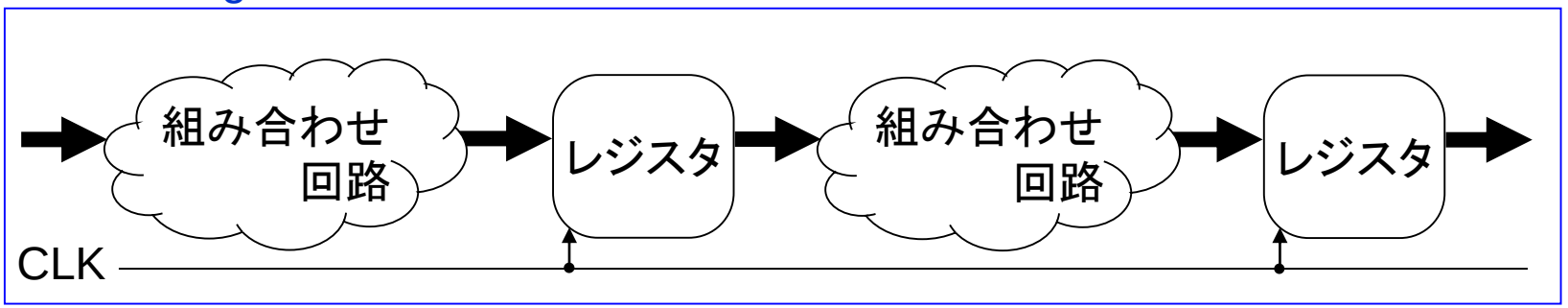
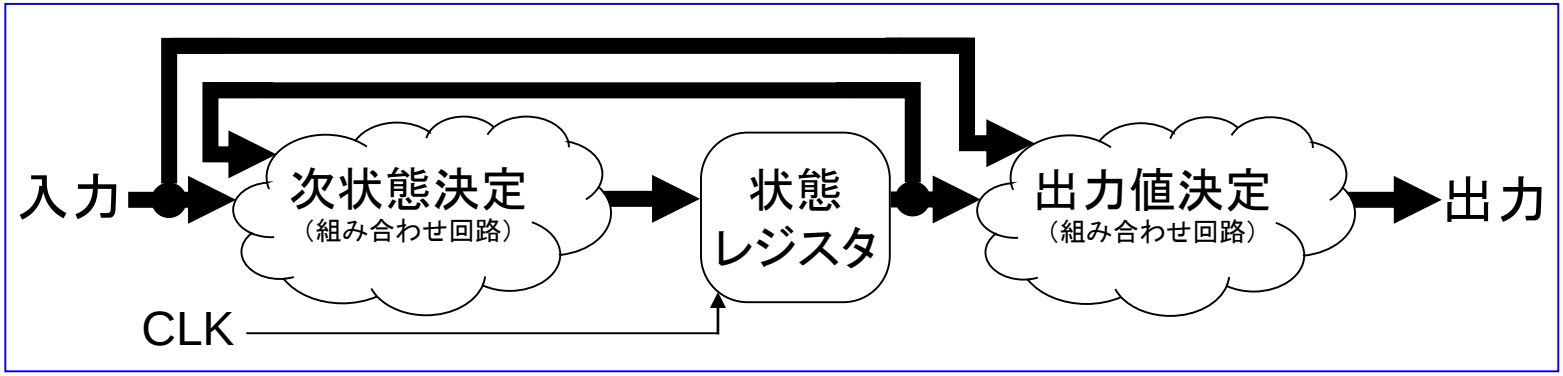
有限状態機械（FSM：Finite State Machine）の設計　～RTLでの機能図～

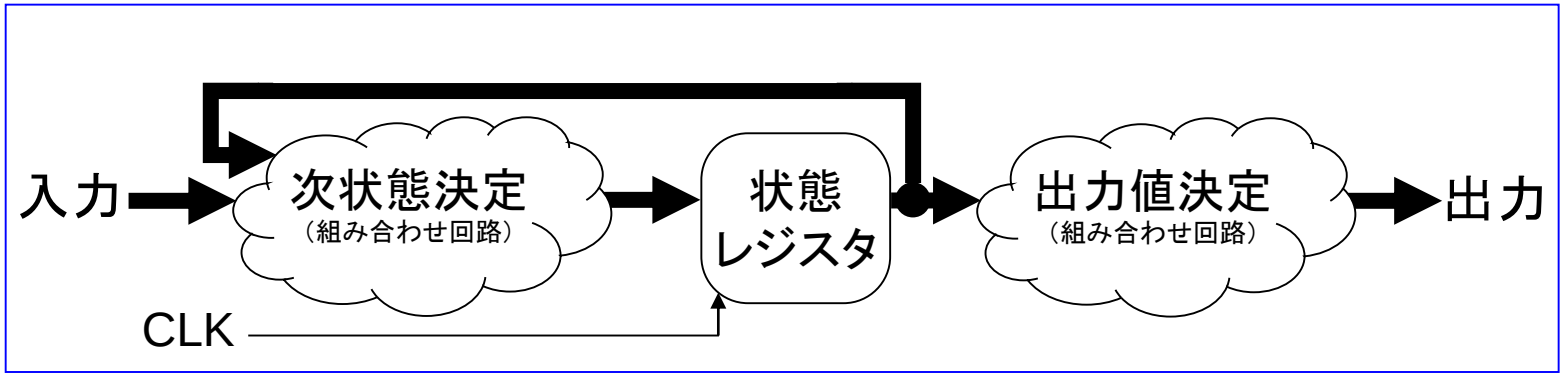
RTL（Register Transfer Level:レジスタ転送レベル）での一般的回路機能図



RTL（Register Transfer Level:レジスタ転送レベル）でのMealy型FSMの機能図



RTL（Register Transfer Level:レジスタ転送レベル）でのMoore型FSMの機能図

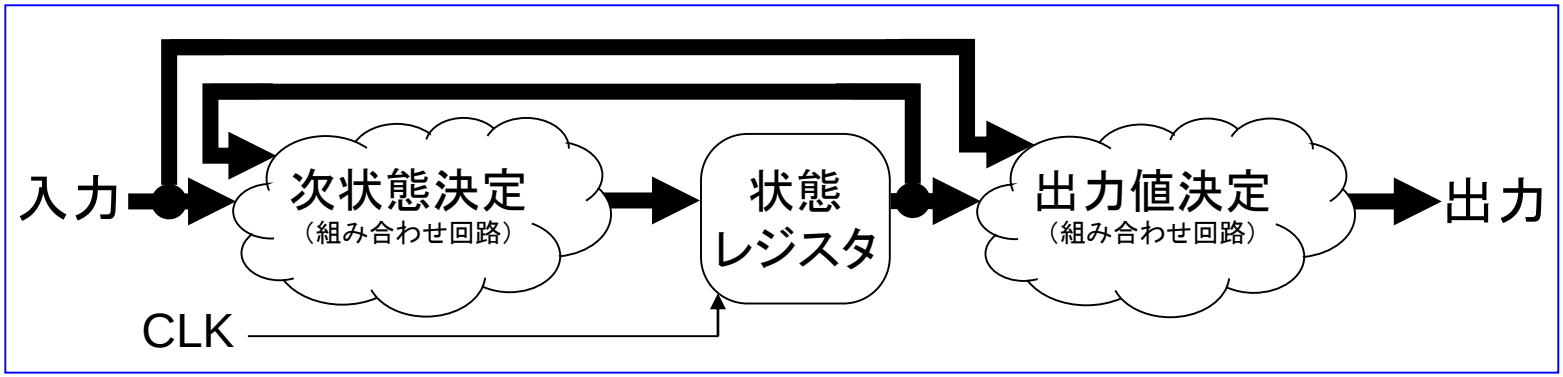


FSMは
順序回路の
一般形。

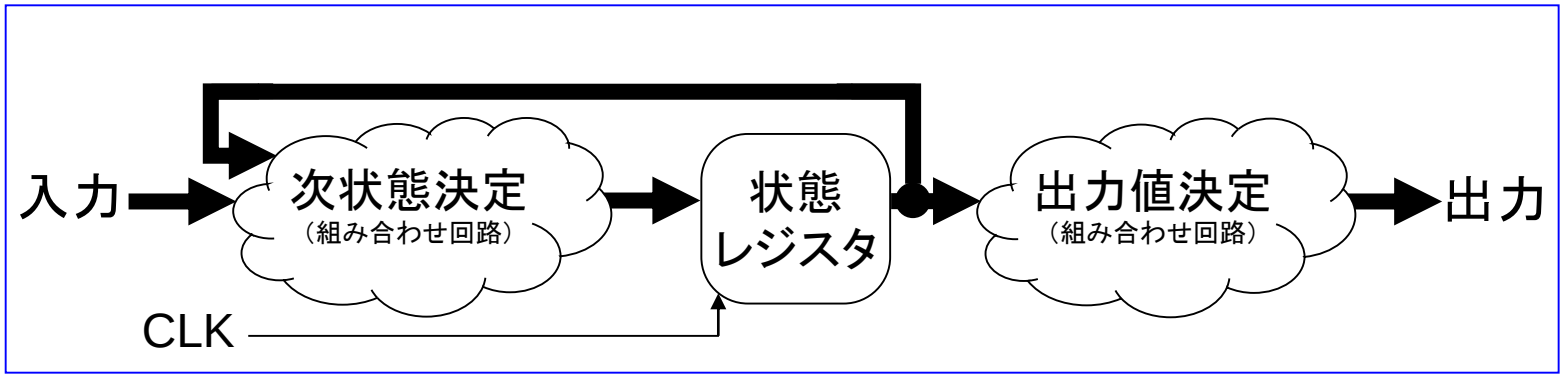
主に、
制御回路の
設計に多用
される。

Mealy型とMoore型の特徴

RTL（Register Transfer Level:レジスタ転送レベル）でのMealy型FSMの機能図



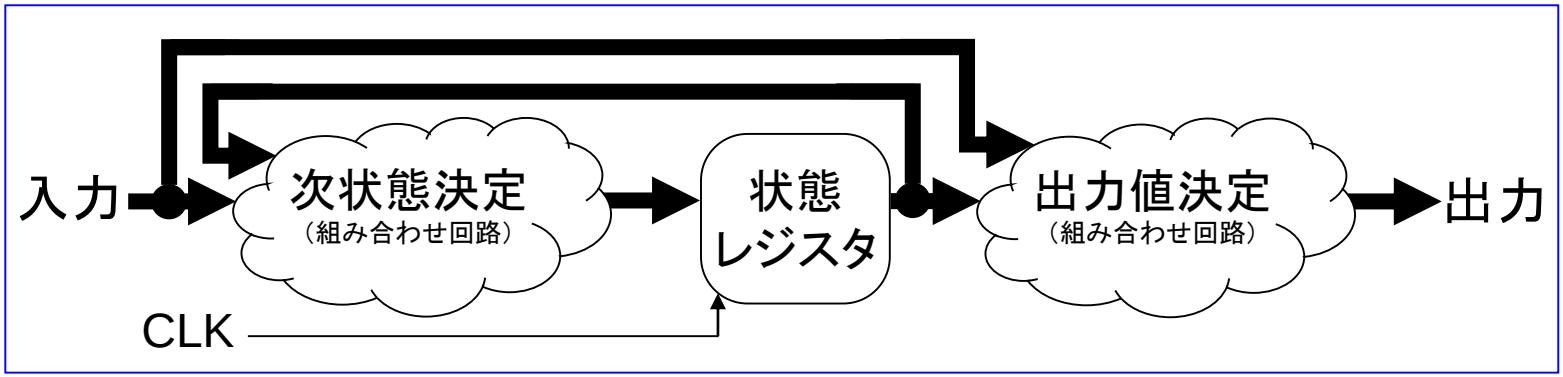
RTL（Register Transfer Level:レジスタ転送レベル）でのMoore型FSMの機能図



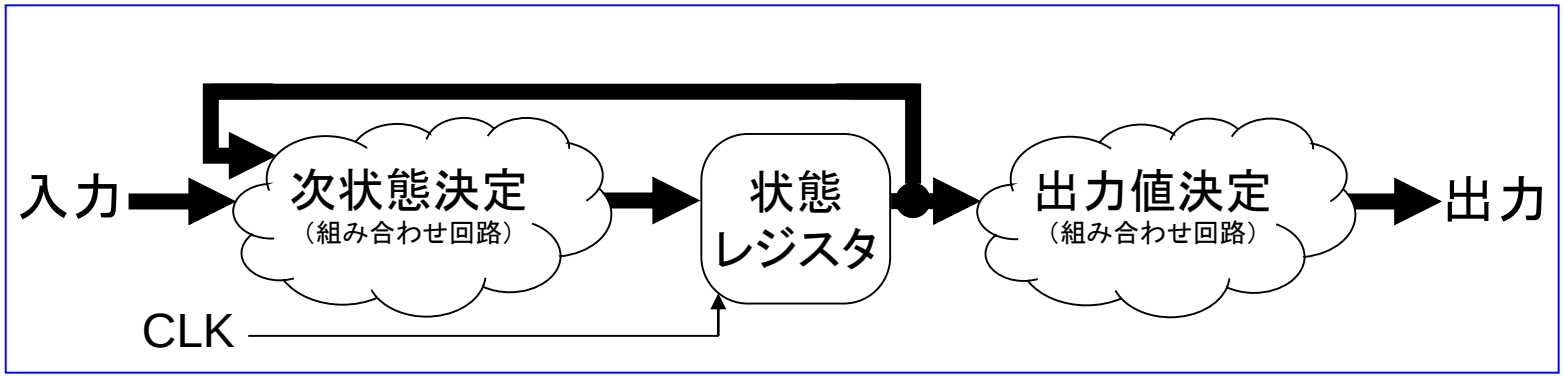
特徴	Mealy型	Moore型
次状態決定	入力値と状態から決まる。	
出力値決定	状態と入力値から決まる。	状態のみから決まる。
状態数（⇔状態レジスタのビット数）	比較的少ない。	比較的多い。
出力におけるハザード	比較的生じやすい。	比較的生じにくい。

FSMのエンティティの基本形

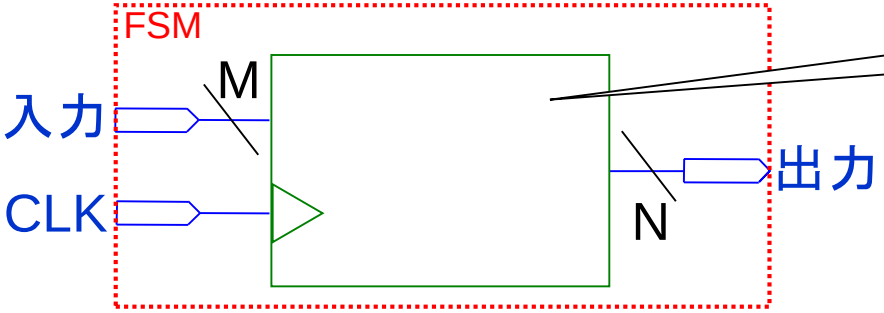
RTL（Register Transfer Level:レジスタ転送レベル）でのMealy型FSMの機能図



RTL（Register Transfer Level:レジスタ転送レベル）でのMoore型FSMの機能図



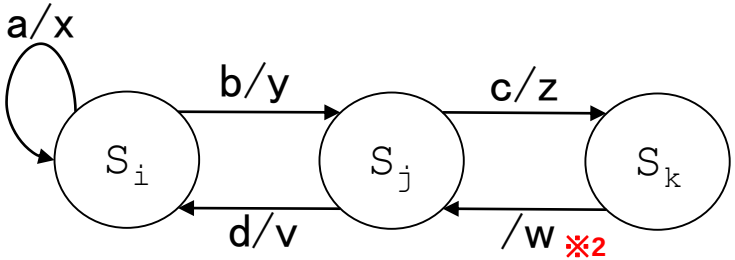
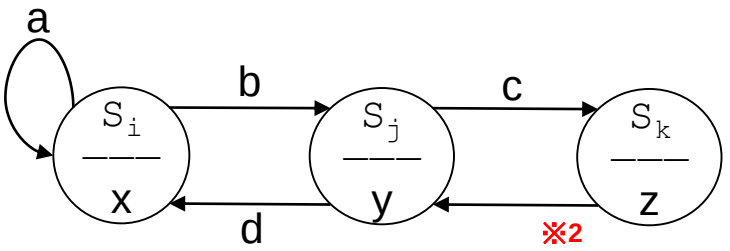
Mealy型とMoore型で、エンティティの形は同じ。



Mealy型とMoore型で、回路機能が異なる。

回路機能の仕様の表現手法として、
状態遷移図（Mealy型、Moore型）が
多用される。

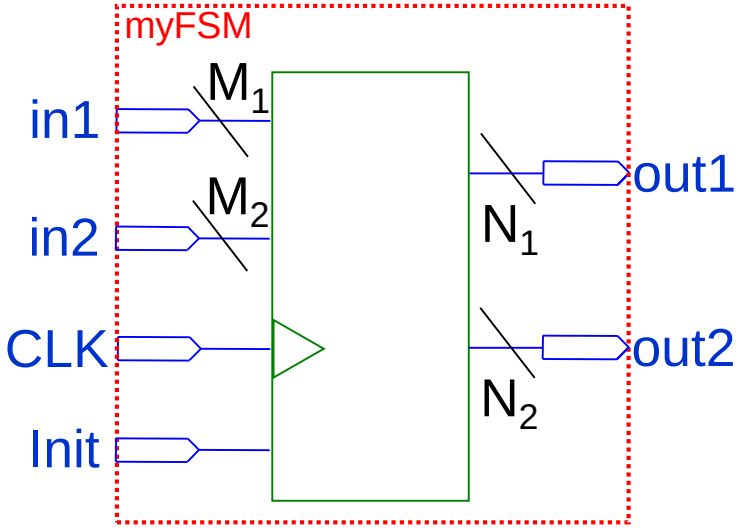
Mealy型とMoore型の状態遷移図（1/2）

	Mealy型	Moore型
描図例※1		
丸	状態を表す。	
丸中の文字	状態名（状態レジスタの値の記号化）	状態名（状態レジスタの値の記号化） 対応する出力値
矢印	クロックエッジのタイミングでの遷移を表す。	
矢印上の文字	入力値／対応する出力値	入力値
図の読み方例	状態S_iにいるとき、 入力値=a ⇒ 出力値=x で、 そのときクロックエッジがあったら、 状態S_iに遷移。 出力値は 状態と入力値から決まる 状態S_iにいるとき、 入力値=b ⇒ 出力値=y で、 そのときクロックエッジがあったら、 状態S_jに遷移。	状態S_iにいるとき、 出力値=x で、 出力値は 状態のみから決まる 入力値=aでクロックエッジがあったら状態S_iに遷移。 入力値=bでクロックエッジがあったら状態S_jに遷移。

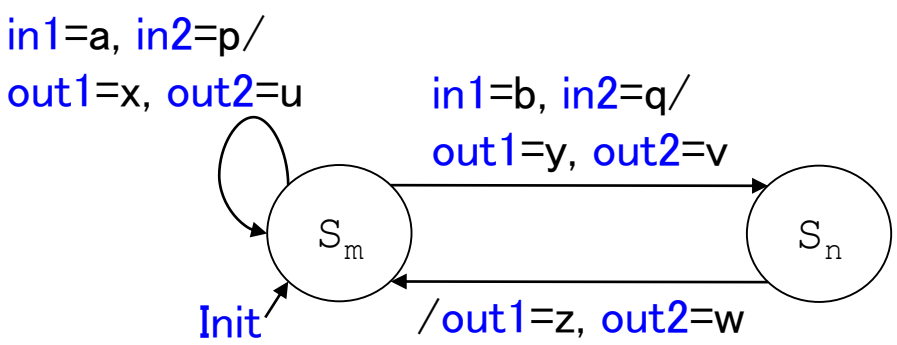
【※1】上記2つの図例は、同じ機能のFSMを表わしているわけではない。 【※2】一意の遷移は、入力に寄らない。

Mealy型とMoore型の状態遷移図(2/2)

