９）の内容となる．読みだされたデータ（00111011）は入力バス DBi を通って ALU へ送られる．第 1 オペランドの ACC の内容（01101100）も ALU へ送られ，引き算が行なわれる．結果（00110001）は DBo を経由して ACC に格納される．また，演算結果にしたがってフラグがセットされる．

1.5 テスト結果

現在までに，以下の設計誤りが検出されている．

1. シフト命令 (Ssm : Shift sm) と巡回シフト命令 (Rsm : Rotate sm) を，SP(Single Phase) 入力を用いて 1 フェーズずつ実行させた場合，CF(Carry Flag) と VF(oVerflow Flag) の値は保証されない．

1.6 KUE-CHIP2 ボード

KUE-CHIP2 の動作を調べる実験は，KUE-CHIP2 を搭載した KUE-CHIP2 ボードの上で行なう．KUE-CHIP2 ボードは図 8 に示すように観測や実験のためのスイッチ，表示用 LED，コネクタを備えている．また，KUE-CHIP2 のほかに，これらのユーザーインタフェースのための多くの IC も搭載している．

KUE-CHIP2 ボードには，実験を円滑に行なうため，外部に 512 バイトのメモリバンクを 16 個搭載できるようになっている．これらのメモリバンクはすべて独立で，スイッチによってその 1 つを選べるようになっている．即ち，KUE-CHIP2 のメモリアドレスバスは 9 ビットであるが，ボード上では 13 ビットのアドレスバスが使われる．メモリの選択が内部メモリの時（MEM スイッチで選択）は KUE-CHIP2 のチップ上のメモリ（512 バイト）が使われ，外部メモリを選択したとき，外部の 16 個のバンクの中の 1 つ（ADDRESS スイッチの上位 4 ビット:CBA9）が使われる．この外部メモリバンクのほかに 16 バンク（1 バンクは 512 バイト）の ROM も搭載できる．ROM に保存してあるプログラムやデータを内部メモリバンクや外部メモリバンクへコピーするのに便利な機能も用意している．

また，KUE-CHIP2 ボードには，KUE-CHIP2 と同じ仕様の回路をボードの外部に作製し，その動作を KUE-CHIP2 ボードのユーザーインタフェースを利用して観測するための機能も付加されている．

KUE-CHIP2 ボードについての詳細は，KUE-CHIP2 教育用ボードリファレンスマニュアルを参照されたい．

1.7 KUE-CHIP2 の諸元

チップの形状

84 ピン Leadless Chip Carrier (LCC)

面積

array area	:	$4.02 \times 3.19 = 12.84mm^2$
active chip area	:	$5.18 \times 4.38 = 22.65mm^2$
die size	:	$5.39 \times 4.59 = 24.70mm^2$

ゲート数

論理素子	:	1597 個
フリップ・フロップ	:	68 個
パッド	:	76 個

動作周波数

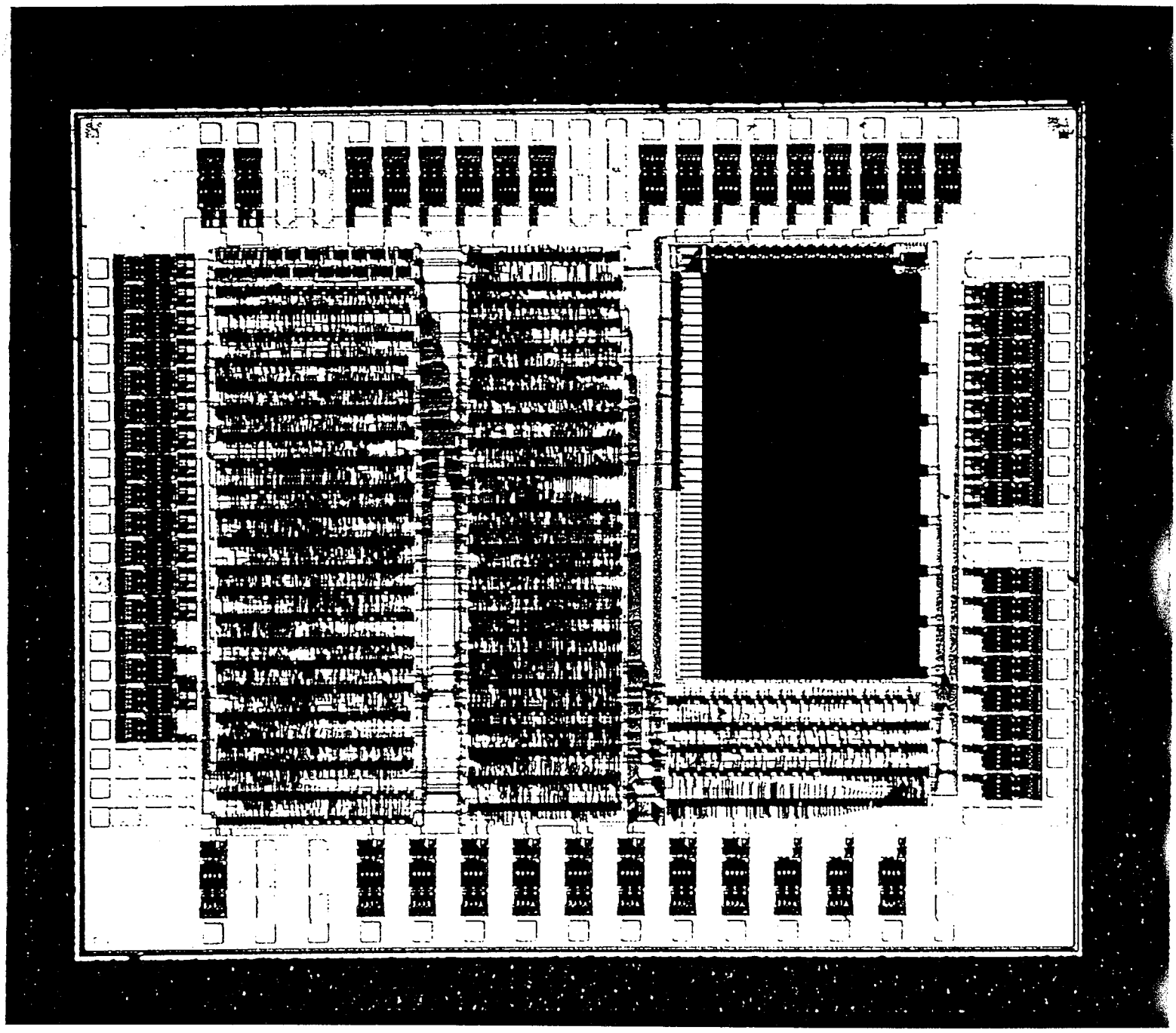
0.033Hz ～ 15MHz

製造プロセス

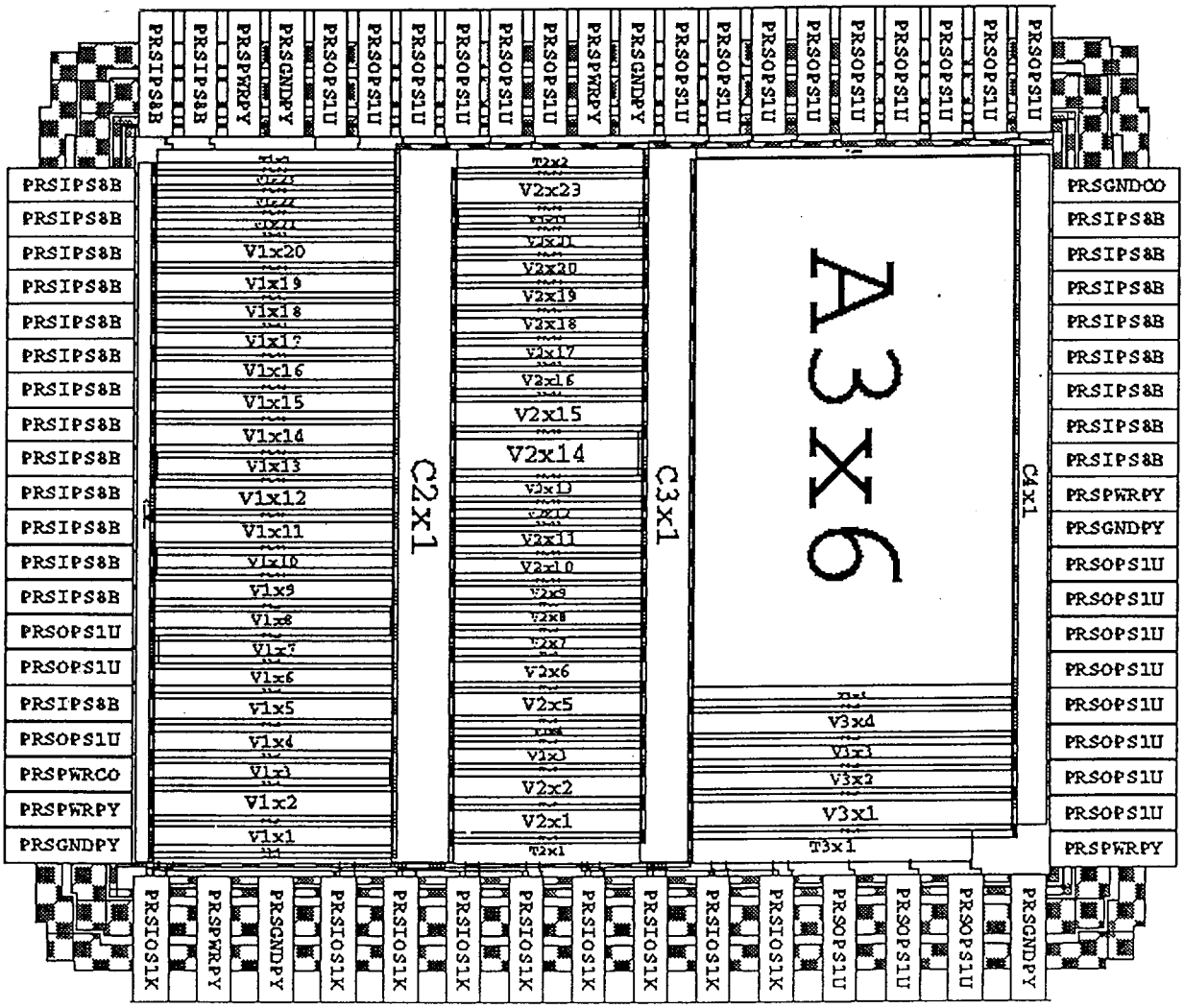
ES2 (European Silicon Structure) 社 1.2μ CMOS プロセス (ecpd12)

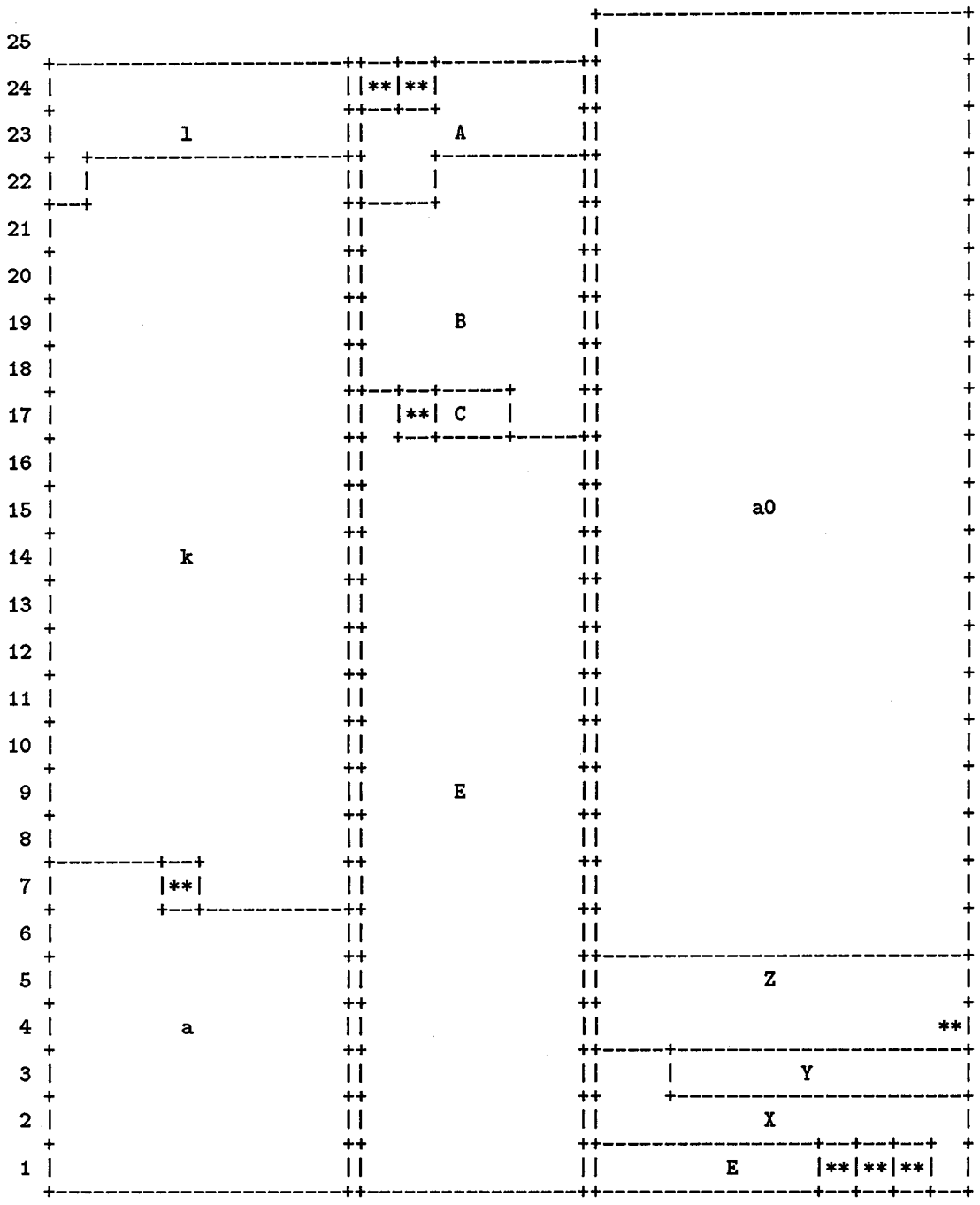
パッケージの写真

レイアウトの写真



レイアウトの写真の説明





Index of Parts shown:

	Instance Name	Model Name	
a	/KUE_CHIP_2/PC_MAR	pcmar	
k	/KUE_CHIP_2/ALU_ACC_IX	aluaccix	
l	/KUE_CHIP_2/SYNCHRONIZER	sync_mod	
A	/KUE_CHIP_2/CLOCK_GEN	clkgen_main	
B	/KUE_CHIP_2/IR_IDC	iridc	
C	/KUE_CHIP_2/BUF_CLOCK	nibuf6	
E	/KUE_CHIP_2/CONTROLLER	control	
X	/KUE_CHIP_2/OB_BUF_OF_DBI_DBO	obblk	
Y	/KUE_CHIP_2/DBI_BUF_OF_EXT_DBI	dbi_buf	
Z	/KUE_CHIP_2/INTER_MEM	imem	
a0	/KUE_CHIP_2/INTER_MEM/RAM	ram8x512	* block *

第 2 章

命令仕様

2.1 アセンブラ文法

KUE-CHIP2 のアドレスモード

ACC	アキュムレータ
IX	インデックスレジスタ
d	即値アドレス
[d]	絶対アドレス (プログラム領域)
(d)	絶対アドレス (データ領域)
[IX+d]	インデックス修飾アドレス (プログラム領域)
(IX+d)	インデックス修飾アドレス (データ領域)

KUE-CHIP2 論理アドレスモード

{ea}	全てのアドレスモード
{reg}	レジスタアドレスモード
{imm}	即値アドレスモード
{ma}	メモリ参照アドレスモード

アドレスモード対応表

	ACC	IX	d	[d]	(d)	[IX+d]	(IX+d)
{ea}	○	○	○	○	○	○	○
{reg}	○	○	×	×	×	×	×
{imm}	×	×	○	×	×	×	×
{ma}	×	×	×	○	○	○	○

KUE-CHIP2 命令表

略記号	アドレスモード	命令機能
HLT		Halt
NOP		No Operation
IN		INput
OUT		OUTput
SCF		Set Carry Flug
RCF		Reset Carry Flug
LD	{reg},{ea}	LoaD
ST	{reg},{ma}	STore
ADD	{reg},{ea}	ADD
ADC	{reg},{ea}	ADd with Carry
SUB	{reg},{ea}	SUBtract
SBC	{reg},{ea}	SuBtract with Carry
CMP	{reg},{ea}	CoMPare
AND	{reg},{ea}	AND
OR	{reg},{ea}	OR
EOR	{reg},{ea}	Exclusive OR
S*sm*	{reg}	Shift
R*sm*	{reg}	Rotate
RA		Right Arithmetically
LA		Left Arithmetically
RL		Right Logically
LL		Left Logically
B*cc*	{imm}	Branch
A		Always
VF		on oVerFlow
NZ		on Not Zero
Z		on Zero
ZP		on Zero or Positive
N		on Negative
P		on Positive
ZN		on Zero or Negative
NI		on No Input
NO		on No Output
NC		on No Carry
C		on Carry
E		on Greater than or Equal
LT		on Less Than
GT		on Greater Than
LE		on Less than or Equal

2.2 命令セット

略記号	命令コード (1 語目)								B'(2 語目)	命令機能の概略	
NOP	0	0	0	0	0	-	-	-	×	No OPeration	
HLT	0	0	0	0	1	-	-	-	×	HaLT	停止
	0	1	0	1	-	-	-	-	×		未使用 (HLT)
OUT	0	0	0	1	0	-	-	-	×	OUTput	$(ACC) \rightarrow OBUF$
IN	0	0	0	1	1	-	-	-	×	INput	$(IBUF) \rightarrow ACC$
RCF	0	0	1	0	0	-	-	-	×	Reset CF	$0 \rightarrow CF$
SCF	0	0	1	0	1	-	-	-	×	Set CF	$1 \rightarrow CF$
Bcc	0	0	1	1	cc				◎	Branch cc	条件が成立すれば $B' \rightarrow PC$
Ssm	0	1	0	0	A	0	sm		×	Shift sm	$(A) \rightarrow$ shift , rotate $\rightarrow A$
Rsm	0	1	0	0	A	1	sm		×	Rotate sm	はみ出したビット $\rightarrow CF$
LD	0	1	1	0	A		B		○	LoaD	$(B) \rightarrow A$
ST	0	1	1	1	A		B		◎	STore	$(A) \rightarrow B$
SBC	1	0	0	0	A		B		○	SuB with Carry	$(A) - (B) - CF \rightarrow A$
ADC	1	0	0	1	A		B		○	ADd with Carry	$(A) + (B) + CF \rightarrow A$
SUB	1	0	1	0	A		B		○	SUBtract	$(A) - (B) \rightarrow A$
ADD	1	0	1	1	A		B		○	ADD	$(A) + (B) \rightarrow A$
EOR	1	1	0	0	A		B		○	Exclusive OR	$(A) \oplus (B) \rightarrow A$
OR	1	1	0	1	A		B		○	OR	$(A) \vee (B) \rightarrow A$
AND	1	1	1	0	A		B		○	AND	$(A) \wedge (B) \rightarrow A$
CMP	1	1	1	1	A		B		○	CoMPare	$(A) - (B)$

cc : Condition Code

A	0	0	0	0	Always	常に成立
VF	1	0	0	0	on oVerflow	桁あふれ $VF = 1$
NZ	0	0	0	1	on Not Zero	$\neq 0$ $ZF = 0$
Z	1	0	0	1	on Zero	$= 0$ $ZF = 1$
ZP	0	0	1	0	on Zero or Positive	≥ 0 $NF = 0$
N	1	0	1	0	on Negative	< 0 $NF = 1$
P	0	0	1	1	on Positive	> 0 $(NF \vee ZF) = 0$
ZN	1	0	1	1	on Zero or Negative	≤ 0 $(NF \vee ZF) = 1$
NI	0	1	0	0	on No Input	$IBUF\_FLG\_IN = 0$
NO	1	1	0	0	on No Output	$OBUF\_FLG\_IN = 1$
NC	0	1	0	1	on Not Carry	$CF = 0$
C	1	1	0	1	on Carry	$CF = 1$
GE	0	1	1	0	on Greater than or Equal	≥ 0 $(VF \oplus NF) = 0$
LT	1	1	1	0	on Less Than	< 0 $(VF \oplus NF) = 1$
GT	0	1	1	1	on Greater Than	> 0 $((VF \oplus NF) \vee ZF) = 0$
LE	1	1	1	1	on Less than or Equal	≤ 0 $((VF \oplus NF) \vee ZF) = 1$

sm : Shift Mode

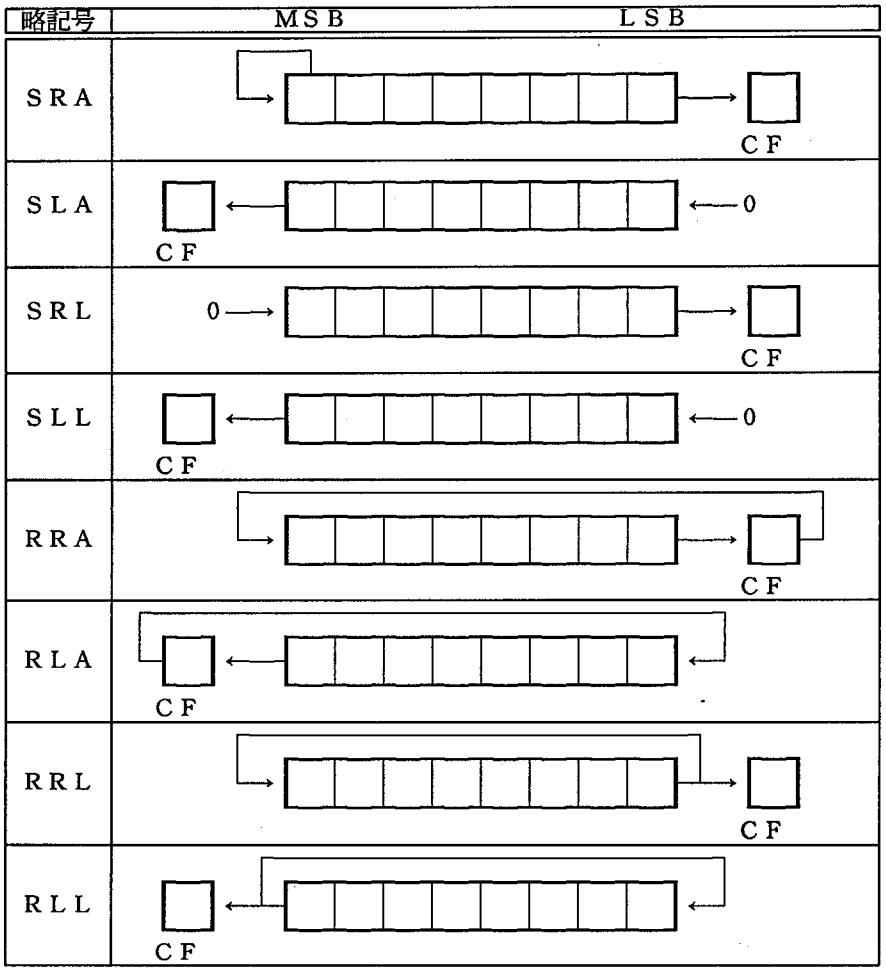
RA	0	0	Right Arithmetically
LA	0	1	Left Arithmetically
RL	1	0	Right Logically
LL	1	1	Left Logically

A = 0 : ACC
A = 1 : IX

B'(2 語目)
× : 不用
○ : 不用 or 必要
◎ : 必要

B = 000 : ACC
B = 001 : IX
B = 01– : d (即値アドレス)
B = 100 : [d] (絶対アドレス)
B = 101 : (d) (絶対アドレス)
B = 110 : [IX+d] (インデックス修飾アドレス)
B = 111 : (IX+d) (インデックス修飾アドレス)

2.3 Shift / Rotate 命令の機能



ＳＬＡとＳＬＬでは，実行後のフラグの状態が異なる (2.4節を参照されたい)

2.4 フラグ機能

CF : Carry Flag　VF : oVerflow Flag　NF : Negative Flag　ZF : Zero Flag

略記号	命令機能の概略	実行への影響 †				実行後の状態 ‡			
		CF	VF	NF	ZF	CF	VF	NF	ZF
NOP	No OPeration	—	—	—	—	—	—	—	—
HLT	HaLT	—	—	—	—	—	—	—	—
OUT	OUTput	—	—	—	—	—	—	—	—
IN	INput	—	—	—	—	—	—	—	—
RCF	Reset Carry Flag	—	—	—	—	0	—	—	—
SCF	Set Carry Flag	—	—	—	—	1	—	—	—
SRA	Shift Right Arithmetically	—	—	—	—	$b0$	0	N	Z
SLA	Shift Left Arithmetically	—	—	—	—	$b7$	V	N	Z
SRL	Shift Right Logically	—	—	—	—	$b0$	0	N	Z
SLL	Shift Left Logically	—	—	—	—	$b7$	0	N	Z
RRA	Rotate Right Arithmetically	$b7$	—	—	—	$b0$	0	N	Z
RLA	Rotate Left Arithmetically	$b0$	—	—	—	$b7$	V	N	Z
RRL	Rotate Right Logically	—	—	—	—	$b0$	0	N	Z
RLL	Rotate Left Logically	—	—	—	—	$b7$	0	N	Z
LD	LoaD	—	—	—	—	—	—	—	—
ST	STore	—	—	—	—	—	—	—	—
SBC	SuBtract with Carry	c	—	—	—	C	V	N	Z
ADC	ADd with Carry	c	—	—	—	C	V	N	Z
SUB	SUBtract	—	—	—	—	—	V	N	Z
ADD	ADD	—	—	—	—	—	V	N	Z
EOR	Exclusive OR	—	—	—	—	—	0	N	Z
OR	OR	—	—	—	—	—	0	N	Z
AND	AND	—	—	—	—	—	0	N	Z
CMP	CoMPare	—	—	—	—	—	V	N	Z
BA	Branch Always	—	—	—	—	—	—	—	—
BVF	Branch on oVerFlow	—	VF	—	—	—	—	—	—
BNZ	Branch on Not Zero	—	—	—	$\overline{ZF}$	—	—	—	—
BZ	Branch on Zero	—	—	—	ZF	—	—	—	—
BZP	Branch on Zero or Positive	—	—	$\overline{NF}$	—	—	—	—	—
BN	Branch on Negative	—	—	NF	—	—	—	—	—
BP	Branch on Positive	—	—	$\overline{NF \vee ZF}$		—	—	—	—
BZN	Branch on Zero or Negative	—	—	$NF \vee ZF$		—	—	—	—
BNI	Branch on No Input	—	—	—	—	—	—	—	—
BNO	Branch on No Output	—	—	—	—	—	—	—	—
BNC	Branch on Not Carry	$\overline{CF}$	—	—	—	—	—	—	—
BC	Branch on Carry	CF	—	—	—	—	—	—	—
BGE	Branch on Greater than or Equal	—	$\overline{VF \oplus NF}$		—	—	—	—	—
BLT	Branch on Less Than	—	$VF \oplus NF$		—	—	—	—	—
BGT	Branch on Greater Than	—	$\overline{(VF \oplus NF) \vee ZF}$			—	—	—	—
BLE	Branch on Less than or Equal	—	$(VF \oplus NF) \vee ZF$			—	—	—	—

†実行への影響

- c : Referred to as carry/borrow input.
- $b7$: Substituted into bit 7 of the operand A.
- $b0$: Substituted into bit 0 of the operand A.
- Exp. : Condition which causes branch.
- — : No effect.

‡実行後の状態

- C : Set to 1 iff carry/borrow occur.
- V : Set to 1 iff overflow occur.
- N : Set to 1 iff bit 7 of the result is 1.
- Z : Set to 1 iff all bits of the result are 0.
- $b7$: Set to 1 iff bit 7 of the operand A was 1.
- $b0$: Set to 1 iff bit 0 of the operand A was 1.
- 0 : Set to 0.
- 1 : Set to 1.
- — : Not modified.

2.5 命令コード早見表

ACC に対する代入および算術命令

	ACC	IX	d	[d]	(d)	[IX+d]	(IX+d)
LD ACC,	60	61	62	64	65	66	67
ST ACC,	–	–	–	74	75	76	77
SBC ACC,	80	81	82	84	85	86	87
ADC ACC,	90	91	92	94	95	96	97
SUB ACC,	A0	A1	A2	A4	A5	A6	A7
ADD ACC,	B0	B1	B2	B4	B5	B6	B7
EOR ACC,	C0	C1	C2	C4	C5	C6	C7
OR ACC,	D0	D1	D2	D4	D5	D6	D7
AND ACC,	E0	E1	E2	E4	E5	E6	E7
CMP ACC,	F0	F1	F2	F4	F5	F6	F7

IX に対する代入および算術命令

	ACC	IX	d	[d]	(d)	[IX+d]	(IX+d)
LD IX,	68	69	6A	6C	6D	6E	6F
ST IX,	–	–	–	7C	7D	7E	7F
SBC IX,	88	89	8A	8C	8D	8E	8F
ADC IX,	98	99	9A	9C	9D	9E	9F
SUB IX,	A8	A9	AA	AC	AD	AE	AF
ADD IX,	B8	B9	BA	BC	BD	BE	BF
EOR IX,	C8	C9	CA	CC	CD	CE	CF
OR IX,	D8	D9	DA	DC	DD	DE	DF
AND IX,	E8	E9	EA	EC	ED	EE	EF
CMP IX,	F8	F9	FA	FC	FD	FE	FF

シフト命令

	ACC	IX
SRA	40	48
SLA	41	49
SRL	42	4A
SLL	43	4B
RRA	44	4C
RLA	45	4D
RRL	46	4E
RLL	47	4F

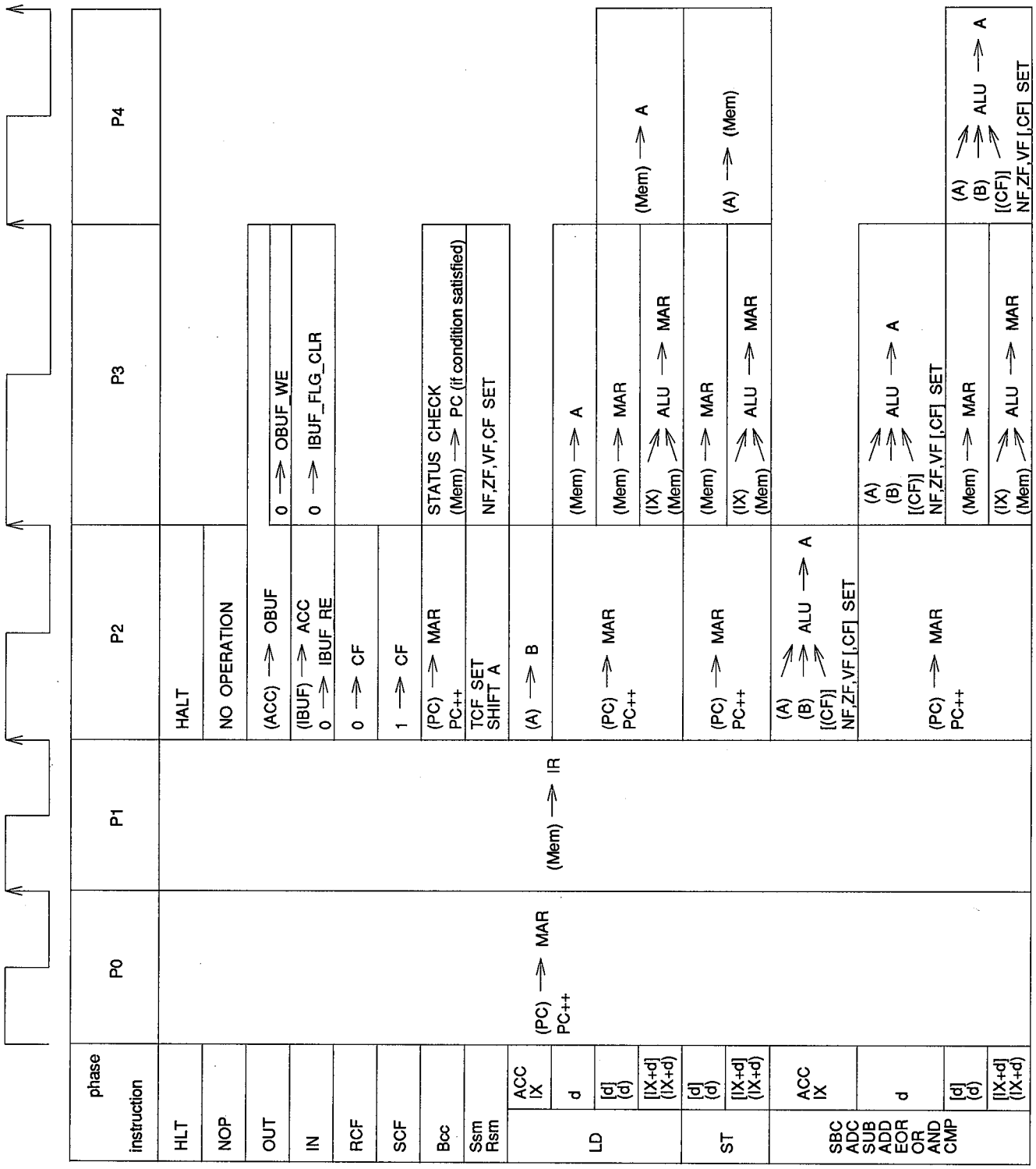
分岐命令

BA	30		
BVF	38		
BNZ	31	BZ	39
BZP	32	BN	3A
BP	33	BZN	3B
BNI	34	BNO	3C
BNC	35	BC	3D
BGE	36	BLT	3E
BGT	37	BLE	3F

制御命令

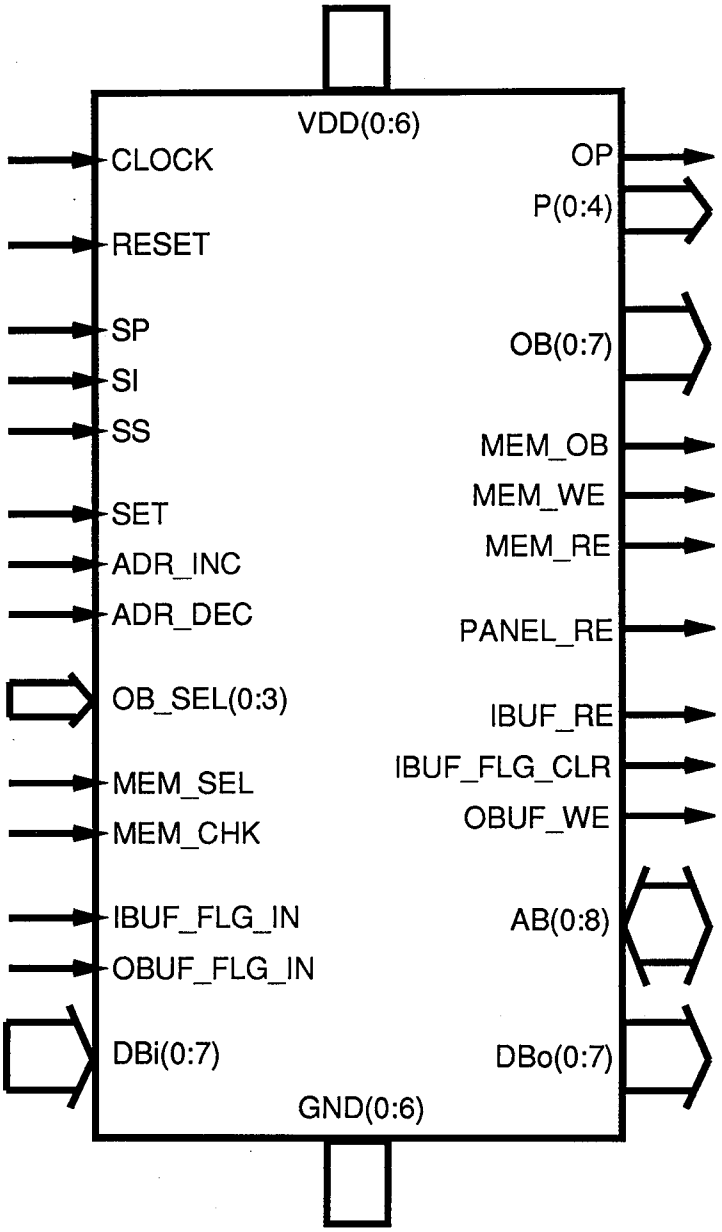
NOP	00
HLT	0F
OUT	10
IN	1F
RCF	20
SCF	2F