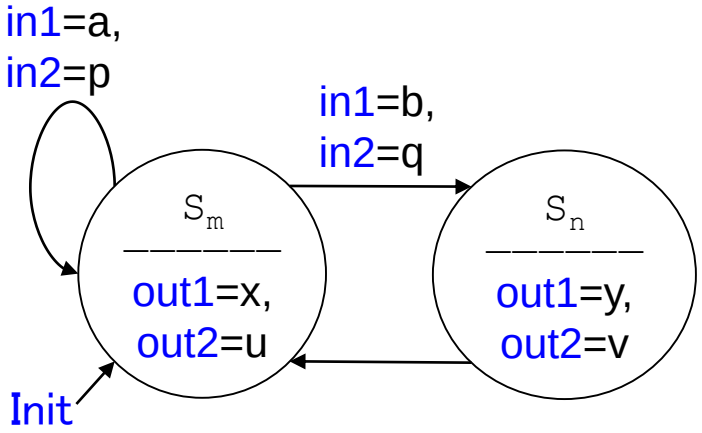
① 初期化機能(Init)の追加
② 入出力ポート名つき状態遷移図



Mealy型

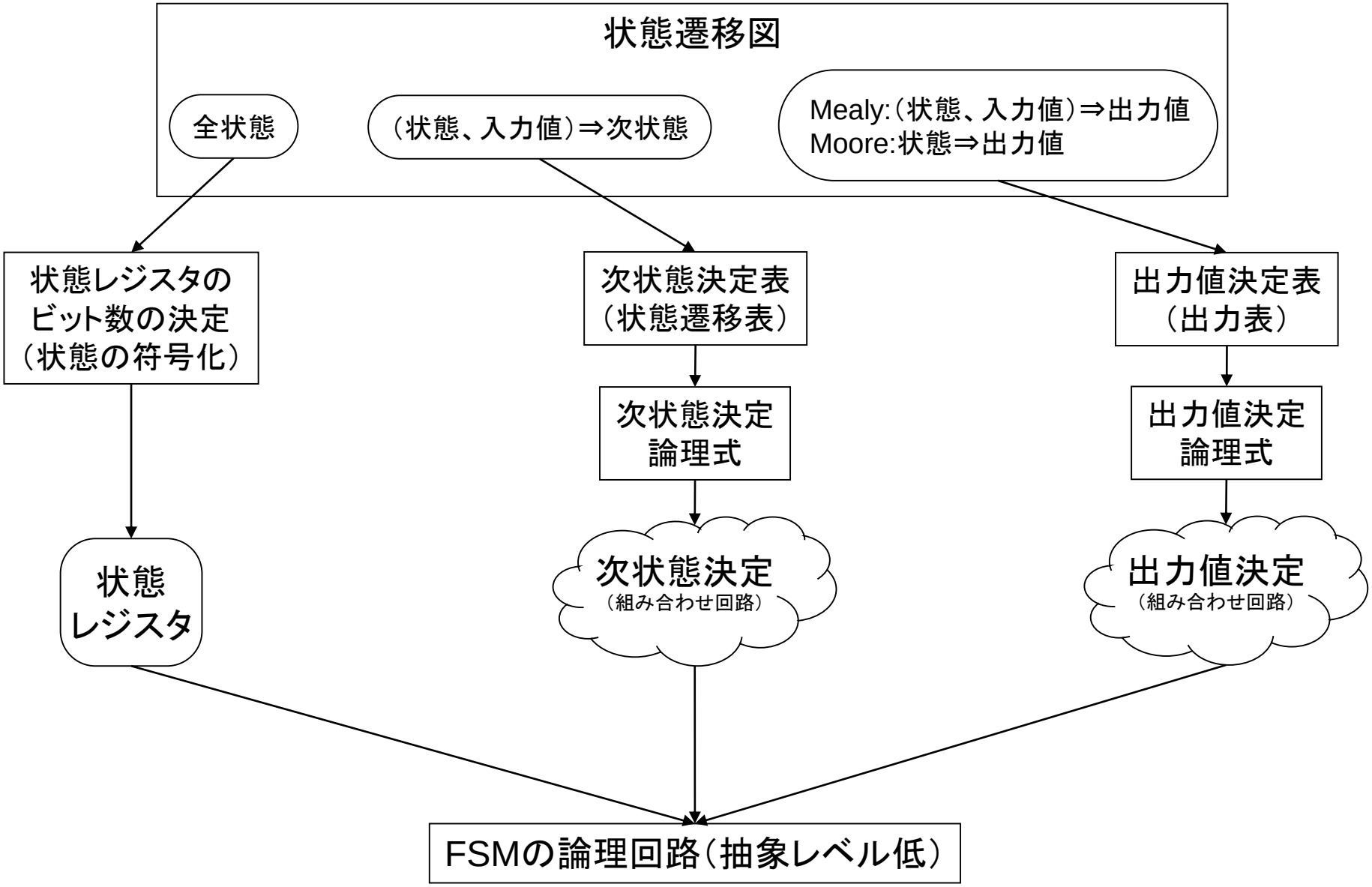


Moore型



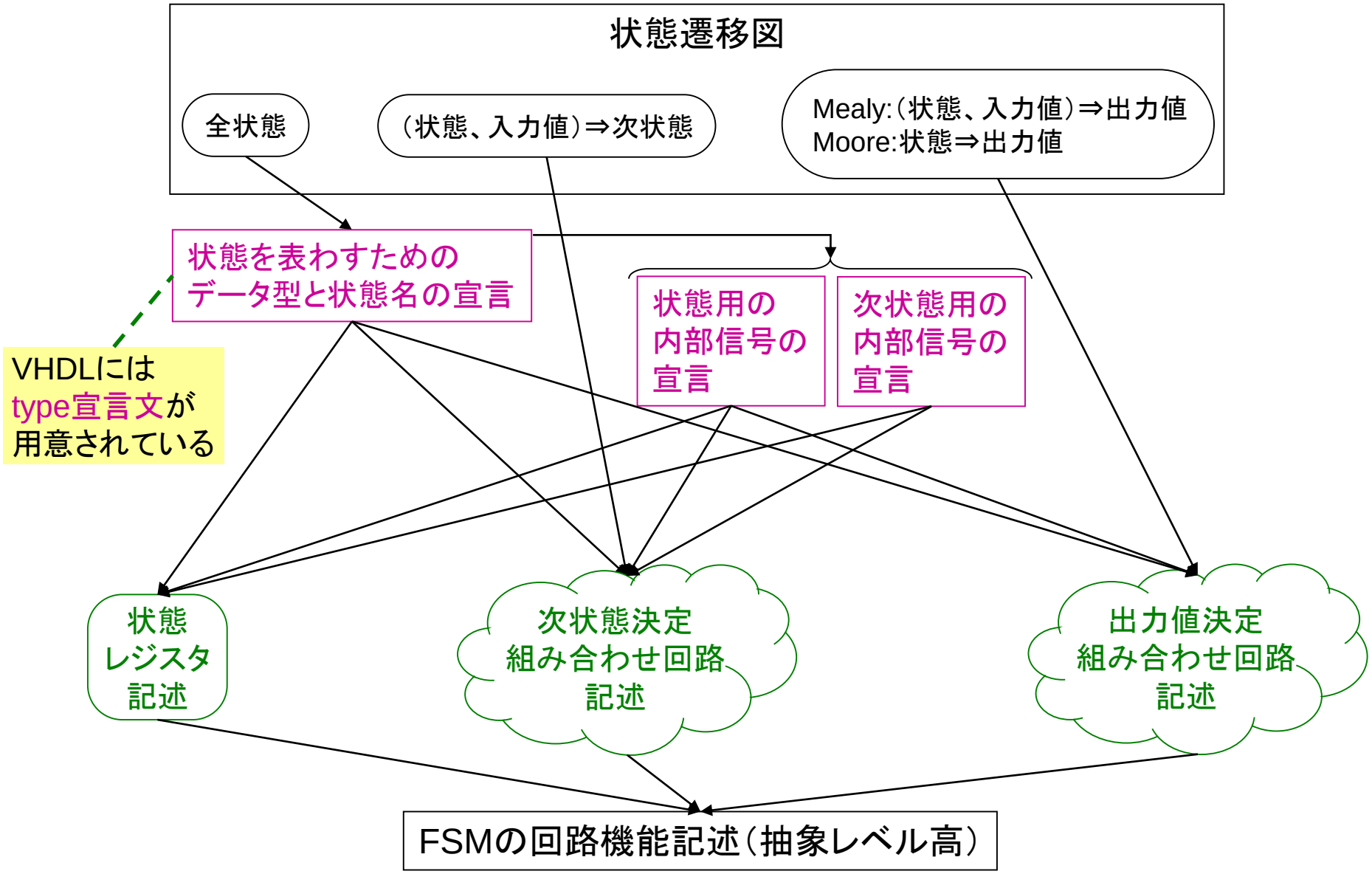
【注】上記2つの図例は、同じ機能のFSMを表わしているわけではない。

状態遷移図→FSMの回路機能の設計（抽象レベル低）



ここでは、この方法での設計については紹介しない。

状態遷移図→FSMの回路機能の設計（抽象レベル高）



ここでは、この方法での設計について紹介する。

type宣言文

●type宣言文：新しい（ユーザ定義の）データ型の宣言

書式　`type` データ型の名前 `is` データ型の定義`;`

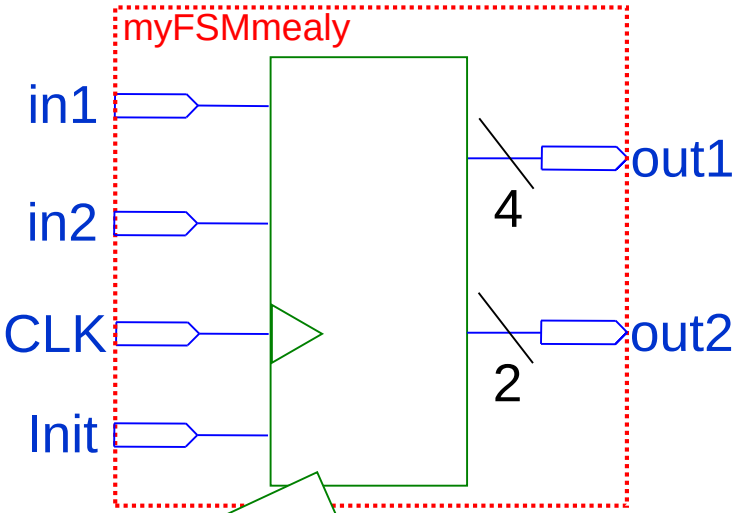
- 先頭文字が半角英文字の半角英数文字列。
- 大小文字を区別しない。
- VHDL予約語は使えない。

◆列挙型定義
書式：（列挙要素，列挙要素，・・・，列挙要素）

◆配列型定義
書式：`array`（添え字範囲）`of` 配列の要素となるデータ型

【注】
- 列挙型定義、配列方定義以外の定義もあるが、ここでは省略する。
- 大規模な配列型定義は、PLD内の論理素子を大量に要するので、あまり行われない。

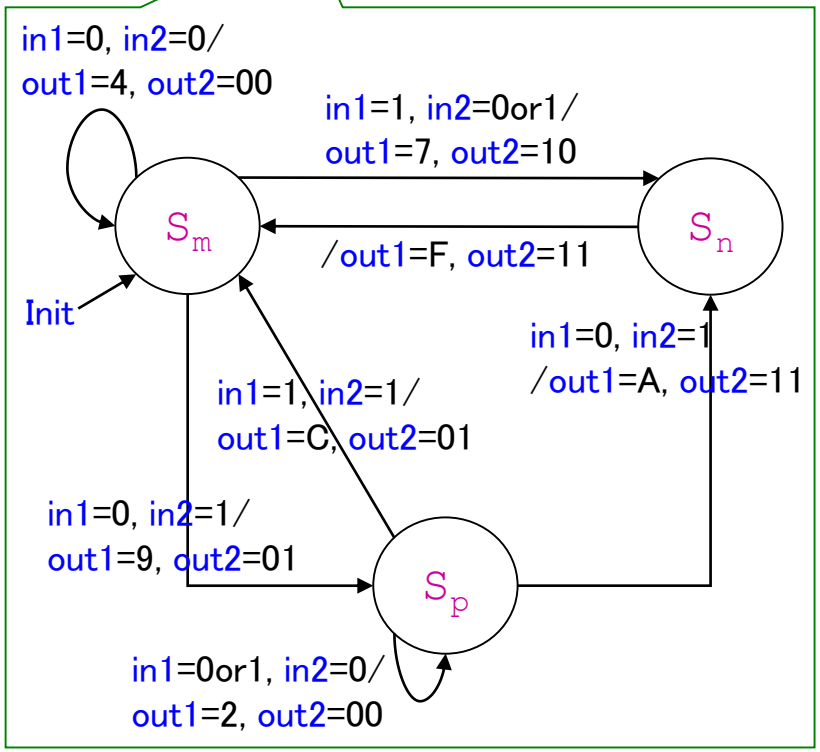
Mealy型FSMの記述例(1/4)



入出力ポートの記述

```
port(
  Init, CLK, in1, in2 : in std_logic;
  out1 : out std_logic_vector(3 downto 0);
  out2 : out std_logic_vector(1 downto 0)
);
```

【※】初期化機能は、Init=1で有効とする。



全状態

状態を表わすための
データ型と状態名の宣言

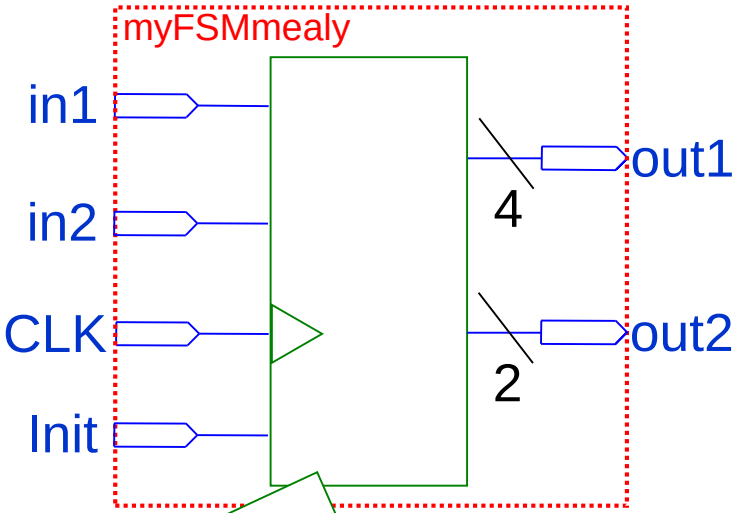
状態用の
内部信号の
宣言

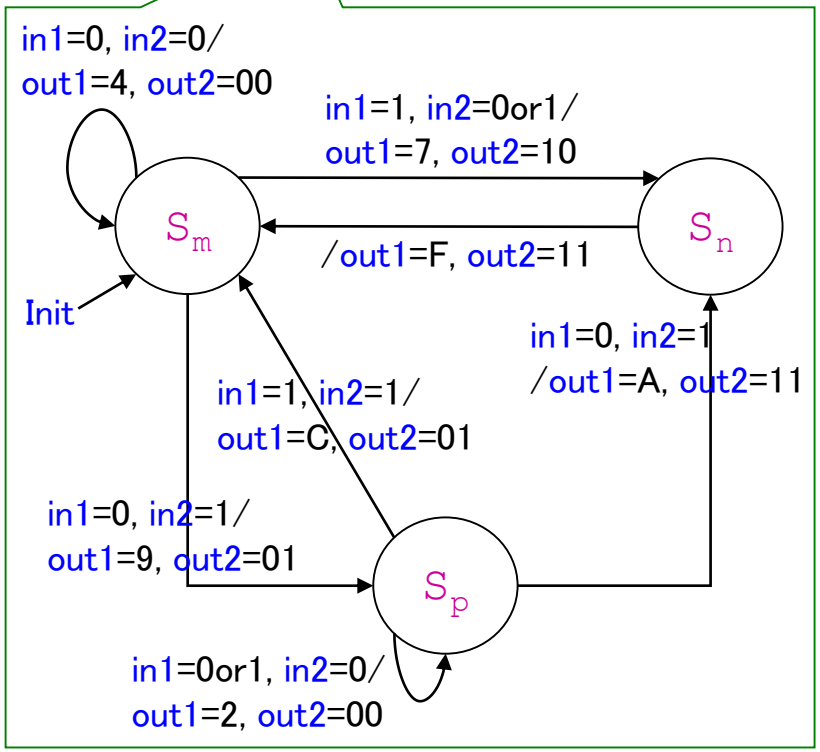
次状態用の
内部信号の
宣言

各種宣言の記述

```
type state_type is (Sm,Sn,Sp);
signal state, nextstate : state_type;
```

Mealy型FSMの記述例(2/4)



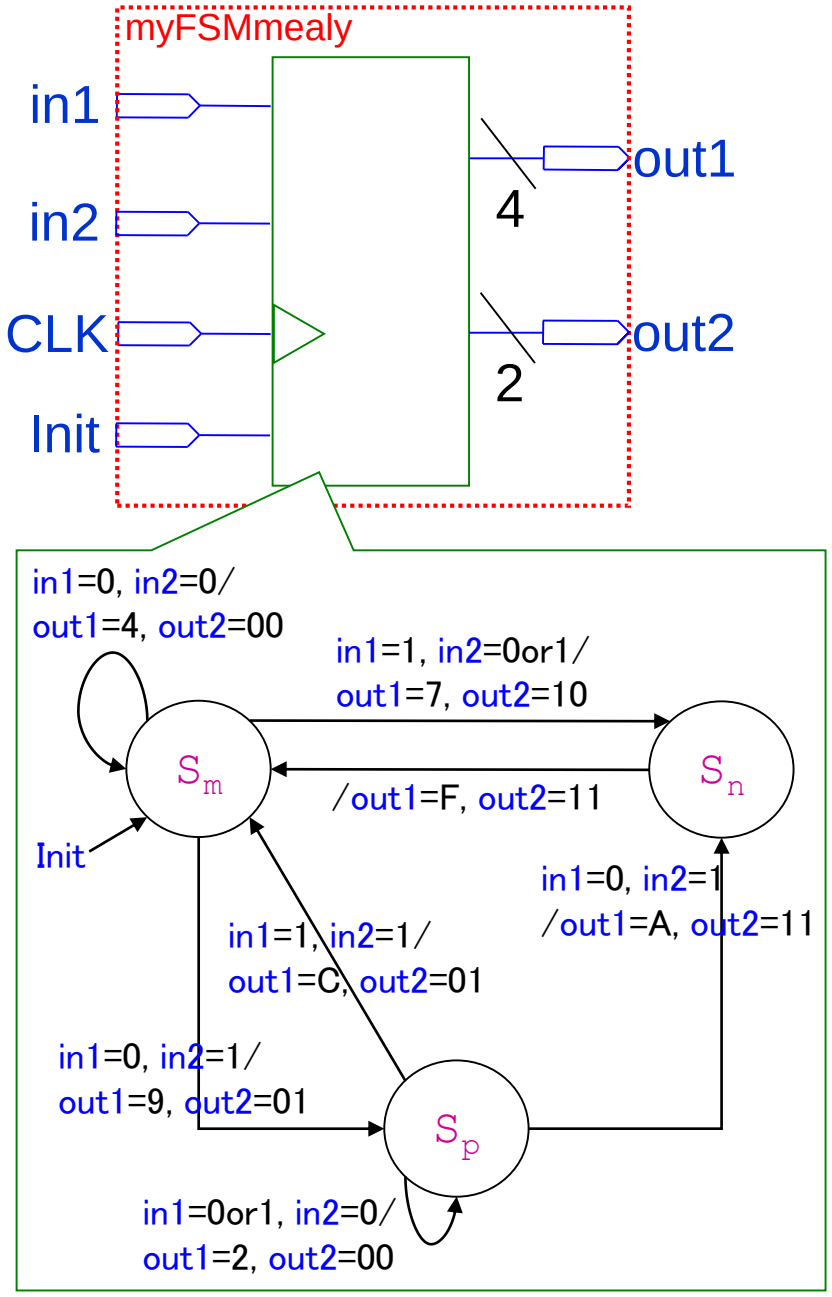


状態
レジスタ
記述

回路機能の記述 その1:状態レジスタ

```
process(Init,CLK)
begin
  if(Init='1')then
    state <= Sm;
  elsif(CLK'event and CLK='1')then
    state <= nextstate;
  end if;
end process;
```

Mealy型FSMの記述例(3/4)