2.6 命令実行フェーズ


phase
instruction
P0
P1
P2
P3
P4
HLT
HALT
NOP
NO OPERATION
OUT
(ACC) → OBUF
0 → OBUF_WE
IN
(IBUF) → ACC
0 → IBUF_RE
0 → IBUF_FLG_CLR
RCF
0 → CF
SCF
1 → CF
Bcc
(PC) → MAR
PC++
STATUS CHECK
(Mem) → PC (if condition satisfied)
Ssm
Rsm
TCF SET
SHIFT A
NF,ZF,VF,CF SET
LD
ST
SBC
ADC
SUB
ADD
EOR
OR
AND
CMP
ACC
IX
d
[d]
(d)
[IX+d]
(IX+d)
(PC) → MAR
PC++
(Mem) → IR
(A) → B
(Mem) → A
(Mem) → MAR
(IX)
(Mem)
ALU → MAR
(A) → (Mem)
(A)
(B)
[(CF)]
ALU → A
NF,ZF,VF [,CF] SET

第 3 章

外部ピン仕様


VDD(0:6)
CLOCK
RESET
SP
SI
SS
SET
ADR_INC
ADR_DEC
OB_SEL(0:3)
MEM_SEL
MEM_CHK
IBUF_FLG_IN
OBUF_FLG_IN
DBi(0:7)
GND(0:6)
OP
P(0:4)
OB(0:7)
MEM_OB
MEM_WE
MEM_RE
PANEL_RE
IBUF_RE
IBUF_FLG_CLR
OBUF_WE
AB(0:8)
DBo(0:7)

3.1 ピン名と機能

VDD1, VDD2, VDD3, VDD4, VDD5, VDD6, VDD7

機能 : 電源
入力 : +5.0V

GND1, GND2, GND3, GND4, GND5, GND6, GND7

機能 : グラウンド
入力 : 0.0V

CLOCK

機能 : マスタークロック
入力 : 方形波

RESET

機能 : 全てのフリップフロップを非同期的にリセット（0）にする
入力 :  でリセット

SP (Single Phase)

機能 : 1フェーズだけ実行して停止
入力 :  チャタリングは除去されていること

SI (Single Instruction)

機能 : 1命令だけ実行して停止
入力 :  チャタリングは除去されていること

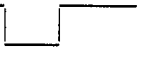
SS (Start/Stop)

機能 : 停止時，PC の指すメモリーアドレスから命令の実行を開始
命令実行時，命令の実行を停止する
（押した時点に実行中の命令の実行の終了を待って停止する）
入力 :  チャタリングは除去されていること

SET

機能 : データスイッチの内容を，OB_SEL(0:3) で指定した記憶素子へセットする
入力 :  チャタリングは除去されていること

ADR_INC

機能 : MAR の値を1つインクリメントする
入力 :  チャタリングは除去されていること

ADR_DEC

機能 : MAR の値を1つデクリメントする
入力 :  チャタリングは除去されていること

OB_SEL(0:3) (OBserve SELect)

機能 : オブザーババス OB(0:7) を通じて中身を観測する
または入力データバス DBi(0:7) を通じてデータの書き込みを行う記憶素子を選択する
入力 : 4 ビット

(3)	(2)	(1)	(0)	記憶素子または内部制御線
0	0	0	0	MC_P (Memory Contents in Program region)
0	0	0	1	MC_D (Memory Contents in Data region)
0	0	1	0	PC (Program Counter)
0	0	1	1	FLAG (FLAG register)
0	1	0	0	ACC (ACCumlator)
0	1	0	1	IX (IndeX register)
0	1	1	0	DBi (Data Bus for Input)
0	1	1	1	DBo (Data Bus for Output)
1	0	0	0	MAR (Memory Address Register)
1	0	0	1	IR (Instruction Register)
1	0	1	0	制御線 imem(0:2) , mar(0:4)
1	0	1	1	制御線 pc(0:2) , ir(0:1) , ios(0:1) , flag_ob
1	1	0	0	制御線 obc(0:1) , cfset(0:2) , vfset(0:2)
1	1	0	1	制御線 phasecut , halt , nfset(0:2) , zfset(0:2)
1	1	1	0	制御線 alu , alu_ope(0:3) , reg(0:2)
1	1	1	1	制御線 acc(0:3) , ix(0:3)

MEM_SEL (MEMory SELect)

機能 : 内部メモリと外部メモリの選択
入力 : 1 : 内部メモリ Int_Mem を選択
0 : 外部メモリ Ext_Mem を選択

MEM_CHK (MEMory CHecK)

機能 :
入力 : 1 : アドレスバスは, MAR が与える (KUE-CHIP2 で)
0 : アドレスバスは, 外部ピン AB(0:8) を通じて外部から入力する

IBUF_FLG_IN

機能 : IBUF のフラグ IBUF_FLAG を読み込む
入力 : 1 : IBUF はデータを書き込まれていてかつまだ読み出されていない
0 : IBUF はデータを読み出された後, データを新たに書き込まれていない

OBUF_FLG_IN

機能 : OBUF のフラグ OBUF_FLAG を読み込む
入力 : 1 : OBUF はデータを書き込まれていてかつまだ読み出されていない
0 : OBUF はデータを読み出された後, データを新たに書き込まれていない

DBi(0:7) (Data Bus for Input)

機能 : 入力データバス
入力 : IBUF または Ext_Mem または Panel_Sw のデータを取り込む

OP (OPeration)

機能 : CPU が動作中か停止中かを示す
出力 : 1 : CPU は動作中
0 : CPU は停止中

P(0:4) (Phase)

機能 : CPU が動作あるいは停止している状態でのクロックフェーズを表示
出力 : 1 : 現在のクロックフェーズ

OB(0:7) (OBserver bus)

機能 : オブサーババス
出力 : OB_SEL(0:3) で選んだレジスタ，カウンタ，フラグなどのデータを出力する

MEM_OB (external MEMory : OBserve enable)

機能 : Ext_Mem のデータをオブサーババス OB(0:7) に出力する
(Ext_Mem とオブサーババス OB(0:7) の間の 3_state_buffer を制御する)
出力 : 0 : 外部メモリを観測する (OB_SEL(0:3) == 0000 または 0001 で MEM_SEL == 0) 状態

MEM_WE (external MEMory : Write Enable)

機能 : DBo(0:7) のデータを Ext_Mem へ書き込む
出力 : 0 : Ext_Mem へデータを書き込みを行うフェーズ

MEM_RE (external MEMory : Read Enable)

機能 : Ext_Mem のデータを DBi(0:7) に出力する
(Ext_Mem と DBi(0:7) 間の 3_state_buffer を制御する)
出力 : 0 : 外部メモリからデータを読み込みを行うフェーズ

PANEL_RE (PANEL switch : Read Enable)

機能 : 8 ビットトグルスイッチのデータを入力データバス DBi(0:7) に出力する
(Panel_Sw と DBi(0:7) 間の 3_state_buffer を制御する)
出力 : 0 : 8 ビットトグルスイッチからデータを読み込みを行う状態

IBUF_RE (IBUF : Read Enable)

機能 : 入力バッファ IBUF のデータを DBi(0:7) に出力する
(IBUF と DBi(0:7) の間の 3_state_buffer を制御する)
出力 : 0 : IBUF → DBi(0:7) に転送するフェーズ

IBUF_FLG_CLR (IBUF : FLaG CLeaR)

機能 : 入力バッファ IBUF のフラグ IBUF_FLAG をリセットする
(非同期的に 0 をセットする)
出力 : 0 : IBUF_FLAG をクリアするフェーズ

OBUF_WE (OBUF : Write Enable)

機能 : 出力データバス DBo(0:7) 上のデータを，出力バッファ OBUF へ書き込む
同時に出力バッファのフラグ OBUF_FLAG を立てる (1 にする)
出力 : 0 : DBo(0:7) 上のデータを OBUF へ書き込むフェーズ (↓で)

AB(0:8) (Addres Bus)

機能 : アドレスバス
入力 : MEM_CHK == 0 のとき，外部からアドレスを与える
出力 : MEM_CHK == 1 のとき， MAR を出力する
（AB(8) は MAR と切り離されている）

DBo(0:7) (Data Bus for Output)

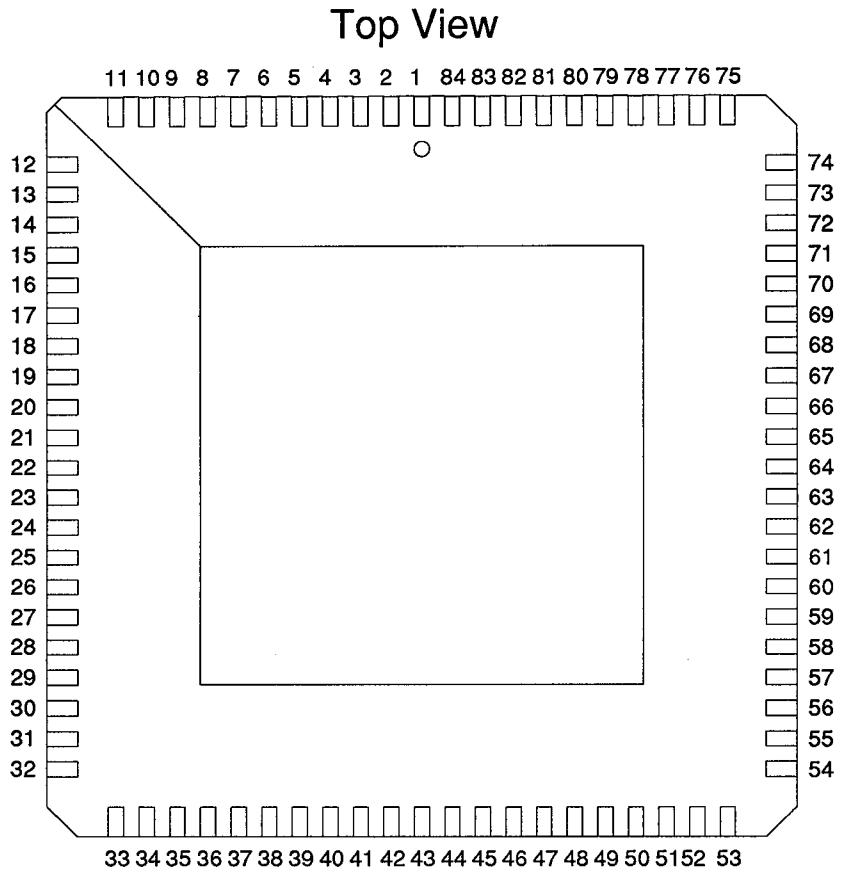
機能 : 出力データバス
出力 : OBUF または Ext_Mem へ書き込むデータを出力する

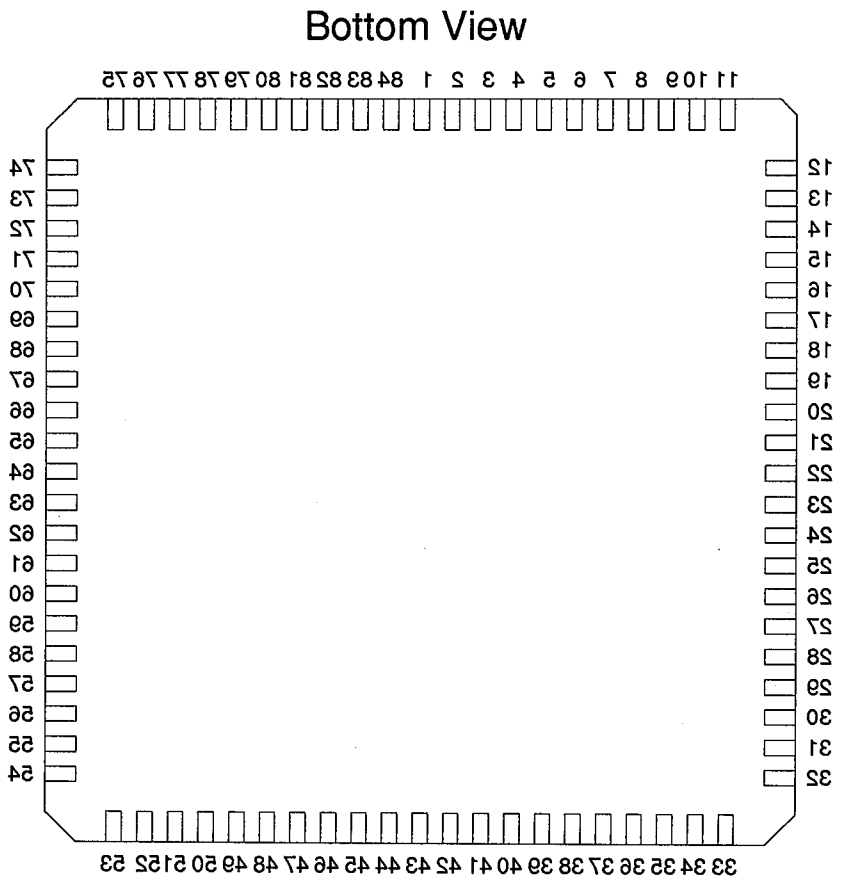
3.2 ピン番号とピン名

ピン番号	ピン名	ピン番号	ピン名
1	GND4	43	OB_SEL0
2	VDD4	44	MEM_SEL
3	DBI7	45	MEM_CHK
4	DBI6	46	IBUF_FLG_IN
5	DBI5	47	IBUF_FLG_CLR
6	DBI4	48	IBUF_RE
7	DBI3	49	OBUF_FLG_IN
8	DBI2	50	OBUF_WE
9	DBI1	51	VDD6
10	DBI0	52	VDD2
11	GND6	53	GND2
12	PANEL_RE	54	—
13	OB0	55	—
14	OB1	56	—
15	OB2	57	—
16	OB3	58	—
17	OB4	59	—
18	OB5	60	—
19	OB6	61	AB8
20	OB7	62	VDD3
21	GND1	63	GND3
22	VDD1	64	AB7
23	OP	65	AB6
24	P0	66	AB5
25	P1	67	AB4
26	P2	68	AB3
27	P3	69	AB2
28	P4	70	AB1
29	GND5	71	AB0
30	VDD5	72	MEM_WE
31	RESET	73	MEM_RE
32	CLOCK	74	MEM_OB
33	—	75	GND0
34	SS	76	VDD0
35	SI	77	DBO7
36	SP	78	DBO6
37	SET	79	DBO5
38	ADR_INC	80	DBO4
39	ADR_DEC	81	DBO3
40	OB_SEL3	82	DBO2
41	OB_SEL2	83	DBO1
42	OB_SEL1	84	DBO0

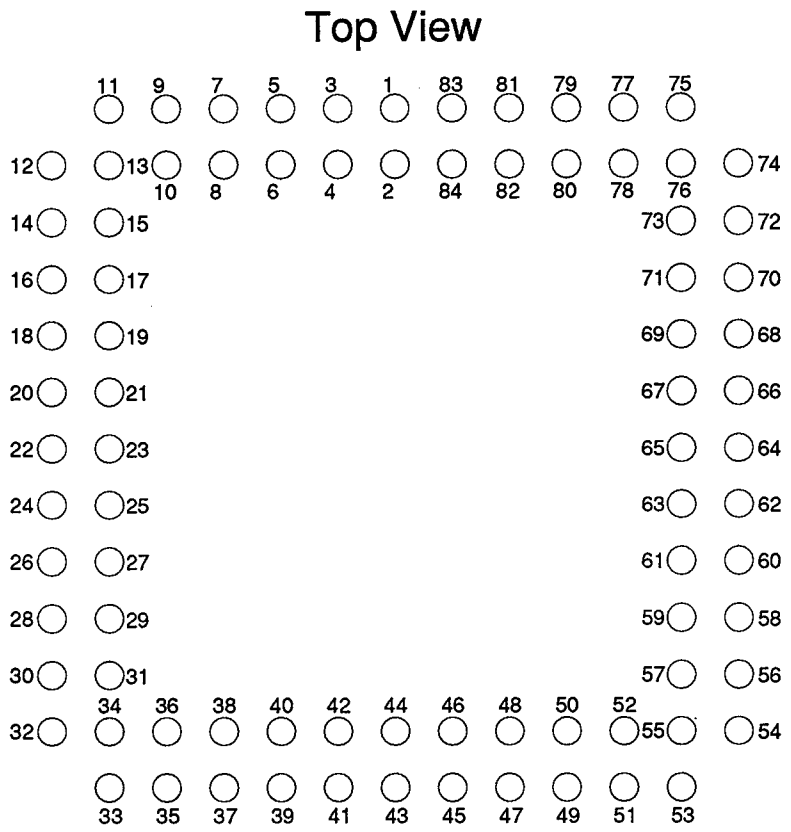
—：未使用

3.3 チップ外形とピン番号



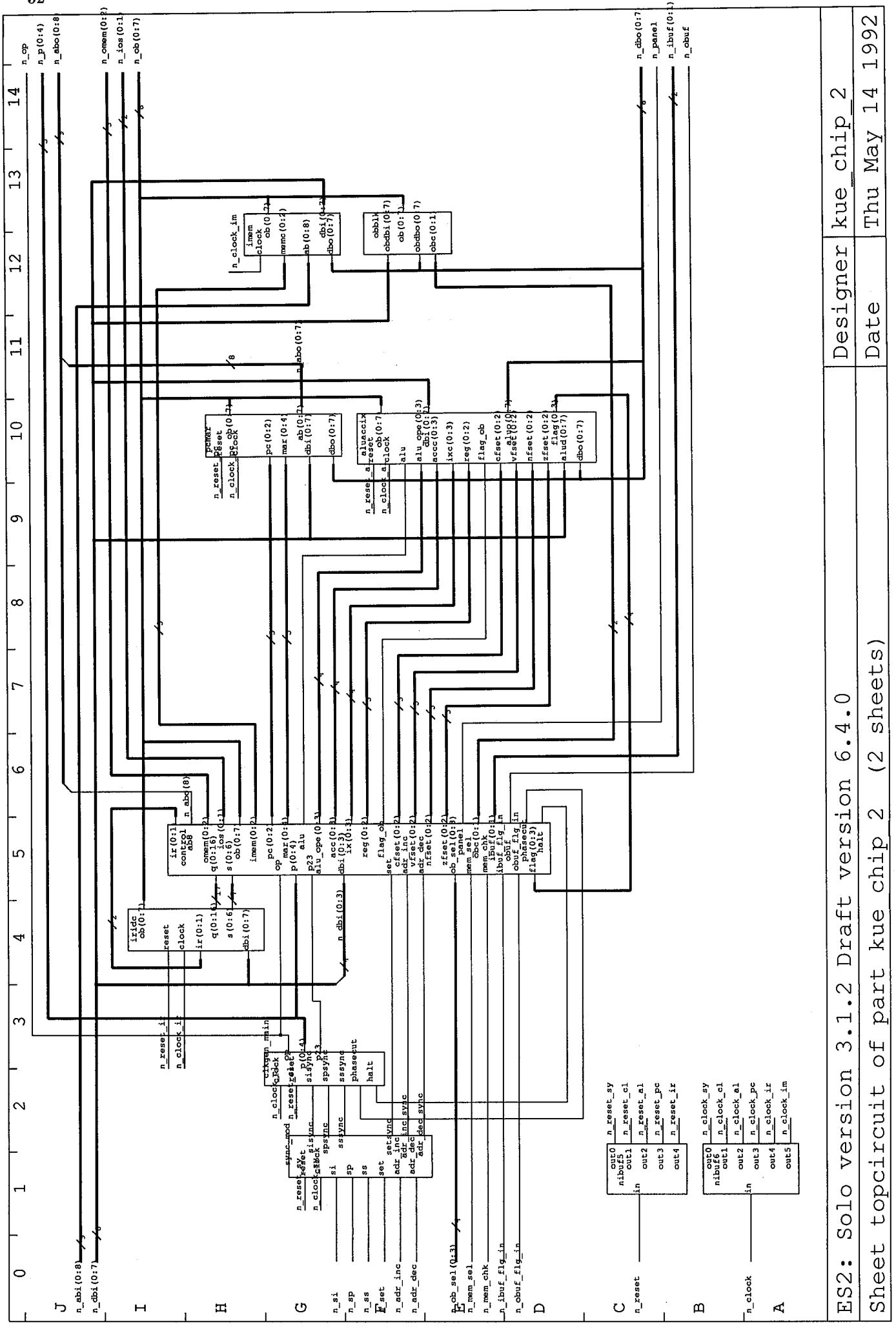


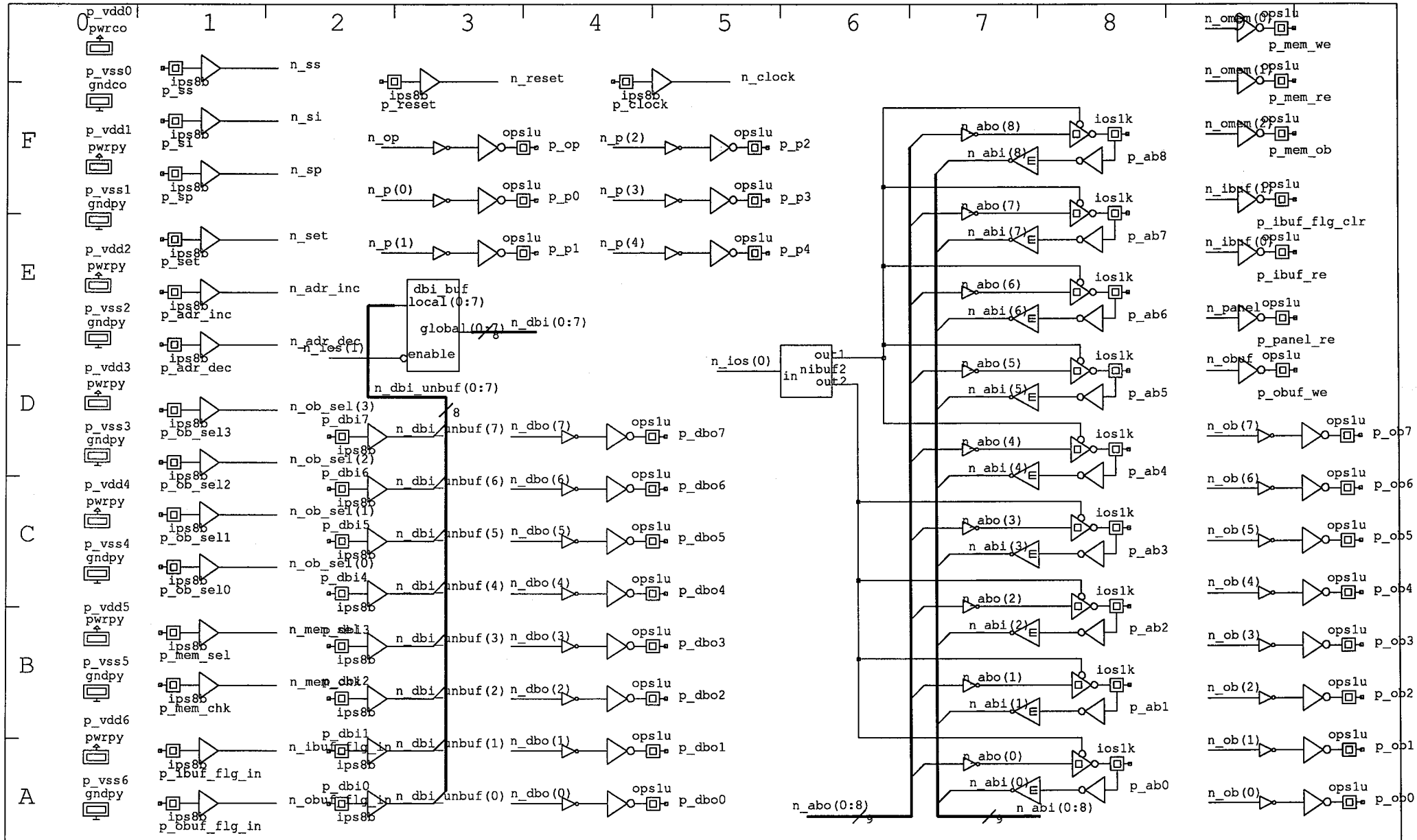
3.4 ソケットのピン配置



第 4 章

最上位モジュール回路図

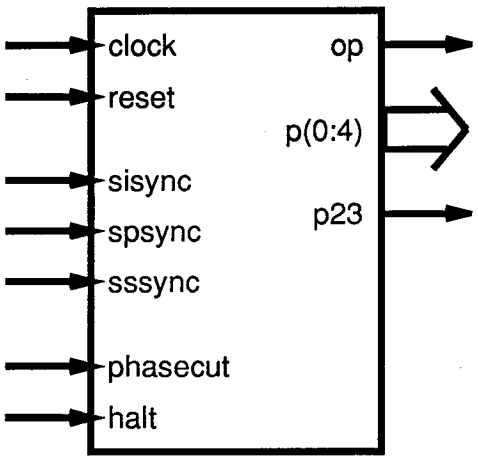

ES2: Solo version 3.1.2 Draft version 6.4.0
Designer kue_chip_2
Sheet topcircuit of part kue_chip_2 (2 sheets)
Date Thu May 14 1992



ES2: Solo version 3.1.2 Draft version 6.4.0		Designer	kue_chip_2
Sheet toppad of part kue_chip_2 (2 sheets)		Date	Thu May 14 1992

第 5 章

クロックジェネレータ部仕様



5.1 入出力仕様

5.1.1 入力仕様

パネルから

clock マスタークロック
reset リセット信号

シンクロナイザから

sisync 1でsiキーの入力
spsync 1でspキーの入力
sssync 1でssキーの入力

コントローラから

phasecut 0でP0に戻る.
halt 0でopを0にする.

5.1.2 出力仕様

op

CPUが動作中かどうかを示す.
0 CPUは停止中
1 CPUは動作中

p(0:4)

CPUがどのクロックフェイズにあるかを示す.
1 CPUはそのクロックフェイズ.

p23

OUT命令の実行において, CPUが2または3フェイズ目にあることを示す.
1 CPUは2または3クロックフェイズ.

5.2 内部信号線仕様

5.2.1 s_0, s_1 : sisync,spsync,sssync

	s_0	s_1
no input	0	0
sssync	0	1
sisync	1	1
spsync	1	0

s_0 = sisync + spsync
s_1 = sssync + sisync

5.2.2 op : s_0, s_1,halt,phasecut

op : f_0, f_1

CPU の動作状態

f_0	f_1	動作状態	op
0	0	停止	0
0	1	通常動作	1
1	1	シングルインストラクション動作	1
1	1	シングルフェイズ動作	1

$op = f_0 + f_1$

f_1, f_0 : halt,phasecut,s_0, s_1

(f_0, f_1)=(0,0) の時の次状態

		No	ss	si	sp
$\overline{h}$	$\overline{p}$	00	01	11	10
0	0	00	01	11	10
0	1	00	00	00	00
1	1	00	00	00	00
1	0	00	00	00	00

(f_0, f_1)=(0,1) の時の次状態

		No	ss	si	sp
$\overline{h}$	$\overline{p}$	00	01	11	10
0	0	01	11	01	01
0	1	01	11	01	01
1	1	00	00	00	00
1	0	00	00	00	00

(f_0, f_1)=(1,1) の時の次状態

		No	ss	si	sp
$\overline{h}$	$\overline{p}$	00	01	11	10
0	0	11	11	11	11
0	1	00	00	00	00
1	1	00	00	00	00
1	0	00	00	00	00

(f_0, f_1)=(1,0) の時の次状態

		No	ss	si	sp
$\overline{h}$	$\overline{p}$	00	01	11	10
0	0	00	00	00	10
0	1	00	00	00	10
1	1	00	00	00	00
1	0	00	00	00	00

f_0(D-ff) の入力:$d_0 = halt * (f_0 * f_1 * phasecut + \overline{f_0} * \overline{f_1} * s_0 * phasecut + \overline{f_0} * f_1 * \overline{s_0} * s_1 + f_0 * \overline{f_1} * s_0 * \overline{s_1})$
f_1(D-ff) の入力:$d_1 = halt * (\overline{f_0} * f_1 + f_1 * phasecut + \overline{f_0} * phasecut * s_1)$

5.2.3 p(0:4),p23 : op,phasecut

p(0:4),p23 : q_0, q_1, q_2

	q_2	q_1	q_0
p(0)	0	0	0
p(1)	0	0	1
p(2)	0	1	1
p(3)	1	1	1
p(4)	1	1	0

q_2	q_1	q_0	
0	0	0	p(0)
0	0	1	p(1)
0	1	1	p(2)
0	1	0	p(0)
1	1	0	p(4)
1	1	1	p(3)
1	0	1	p(0)
1	0	0	p(0)

p(0) = $q_2 * \overline{q_1} + \overline{q_2} * \overline{q_0}$
p(1) = $\overline{q_2} * \overline{q_1} * q_0$
p(2) = $\overline{q_2} * q_1 * q_0$
p(3) = $q_2 * q_1 * q_0$
p(4) = $q_2 * q_1 * \overline{q_0}$
p23 = $q_1 * q_0$

q_2, q_1, q_0 : **op,phasecut**

	op=1,phasecut=0	op=1,phasecut=1	op=0
p0	p0	p1	p0
p1	p0	p2	p1
p2	p0	p3	p2
p3	p0	p4	p3
p4	p0	p0	p4

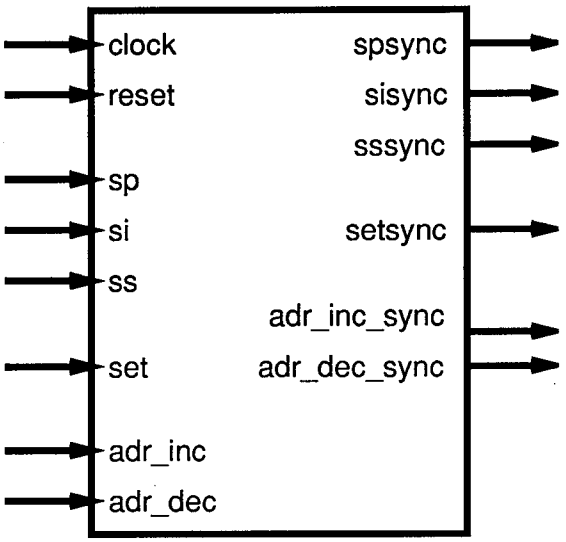
q_0(D-ff) の入力: $d_0 = op * phasecut * (\overline{q_2} + \overline{q_1}) + \overline{op} * q_0 * (\overline{q_2} + q_1)$

q_1(D-ff) の入力: $d_1 = op * phasecut * q_0 * (\overline{q_2} + q_1) + \overline{op} * q_1 * (q_2 + q_0)$

q_2(D-ff) の入力: $d_2 = op * phasecut * q_1 * q_0 + \overline{op} * q_2 * q_1$

第 6 章

シンクロナイザ部仕様



6.1 入出力仕様

6.1.1 入力仕様

パネルから

clock	マスタークロック
reset	リセット信号
sp	0 の時 sp のキーが押された.
si	0 の時 si のキーが押された.
ss	0 の時 ss のキーが押された.
set	0 の時 set のキーが押された.
adr_inc	0 の時 adr_inc のキーが押された.
adr_dec	0 の時 adr_dec のキーが押された.

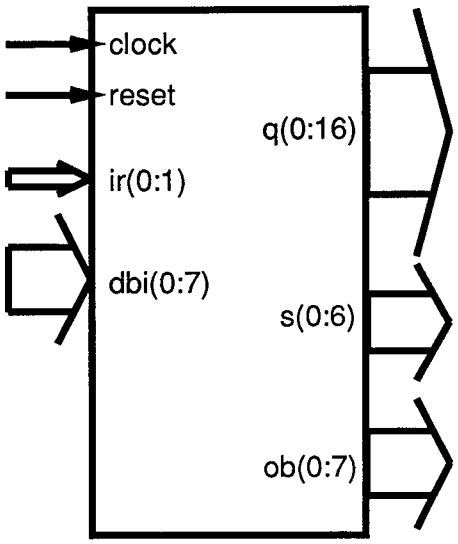
6.1.2 出力仕様

spsync,sisync,sssync,setsync,adr_inc_sync,adr_dec_sync

それぞれシンクロナイズされた sp,si,ss,set,adr_inc,adr_dec
1 (1 クロック周期) そのキーが押された.