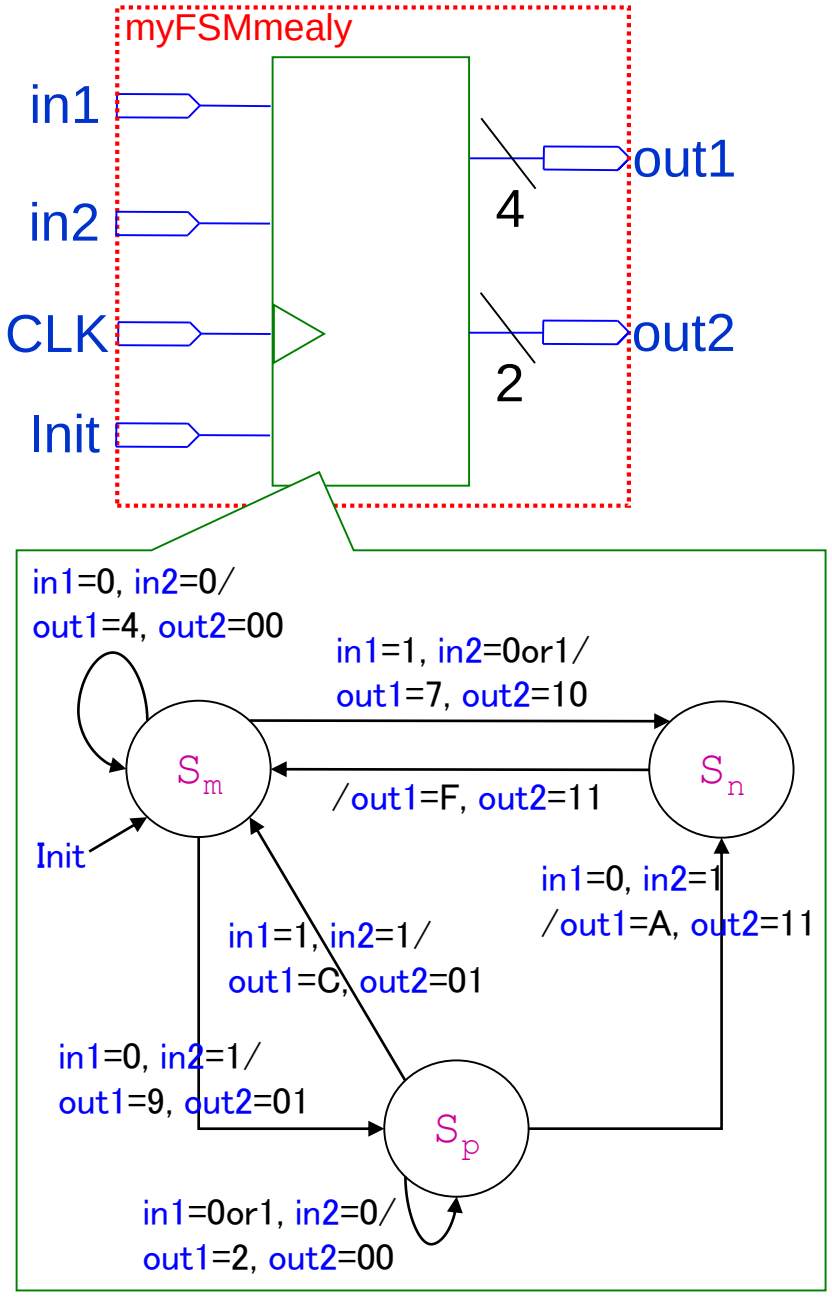


次状態決定
組み合わせ回路
記述

回路機能の記述 その2:次状態決定回路

```
process(state,in1,in2)
begin
  case state is
    when Sm => if(in1='0' and in2='0')then
                 nextstate <= Sm;
               elsif(in1='0' and in2='1')then
                 nextstate <= Sp;
               else
                 nextstate <= Sn;
               end if;
    when Sp => if(in1='0' and in2='1')then
                 nextstate <= Sn;
               elsif(in1='1' and in2='1')then
                 nextstate <= Sm;
               else
                 nextstate <= Sp;
               end if;
    when Sn => nextstate <= Sm;
  end case;
end process;
```

Mealy型FSMの記述例(4/4)



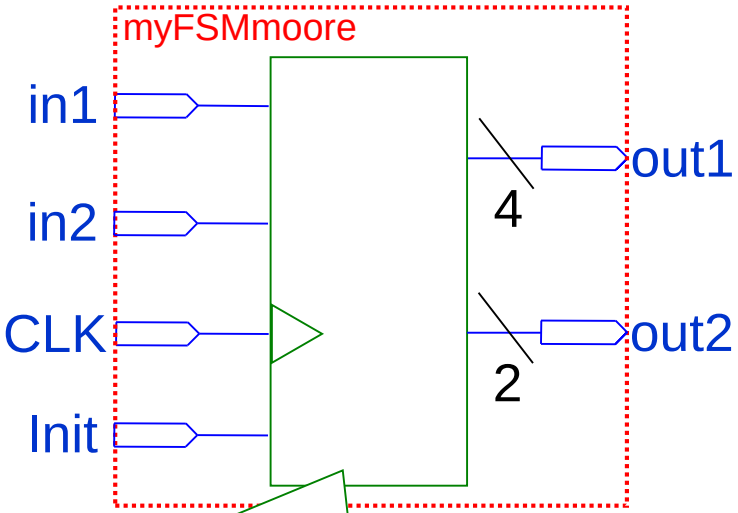
出力値決定
組み合わせ回路
記述

回路機能の記述 その3:出力値決定回路

出力値は
状態と入力値から決まる

```
process(state,in1,in2)
begin
  case state is
    when Sm => if(in1='0' and in2='0')then
                 out1 <= X"4"; out2 <= "00";
               elsif(in1='0' and in2='1')then
                 out1 <= X"9"; out2 <= "01";
               else
                 out1 <= X"7"; out2 <= "10";
               end if;
    when Sp => if(in1='0' and in2='1')then
                 out1 <= X"A"; out2 <= "11";
               elsif(in1='1' and in2='1')then
                 out1 <= X"C"; out2 <= "01";
               else
                 out1 <= X"2"; out2 <= "00";
               end if;
    when Sn => out1 <= X"F"; out2 <= "11";
  end case;
end process;
```

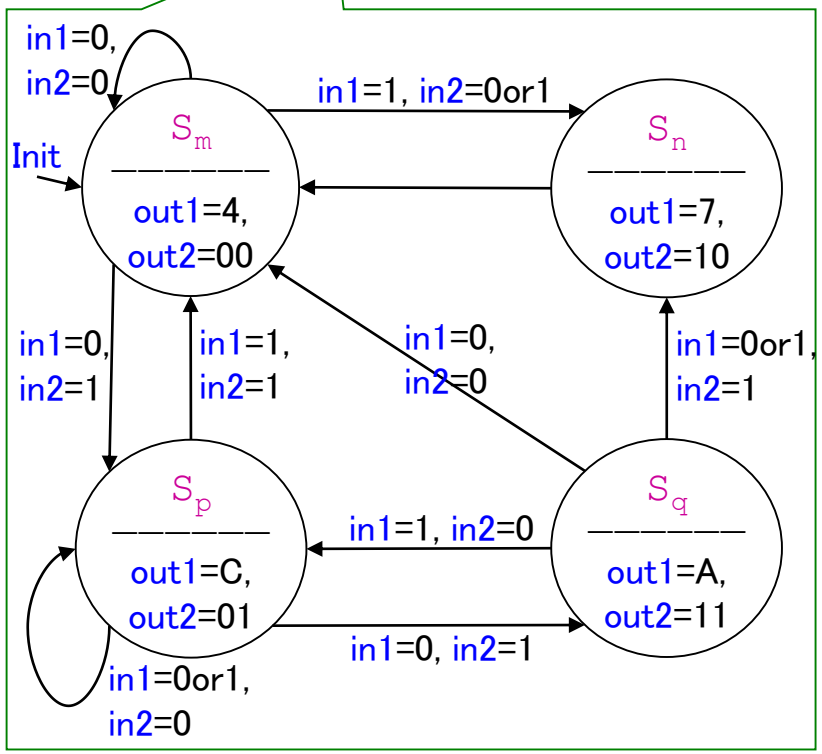
Moore型FSMの記述例(1/4)



入出力ポートの記述

```
port(
  Init, CLK, in1, in2 : in std_logic;
  out1 : out std_logic_vector(3 downto 0);
  out2 : out std_logic_vector(1 downto 0)
);
```

【※】初期化機能は、Init=1で有効とする。



全状態

状態を表わすための
データ型と状態名の宣言

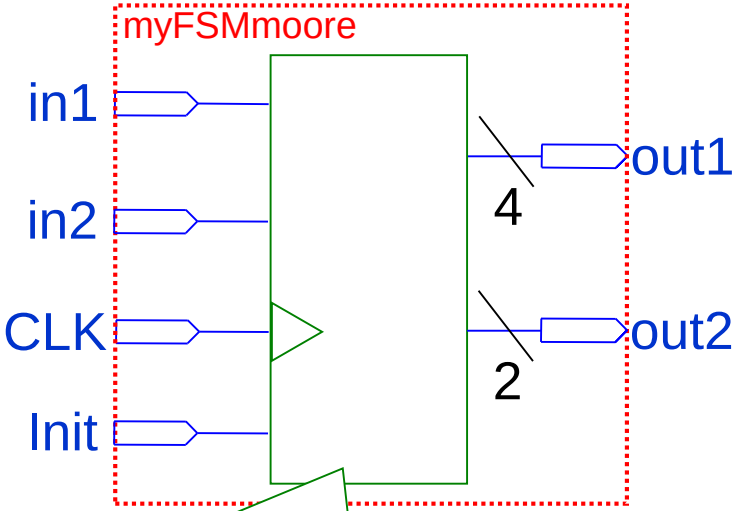
状態用の
内部信号の
宣言

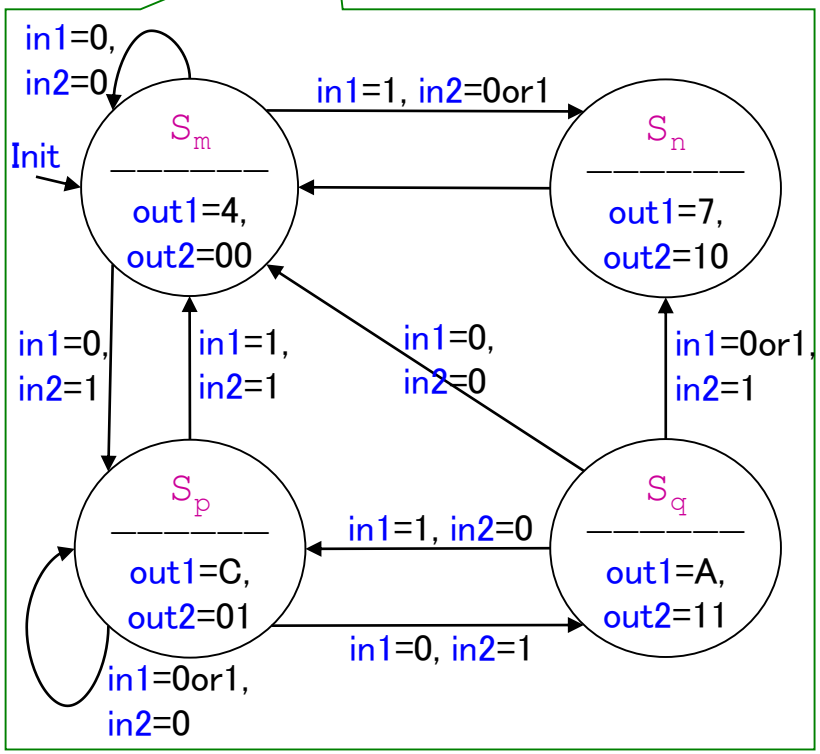
次状態用の
内部信号の
宣言

各種宣言の記述

```
type state_type is (Sm,Sn,Sp,Sq);
signal state, nextstate : state_type;
```

Moore型FSMの記述例(2/4)



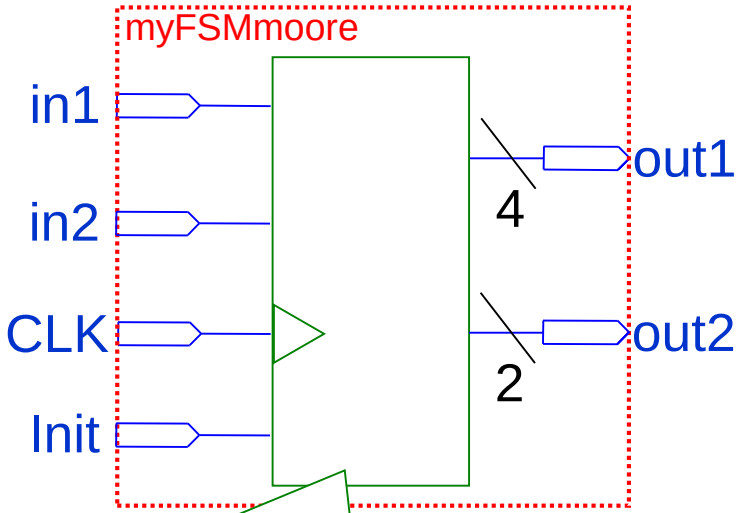


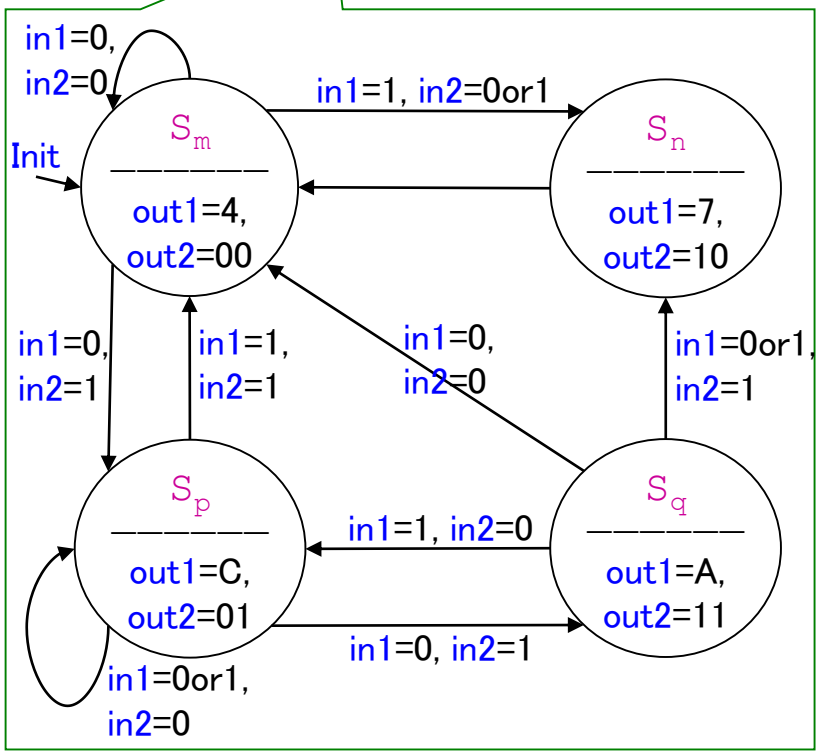
状態
レジスタ
記述

回路機能の記述 その1:状態レジスタ

```
process(Init,CLK)
begin
  if(Init='1')then
    state <= Sm;
  elsif(CLK'event and CLK='1')then
    state <= nextstate;
  end if;
end process;
```

Moore型FSMの記述例(3/4)



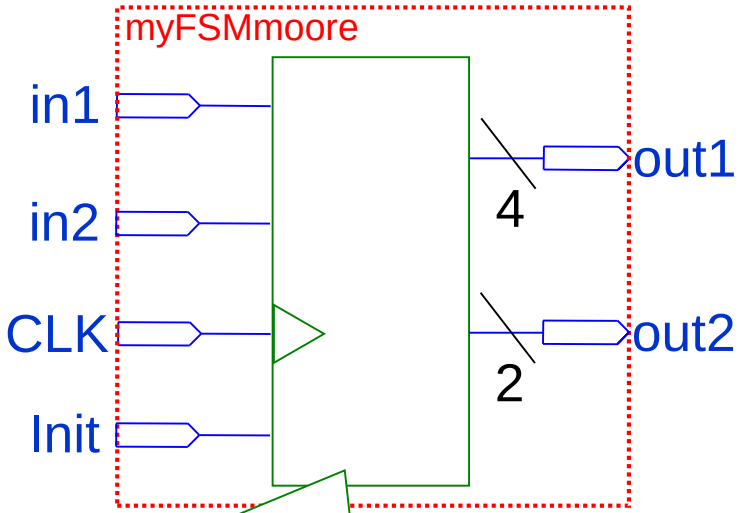


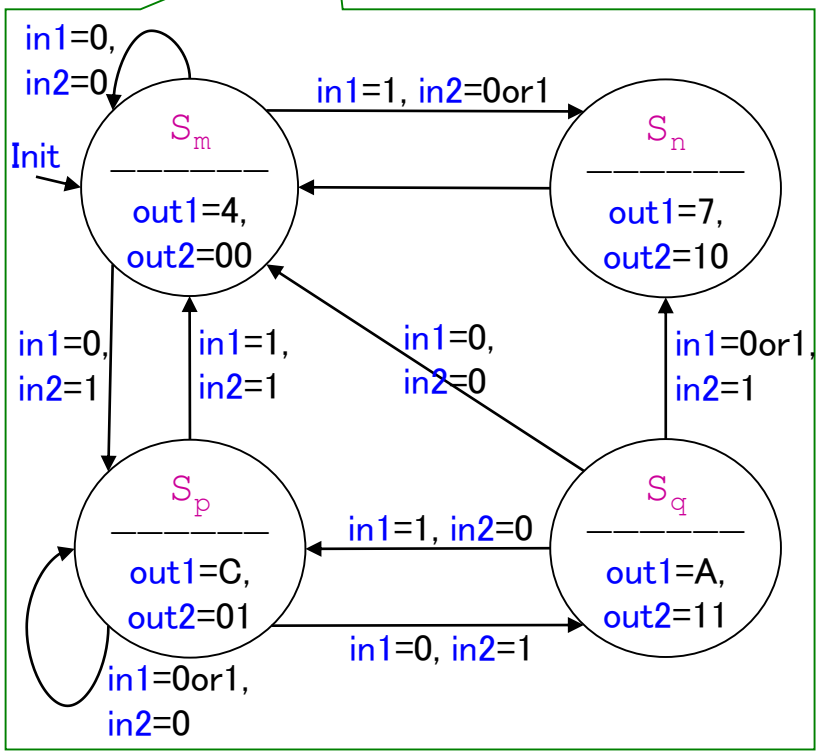
次状態決定
組み合わせ回路
記述

回路機能の記述
その2:次状態決定回路

```
process(state,in1,in2)
begin
  case state is
    when Sm => if(in1='0' and in2='0')then
                 nextstate <= Sm;
               elsif(in1='0' and in2='1')then
                 nextstate <= Sp;
               else
                 nextstate <= Sn;
               end if;
    when Sp => if(in1='0' and in2='1')then
                 nextstate <= Sq;
               elsif(in1='1' and in2='1')then
                 nextstate <= Sm;
               else
                 nextstate <= Sp;
               end if;
    when Sq => if(in1='0' and in2='0')then
                 nextstate <= Sm;
               elsif(in1='1' and in2='0')then
                 nextstate <= Sp;
               else
                 nextstate <= Sn;
               end if;
    when Sn => nextstate <= Sm;
  end case;
end process;
```

Moore型FSMの記述例(4/4)





出力値決定
組み合わせ回路
記述

回路機能の記述 その3-A:出力値決定回路

```
process(state)
begin
  case state is
    when Sm => out1 <= X"4"; out2 <= "00";
    when Sp => out1 <= X"C"; out2 <= "01";
    when Sq => out1 <= X"A"; out2 <= "11";
    when Sn => out1 <= X"7"; out2 <= "10";
  end case;
end process;
```

出力値は
状態のみから決まる

回路機能の記述 その3-B:出力値決定回路

```
out1 <= X"4" when (state=Sm) else
        X"C" when (state=Sp) else
        X"A" when (state=Sq) else
        X"7";

out2 <= "00" when (state=Sm) else
        "01" when (state=Sp) else
        "11" when (state=Sq) else
        "10";
```

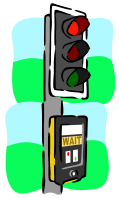
出力値は
状態のみから決まる

その3-A と その3-B の 機能は同一

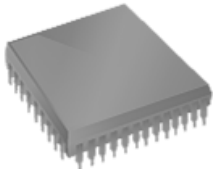
FSMによる制御回路の設計例

●自動販売機の制御回路
投入金額に応じて、
商品やおつりを投下する内部装置を、適切に制御する。

●信号機の制御回路
タイマーや押しボタンに応じて、
ランプを切り替える内部装置を、適切に制御する。



●マイクロプロセッサ（CPU）の制御回路
ISAで定められた各機械語命令に応じて、
レジスタやALUなどの内部装置を、命令が実行されるように制御する。



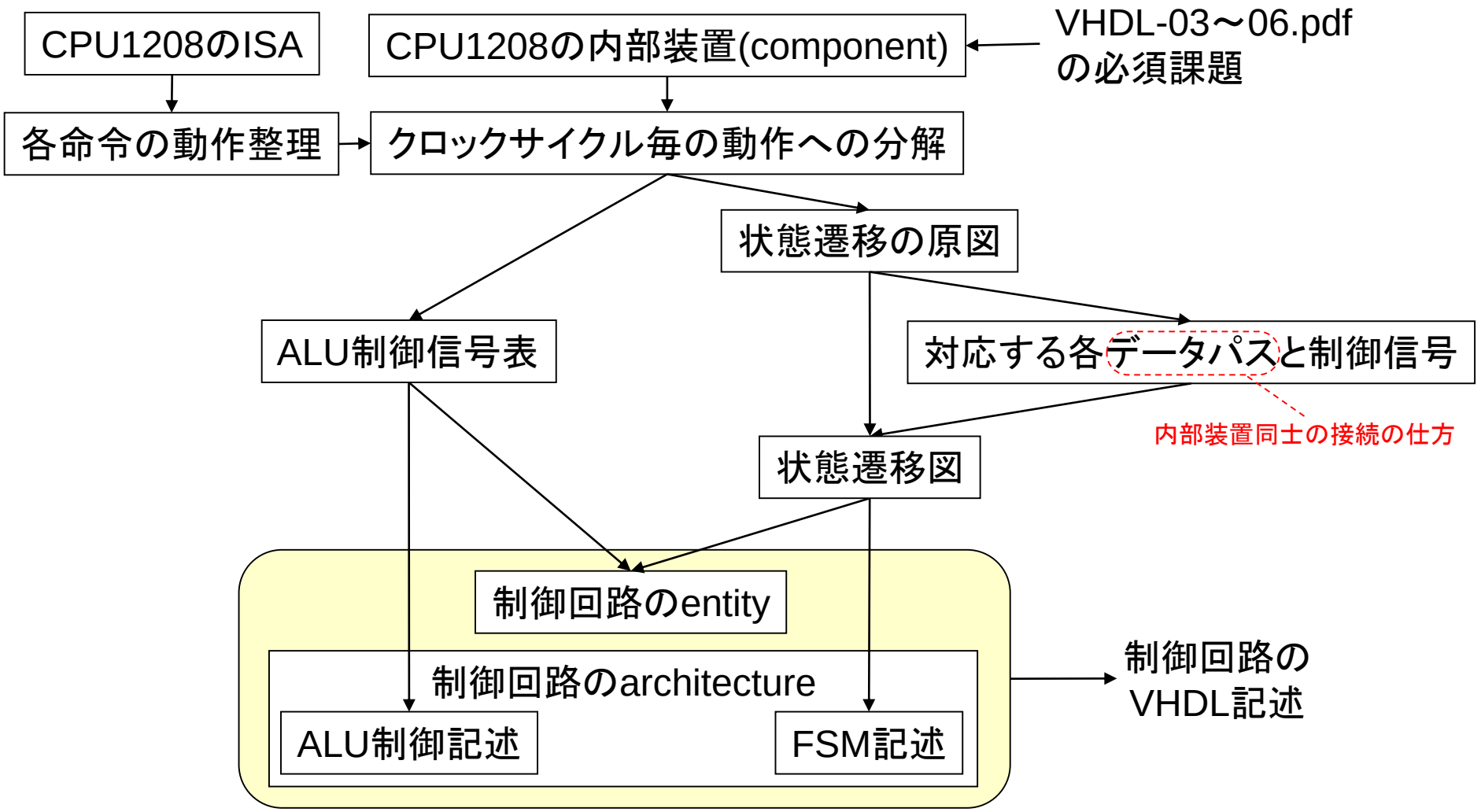
その他、ロボット、ロケット、etc・・・

ここでは、FSMの設計課題として、これを取り上げる。

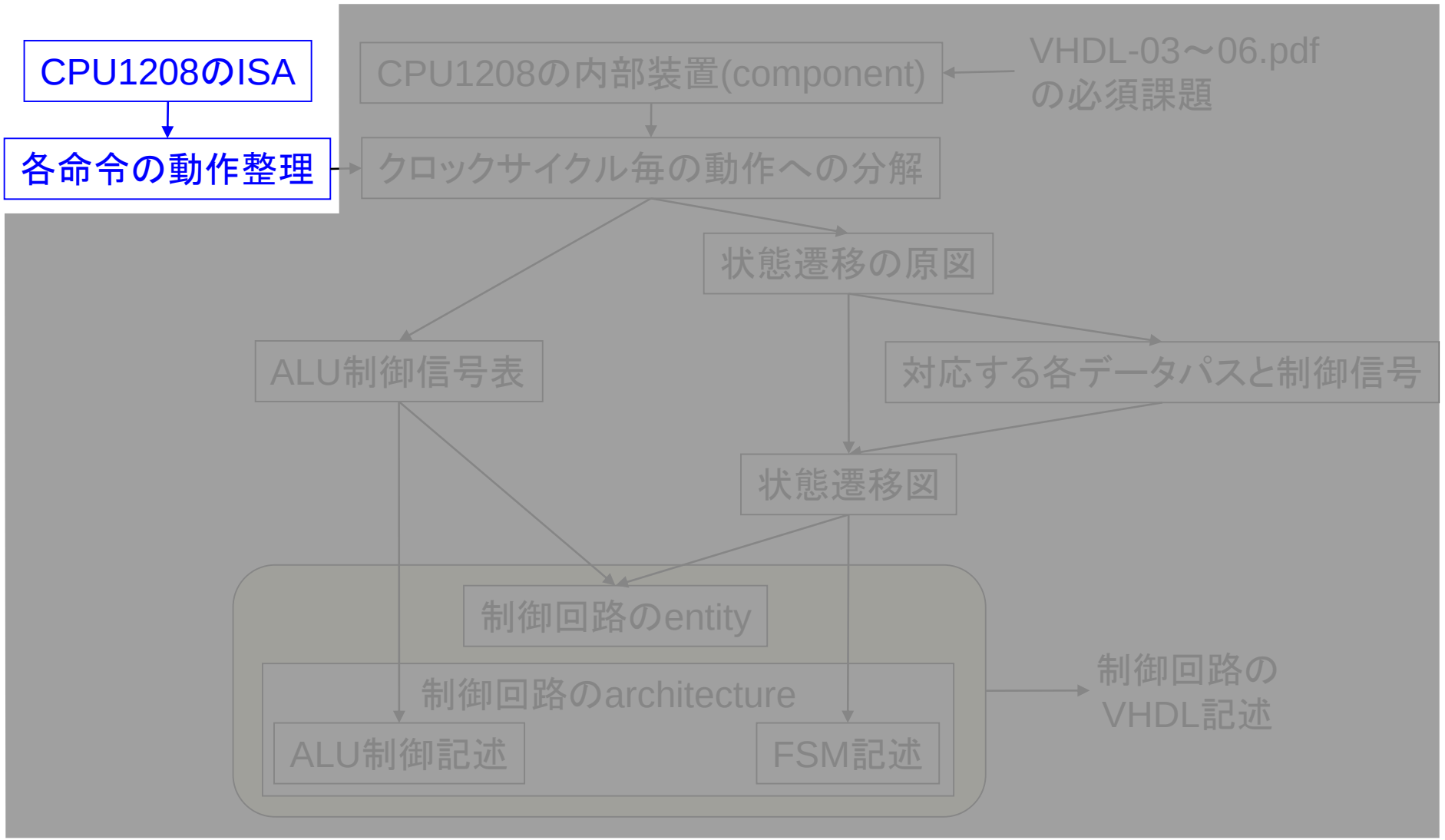
どんなISA、どんな内部装置をもつマイクロプロセッサ（CPU）を前提にするか？
→単純な構成のCPUがよい。

マイクロプロセッサ（CPU）の制御回路の設計に向けて

単純な構成のCPUとして、CPU1208なるマイクロプロセッサを考え、
以下の手順で、CPU1208の制御回路（以下、単に制御回路）を設計することを目指す。

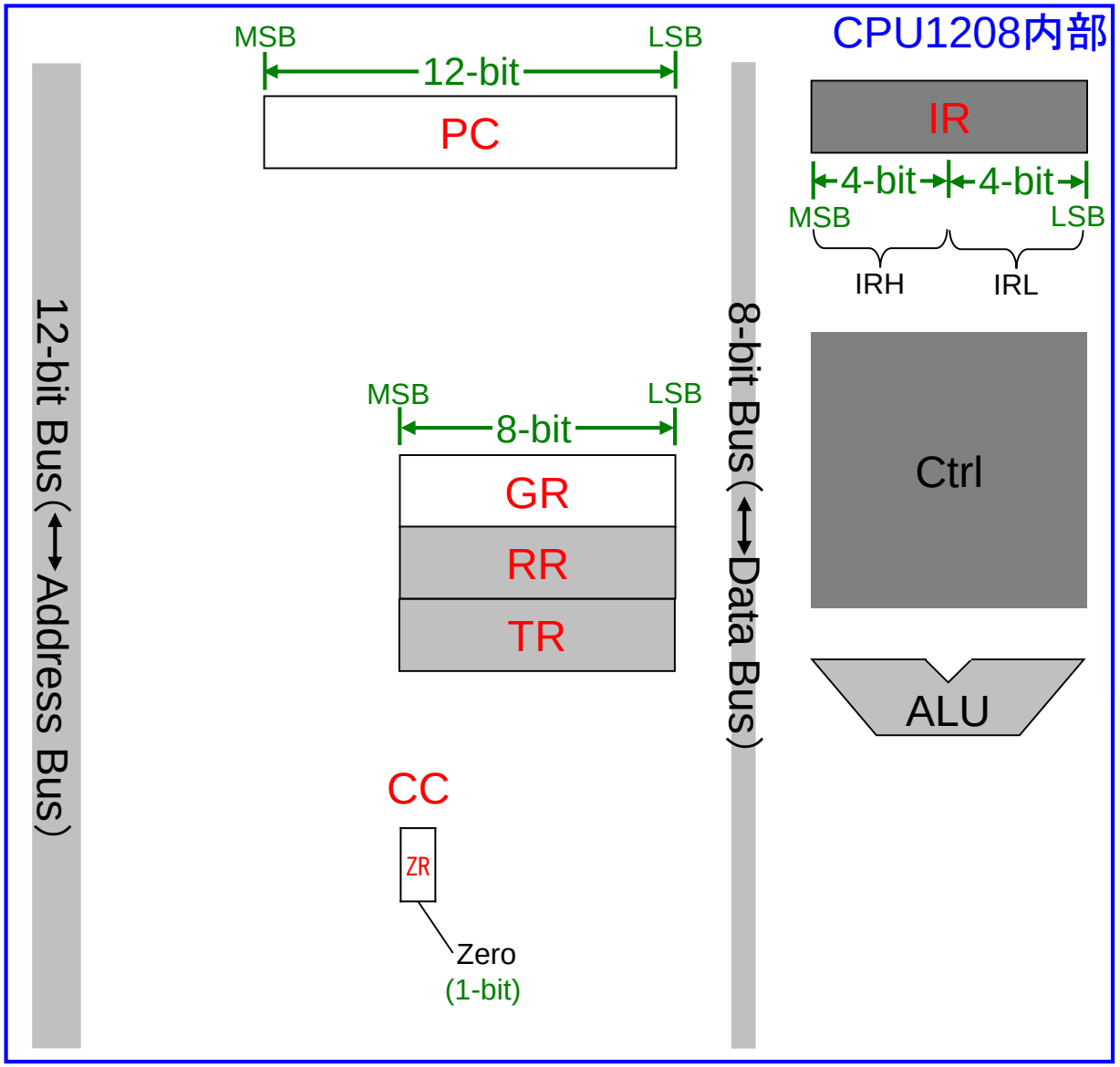


手順


CPU1208のISA
各命令の動作整理
CPU1208の内部装置(component)
VHDL-03～06.pdf
の必須課題
クロックサイクル毎の動作への分解
状態遷移の原図
ALU制御信号表
対応する各データパスと制御信号
状態遷移図
制御回路のentity
制御回路のarchitecture
ALU制御記述
FSM記述
制御回路の
VHDL記述

CPU1208のISA： 内部記憶装置

ISA：プロセッサ内部の記憶装置、命令（種類、形式）、アドレッシングモード、メモリ



GR：汎用レジスタ
CC：Condition Code Register
(ZRと表わす)

PC：Program Counter

TR：一時レジスタ
RR：結果レジスタ

ALU：Arithmetic Logic Unit
IR：命令レジスタ
Ctrl：Control Unit

赤文字の装置：記憶装置

装置の地の色：白 ⟷ 濃い灰
重要度：高 ⟷ 低

【注】この『重要度』は、ISAの意味での。

CPU1208のISA：命令

ISA:プロセッサ内部の記憶装置、命令（種類、形式）、アドレッシングモード、メモリ

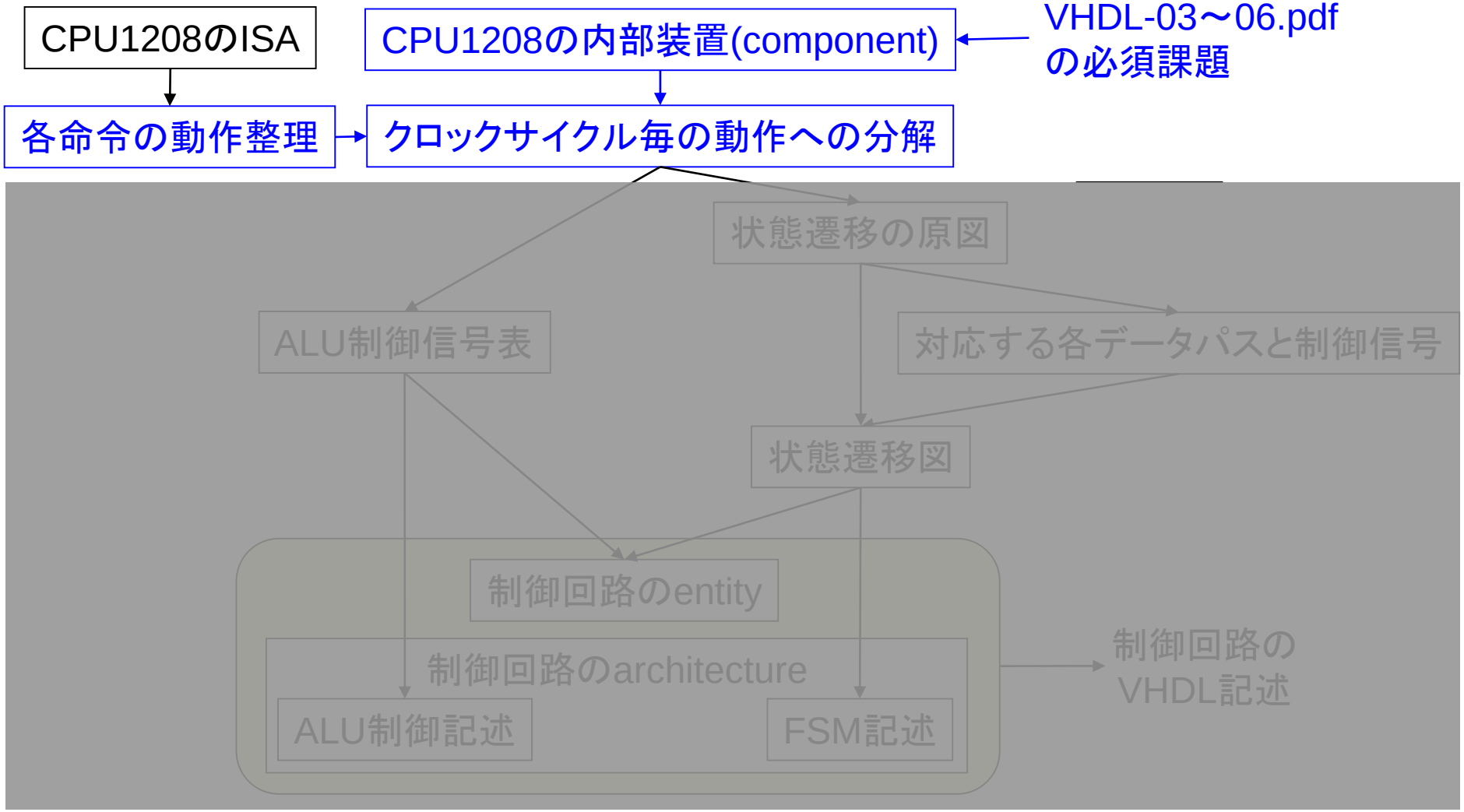
基本命令分類	基本命令機能	基本命令名	機械語オペコード	ポストニブル	機械語オペランド	機械語命令	機械語命令長	実行サイクル数	ゼロフラグビット	アドレッシングモード	動作説明
									Z		
データ転送	ロード	LD	0	0	hh	00hh	2	2	●	Immediate	hh→GR
テスト	ビットテスト	BIT	1	0	hh	10hh	2	2	↑	Immediate	GR and hh, regist Z
テスト	等価テスト	EQU	3	0	hh	30hh	2	2	↑	Immediate	GR eor hh, regist Z
算術演算	加算	ADD	4	0	hh	40hh	2	3	↑	Immediate	GR + hh→GR, regist Z
論理演算	論理積	AND	5	0	hh	50hh	2	3	↑	Immediate	GR and hh→GR, regist Z
論理演算	論理和	OR	6	0	hh	60hh	2	3	↑	Immediate	GR or hh→GR, regist Z
論理演算	排他的論理和	EOR	7	0	hh	70hh	2	3	↑	Immediate	GR eor hh→GR, regist Z
算術演算	加算	ADD	8		hhh	8hhh	2	4	↑	Extended	GR + M[hhh]→GR, regist Z
論理演算	論理積	AND	9		hhh	9hhh	2	4	↑	Extended	GR and M[hhh]→GR, regist Z
論理演算	論理和	OR	A		hhh	Ahhh	2	4	↑	Extended	GR or M[hhh]→GR, regist Z
論理演算	排他的論理和	EOR	B		hhh	Bhhh	2	4	↑	Extended	GR eor M[hhh]→GR, regist Z
実行順序変更	ジャンプ	JZE	C		hhh	Chhh	2	3	●	Extended	if Z=1 then hhh→PC
実行順序変更	ジャンプ	JMP	D		hhh	Dhhh	2	3	●	Extended	hhh→PC
データ転送	ロード	LD	E		hhh	Ehhh	2	3	●	Extended	M[hhh]→GR
データ転送	ストア	ST	F		hhh	Fhhh	2	3	●	Extended	GR→M[hhh]