第 7 章

IR,IDC 部仕様



7.1 入出力仕様

7.1.1 入力仕様

コントローラから

ir(1) 0 の時, IR は 観測バス OB にデータを出力する
ir(0) 0 の時, IR は データバス DBi 上のデータをラッチする

パネル (外部ピン) から

clock マスタークロック
reset リセット信号

内部バスから

dbi(0:7) データバス DBi

7.1.2 出力仕様

q(0:16)

q(0)	NOP	
q(1)	HLT, 未定義命令, 不正命令	
q(2)	OUT	
q(3)	IN	
q(4)	RCF, SCF	
q(5)	Bcc	
q(6)	SRA, SLA, SRL, SLL, RRA, RLA, RRA, RLL	
q(7)	LD	オペランド B が ACC/IX
q(8)	LD	オペランド B が即値アドレス
q(9)	LD	オペランド B が絶対アドレス
q(10)	LD	オペランド B がインデックス修飾アドレス
q(11)	ST	オペランド B が絶対アドレス
q(12)	ST	オペランド B がインデックス修飾アドレス
q(13)	SBC, ADC, SUB, ADD, EOR, OR, AND, CMP	オペランド B が ACC/IX
q(14)	SBC, ADC, SUB, ADD, EOR, OR, AND, CMP	オペランド B が即値アドレス
q(15)	SBC, ADC, SUB, ADD, EOR, OR, AND, CMP	オペランド B が絶対アドレス
q(16)	SBC, ADC, SUB, ADD, EOR, OR, AND, CMP	オペランド B がインデックス修飾アドレス

s(0:6)

s(6) 命令コードの右から 7 ビット目
s(5) 命令コードの右から 6 ビット目
s(4) 命令コードの右から 5 ビット目
s(3) 命令コードの右から 4 ビット目
s(2) 命令コードの右から 3 ビット目
s(1) 命令コードの右から 2 ビット目
s(0) 命令コードの右から 1 ビット目

ob(0:7)

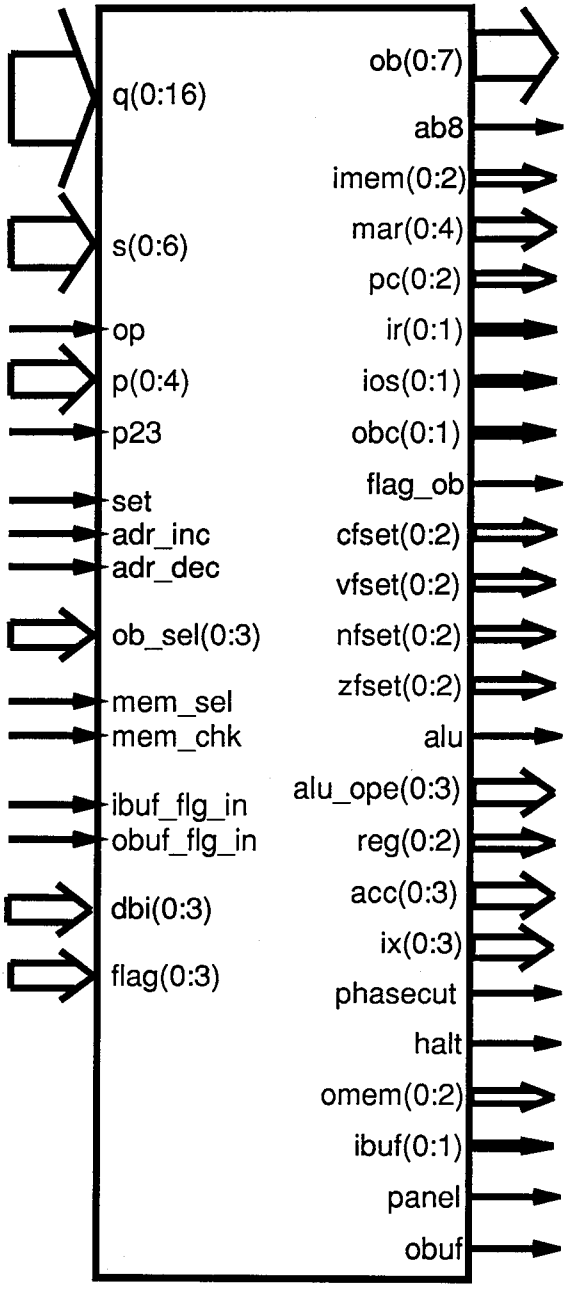
ob(0:7) 観測バス OB

7.2 動作

	$\overline{\text{op}}$·set	op·p(0)	op·p(1)	op·p(2)	op·p(3)	op·p(4)
全ての命令	(データスイッチ) → IR		(Mem) → IR			

第 8 章

コントローラ部仕様



8.1 入出力仕様

8.1.1 入力仕様

IR,IDC 部から

q(0:16) 命令の種類 (1)
s(0:6) 命令コード

クロックジェネレータから

op CPU が動作中 (1) か停止中 (0) か
p(0:4) クロックフェーズを表示 (1)
p23 クロックフェーズがフェーズ 2 とフェーズ 3 でずっと 1 (ハザードなし)

シンクロナイザから (クロック 1 周期分)

set データスイッチの内容を ob_sel(0:3) で指定した記憶素子にセットする (1)
adr_inc MAR を 1 インクリメントする (1)
adr_dec MAR を 1 デクリメントする (1)

パネル (外部ピン) から

ob_sel(0:3) 記憶素子または内部制御線を選択する (下記参照)

(3)	(2)	(1)	(0)	記憶素子または内部制御線
0	0	0	0	mc_p
0	0	0	1	mc_d
0	0	1	0	pc
0	0	1	1	flag
0	1	0	0	acc
0	1	0	1	ix
0	1	1	0	dbi
0	1	1	1	dbo
1	0	0	0	mar
1	0	0	1	ir
1	0	1	0	imem(0:2) , mar(0:4)
1	0	1	1	pc(0:2) , ir(0:1) , ios(0:1) , flag_ob
1	1	0	0	obc(0:1) , cfset(0:2) , vfset(0:2)
1	1	0	1	phasecut , halt , nfset(0:2) , zfset(0:2)
1	1	1	0	alu , alu_ope(0:3) , reg(0:2)
1	1	1	1	acc(0:3) , ix(0:3)

mem_sel 内部メモリ (1) と外部メモリ (0) の選択
mem_chk アドレスバスのデータは MAR(1), 外部ピン (0) が与える
ibuf_flg_in 読み出されていない (1), 書き込まれていない (0)
obuf_flg_in 読み出されていない (1), 書き込まれていない (0)
dbi(0:3) データバス DBi (フラグをセットするため)

ALU,ACC,IX 部から

flag(3) carry flag
flag(2) overflow flag
flag(1) negative flag
flag(0) zero flag

8.1.2 出力仕様

ob(0:7)

ob(0:7) : ob_sel(0:3) で選択された内部制御線を出力する

ab8

ab8 : 0 の時, プログラム領域
1 の時, データ領域

imem(0:2)

imem(2) : 0 の時, OB に内蔵メモリのデータを出力する
imem(1) : 0 の時, DBi に内蔵メモリのデータを出力する
imem(0) : 0 の時, DBo 上のデータを内蔵メモリに書き込む

mar(0:4)

mar(4) : 0 の時, MAR は そのデータを 1 デクリメントする
mar(3) : 0 の時, MAR は そのデータを 1 インクリメントする
mar(2) : 0 の時, MAR は OB にそのデータを出力する
mar(1) : 0 の時, Sel. は DBo を MAR に接続
1 の時, Sel. は PC を MAR に接続
mar(0) : 0 の時, MAR は PC または DBo 上のデータをラッチする
(PC と DBo の選択は, mar(1) で指定する)

pc(0:2)

pc(2) : 0 の時, PC は そのデータを OB に出力する
pc(1) : 0 の時, PC は そのデータを 1 インクリメントする
pc(0) : 0 の時, PC は DBi 上のデータをラッチする

ir(0:1)

ir(1) : 0 の時, IR は 観測バス OB にデータを出力する
ir(0) : 0 の時, IR は DBi 上のデータをラッチする

ios(0:1)

ios(1) : 0 の時, DBi はチップの外部から駆動される
ios(0) : 0 の時, アドレスバス AB は MAR に接続
1 の時, アドレスバス AB はチップの外部から駆動される

obc(0:1)

obc(1) : 0 の時, DBi 上のデータを OB に出力する
obc(0) : 0 の時, DBo 上のデータを OB に出力する

flag_ob

flag_ob : 0 の時, FLAG を OB に出力 (形式は右記参照)

TC	-	-	-	C	V	N	Z

cfset(0:2)

cfset(2) : 0 の時, CF はデータをラッチする
cfset(1) : 右記参照
cfset(0) : 右記参照

(1)	(0)	
0	0	CF は 0 に接続する
0	1	1 に接続する
1	0	ALU の Carry に接続する
1	1	TCF に接続する

vfset(0:2)

vfset(2) : 0 の時, VF はデータをラッチする
vfset(1) : 右記参照
vfset(0) : 右記参照

(1)	(0)	
0	0	VF は 0 に接続する
0	1	1 に接続する
1	0	ALU の Overflow に接続する
1	1	$(TCF \oplus b7)$ に接続する

nfset(0:2)

nfset(2) : 0 の時, NF はデータをラッチする
nfset(1) : 右記参照
nfset(0) : 右記参照

(1)	(0)	
0	0	NF は 0 に接続する
0	1	1 に接続する
1	–	b7 に接続する

zfset(0:2)

zfset(2) : 0 の時, ZF はデータをラッチする
zfset(1) : 右記参照
zfset(0) : 右記参照

(1)	(0)	
0	0	ZF は 0 に接続する
0	1	1 に接続する
1	–	(b0 – b7) の NOR に接続する

alu

alu : 0 の時, Sel. は ACC を ALU につなぐ
1 の時, Sel. は IX を ALU につなぐ

alu_ope(0:3)

alu_ope(3) : 右記参照
alu_ope(2) : 右記参照
alu_ope(1) : 右記参照
alu_ope(0) : 右記参照

(3)	(2)	(1)	(0)			
0	–	–	–	ALU の演算は	DBi.	をそのまま出力
1	0	0	0		SBC	$(Sel. - DBi - CF)$
1	0	0	1		ADC	$(Sel. + DBi + CF)$
1	0	1	0		SUB	$(Sel. - DBi)$
1	0	1	1		ADD	$(Sel. + DBi)$
1	1	0	0		EOR	$(Sel. \oplus DBi)$
1	1	0	1		OR	$(Sel. \vee DBi)$
1	1	1	0		AND	$(Sel. \wedge DBi)$
1	1	1	1		SUB	$(Sel. - DBi)$

reg(0:2)

reg(2) : 右記参照
reg(1) : 右記参照
reg(0) : 右記参照

(2)	(1)	(0)				
0	0	0	シフトの種類は	Right	Arithmetically	(TCF := b0)
0	0	1		Left	Arithmetically	(TCF := b7)
0	1	0		Right	Logically	(TCF := b0)
0	1	1		Left	Logically	(TCF := b7)
1	0	0	ローテートの種類は	Right	Arithmetically	(TCF := b0)
1	0	1		Left	Arithmetically	(TCF := b7)
1	1	0		Right	Logically	(TCF := b0)
1	1	1		Left	Logically	(TCF := b7)

acc(0:3)

acc(3) : 0 の時, ACC は OB にそのデータを出力する
acc(2) : 0 の時, ACC は DBi にそのデータを出力する
acc(1) : 0 の時, ACC は そのデータをシフトする.
TCF は ACC の b7 または b0 をラッチする
(シフトの種類と TCF は, reg(0:2) で指定する)
acc(0) : 0 の時, ACC は DBo 上のデータをラッチする

ix(0:3)

ix(3) : 0 の時, IX は 観測バス OB にそのデータを出力する
ix(2) : 0 の時, IX は データバス DBi にそのデータを出力する
ix(1) : 0 の時, IX は そのデータの値をシフトする
TCF は IX の b7 または b0 をラッチする
(シフトの種類と TCF は, reg(0:2) で指定する)
ix(0) : 0 の時, IX は DBo 上のデータをラッチする

phasecut

phasecut : 0 の時, クロックジェネレータは P0 に戻る
(各命令実行の最後のクロックフェーズで 0)

halt

halt : 0 の時, OP は 0 にセットする
(Halt 命令実行の最後のクロックフェーズ P2 あるいは 各命令の未定義フェーズで 0)

omem(0:2)

omem(2) : 1 の時, OB に外部メモリのデータを出力する
(MEM_OB に invert して接続)
omem(1) : 1 の時, DBi に外部メモリのデータを出力する
(MEM_RE に invert して接続)
omem(0) : 1 の時, DBo のデータが外部メモリに書き込まれる
(MEM_WE に invert して接続)

ibuf(0:1)

ibuf(1) : 1 の時, IBUF のフラグをリセットする
(IBUF_FLG_CLR に invert して接続)
ibuf(0) : 1 の時, IBUF はデータバス DBi にそのデータを出力する
(IBUF_RE に invert して接続)

panel

panel : 1 の時, DATA_SW はデータバス DBi にそのデータを出力する
(DATA_RE に invert して接続)

obuf

obuf : 0 から 1 に変化する時, OBUF にデータバス DBo のデータがセットされる
(OBUF_WE に invert して接続)

8.2 内部論理式

8.2.1 内部変数

b(0:7)

$$b(0) = \overline{s(3)} \cdot \overline{s(2)} \cdot \overline{s(1)} \cdot \overline{s(0)}$$
$$b(1) = s(3) \cdot \overline{s(2)} \cdot \overline{s(1)} \cdot \overline{s(0)}$$
$$b(2) = \overline{s(3)} \cdot s(2) \cdot \overline{s(1)} \cdot \overline{s(0)}$$
$$b(3) = s(3) \cdot s(2) \cdot \overline{s(1)} \cdot \overline{s(0)}$$
$$b(4) = s(2) \cdot \overline{s(1)} \cdot s(0)$$
$$b(5) = s(2) \cdot s(1)$$
$$b(6) = s(1)$$
$$b(7) = s(0) \cdot [\ \overline{s(2)} + s(1)\ ]$$

b(0)	b(1)	b(2)	b(3)	b(4)	b(5)	b(6)	b(7)	
1	0	0	0	0	0	0	0	BA
0	1	0	0	0	0	0	0	BVF
0	0	1	0	0	0	0	0	BNI
0	0	0	1	0	0	0	0	BNO
0	0	0	0	1	0	0	0	BNC, BC
0	0	0	0	0	0	0	1	BNZ, BZ
0	0	0	0	0	0	1	0	BZP, BN
0	0	0	0	0	0	1	1	BP, BZN
0	0	0	0	0	1	1	0	BGE, BLT
0	0	0	0	0	1	1	1	BGT, BLE

cond

$$\begin{aligned} cond = {} & b(0) + b(1) \cdot flag(2) + b(2) \cdot \overline{ibuf\_flg\_in} + b(3) \cdot obuf\_flg\_in \\ & + [\ b(4) \cdot flag(3) + [\ b(5) \cdot flag(2) \oplus b(6) \cdot flag(1)\ ] + b(7) \cdot flag(0)\ ] \oplus [\ \overline{s(3)} \cdot [\ s(0) + s(1)\ ]\ ] \end{aligned}$$

ob_sel(0:3) のデコード結果

		(3)	(2)	(1)	(0)	
ob_mc_p	=	0	0	0	0	
ob_mc_d	=	0	0	0	1	
ob_pc	=	0	0	1	0	
ob_flag	=	0	0	1	1	
ob_acc	=	0	1	0	0	
ob_ix	=	0	1	0	1	
ob_dbi	=	0	1	1	0	
ob_dbo	=	0	1	1	1	
ob_mar	=	1	0	0	0	
ob_ir	=	1	0	0	1	
ob_cont0	=	1	0	1	0	imem(0:2) , mar(0:4)
ob_cont1	=	1	0	1	1	pc(0:2) , ir(0:1) , ios(0:1) , flag_ob
ob_cont2	=	1	1	0	0	obc(0:1) , cfset(0:2) , vfset(0:2)
ob_cont3	=	1	1	0	1	phasecut , halt , nfset(0:2) , zfset(0:2)
ob_cont4	=	1	1	1	0	alu , alu_ope(0:3) , reg(0:2)
ob_cont5	=	1	1	1	1	acc(0:3) , ix(0:3)

その他

panel_set	=	$\overline{op} \cdot set$
direct_index	=	q(9,10,11,12,15,16)
pc_mar	=	p(0) + p(2) · [q(5,8,14) + direct_index]
arith_logic	=	p(2) · q(13) + p(3) · q(14) + p(4) · q(15,16)
index	=	p(3) · q(10,12,16)
reg_load	=	p(2) · q(7) + p(3) · q(8) + p(4) · q(9,10) + arith_logic · $\overline{s(6) \cdot s(5) \cdot s(4)}$
set_flag	=	panel_set · ob_flag
shift_flag	=	p(3) · q(6)

8.2.2 出力変数

A 観測バス OB の制御信号
B データバス DBi の制御信号
C データ更新用制御信号
D その他

ob(0:7)

D 6 個の 3-state buffer を ob_cont0~5 で制御して， wired-or をとる

ab8

D $ab8 = \overline{op} \cdot ob\_mc\_d + \overline{\overline{op} \cdot ob\_mc\_p} \cdot p(4) \cdot direct\_index \cdot s(0)$

imem(0:2)

A $\overline{imem(2)} = ob\_mc\_p + ob\_mc\_d$
B $\overline{imem(1)} = \overline{panel\_set} \cdot acc(2) \cdot ix(2) \cdot \overline{ibuf(0)} \cdot mem\_sel$
C $\overline{imem(0)} = panel\_set \cdot [\ ob\_mc\_p + ob\_mc\_d\ ] \cdot mem\_sel + op \cdot p(4) \cdot q(11,12) \cdot mem\_sel$

mar(0:4)

C $\overline{mar(4)} = \overline{op} \cdot adr\_dec$
C $\overline{mar(3)} = \overline{op} \cdot adr\_inc$
A $\overline{mar(2)} = ob\_mar$
D $mar(1) = pc\_mar \cdot \overline{panel\_set \cdot ob\_mar}$
C $\overline{mar(0)} = panel\_set \cdot ob\_mar + op \cdot [\ pc\_mar + p(3) \cdot direct\_index\ ]$

pc(0:2)

A $\overline{pc(2)} = ob\_pc$
C $\overline{pc(1)} = op \cdot pc\_mar$
C $\overline{pc(0)} = panel\_set \cdot ob\_pc + op \cdot p(3) \cdot q(5) \cdot cond$

ir(0:1)

A $\overline{ir(1)} = ob\_ir$
C $\overline{ir(0)} = panel\_set \cdot ob\_ir + op \cdot p(1)$

ios(0:1)

B $\overline{\text{ios}(1)} = \text{omem}(1) + \text{ibuf}(0) + \text{panel}$

D $\overline{\text{ios}(0)} = \text{mem\_chk}$

obc(0:1)

A $\overline{\text{obc}(1)} = \text{ob\_dbi}$

A $\overline{\text{obc}(0)} = \text{ob\_dbo}$

flag_ob

A $\overline{\text{flag\_ob}} = \text{ob\_flag}$

cfset(0:2)

C $\overline{\text{cfset}(2)} = \text{set\_flag} + \text{op} \cdot [\ \text{p}(2) \cdot \text{q}(4) + \text{shift\_flag} + \text{arith\_logic} \cdot \overline{\text{s}(6)} \cdot \overline{\text{s}(5)}\ ]$

D $\overline{\text{cfset}(1)} = \text{set\_flag} + \text{p}(2) \cdot \text{q}(4)$

D $\text{cfset}(0) = \text{set\_flag} \cdot \text{dbi}(3) + \overline{\text{set\_flag} \cdot \overline{\text{dbi}(3)}} \cdot [\ \text{p}(2) \cdot \text{q}(4) \cdot \text{s}(3) + \text{shift\_flag}\ ]$

vfset(0:2)

C $\overline{\text{vfset}(2)} = \text{set\_flag} + \text{op} \cdot [\ \text{shift\_flag} + \text{arith\_logic}\ ]$

D $\overline{\text{vfset}(1)} = \text{set\_flag} + \text{shift\_flag} \cdot [\ \text{s}(1) + \overline{\text{s}(0)}\ ] + \text{arith\_logic} \cdot [\ \text{s}(6) \cdot [\ \overline{\text{s}(5)} + \overline{\text{s}(4)}\ ]\ ]$

D $\text{vfset}(0) = \text{set\_flag} \cdot \text{dbi}(2) + \overline{\text{set\_flag} \cdot \overline{\text{dbi}(2)}} \cdot \text{shift\_flag} \cdot \overline{\text{s}(1)} \cdot \text{s}(0)$

nfset(0:2)

C $\overline{\text{nfset}(2)} = \text{set\_flag} + \text{op} \cdot [\ \text{shift\_flag} + \text{arith\_logic}\ ]$

D $\overline{\text{nfset}(1)} = \text{set\_flag}$

D $\text{nfset}(0) = \text{set\_flag} \cdot \text{dbi}(1)$

zfset(0:2)

C $\overline{\text{zfset}(2)} = \text{set\_flag} + \text{op} \cdot [\ \text{shift\_flag} + \text{arith\_logic}\ ]$

D $\overline{\text{zfset}(1)} = \text{set\_flag}$

D $\text{zfset}(0) = \text{set\_flag} \cdot \text{dbi}(0)$

alu

D $\text{alu} = \text{s}(3) + \text{index}$

alu_ope(0:3)

D $\text{alu\_ope}(3) = \overline{\text{panel\_set}} \cdot [\ \text{arith\_logic} + \text{index}\ ]$

D $\text{alu\_ope}(2) = \overline{\text{index}} \cdot \text{s}(6)$

D $\text{alu\_ope}(1) = \text{s}(5) + \text{index}$

D $\text{alu\_ope}(0) = \text{s}(4) + \text{index}$

reg(0:2)

D $reg(2) = s(2)$

D $reg(1) = s(1)$

D $reg(0) = s(0)$

acc(0:3)

A $\overline{acc(3)} = ob\_acc$

B $\overline{acc(2)} = \overline{panel\_set} \cdot [\ p23 \cdot q(2) + [\ shift\_flag + p(4) \cdot q(11,12)\ ] \cdot \overline{s(3)} + p(2) \cdot q(7,13) \cdot \overline{s(0)}\ ]$

C $\overline{acc(1)} = op \cdot p(2) \cdot q(6) \cdot \overline{s(3)}$

C $\overline{acc(0)} = panel\_set \cdot ob\_acc + op \cdot [\ p(2) \cdot q(3) + reg\_load \cdot \overline{s(3)}\ ]$

ix(0:3)

A $\overline{ix(3)} = ob\_ix$

B $\overline{ix(2)} = \overline{panel\_set} \cdot [\ [\ shift\_flag + p(4) \cdot q(11,12)\ ] \cdot s(3) + p(2) \cdot q(7,13) \cdot s(0)\ ]$

C $\overline{ix(1)} = op \cdot p(2) \cdot q(6) \cdot s(3)$

C $\overline{ix(0)} = panel\_set \cdot ob\_ix + op \cdot reg\_load \cdot s(3)$

phasecut

C $\overline{phasecut} = op \cdot [\ p(2) \cdot q(0,1,4,7,13) + p(3) \cdot q(2,3,5,6,8,14) + p(4) \cdot direct\_index\ ]$

halt

C $\overline{halt} = op \cdot [\ p(2) \cdot q(1) + p(3) \cdot q(0,1,4,7,13) + p(4) \cdot [\ q(0,1,4,7,13) + q(2,3,5,6,8,14)\ ]\ ]$

omem(0:2)

A $omem(2) = [\ ob\_mc\_p + ob\_mc\_d\ ] \cdot \overline{mem\_sel}$

B $omem(1) = \overline{panel\_set} \cdot acc(2) \cdot ix(2) \cdot \overline{ibuf(0)} \cdot \overline{mem\_sel}$

C $omem(0) = panel\_set \cdot [\ ob\_mc\_p + ob\_mc\_d\ ] \cdot \overline{mem\_sel} + op \cdot p(4) \cdot q(11,12) \cdot \overline{mem\_sel}$

ibuf(0:1)

C $ibuf(1) = op \cdot p(3) \cdot q(3)$

B $ibuf(0) = \overline{panel\_set} \cdot p(2) \cdot q(3)$

panel

B $panel = panel\_set$

obuf

C $obuf = op \cdot p(3) \cdot q(2)$

8.3 出力表

8.3.1 観測バス OB の制御信号

$$\overline{\text{imem(2)}} = \text{ob\_mc\_p} + \text{ob\_mc\_d}$$
$$\overline{\text{mar(2)}} = \text{ob\_mar}$$
$$\overline{\text{pc(2)}} = \text{ob\_pc}$$
$$\overline{\text{ir(1)}} = \text{ob\_ir}$$
$$\overline{\text{obc(1)}} = \text{ob\_dbi}$$
$$\overline{\text{obc(0)}} = \text{ob\_dbo}$$
$$\overline{\text{flag\_ob}} = \text{ob\_flag}$$
$$\overline{\text{acc(3)}} = \text{ob\_acc}$$
$$\overline{\text{ix(3)}} = \text{ob\_ix}$$
$$\text{omem(2)} = [\ \text{ob\_mc\_p} + \text{ob\_mc\_d}\ ] \cdot \overline{\text{mem\_sel}}$$

$$\text{ob\_cont0} = \text{ob\_cont0}$$
$$\text{ob\_cont1} = \text{ob\_cont1}$$
$$\text{ob\_cont2} = \text{ob\_cont2}$$
$$\text{ob\_cont3} = \text{ob\_cont3}$$
$$\text{ob\_cont4} = \text{ob\_cont4}$$
$$\text{ob\_cont5} = \text{ob\_cont5}$$

8.3.2 データバス DBi の制御信号

$$\text{panel} = \text{panel\_set}$$
$$\text{ibuf(0)} = \overline{\text{panel\_set}} \cdot \text{p(2)} \cdot \text{q(3)}$$
$$\overline{\text{acc(2)}} = \overline{\text{panel\_set}} \cdot [\ \text{p23} \cdot \text{q(2)} + [\ \text{shift\_flag} + \text{p(4)} \cdot \text{q(11,12)}\ ] \cdot \overline{\text{s(3)}} + \text{p(2)} \cdot \text{q(7,13)} \cdot \overline{\text{s(0)}}\ ]$$
$$\overline{\text{ix(2)}} = \overline{\text{panel\_set}} \cdot [\ [\ \text{shift\_flag} + \text{p(4)} \cdot \text{q(11,12)}\ ] \cdot \text{s(3)} + \text{p(2)} \cdot \text{q(7,13)} \cdot \text{s(0)}\ ]$$
$$\overline{\text{imem(1)}} = \overline{\text{panel\_set}} \cdot \text{acc(2)} \cdot \text{ix(2)} \cdot \overline{\text{ibuf(0)}} \cdot \text{mem\_sel}$$
$$\text{omem(1)} = \overline{\text{panel\_set}} \cdot \text{acc(2)} \cdot \text{ix(2)} \cdot \overline{\text{ibuf(0)}} \cdot \overline{\text{mem\_sel}}$$
$$\overline{\text{ios(1)}} = \text{omem(1)} + \text{ibuf(0)} + \text{panel}$$

	panel_set	$\overline{\text{panel\_set}} \cdot \text{p(0)}$	$\overline{\text{panel\_set}} \cdot \text{p(1)}$	$\overline{\text{panel\_set}} \cdot \text{p(2)}$	$\overline{\text{panel\_set}} \cdot \text{p(3)}$	$\overline{\text{panel\_set}} \cdot \text{p(4)}$
q(0)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		—	—
q(1)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		—	—
q(2)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{acc(2)}}$		—
q(3)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	ibuf(0)		—
q(4)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		—	—
q(5)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	—
q(6)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{acc(2)}} \oplus \overline{\text{ix(2)}}$	—
q(7)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{acc(2)}} \oplus \overline{\text{ix(2)}}$	—	—
q(8)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	—
q(9)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{imem(1)}} \oplus \text{omem(1)}$
q(10)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{imem(1)}} \oplus \text{omem(1)}$
q(11)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{acc(2)}} \oplus \overline{\text{ix(2)}}$
q(12)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{acc(2)}} \oplus \overline{\text{ix(2)}}$
q(13)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{acc(2)}} \oplus \overline{\text{ix(2)}}$	—	—
q(14)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	—
q(15)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{imem(1)}} \oplus \text{omem(1)}$
q(16)	panel		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$		$\overline{\text{imem(1)}} \oplus \text{omem(1)}$	$\overline{\text{imem(1)}} \oplus \text{omem(1)}$

空白と — は $\overline{\text{imem(1)}} \oplus \text{omem(1)}$

8.3.3 データ更新用制御信号

$\overline{\text{imem(0)}} = \text{panel\_set} \cdot [\ \text{ob\_mc\_p} + \text{ob\_mc\_d}\ ] \cdot \text{mem\_sel} + \text{op} \cdot \text{p(4)} \cdot \text{q(11,12)} \cdot \text{mem\_sel}$

$\overline{\text{mar(4)}} = \overline{\text{op}} \cdot \text{adr\_dec}$

$\overline{\text{mar(3)}} = \overline{\text{op}} \cdot \text{adr\_inc}$

$\overline{\text{mar(0)}} = \text{panel\_set} \cdot \text{ob\_mar}$

$\overline{\text{pc(1)}} = \text{op} \cdot \text{pc\_mar}$

$\overline{\text{pc(0)}} = \text{panel\_set} \cdot \text{ob\_pc} + \text{op} \cdot \text{p(3)} \cdot \text{q(5)} \cdot \text{cond}$

$\overline{\text{ir(0)}} = \text{panel\_set} \cdot \text{ob\_ir} + \text{op} \cdot \text{p(1)}$

$\overline{\text{cfset(2)}} = \text{set\_flag} + \text{op} \cdot [\ \text{p(2)} \cdot \text{q(4)} + \text{shift\_flag} + \text{arith\_logic} \cdot \overline{\text{s(6)}} \cdot \overline{\text{s(5)}}\ ]$

$\overline{\text{vfset(2)}} = \text{set\_flag} + \text{op} \cdot [\ \text{shift\_flag} + \text{arith\_logic}\ ]$

$\overline{\text{nfset(2)}} = \text{set\_flag} + \text{op} \cdot [\ \text{shift\_flag} + \text{arith\_logic}\ ]$

$\overline{\text{zfset(2)}} = \text{set\_flag} + \text{op} \cdot [\ \text{shift\_flag} + \text{arith\_logic}\ ]$

$\overline{\text{acc(1)}} = \text{op} \cdot \text{p(2)} \cdot \text{q(6)} \cdot \overline{\text{s(3)}}$

$\overline{\text{acc(0)}} = \text{panel\_set} \cdot \text{ob\_acc} + \text{op} \cdot [\ \text{p(2)} \cdot \text{q(3)} + \text{reg\_load} \cdot \overline{\text{s(3)}}\ ]$

$\overline{\text{ix(1)}} = \text{op} \cdot \text{p(2)} \cdot \text{q(6)} \cdot \text{s(3)}$

$\overline{\text{ix(0)}} = \text{panel\_set} \cdot \text{ob\_ix} + \text{op} \cdot \text{reg\_load} \cdot \text{s(3)}$

$\overline{\text{phasecut}} = \text{op} \cdot [\ \text{p(2)} \cdot \text{q(0,1,4,7,13)} + \text{p(3)} \cdot \text{q(2,3,5,6,8,14)} + \text{p(4)} \cdot \text{direct\_index}\ ]$

$\overline{\text{halt}} = \text{op} \cdot \text{p(2)} \cdot \text{q(1)}$

$\text{omem(0)} = \text{panel\_set} \cdot [\ \text{ob\_mc\_p} + \text{ob\_mc\_d}\ ] \cdot \overline{\text{mem\_sel}} + \text{op} \cdot \text{p(4)} \cdot \text{q(11,12)} \cdot \overline{\text{mem\_sel}}$

$\text{ibuf(1)} = \text{op} \cdot \text{p(3)} \cdot \text{q(3)}$

$\text{obuf} = \text{op} \cdot \text{p(3)} \cdot \text{q(2)}$

op = 0

	ob_mc_p	ob_mc_d	ob_mar	ob_pc	ob_ir	ob_flag	ob_acc	ob_ix	その他
set	$\overline{\text{imem(0)}} \oplus \text{omem(0)}$	$\overline{\text{imem(0)}} \oplus \text{omem(0)}$	$\overline{\text{mar(0)}}$	$\overline{\text{pc(0)}}$	$\overline{\text{ir(0)}}$	$\overline{\text{fset(2)}}$	$\overline{\text{acc(0)}}$	$\overline{\text{ix(0)}}$	
adr_dec	$\overline{\text{mar(4)}}$								
adr_inc	$\overline{\text{mar(3)}}$								

$\overline{\text{fset(2)}} = \overline{\text{cfset(2)}} \cdot \overline{\text{vfset(2)}} \cdot \overline{\text{nfset(2)}} \cdot \overline{\text{zfset(2)}}$

op = 1

	p(0)	p(1)	p(2)	p(3)	p(4)
q(0)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$		—	—
q(1)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{halt}}$	—	—
q(2)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$		obuf	—
q(3)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{acc}(0)}$	ibuf(1)	—
q(4)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{cfset}(2)}$	—	—
q(5)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{pc}(0)}$	—
q(6)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{acc}(1)} \oplus \overline{\text{ix}(1)}$	$\overline{\text{fset}(2)}$	—
q(7)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}$	—	—
q(8)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}$	—
q(9)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{mar}(0)}$	$\overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}$
q(10)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{mar}(0)}$	$\overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}$
q(11)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{mar}(0)}$	$\overline{\text{imem}(0)} \oplus \text{omem}(0)$
q(12)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{mar}(0)}$	$\overline{\text{imem}(0)} \oplus \text{omem}(0)$
q(13)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$[\ \overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}\ ] \cdot \overline{\text{fset}(2)}$	—	—
q(14)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$[\ \overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}\ ] \cdot \overline{\text{fset}(2)}$	—
q(15)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{mar}(0)}$	$[\ \overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}\ ] \cdot \overline{\text{fset}(2)}$
q(16)	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{ir}(0)}$	$\overline{\text{mar}(0)} \cdot \overline{\text{pc}(1)}$	$\overline{\text{mar}(0)}$	$[\ \overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}\ ] \cdot \overline{\text{fset}(2)}$