各命令動作の整理

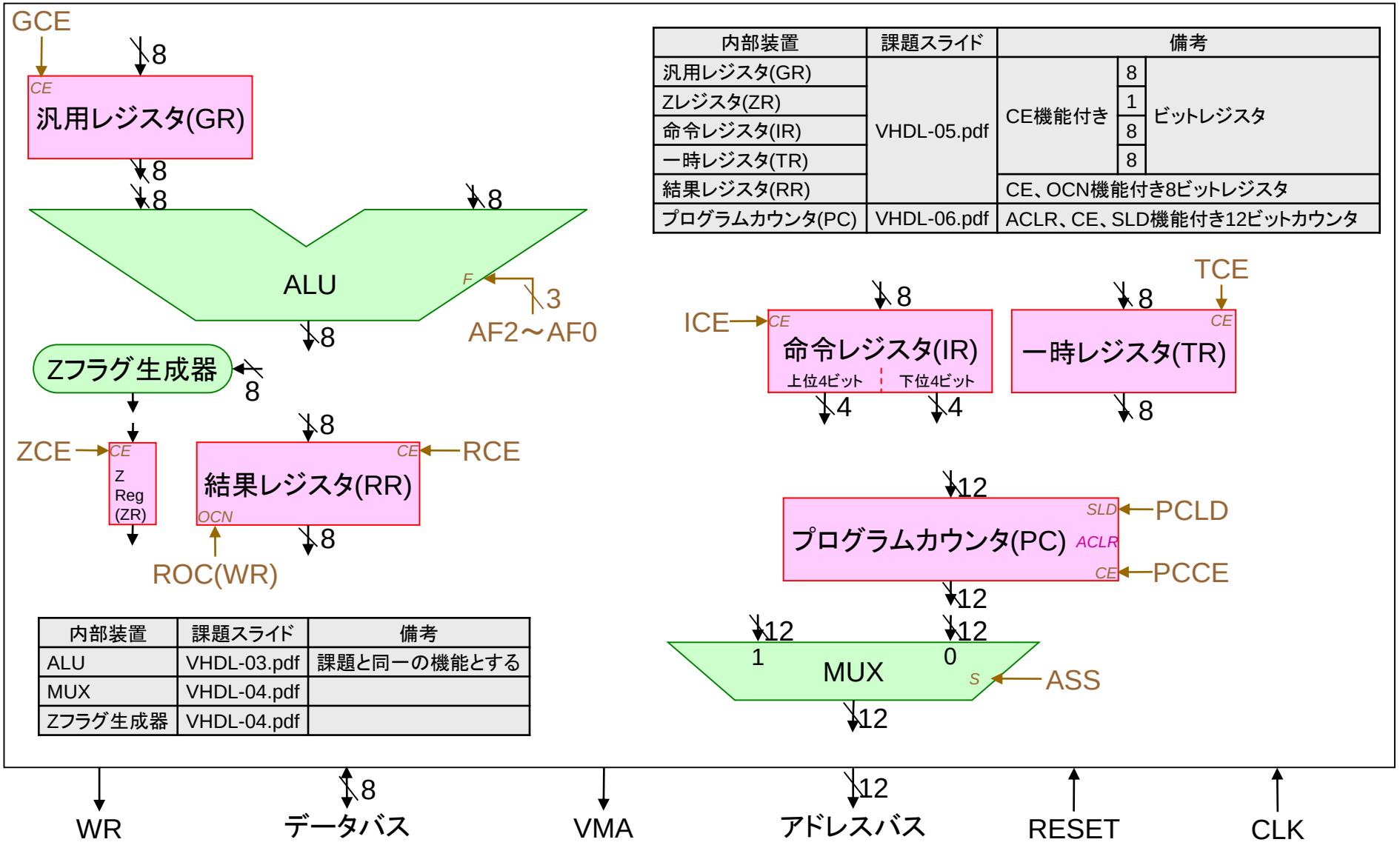
基本命令分類	基本命令機能	基本命令名	機械語オペコード	ポストニブル	機械語オペランド	機械語命令	機械語命令長	実行サイクル数	ゼロフラグビット	アドレッシングモード	動作説明	動作説明（整理版）	
									Z				
データ転送	ロード	LD	0	0	hh	00hh	2	2	●	Immediate	hh→GR	hh→GR	
テスト	ビットテスト	BIT	1	0	hh	10hh	2	2	↑	Immediate	GR and hh, regist Z	GR △ hh, regist Z	△⇔and
テスト	等価テスト	EQU	3	0	hh	30hh	2	2	↑	Immediate	GR eor hh, regist Z		△⇔eor
算術演算	加算	ADD	4	0	hh	40hh	2	3	↑	Immediate	GR + hh→GR, regist Z	GR △ hh→GR, regist Z	△⇔+
論理演算	論理積	AND	5	0	hh	50hh	2	3	↑	Immediate	GR and hh→GR, regist Z		△⇔and
論理演算	論理和	OR	6	0	hh	60hh	2	3	↑	Immediate	GR or hh→GR, regist Z		△⇔or
論理演算	排他的論理和	EOR	7	0	hh	70hh	2	3	↑	Immediate	GR eor hh→GR, regist Z		△⇔eor
算術演算	加算	ADD	8		hhh	8hhh	2	4	↑	Extended	GR + M[hhh]→GR, regist Z	GR △ M[hhh]→GR, regist Z	△⇔+
論理演算	論理積	AND	9		hhh	9hhh	2	4	↑	Extended	GR and M[hhh]→GR, regist Z		△⇔and
論理演算	論理和	OR	A		hhh	Ahhh	2	4	↑	Extended	GR or M[hhh]→GR, regist Z		△⇔or
論理演算	排他的論理和	EOR	B		hhh	Bhhh	2	4	↑	Extended	GR eor M[hhh]→GR, regist Z		△⇔eor
実行順序変更	ジャンプ	JZE	C		hhh	Chhh	2	3	●	Extended	if Z=1 then hhh→PC	if Z=1 then hhh→PC	
実行順序変更	ジャンプ	JMP	D		hhh	Dhhh	2	3	●	Extended	hhh→PC	hhh→PC	
データ転送	ロード	LD	E		hhh	Ehhh	2	3	●	Extended	M[hhh]→GR	M[hhh]→GR	
データ転送	ストア	ST	F		hhh	Fhhh	2	3	●	Extended	GR→M[hhh]	GR→M[hhh]	

共有項整理

手順


CPU1208のISA
CPU1208の内部装置(component)
VHDL-03～06.pdf
の必須課題
各命令の動作整理
クロックサイクル毎の動作への分解
状態遷移の原図
ALU制御信号表
対応する各データパスと制御信号
状態遷移図
制御回路のentity
制御回路のarchitecture
ALU制御記述
FSM記述
制御回路の
VHDL記述

CPU1208　外部インタフェース　内部装置（component）



内部装置	課題スライド	備考
汎用レジスタ(GR)	VHDL-05.pdf	CE機能付き 8 1 8 8 ビットレジスタ
Zレジスタ(ZR)		
命令レジスタ(IR)		
一時レジスタ(TR)		
結果レジスタ(RR)		CE、OCN機能付き8ビットレジスタ
プログラムカウンタ(PC)	VHDL-06.pdf	ACLR、CE、SLD機能付き12ビットカウンタ

内部装置	課題スライド	備考
ALU	VHDL-03.pdf	課題と同一の機能とする
MUX	VHDL-04.pdf	
Zフラグ生成器	VHDL-04.pdf	

組合せ回路　順序回路

VMA (Valid Memory Address)：アドレス有効信号、WR (Write)：メモリ書込み信号　※共に1で有効

制御(回路の出力)信号	VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD	AF2	AF1	AF0

各命令動作→クロックサイクル毎の動作への分解

基本命令名	機械語命令	実行サイクル数	ゼロフラグビット Z	アドレッシングモード	動作説明(整理版)		1cyc	2cyc		3cyc		4cyc
LD	00hh	2	●	Immediate	hh→GR		M[PC]→IR, PC+1→PC	M[PC]→GR, PC+1→PC		—		—
BIT	10hh	2	↑	Immediate	GR △ hh, regist Z	△⇔and		GR △ M[PC], regist ZR, PC+1→PC	△⇔and			
EQU	30hh	2	↑	Immediate		△⇔eor			△⇔eor			
ADD	40hh	3	↑	Immediate	GR △ hh→GR, regist Z	△⇔+		GR △ M[PC] → RR, regist ZR, PC+1→PC	△⇔+	RR→GR		
AND	50hh	3	↑	Immediate		△⇔and			△⇔and			
OR	60hh	3	↑	Immediate		△⇔or			△⇔or			
EOR	70hh	3	↑	Immediate		△⇔eor			△⇔eor			
ADD	8hhh	4	↑	Extended	GR △ M[hhh]→GR, regist Z	△⇔+		M[PC]→TR, PC+1→PC		GR △ M[IRL&TR] → RR, regist ZR	△⇔+	RR→GR
AND	9hhh	4	↑	Extended		△⇔and					△⇔and	
OR	Ahhh	4	↑	Extended		△⇔or					△⇔or	
EOR	Bhhh	4	↑	Extended		△⇔eor					△⇔eor	
JZE	Chhh	3	●	Extended	if Z=1 then hhh→PC			M[PC]→TR, GR→RR, PC+1→PC	△⇔thr	if ZR=1 then IRL&TR→PC		—
JMP	Dhhh	3	●	Extended	hhh→PC					IRL&TR→PC		
LD	Ehhh	3	●	Extended	M[hhh]→GR					M[IRL&TR]→GR		
ST	Fhhh	3	●	Extended	GR→M[hhh]					RR→M[IRL&TR]		

先頭4ビットがOPcode

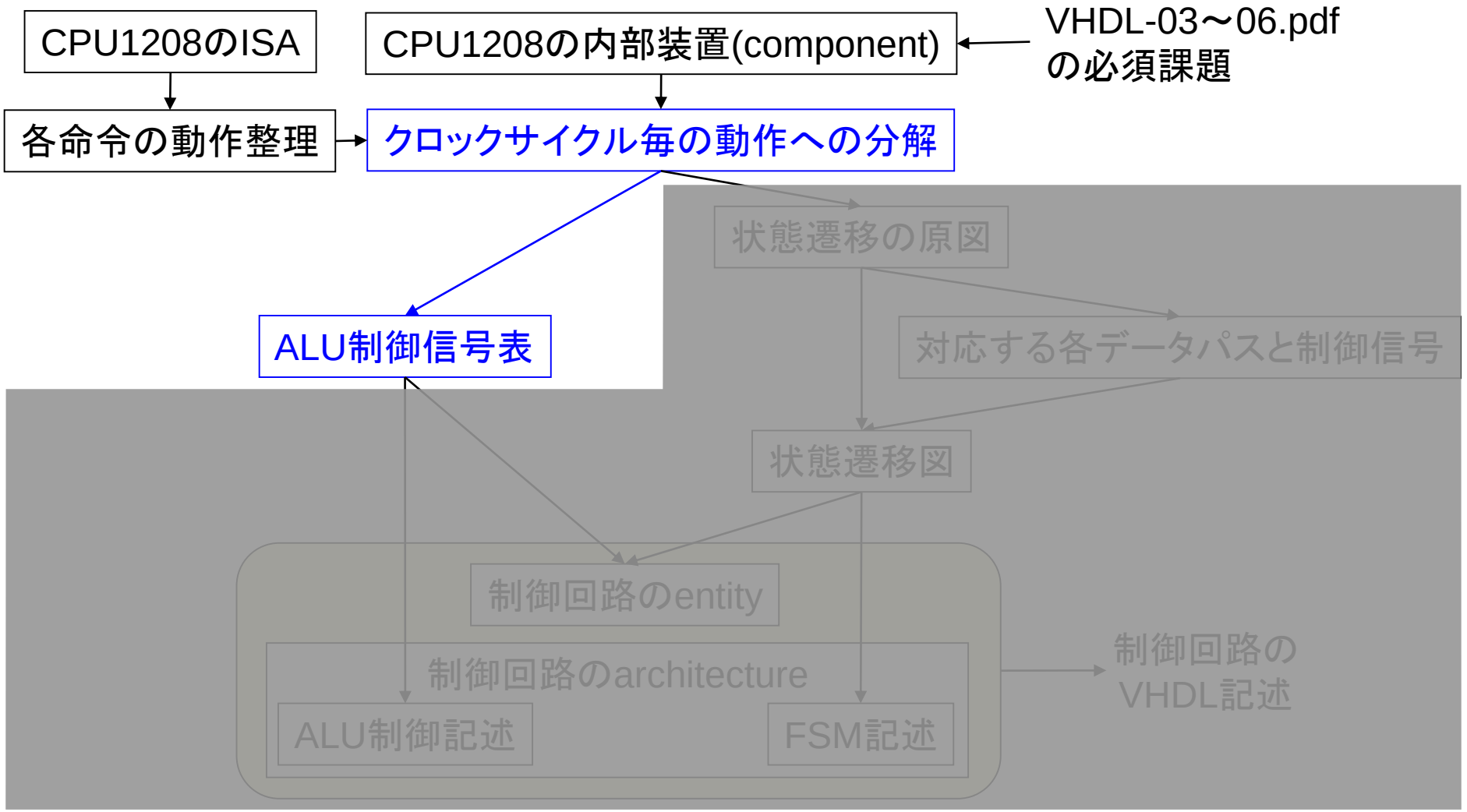
命令フェッチ開始

OPcodeは、IRHに格納される。

この先、IRHの値によってそれぞれ、5、4分岐する。

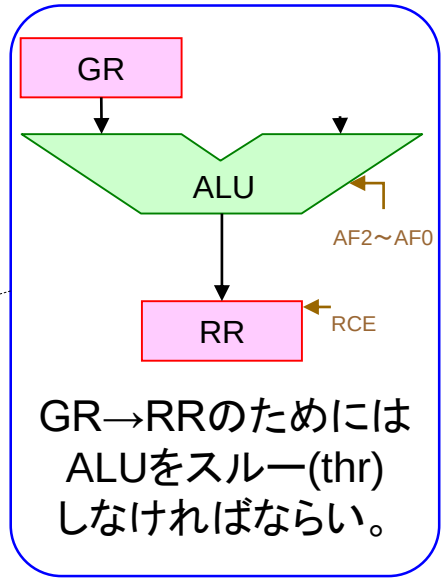
共通項

手順


CPU1208のISA
CPU1208の内部装置(component)
VHDL-03～06.pdf
の必須課題
各命令の動作整理
クロックサイクル毎の動作への分解
状態遷移の原図
ALU制御信号表
対応する各データパスと制御信号
状態遷移図
制御回路のentity
制御回路のarchitecture
ALU制御記述
FSM記述
制御回路の
VHDL記述

クロックサイクル毎の動作への分解→ALU制御信号表

		1cyc	2cyc		3cyc		4cyc
LD	00hh	M[PC]→IR,PC+1→PC	M[PC]→GR, PC+1→PC		—		—
BIT	10hh		GR △ M[PC], regist ZR, PC+1→PC	△⇔and			
EQU	30hh			△⇔eor			
ADD	40hh		GR △ M[PC] → RR, regist ZR, PC+1→PC	△⇔+	RR→GR		
AND	50hh			△⇔and			
OR	60hh			△⇔or			
EOR	70hh			△⇔eor			
ADD	8hhh		M[PC]→TR, PC+1→PC		GR △ M[IRL&TR] → RR, regist ZR	△⇔+	RR→GR
AND	9hhh					△⇔and	
OR	Ahhh					△⇔or	
EOR	Bhhh					△⇔eor	
JZE	Chhh		M[PC]→TR, GR→RR, PC+1→PC	△⇔thr	if ZR=1 then IRL&TR→PC		—
JMP	Dhhh				IRL&TR→PC		
LD	Ehhh				M[IRL&TR]→GR		
ST	Fhhh				RR→M[IRL&TR]		



GR→RRのためには
ALUをスルー(thr)
しなければならい。

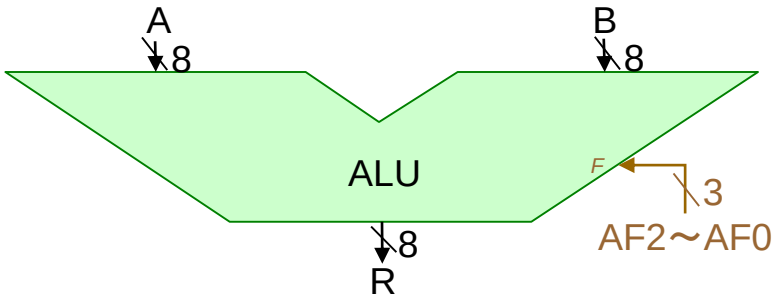
IRH

ALU機能の制御信号はIRHより決まる。

入力 IRH	出力 AF2～AF0	備考(△)	備考(基本命令命)
4	Think1	+	ADD
8			
1		and	BIT
5			AND
9			
6		or	OR
A			
3		eor	EQU
7			EOR
B			
0		thr	LD
C			JZE
D			JMP
E			LD
F			ST

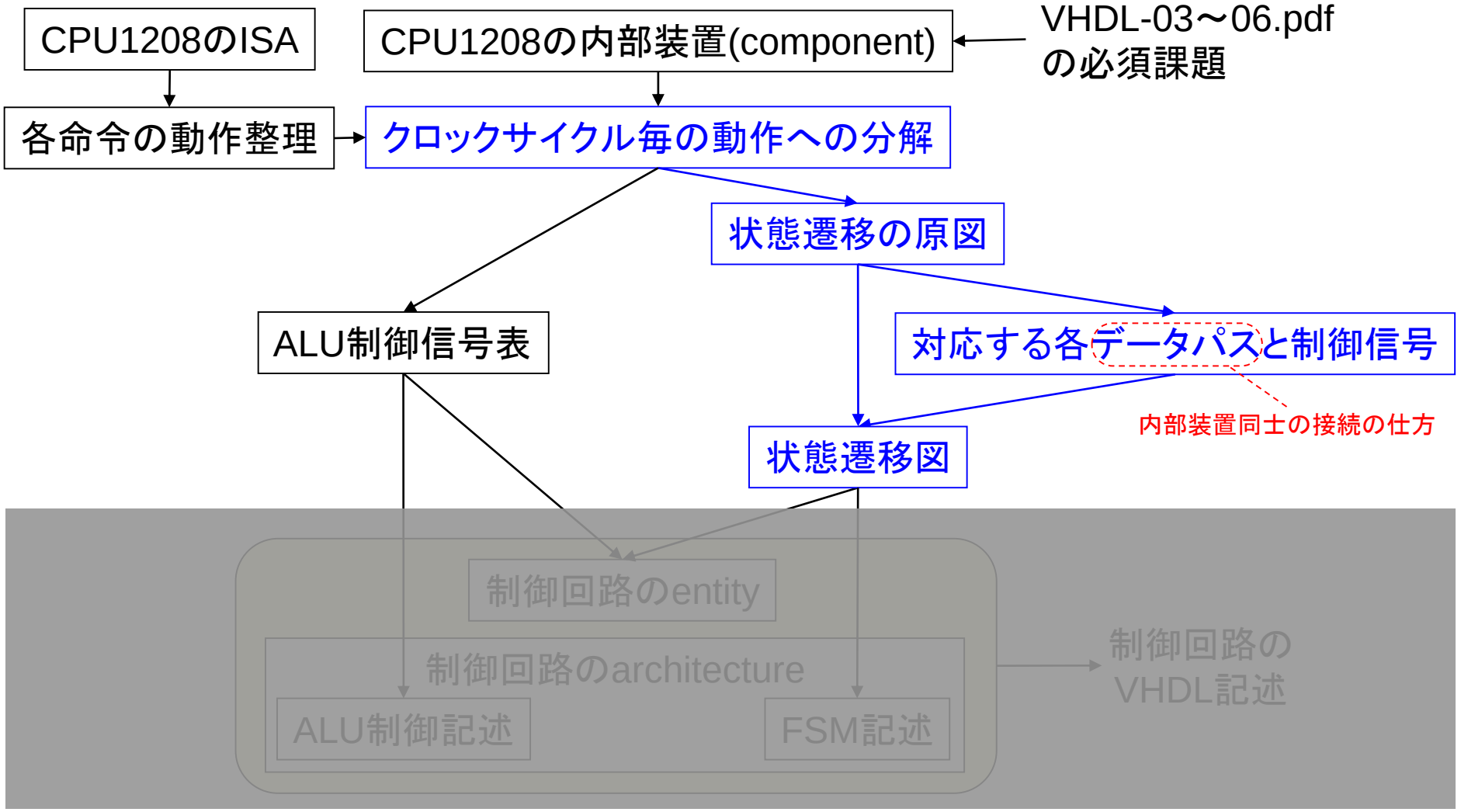
これは、原理的に
任意でよいので、
ここに定めた。

参考：VHDL-03.pdfより



入力			出力
AF2～AF0			R
0	0	0	AとBの算術加算値(キャリー出力なし)
0	0	1	AとBの論理積の値
0	1	0	AとBの論理和の値
0	1	1	AとBの排他的論理和の値
その他			Aの値

手順


CPU1208のISA
CPU1208の内部装置(component)
VHDL-03～06.pdf
の必須課題
各命令の動作整理
クロックサイクル毎の動作への分解
状態遷移の原図
ALU制御信号表
対応する各データパスと制御信号
内部装置同士の接続の仕方
状態遷移図
制御回路のentity
制御回路のarchitecture
ALU制御記述
FSM記述
制御回路の
VHDL記述

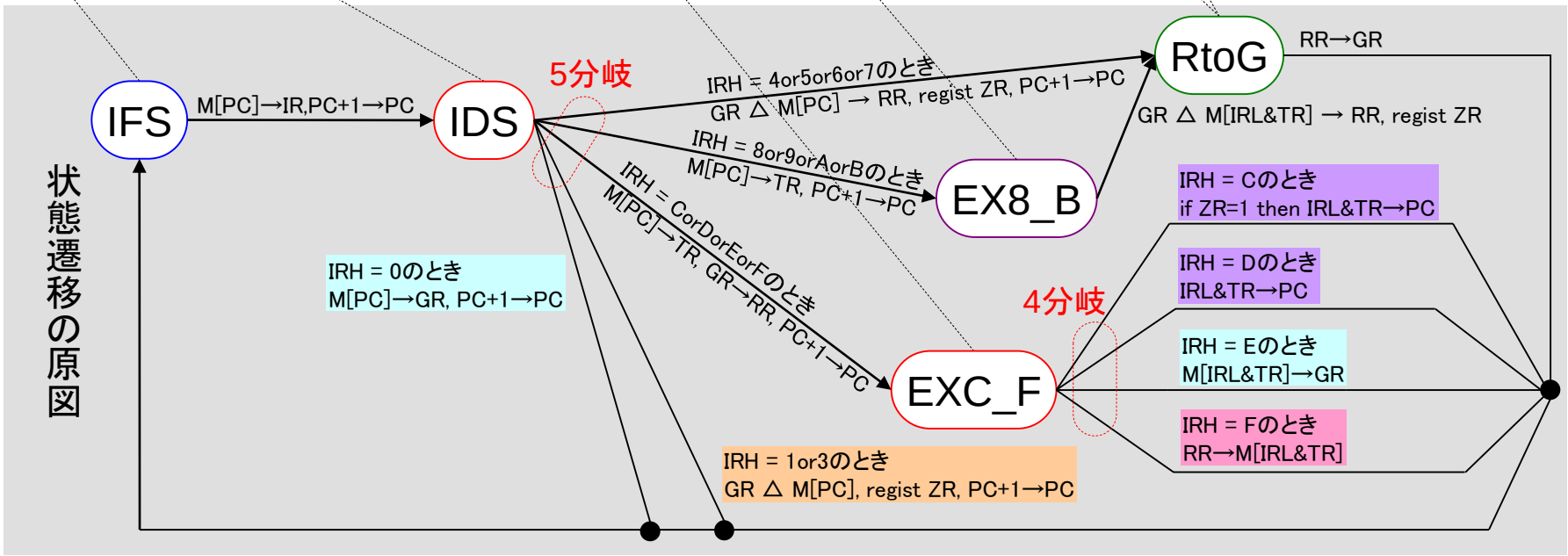
クロックサイクル毎の動作への分解→状態遷移の原図

		1cyc	2cyc		3cyc		4cyc
LD	00hh	M[PC]→IR,PC+1→PC	M[PC]→GR, PC+1→PC		—		—
BIT	10hh		GR △ M[PC], regist ZR, PC+1→PC	△⇔and			
EQU	30hh			△⇔eor			
ADD	40hh		GR △ M[PC] → RR, regist ZR, PC+1→PC	△⇔+	RR→GR		
AND	50hh			△⇔and			
OR	60hh			△⇔or			
EOR	70hh			△⇔eor			
ADD	8hhh		M[PC]→TR, PC+1→PC		GR △ M[IRL&TR] → RR, regist ZR	△⇔+	RR→GR
AND	9hhh					△⇔and	
OR	Ahhh					△⇔or	
EOR	Bhhh					△⇔eor	
JZE	Chhh		M[PC]→TR, GR→RR, PC+1→PC	△⇔thr	if ZR=1 then IRL&TR→PC		—
JMP	Dhhh				IRL&TR→PC		
LD	Ehhh				M[IRL&TR]→GR		
ST	Fhhh				RR→M[IRL&TR]		

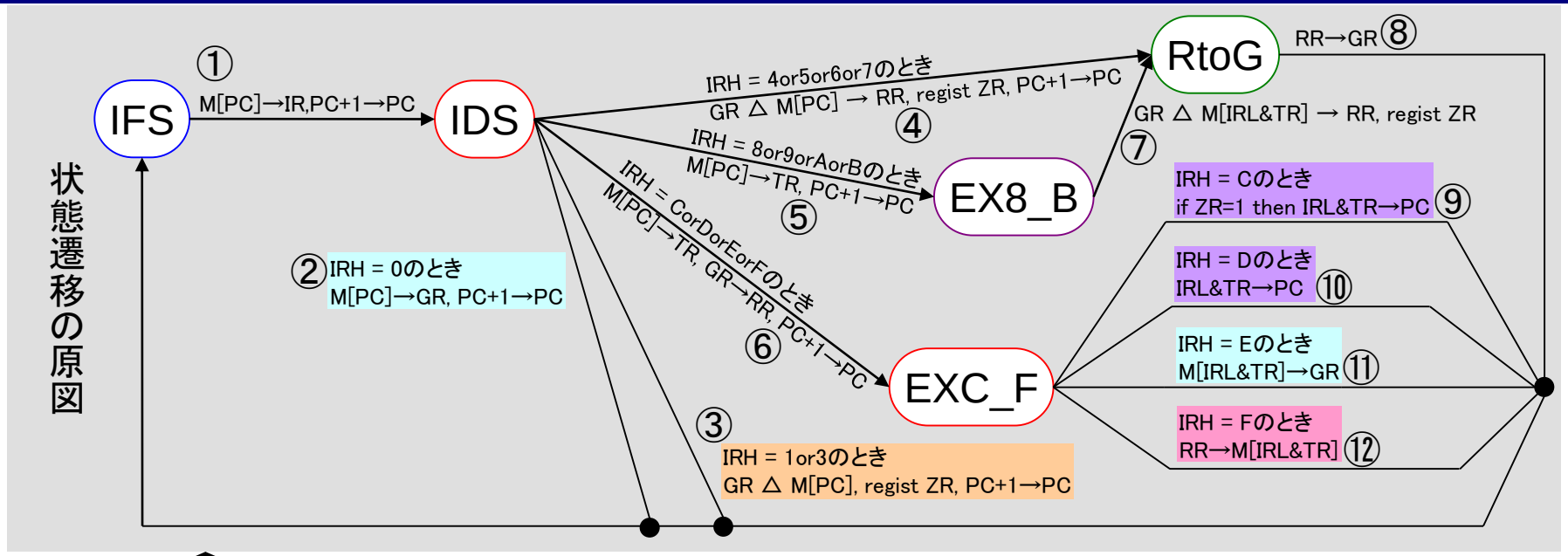
次のcycがない
↓
1cycへ戻る

IRH

状態遷移の原図



状態遷移の原図→対応する各データパスと制御信号



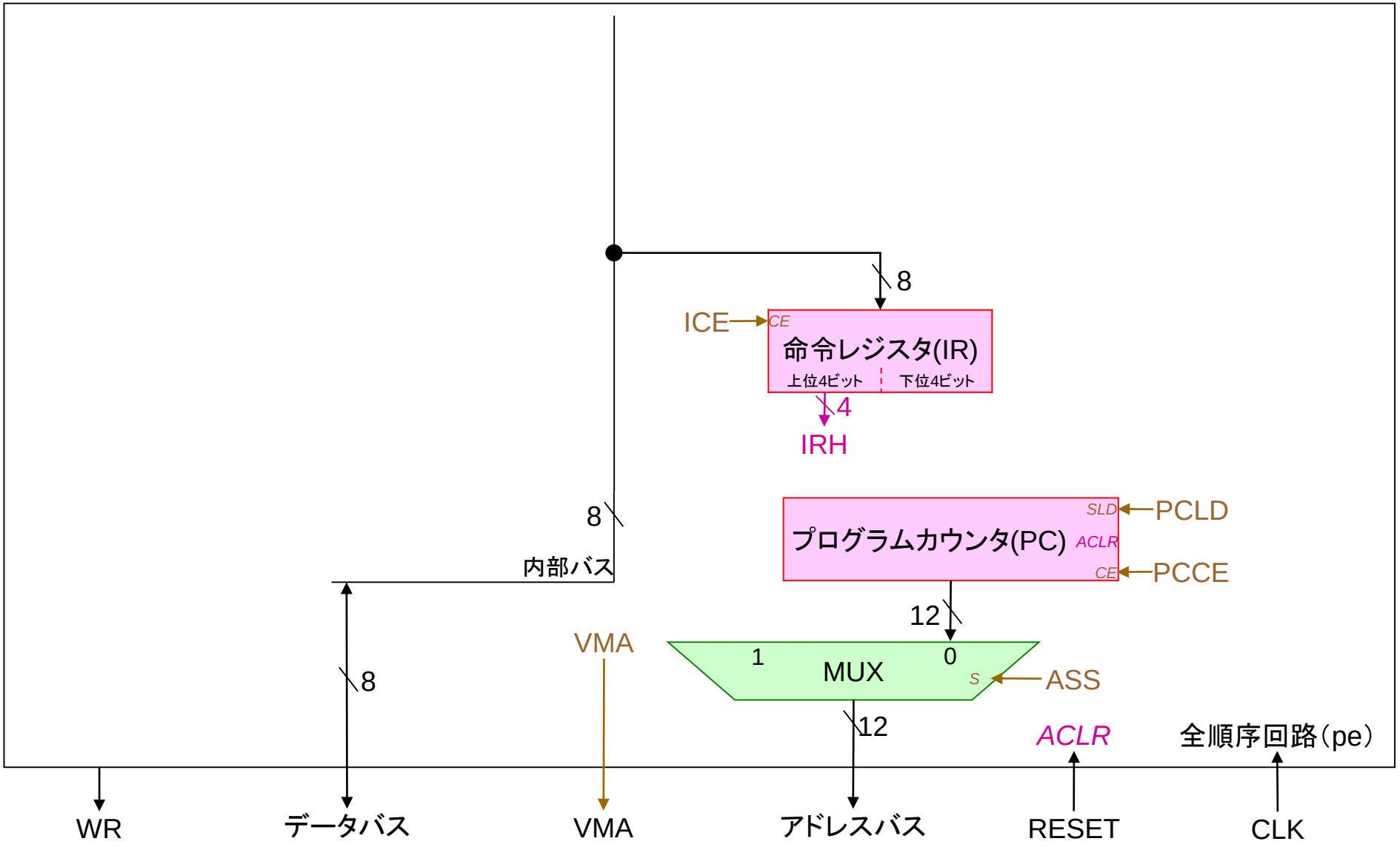
M[PC]→IR, PC+1→PC	①
M[PC]→GR, PC+1→PC	②
GR △ M[PC], regist ZR, PC+1→PC	③
GR △ M[PC] → RR, regist ZR, PC+1→PC	④
M[PC]→TR, PC+1→PC	⑤
M[PC]→TR, GR→RR, PC+1→PC	⑥
GR △ M[IRL&TR] → RR, regist ZR	⑦
RR→GR	⑧
if ZR=1 then IRL&TR→PC	⑨
IRL&TR→PC	⑩
M[IRL&TR]→GR	⑪
RR→M[IRL&TR]	⑫