$\overline{\text{fset}(2)} = \overline{\text{cfset}(2)} \cdot \overline{\text{vfset}(2)} \cdot \overline{\text{nfset}(2)} \cdot \overline{\text{zfset}(2)}$

この他，各命令の最終フェーズで $\overline{\text{phasecut}}$ ，— で $\overline{\text{halt}}$

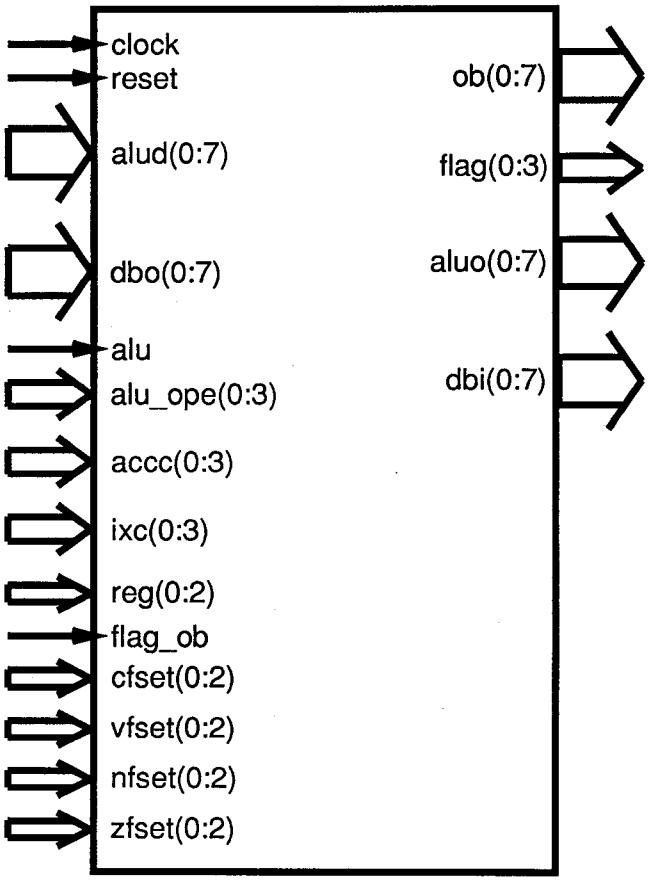
$\overline{\text{pc}(0)}$ は cond = 1 のときのみ成立

$[\ \overline{\text{acc}(0)} \oplus \overline{\text{ix}(0)}\ ] \cdot \overline{\text{fset}(2)}$ は $\overline{\text{s}(6) \cdot \text{s}(5) \cdot \text{s}(4)} = 1$ のときのみ成立 ， $\overline{\text{s}(6) \cdot \text{s}(5) \cdot \text{s}(4)} = 0$ のときは $\overline{\text{fset}(2)}$ となる

arith_logic の $\overline{\text{cfset}(2)}$ は $\overline{\text{s}(6)} \cdot \overline{\text{s}(5)} = 1$ のときのみ成立

第 9 章

ALU,ACC,IX 部仕様



9.1 入出力仕様

9.1.1 入力

clock : マスタークロック
reset : リセット信号
alud(0:7) : ALU の B 入力に直結
dbo(0:7) : ACC と IX の入力に接続
alu : ALU の A 入力の ACC と IX を 選択する
alu_ope(0:3) : ALU での演算を指定する
accc(0:3) : ACC を制御する
ixc(0:3) : IX を制御する
reg(0:2) : シフトの種類を指定する
flag_ob : 0 の時, フラグを OB に出力
cfset(0:2) : CF を制御する
vfset(0:2) : VF を制御する
nfset(0:2) : NF を制御する
zfset(0:2) : ZF を制御する

9.1.2 出力

ob(0:7) : 観測バス OB
flag(0:3) : 各 flag の状態を出力
aluo(0:7) : ALU の出力に直結
dbi(0:7) : ACC と IX の値を出力 (下表参照)

	op·p(0)	op·p(1)	op·p(2)	op·p(3)	op·p(4)
q(0)				—	—
q(1)				—	—
q(2)			$\overline{\text{accc(2)}}$, $\overline{\text{ixc(2)}}$		—
q(3)					—
q(4)				—	—
q(5)					—
q(6)				$\overline{\text{accc(2)}}$, $\overline{\text{ixc(2)}}$	—
q(7)			$\overline{\text{accc(2)}}$, $\overline{\text{ixc(2)}}$	—	—
q(8)					—
q(9)					
q(10)					
q(11)					$\overline{\text{accc(2)}}$, $\overline{\text{ixc(2)}}$
q(12)					$\overline{\text{accc(2)}}$, $\overline{\text{ixc(2)}}$
q(13)			$\overline{\text{accc(2)}}$, $\overline{\text{ixc(2)}}$	—	—
q(14)					—
q(15)					
q(16)					

9.2 内部仕様

9.2.1 blk_register

入力

clock : clock
reset : reset
cf : キャリーフラグ
reg(0:2) : reg(0:2)
control(0:3) : accc(0:3), ixc(0:3)
in(0:7) : dbo(0:7)

出力

out(0:7) : Sel. へ
dbi_out(0:7) : dbi(0:7)
ob(0:7) : ob(0:7)
tcr : シフトのキャリー

参考 : shift register(194r) の制御

reg				194r inp.		
(2)	(1)	(0)	動作	isr	isl	TC
0	0	0	SRA	b7	–	b0
0	0	1	SLA	–	0	b7
0	1	0	SRL	0	–	b0
0	1	1	SLL	–	0	b7
1	0	0	RRA	CF	–	b0
1	0	1	RLA	–	CF	b7
1	1	0	RRL	b0	–	b0
1	1	1	RLL	–	b7	b7

9.2.2 sel8bit

入力

a(0:7) : ACC の値の入力
b(0:7) : IX の値の入力
sl : alu
st : $\overline{\text{alu\_ope(3)}}$ (H の時, 出力を 0 に落す.)

出力

y(0:7) : ALU へ

9.2.3 blk_alu

入力

cf : キャリーフラグ
a(0:7) : Sel. から
b(0:7) : alud(0:7)
ope(0:3) : alu_ope(0:3)

出力

aluo(0:7) : aluo(0:7)
dbi_out(0:7) : dbi(0:7)
ov : ALU のオーバーフロー出力
cr : ALU のキャリー出力
ng : ALU のネガティブ出力
zr : ALU のゼロ出力

参考 : kuealu の制御

alu_ope				ALU の	kuealu control				kuealu	Carry
(3)	(2)	(1)	(0)	演算	S2	S1	S0	Cn	の動作	出力
0	–	–	–	DBi.	1	0	1	–	OR	–
1	0	0	0	SBC	0	0	0	$\overline{CF}$	MINUS	$\overline{cn8}$
1	0	0	1	ADC	0	0	1	CF	PLUS	cn8
1	0	1	0	SUB	0	1	0	1	MINUS	–
1	0	1	1	ADD	0	1	1	0	PLUS	–
1	1	0	0	EOR	1	0	0	–	EOR	–
1	1	0	1	OR	1	0	1	–	OR	–
1	1	1	0	AND	1	1	0	–	AND	–
1	1	1	1	SUB	1	1	1	1	MINUS	–

9.2.4 blk_status

入力

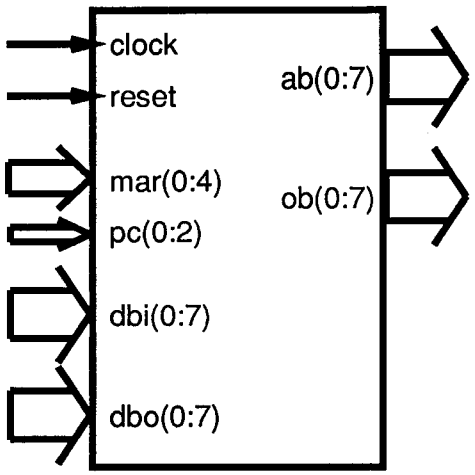
clock : clock
reset : reset
alu_ov : ALU からのオーバーフロー入力
alu_cr : ALU からのキャリー入力
alu_ng : ALU からのネガティブ入力
alu_zr : ALU からのゼロ入力
acc_tcr : ACC からのテンポラリーキャリー入力
ix_tcr : IX からのテンポラリーキャリー入力
flag_ob : flag_ob
cfset(0:2) : cfset(0:2)
vfset(0:2) : vfset(0:2)
nfset(0:2) : nfset(0:2)
zfset(0:2) : zfset(0:2)

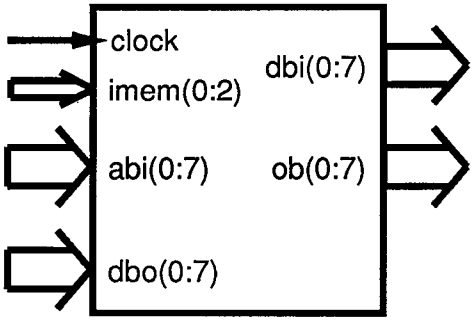
出力

flag(0:3) : flag(0:3)
ob(0:7) : ob(0:7)

第 10 章

PC，MAR 部，IMEM 部仕様





10.1 PC，MAR 部の入出力仕様

10.1.1 入力仕様

コントローラから

- mar(0:4)

  mar(4) : 0 の時, MAR は その値を 1 デクリメントする
  mar(3) : 0 の時, MAR は その値を 1 インクリメントする
  mar(2) : 0 の時, MAR は OB にその値を出力する
  mar(1) : 0 の時, Sel. は DBo を MAR の入力に接続
  1 の時, Sel. は PC を MAR の入力に接続
  mar(0) : 0 の時, MAR は PC または DBo 上のデータをラッチする
  (PC と DBo の選択は, mar(1) で指定する)

- pc(0:2)

  pc(2) : 0 の時, PC は その値を OB に出力する
  pc(1) : 0 の時, PC は その値を 1 インクリメントする
  pc(0) : 0 の時, PC は DBi 上のデータをラッチする

パネル (外部ピン) から

clock マスタークロック
reset リセット信号

内部バスから

dbi(0:7) DBi バス
dbo(0:7) DBo バス

10.1.2 出力仕様

ab(0:7)

ab(0:7) アドレス出力バス ABo

ob(0:7)

ob(0:7) 観測バス OB

10.2 IMEM 部の入出力仕様

10.2.1 入力仕様

コントローラから

- imem(0:2)

  imem(2) : 0 の時, OB に内蔵メモリのデータを出力する
  imem(1) : 0 の時, DBi に内蔵メモリのデータを出力する
  imem(0) : 0 の時, DBo 上のデータを内蔵メモリに書き込む

パネル (外部ピン) から

clock マスタークロック

内部バスから

abi(0:7) アドレス入力バス ABi
dbo(0:7) DBo バス

10.2.2 出力仕様

dbi(0:7)

dbi(0:7) DBi バス

ob(0:7)

ob(0:7) 観測バス OB

10.3 動作

MEM は, IMEM または OMEM(mem_sel=1 のとき前者)

10.3.1 共通事項

ob_mc_p	$\overline{\text{imem(2)}} \oplus \text{omem(2)}$	MEM → OB
ob_mc_d		
ob_mar	$\overline{\text{mar(2)}}$	MAR → OB
ob_pc	$\overline{\text{pc(2)}}$	PC → OB

10.3.2 停止中 (OP=0)

ob_mc_p·set	$\overline{\text{imem(0)}} \oplus \text{omem(0)}$	DBo → MEM
ob_mc_d·set		
ob_mar·set	$\overline{\text{mar(1)}}$ $\overline{\text{mar(0)}}$	DBo → SELMAR SELMAR → MAR
ob_pc·set	$\overline{\text{pc(0)}}$	DBi → PC
adr_inc	$\overline{\text{mar(3)}}$	MAR++
adr_dec	$\overline{\text{mar(4)}}$	MAR−−

10.3.3 動作中または停止待機中 (OP=1 または set=0)

	p(0)	p(1)	p(2)	p(3)	p(4)
q(0)	PC → SELMAR (mar(1)) SELMAR → MAR* ($\overline{\text{mar(0)}}$) PC++* ($\overline{\text{pc(1)}}$)	MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)		—	—
q(1)				—	—
q(2)					—
q(3)					—
q(4)				—	—
q(5)			PC → SELMAR (mar(1)) SELMAR → MAR* ($\overline{\text{mar(0)}}$) PC++* ($\overline{\text{pc(1)}}$)	MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$) DBi → PC* ** ($\overline{\text{pc(0)}}$)	—
q(6)					—
q(7)				—	—
q(8)			PC → SELMAR (mar(1)) SELMAR → MAR* ($\overline{\text{mar(0)}}$) PC++* ($\overline{\text{pc(1)}}$)	MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)	—
q(9)				MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)	MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)
q(10)					
q(11)				DBo → SELMAR ($\overline{\text{mar(1)}}$) SELMAR → MAR* ($\overline{\text{mar(0)}}$)	DBo → MEM* ($\overline{\text{imem(0)}} \oplus \text{omem(0)}$)
q(12)					
q(13)				—	—
q(14)			PC → SELMAR (mar(1)) SELMAR → MAR* ($\overline{\text{mar(0)}}$) PC++* ($\overline{\text{pc(1)}}$)	MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)	—
q(15)				MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$) DBo → SELMAR ($\overline{\text{mar(1)}}$) SELMAR → MAR* ($\overline{\text{mar(0)}}$)	MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)
q(16)					

註 * : op=1 のときのみ

** : cond=1 のときのみ

このほか, $\overline{\text{acc(2)}}$ または $\overline{\text{ix(2)}}$ または ibuf(0) の場合を除き, 常に MEM → DBi ($\overline{\text{imem(1)}} \oplus \text{omem(1)}$)

第 11 章

KUE-CHIP2 設計誤り検出テスト及びテストベクトルの生成

11.1 テストの目的

一般に LSI の設計において，LSI が所望の機能を発揮できるか否かをテストすることはその LSI 自体の回路設計と同等の重要性を持ち，同時に回路設計と同等またはそれ以上の時間及び手間を必要とする．KUE-CHIP2 の設計においても当然のことながら多大なテストがなされている．本章では KUE-CHIP2 の設計段階のテスト及び製造段階のテストとしてどのようなことがなされたかを述べる．

LSI の設計の故障には，設計の誤りによって起こる設計故障 (バグ) と LSI の製造の誤差や動作中の劣化によって起こる物理故障がある．よって LSI のテストの目的を以下の 3 通りに分けて考える必要が生じる．

1. 設計の誤りの検出
2. 製造時の不良品の検出
3. 動作中の誤動作の検出

1,2に関しては以下で詳しく述べるが，3に関しては LSI の設計段階では如何ともし難い．しかし KUE-CHIP2 は，CPU 内に存在するほとんどのレジスタの内容や，主だった信号線の値を随時外部から観測できるように設計されている．よって，3の予測不能な原因による誤動作も容易に観測可能である．

11.2 設計誤り検出テスト

ここでは KUE-CHIP2 の設計において設計の誤りを検出し訂正するためにどのようなテストがなされたかを詳しく挙げる．

11.2.1 CPU 制御部のテスト

reset 信号のテスト

1. 書き込み可能な全てのフリップフロップ即ち，FLAG,ACC,IX,PC,MAR,IR を構成するフリップッフロップに 0 を書き込む．
2. 1で書き込んだフリップフロップが 0 かチェックする．
3. 書き込み可能な全てのフリップフロップに 1 を書き込む．
4. 3で書き込んだフリップフロップが 1 かチェックする．
5. リセットする．
6. 読み込み可能な全てのフリップフロップが 0 かチェックする．

set,adr_inc,adr_dec 信号のテスト

1. リセットする.
2. MAR を読み出し, 00000000 かどうかチェックする.
3. AB に 00000000 が出ているかチェックする.
4. adr_inc を 1 回入力する.
5. MAR を読み出し, 00000001 かどうかチェックする.
6. AB に 00000001 が出ているかチェックする.
7. DBi に 00000000 を乗せ, set を入力する. DATE_RE,MEM_WE を確認する.
8. MC を読み出し, 00000000 かチェックする.
9. DBi に 11111111 を乗せ, set を入力する.
10. MC を読み出し, 11111111 かチェックする.
11. MEM_CHK を 0 にして, 所定のアドレスを外部ピンから入力し, 実際のメモリの内容をチェックする.
12. adr_inc を 100 回入力する.
13. MAR を読み出し, 01100101 かどうかチェックする.
14. AB に 01100101 が出ているかチェックする.
15. 7から 11を繰り返す.
16. adr_dec に対しても同様にチェックする.

halt 信号のテスト

1. リセットする.
2. 0 番地に 00001111(HALT) をかく.
3. ss を入力する.
4. CPU の停止を確認する.