①～⑫の各データパスの構築

制御信号

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
Think2									

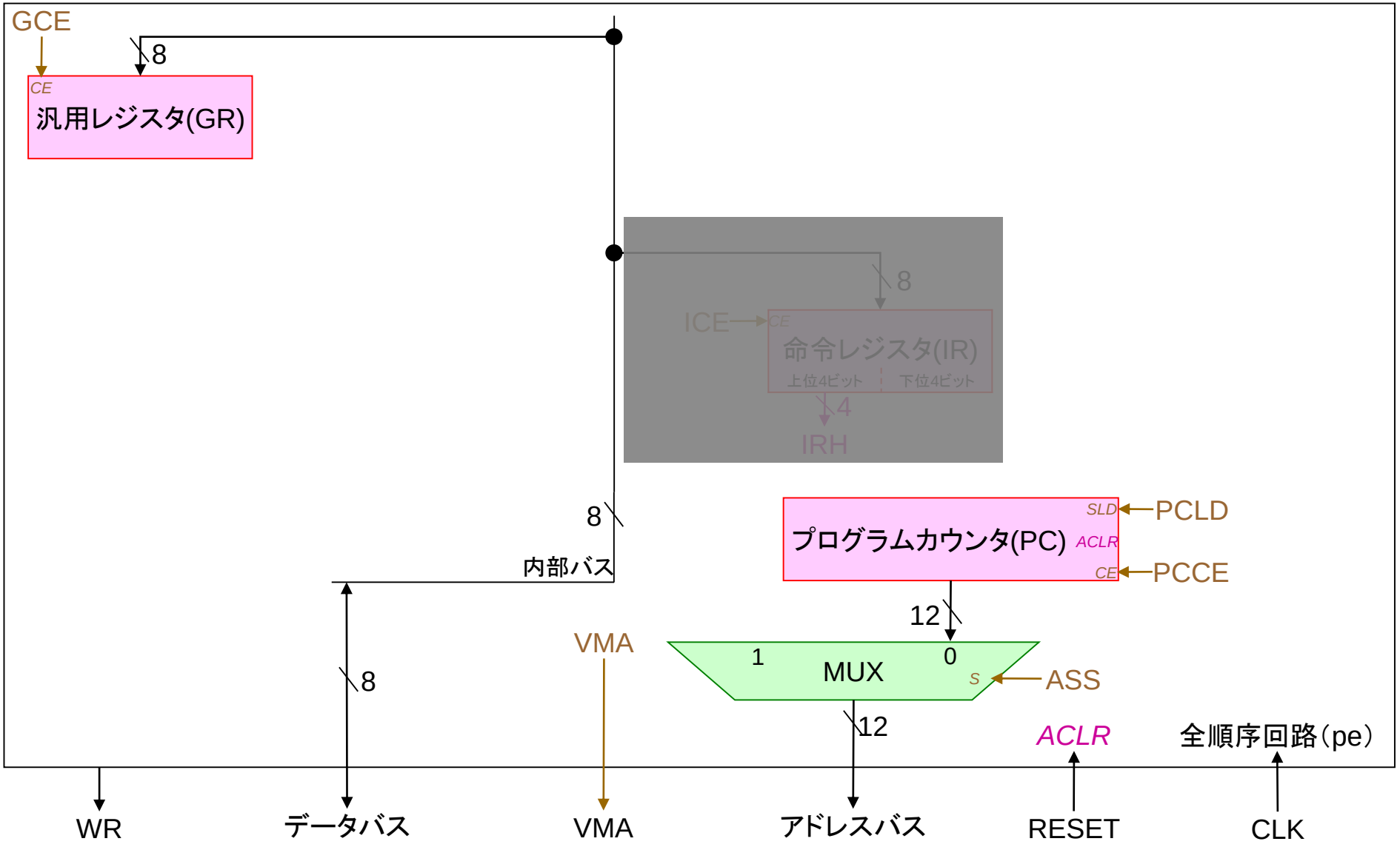
① M[PC]→IR, PC+1→PC のデータパスと制御信号



Think2-①

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0			0	0	0	0		

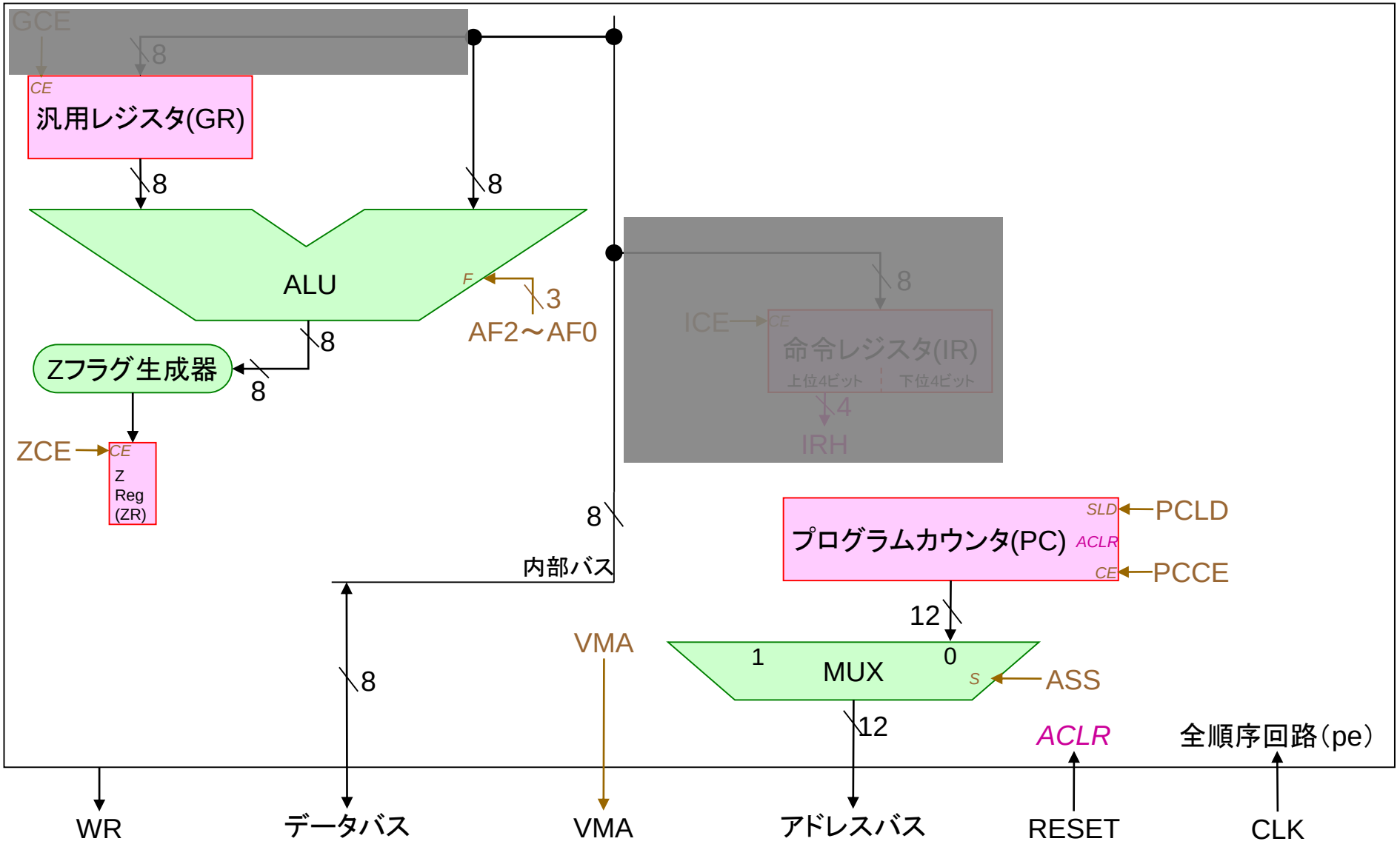
② M[PC]→GR, PC+1→PC のデータパスと制御信号



Think2-②

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0		0	0	0		

③ GR △ M[PC], regist ZR, PC+1→PC のデータパスと制御信号

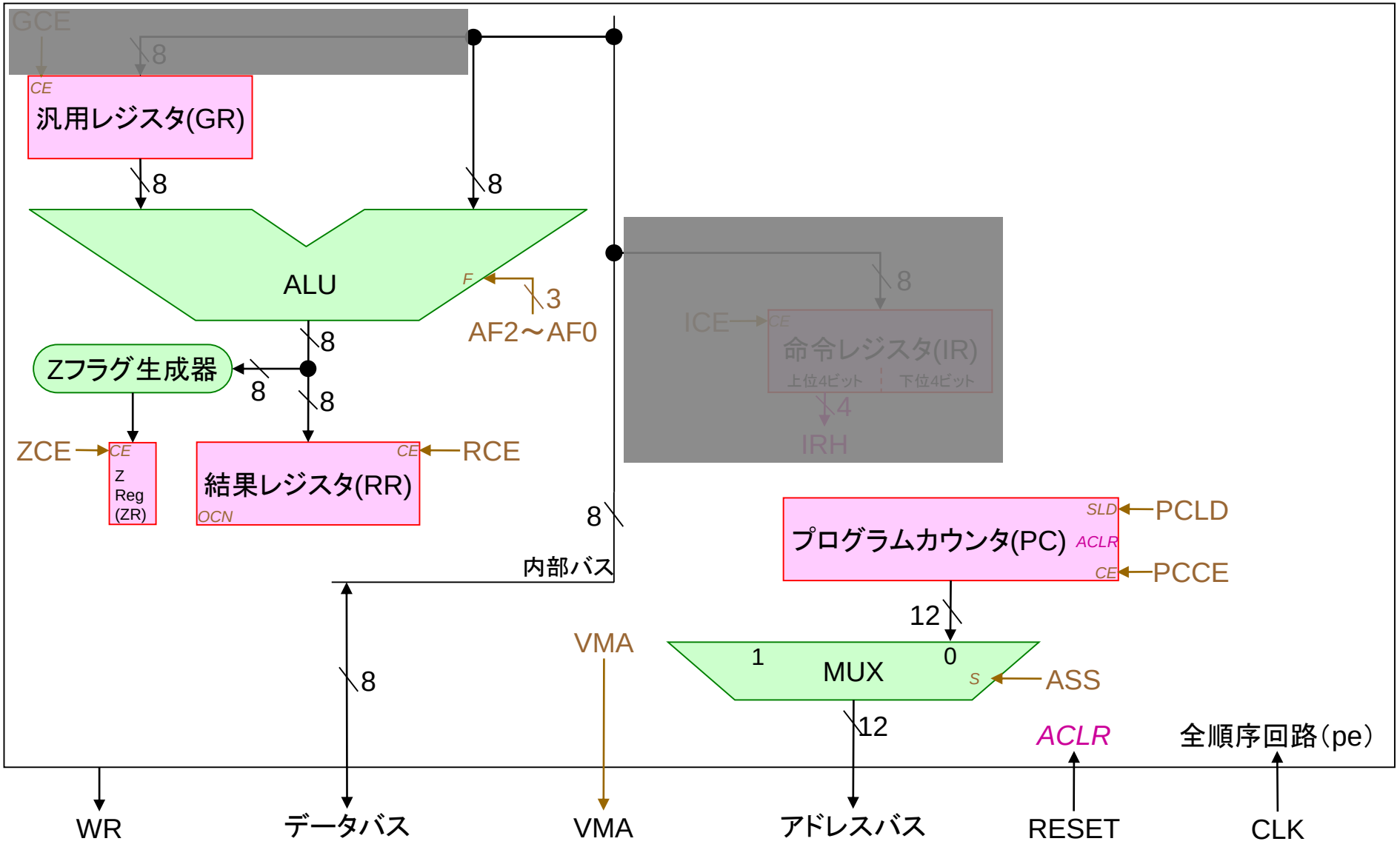


Think2-③

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0	0		0	0		

AF2～AF0については Think1で済。

④ GR △ M[PC] →RR, regist ZR, PC+1→PC のデータパスと制御信号

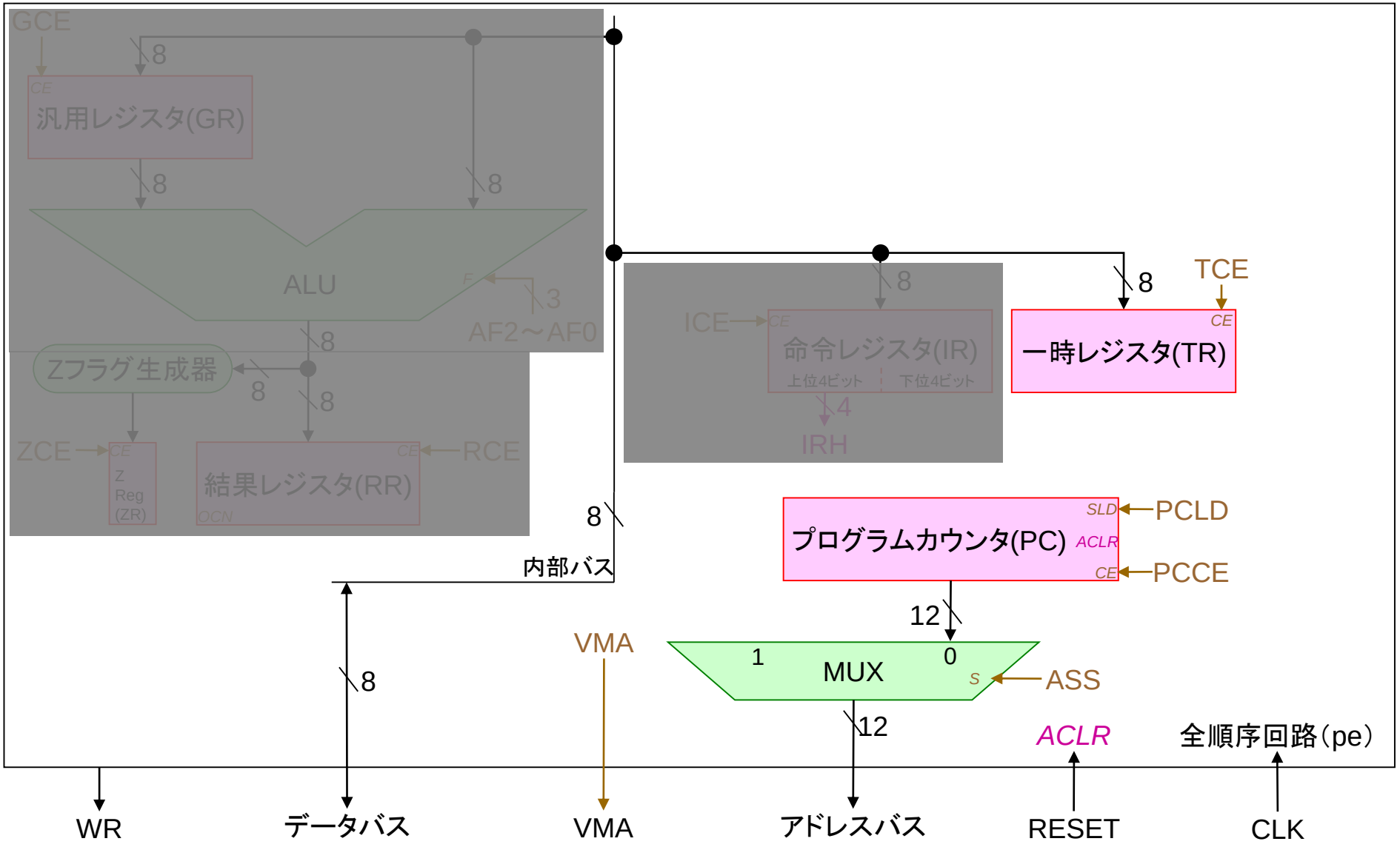


Think2-④

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0	0			0		

AF2～AF0については Think1で済。

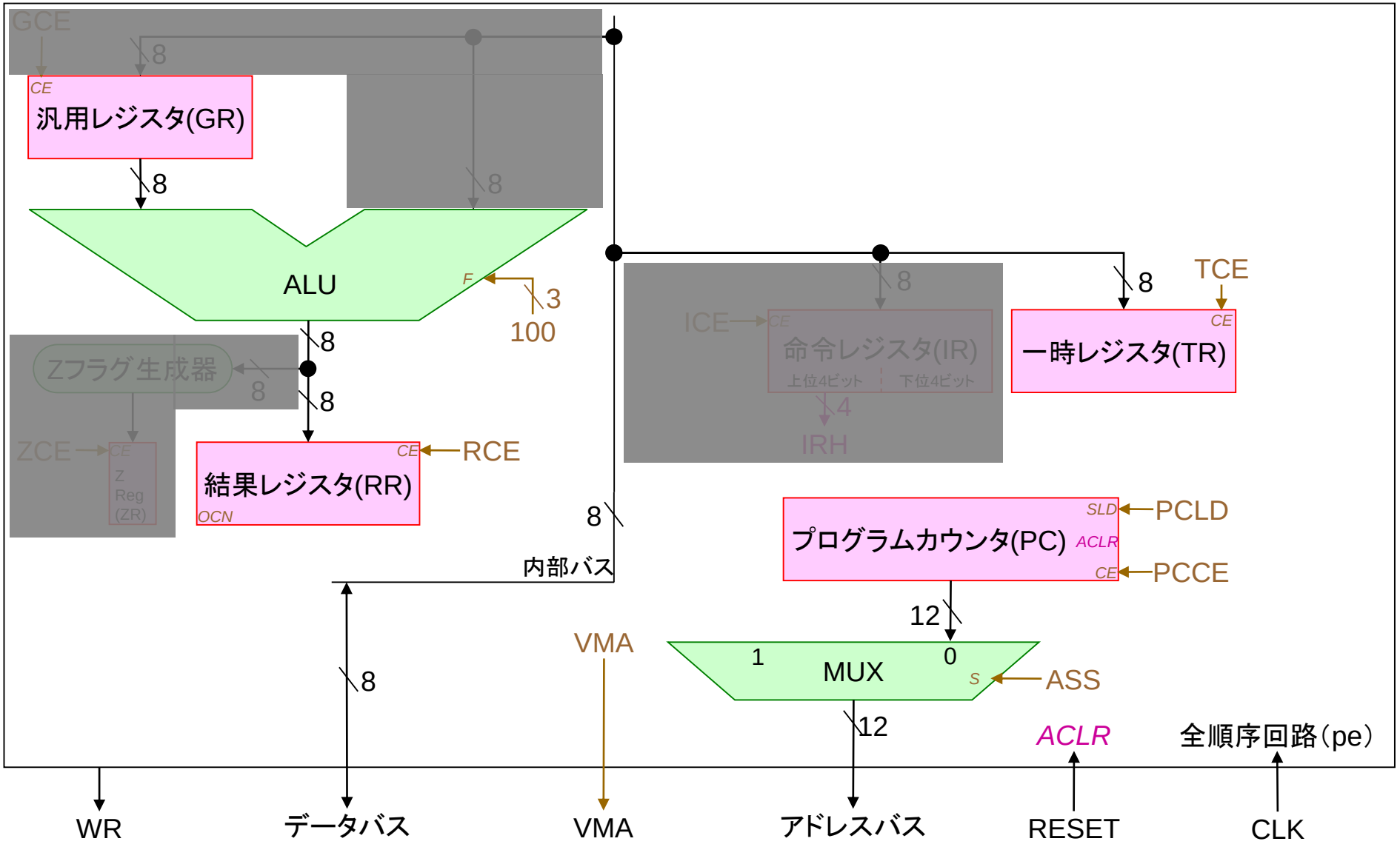
⑤ M[PC]→TR, PC+1→PC のデータパスと制御信号



Think2-⑤

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0	0	0	0			

⑥ M[PC]→TR, GR→RR, PC+1→PC のデータパスと制御信号

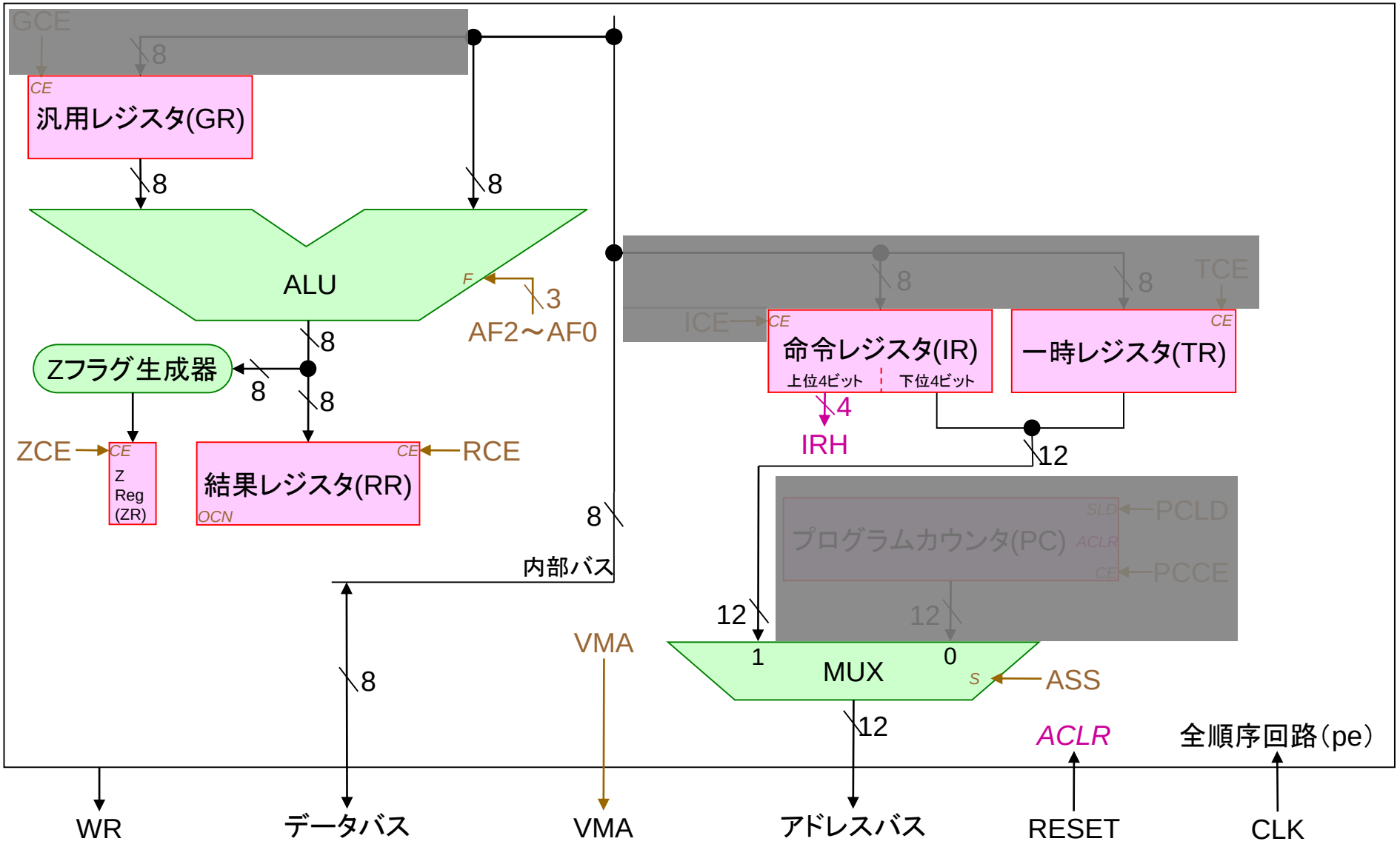


Think2-⑥

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0	0	0				

AF2～AF0については Think1で済。

⑦ GR △ M[IRL&TR] → RR, regist ZR のデータパスと制御信号

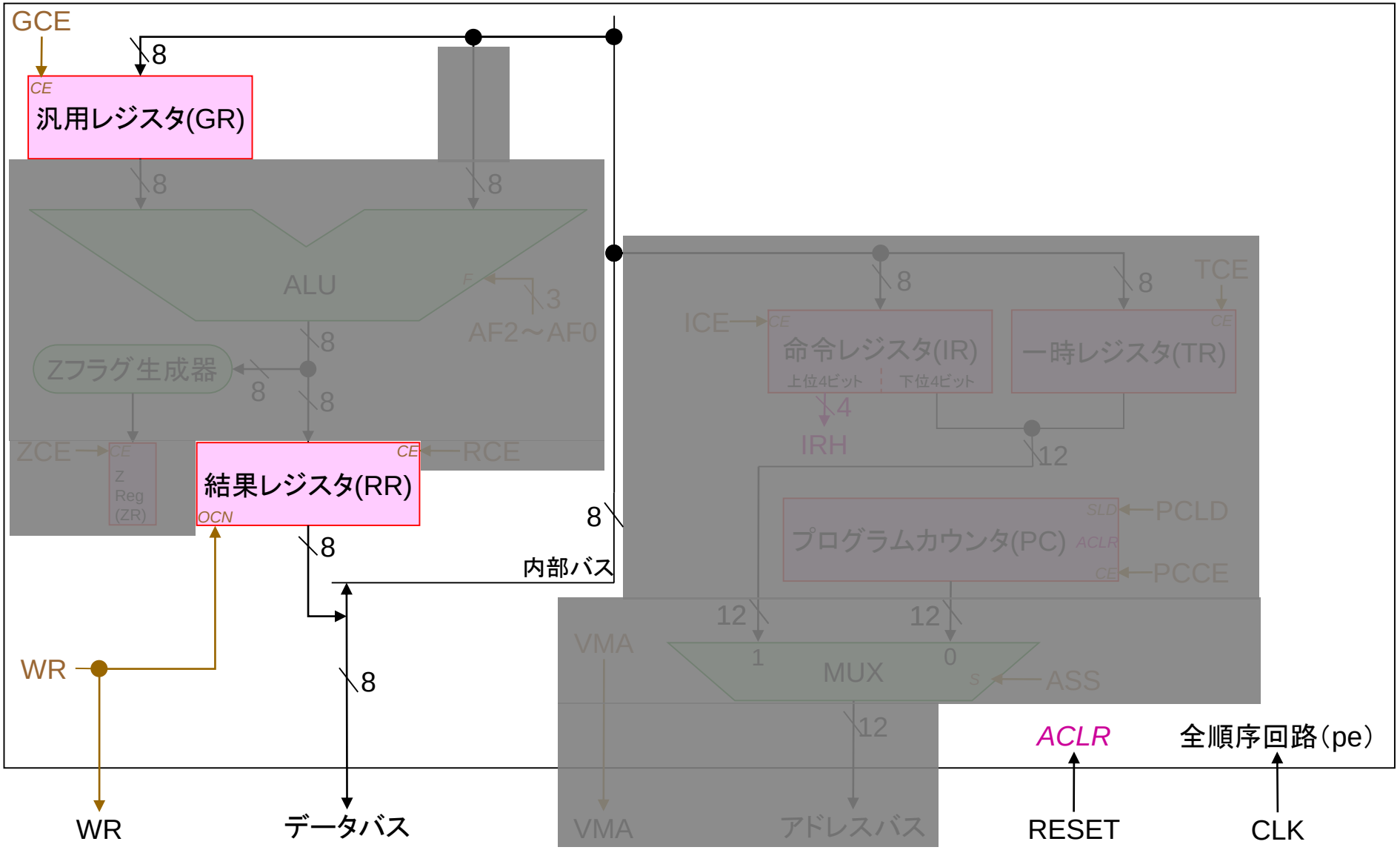


Think2-⑦

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0	0			0	0	0

AF2～AF0については Think1で済。

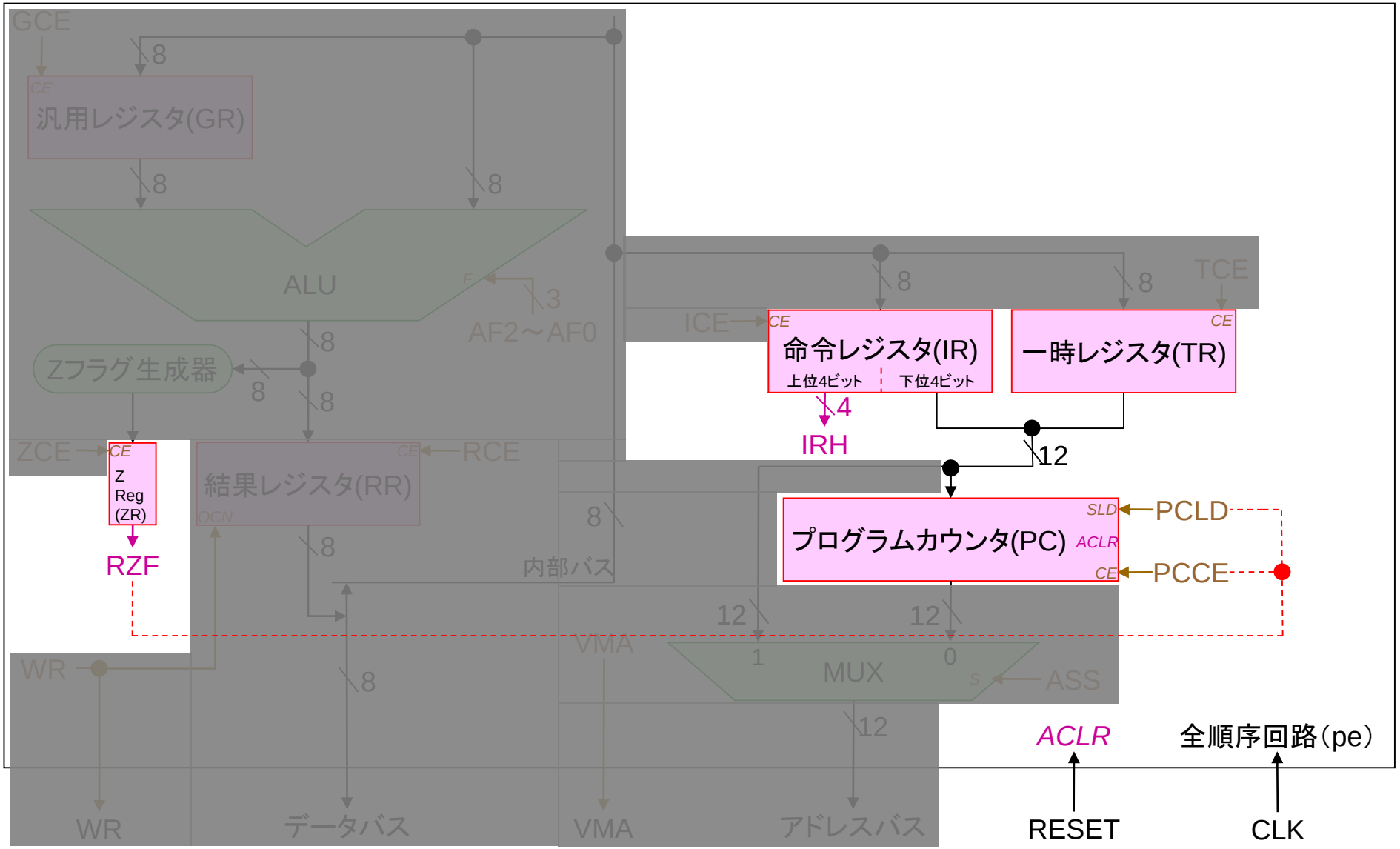
⑧ RR→GR のデータパスと制御信号



Think2-⑧

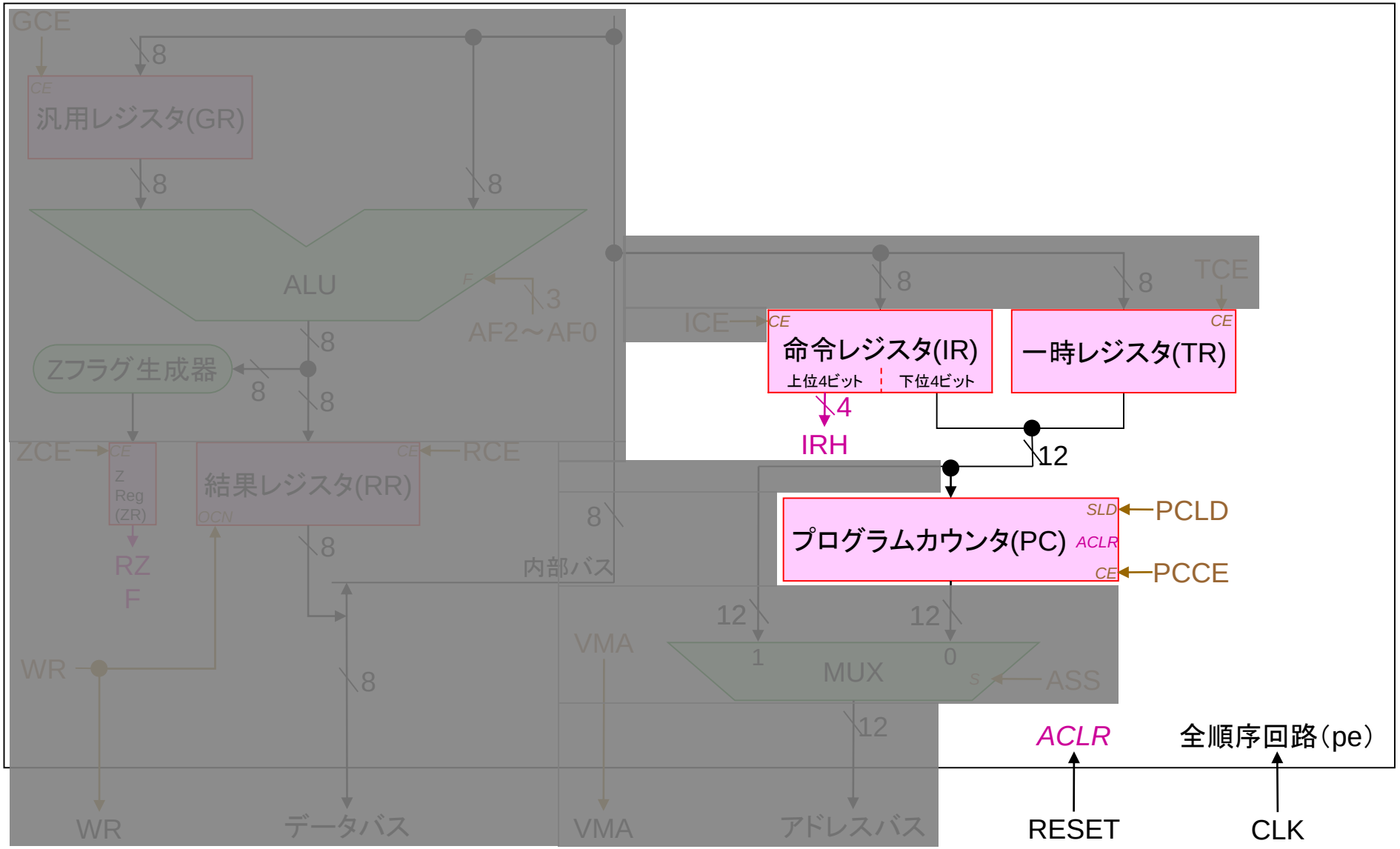
VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
0		0	0		0	0	0	0	0