外部メモリコントロール信号のテスト

1. P0,P1,P2,P3,P4
   5 クロックの命令を実行することでチェックする.
2. MEM_OB
   MEM_SEL=0 とし, OBSEL=0,1 の時のみ 0 となることを確認する.
3. MEM_WE, MEM_RE
   MEM_SEL=0 とし, あるアドレスにデータを書き込み, 同時に MEM_WE,MEM_RE が正しいか確認する.
4. PANEL_RE
   MEM_SEL=1 とし, あるレジスタにデータを書き込み, 同時に PANEL_RE が正しいか確認する.

内部コントロール信号のテスト

1. リセットする.
2. SP を 2 回入力し P2 にする.
3. IR にある命令に対応する値を書き込む.
4. OBSEL が 10~15 の間で変化させ，すべてのコントロール信号をチェックする.
5. 1から 4を 17 種類の命令すべてに対して行なう.
6. 同様に IR に値を書き込み， SS を入力することで CPU の動作状態 (OP=1) でもチェックする.

11.2.2 レジスタのテスト

IR

1. リセットする.
2. ddd = 00,01,···,ff に対して,
   (a) IR に ddd を書き込む.
   (b) IR を読み出し， ddd かどうかチェックする.
   (c) q(0:16) を読み出し，確認する.
3. ddd = ff,fe,···,00 でも繰り返し確認する.

ACC,IX

1. リセットする.
2. ACC に 00000000 を書き込む.
3. ACC を読み出し，値を確認する.
4. ACC に 00000001 を書き込み，値を確認する.
5. ACC に 00000011 を書き込み，値を確認する.
6. 11111111 まで繰り返す.
7. ビットを反転して繰り返す.
8. IX でも同様に行なう.

FLAG

1. リセットする.
2. FLAG を読み出し，値を確認する.
3. FLAG に 0001 を書き込み，値を確認する.
4. 1111 まで繰り返す.
5. ビットを反転して繰り返す.

11.2.3 命令のテスト

RCF,SCF

1. リセットする. (TCF=0 とするため.)
2. FLAG に 00000000 を書き込む.
3. RCF を実行し, FLAG=00000000 を確認する.
4. SCF を実行し, FLAG=00001000 を確認する.
5. FLAG に 11111111 を書き込む. (パネルから)
6. SCF を実行し, FLAG=00001111 を確認する.
7. RCF を実行し, FLAG=00000111 を確認する.

LD,ST, アドレスモード

1. LD ACC,3a ; LD IX,d5 を行ない ACC,IX をチェックする.
2. LD ACC,3a ; LD IX,d5 ; LD IX,ACC を行ない ACC,IX をチェックする.
3. いくつかの adr に対して, LD ACC,00 ; ST ACC,(adr) ; LD IX,(adr) を行ない ACC,IX,(adr) をチェックする.
4. (adr) を [adr] にかえて 3を行なう.
5. ACC と IX を入れ換えて, 2から 4を行なう.
6. LD ACC,3a ; LD IX,aaa ; ST ACC,(IX+ppp) ; ST IX,(IX+qqq) ; LD ACC,(IX+qqq) ; LD IX,(IX+ppp) を行ない, ACC と IX が入れ替わったか確認する.
7. (adr) を [adr] にかえて 6を行なう.
8. これらすべてを, ビットを反転させて繰り返す.

算術演算のテスト

1. xxx,yyy={00000000,01010101,11111111,01111111,10000000} の全ての組み合わせについて,
   FLAG = 0000 ; LD ACC,xxx ; LD IX,yyy ; ADD ACC,IX を行ない, ACC,IX,FLAG が正しいかチェックする.
2. SUB,EOR,OR,AND,CMP,SBC,ADC でも同様にチェックする.
3. IX と ACC を入れ替えて再度チェックする.
4. FLAG=1111 の状態で, 1から 3を再度行なう.
5. SBC,ADC については, FLAG={0000,0001,0011,0111,1111,1110,1100,1000} で行なう.
6. いくつかの xxx,yyy,adr にたいして,
   - LD ACC,xxx ; LD IX,yyy ; ST IX,(adr) ; ADD ACC,(adr) をおこない, 同様にチェックする.
   - LD ACC,xxx ; LD IX,yyy ; ST IX,[adr] ; ADD ACC,[adr] をおこない, 同様にチェックする.

- LD ACC,xxx ; LD IX,yyy ; ST IX,(IX+adr) ; ADD ACC,(IX+adr) をおこない，同様にチェックする.
- LD ACC,xxx ; LD IX,yyy ; ST IX,[IX+adr] ; ADD ACC,[IX+adr] をおこない，同様にチェックする.

シフト命令のテスト

1. xxx={00000000,10101010,11111110,11111111,10000000} について,
   FLAG = 0000 ; LD ACC,xxx ; LD IX,xxx ; SRA ACC を行ない， ACC,FLAG が正しく IX が変化していないか確認する.
2. SLA,SRL,SLL,RRA,RLA,RRL,RLL でも同様にチェックする.
3. IX と ACC を入れ替えて再度チェックする.
4. FLAG = {0111,1000} についても同様にチェックする.

ブランチ命令のテスト

1. リセットする.
2. BA 00 をシングルインストラクションで実行し， PC が 00 であることを確認する.
3. 01,03,07,0f,1f,3f,7f,ff でも同様に確認する.
4. FLAG = 0000 ; BA adr ; を行ない， PC の値を確認する.
5. BVF,BNZ,BZ,BZP,BN,BP,BZN,BNC,BC,BGE,BLT,BGT,BLE でも同様にチェックする.
6. 全ての FLAG の状態 (16 通り) で確認する.
7. IBUF_FLG_IN = 1 とし， BNI adr を実行する． PC = adr を確認する.
8. IBUF_FLG_IN = 0 とし， BNI adr を実行する． PC = PC + 1 を確認する.
9. OBUF_FLG_IN = 1 とし， BNO adr を実行する． PC = adr を確認する.
10. OBUF_FLG_IN = 0 とし， BNO adr を実行する． PC = PC + 1 を確認する.

11.2.4 外部入出力のテスト

IN,OUT

1. リセットする.
2. IN をシングルフェイズで実行する.

   P2 DBi に 00000000 をのせ， IBUF_RE = 0, ACC = 00000000 を確認する.

   P3 IBUF_FLG_CLR = 0 を確認する.
3. 2を 00000001,00000011,$\cdots$,11111111,11111110,$\cdots$,10000000 で繰り返す.
4. ACC = 00000000 とし， OUT をシングルフェイズで実行する.

   P2 DBo = ACC を確認する.

P3 OBUF_WE = 0 を確認する.

5. 4を 00000001,00000011,···,11111111,11111110,···,10000000 で繰り返す.

11.2.5 RAM のテスト

1. MEM_CHK を 0 にし，アドレスバスは外部ピンから入力するようにする.
2. 全ての RAM に 00000000 を書き込む.
3. 全ての RAM を読み出し， 00000000 かチェックする.
4. n=00~ff に対し,
   - n 番地に 10101010 を書き込む.
   - n 番地を読み込み， 10101010 かどうかチェックする.
   - (n-1) 番地が， 10101010 かどうかチェックする.
   - (n+1) 番地が， 00000000 かどうかチェックする.
5. 4を， 01010101 で繰り返す.

11.3 テストベクトルの生成

一般に LSI チップは非常に良品率が低く，製造後の良品チェックは非常に重要である．その良品チェックに用いられる入出力パターンのことをテストベクトルという．ここでは KUE-CHIP2 に製造後どのようなテストベクトルでチェックされたかを詳しく述べる.

なお，テストベクトルはそのチェック機能を落さない範囲でもっとも短いことが望ましい．よって，先の設計誤りの検出とは別に専用のテストベクトルを作成するのが望ましいが，今回は設計誤りの検出プログラムをもとにテストベクトルを生成した．これは，設計誤りの検出プログラムから冗長な部分を削り取り，入出力端子の電流電圧特性を検査するためのベクトルを付加することで実現した.

11.3.1 CPU 制御部のテスト

reset 信号のテスト

設計誤り検出テストと同様である.

set,adr_inc,adr_dec 信号のテスト

設計誤り検出テストと同様である.

外部メモリコントロール信号のテスト

1. MEM_OB
   MEM_SEL=0 とし， OBSEL=0,1 の時のみ 0 となることを確認する.
2. MEM_WE, MEM_RE
   MEM_SEL=0 とし，あるアドレスにデータを書き込み，同時に MEM_WE,MEM_RE が正しいか確認する.
3. PANEL_RE
   MEM_SEL=1 とし，あるレジスタにデータを書き込み，同時に PANEL_RE が正しいか確認する.

内部コントロール信号のテスト

設計誤り検出テストと同様である.

11.3.2 命令のテスト

RCF,SCF

設計誤り検出テストと同様である.

LD,ST, アドレスモード

設計誤り検出テストと同様である.

算術演算のテスト

xxx,yyy={01111111,10000000} のそれぞれの組み合わせについて，設計誤り検出テストと同様のテストを行なう.

シフト命令のテスト

yyy={01111111,10000000} について，設計誤り検出テストと同様のテストを行なう.

ブランチ命令のテスト

設計誤り検出テストと同様である.

11.3.3 外部入出力のテスト

IN,OUT

設計誤り検出テストと同様である.

11.3.4 RAM のテスト

設計誤り検出テストと同様である.

第 12 章

KUE-CHIP2 形式的設計検証

12.1 形式的論理設計検証

論理設計の段階で設計した回路が設計者の意図通りの動作をするかどうかを確認する作業，すなわち論理設計検証は，設計の下流工程からの手戻りを可能な限り防ぐために不可欠な作業である．従来，この目的のために，論理シミュレーション手法を用いて，8章で示したような設計誤り検出テストが行なわれてきた．しかし，KUE-CHIP2のような比較的小規模の回路に対してすら，全ての場合をカバーする入力パターンの生成は困難である．例えば，バスの競合やフロートが，どのような状況でも起こり得ないことを保証するような入力パターンの集合を生成することは，順序回路のような時間動作を含む対象にたいしては，極めて難しい．実際，KUE-CHIP2 においても，以下で述べる形式的検証手法を適用して始めて，バスへの出力が全てハイインピーダンス状態となる場合のあることが発見された[1]．

論理シミュレーションが，与えられたパターンに対してのみの動作の確認に用いられるのに対して，形式的検証手法は意味の明確な言語によって記述された性質を，設計された対象がかならず満足することを保証しようというものである．順序回路を対象とした形式的検証は，主として2つの順序回路の等価性判定を行うものと，時相論理と呼ばれる論理体系を仕様記述言語として用いて仕様が満足されるかどうかを判定するものに分けられる．本章で扱う形式的検証は，時相論理を仕様記述言語として用いたものであり，本質的には全ての場合を網羅的に調べる手法に基づいている．具体的には順序回路から状態遷移図を構成し，この遷移図の性質を調べることにより，仕様を満足するかどうかを調べることになる．フリップフロップを n 個持つ完全同期式の順序回路を考えた場合，最大 2^n 個の状態が遷移図に出現する．すなわち，フリップフロップ数に対する指数爆発が起こる．このため，形式的検証手法は小規模な回路に対してのみ適用可能であった．

一方，論理関数の新しい表現，処理のための基本的な手段として，Akers や Bryant によって定式化された二分決定グラフ (Binary Decision Diagram, BDD) が，用いられるようになってきている [1,2]．これは 高性能の BDD 処理パッケージが開発され，実用的な多くの場合に，論理関数を扱う際の指数爆発を BDD によって緩和しうることが明らかとなってきたためである [3,4]．

BDD の利用により，形式的検証に伴う指数爆発を緩和できれば，より巨大な回路の設計検証を行いうる．Coudert らによって提案された順序回路の等価性判定手法 [5] Burch らによって提案された記号モデル検査法 (symbolic model checking) [6] は，この BDD の特性を利用するものであり，検証対象となる順序回路の状態遷移関数を BDD を用いて表現し，論理関数処理によるアルゴリズムを用いるものである．BDD の利用により，これらの手法では，2^{100} 状態を持つような順序回路の検証に成功している．

本章で紹介する KUE-CHIP2 の形式的設計検証も基本的には Burch らの手法に基づいている．Burch らは，仕様記述言語として計算木論理 (computaion tree logic, CTL) を用いたが，この時相論理は，時間に関して，「次の状態で」，「常に」，「性質 A が 性質 B が満足されるまで成立する」というような限られた性質しか表現すること

[1] この部分はデバッグされた

ができない. KUE-CHIP2 に対して行った設計検証では，仕様を記述する際に，CTL を拡張して記述能力を高めた BRTL (branching time regular temporal logic) を用いた [7,8].

12.2 KUE-CHIP2 の設計検証

本節では，KUE-CHIP2 に対してどのような性質を検証したかを述べる. BRTL の定義，仕様記述例，および，BDD を用いた検証アルゴリズムについては，文献を参照されたい [7,8].

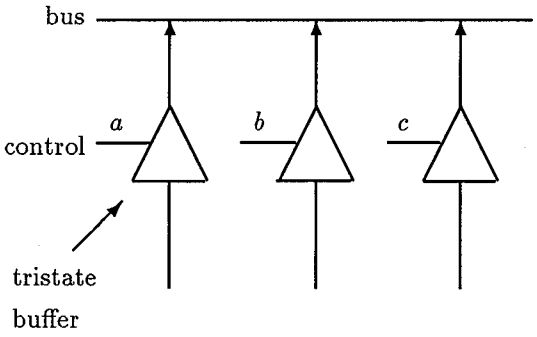
1. バスに対する条件 (バスへの出力が競合していないか，または，すべての出力がハイインピーダンス状態となることがないかどうか).

2. 各命令の動作. これについては次の 3 つの条件を検証した.

   (a) start/stop スイッチがマイクロプロセッサを始動／停止させて，内部信号線 op を 1 (動作中) / 0 (停止中) とする.

   (b) op が 1 の間は，何フェーズ目であるかを表す内部信号 $ph0, ph1, ph2, ph3, ph4$ が正しく動作する.

   (c) op が 1 の間は，各命令が正しく動作する.



ここでは，上図のバス構造を例として取り上げる. 仕様は制御線 a, b, c を用いて，次のように記述される.

$$\forall Always(a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c)$$

$\forall$ は“初期状態から起こり得る全ての動作について”を表し，$Always$ は，“どの時点でも”を意味する. $Always$ は，厳密には無限系列に対して受理／非受理を判定する ω 有限オートマトンを用いて記述されるが，これについても文献にゆずる.

この記述は，直観的には，マイクロプロセッサが取り得るどのような状態においても，バスを制御する信号線のうち，1 本のみ，かつ少なくとも 1 本がオンになっていることを意味している.

12.3 検証結果

C 言語で検証システムを実現し，SPARCstation2 上で，BDD を用いて検証を行い，次の結果が得られた.

1. バスに対する条件：前節で示した仕様に対して，設計誤りが検出された. 誤りの修正前 / 後とも，約 13 秒を要した.

2. 全命令に対する動作：アドレッシング・モード，分岐条件などを区別した場合の約 170 命令全てについて，シングル・フェーズ・モード以外について検証を行った. 検証には約 10 時間を要した.

また，1.5 節で記した設計誤りについては，次の性質を BRTL で記述し検証した[2].

[2]この実験は，LSI チップが製造され，設計誤りのあることが発見されたのちに行ったものである.

```
    n_p#0: 1,1,1,1,0,0,0,1,1,0,0,0,0,0,0,
    n_p#1: 0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,
    n_p#2: 0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,
    n_p#3: 0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,
    n_p#4: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
n_sp_sync: 0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,
n_si_sync: 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,
     n_op: 0,0,0,1,1,1,1,0,1,0,1,0,1,0,0,
     r_tc: 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,
```

"$op = 0$ のとき，全てのフリップフロップは値を保持する．"

この結果，設計誤りが検出された．検証システムは，仕様が満足されない場合には，仕様を満足しない系列を求めることができる[3]．得られた系列を以下に示す．ここで，*n_p#0* ··· *r_tc* は信号線名である．*n_op* (信号線 *op*) が最後の 2 時刻で 0,0 であるにも関わらず，*r_tc* (temorary carry flag) の値が変化していることがわかる．

参考文献

[1] S. B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, June 1978.

[2] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[3] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient Implementation of a BDD Package. *Proceedings of 27th Design Automation Conference*, pages 40–45, 1990.

[4] Shinichi Minato, Nagisa Ishiura, and Shuzo Yajima. Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation. *Proceedings of 27th Design Automation Conference*, pages 52–57, 1990.

[5] O. Coudert, C. Berthet, and J-C. Madre. Verification of Sequential Machines Using Functional Vectors. *Proceedings of IMEC-IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pages 111–128, November 1989.

[6] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill. Sequential Circuit Verification Using Symbolic Model Checking. *Proceedings of 27th Design Automation Conference*, pages 46–51, 1990.

[7] 浜口清治, 平石裕実, 矢島脩三. 論理関数処理を用いた分岐時間正則時相論理による順序機械の設計検証. 情報処理学会研究報告 *DA-60-26*, pages 201–208, 1991.

[8] K. Hamaguchi, H. Hiraishi, and S. Yajima. Design Verification of a Microprocessor Using Branching-Time Regular Temporal Logic. *Workshop on Computer-Aided Verification*, pages 201–212, June 1992.

[3] 作成したシステムでは，この機能は，まだ一般の場合については実現されていない．ここで示した系列は，当該の設計誤り専用のプログラムを作成して求めた．