⑨ if Z=1 then IRL&TR→PC のデータパスと制御信号



VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
0	0	0	0	0	0	0	0	RZF	RZF

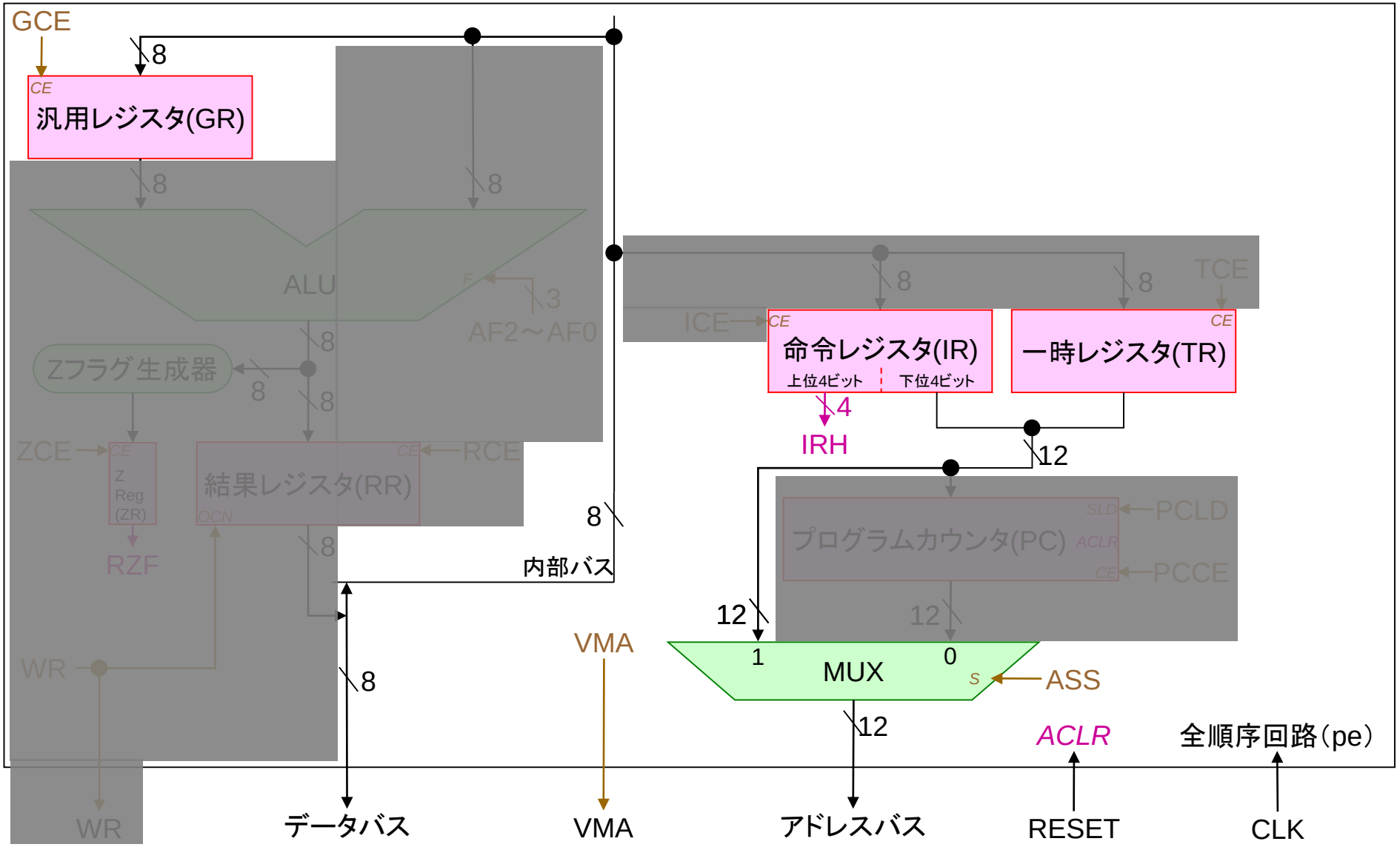
⑩ IRL&TR→PC のデータパスと制御信号



Think2-⑩

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
0	0	0	0	0	0	0	0		

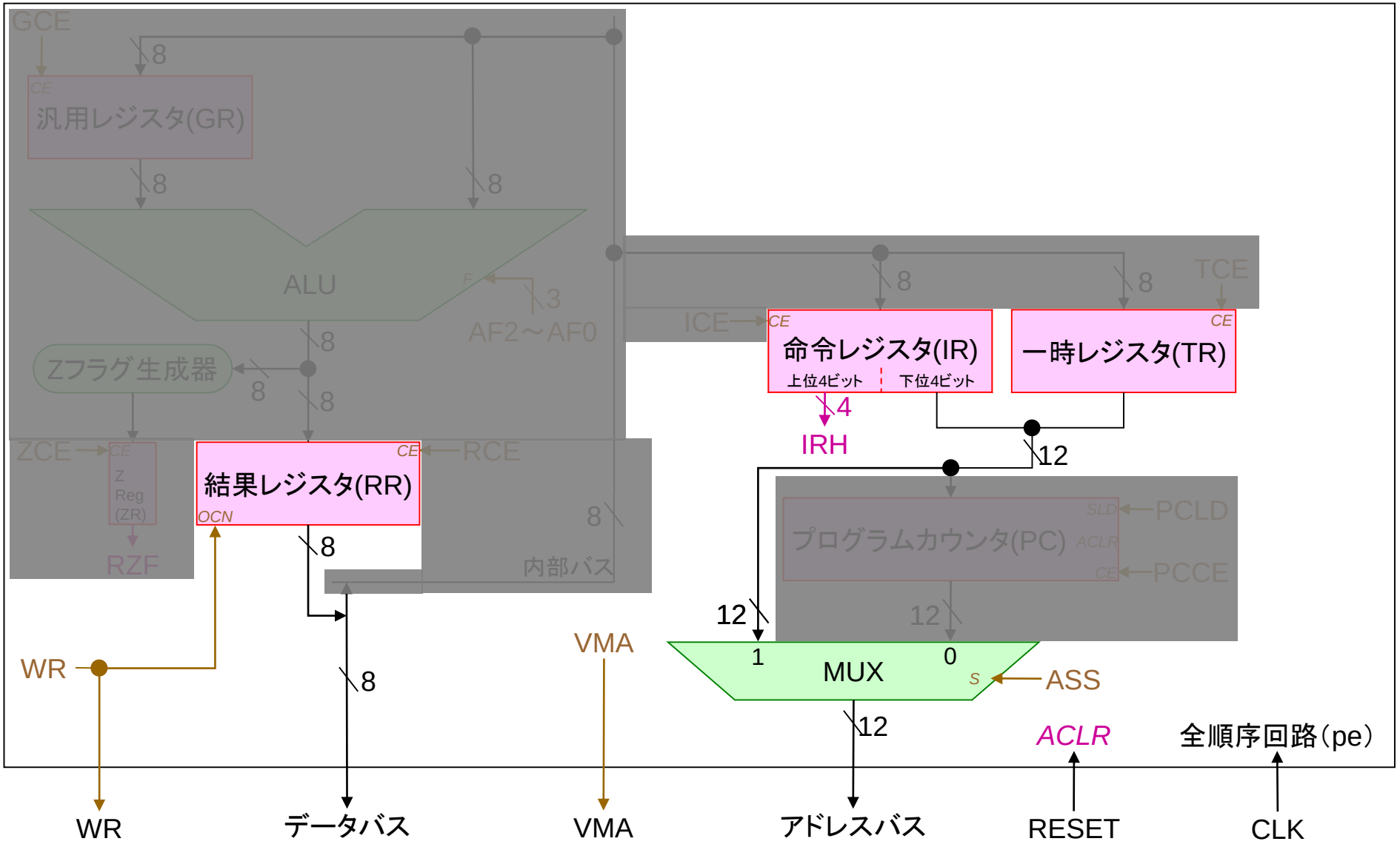
⑪ M[IRL&TR]→GR のデータパスと制御信号



Think2-⑪

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
	0		0		0	0	0	0	0

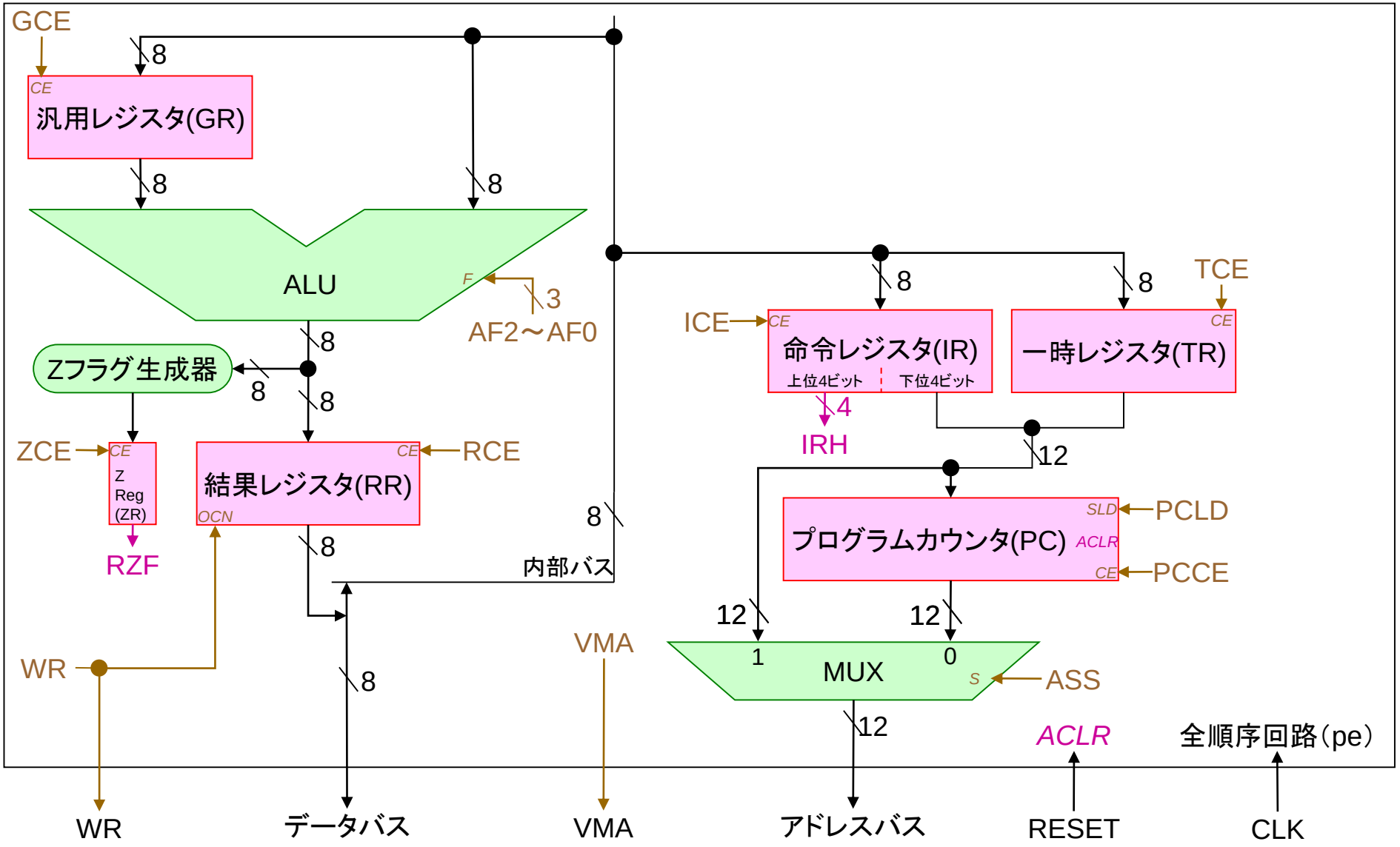
⑫ RR→M[IRL&TR] のデータパスと制御信号



Think2-⑫

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
			0	0	0	0	0	0	0

①～⑫の全データパス


GCE
8
CE
汎用レジスタ(GR)
8
8
ALU
F
3
AF2～AF0
8
Zフラグ生成器
8
8
ZCE
CE
Z
Reg
(ZR)
RZF
CE
RCE
結果レジスタ(RR)
OCN
8
WR
8
8
内部バス
VMA
8
8
TCE
ICE
CE
命令レジスタ(IR)
上位4ビット
下位4ビット
CE
一時レジスタ(TR)
4
IRH
12
SLD
PCLD
プログラムカウンタ(PC)
ACLR
CE
PCCE
12
12
1
0
MUX
S
ASS
12
ACLR
全順序回路(pe)
WR
データバス
VMA
アドレスバス
RESET
CLK

①～⑫の全制御信号

		VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
M[PC]→IR, PC+1→PC	①		0			0	0	0	0		
M[PC]→GR, PC+1→PC	②		0		0		0	0	0		
GR △ M[PC], regist ZR, PC+1→PC	③		0		0	0		0	0		
GR △ M[PC] → RR, regist ZR, PC+1→PC	④		0		0	0			0		
M[PC]→TR, PC+1→PC	⑤		0		0	0	0	0			
M[PC]→TR, GR→RR, PC+1→PC	⑥		0		0	0	0				
GR △ M[IRL&TR] → RR, regist ZR	⑦		0		0	0			0	0	0
RR→GR	⑧	0		0	0		0	0	0	0	0
if ZR=1 then IRL&TR→PC	⑨	0	0	0	0	0	0	0	0	RZF	RZF
IRL&TR→PC	⑩	0	0	0	0	0	0	0	0		
M[IRL&TR]→GR	⑪		0		0		0	0	0	0	0
RR→M[IRL&TR]	⑫				0	0	0	0	0	0	0

Think2

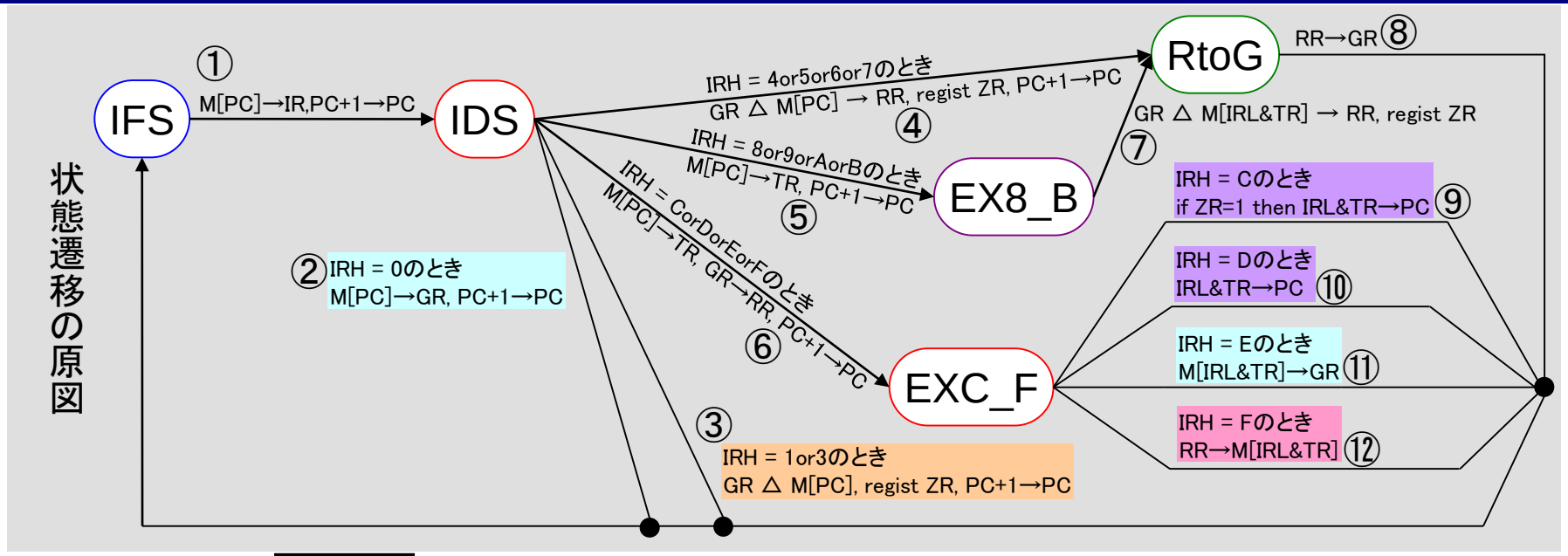
1バイト16進表現

2ビット2進表現

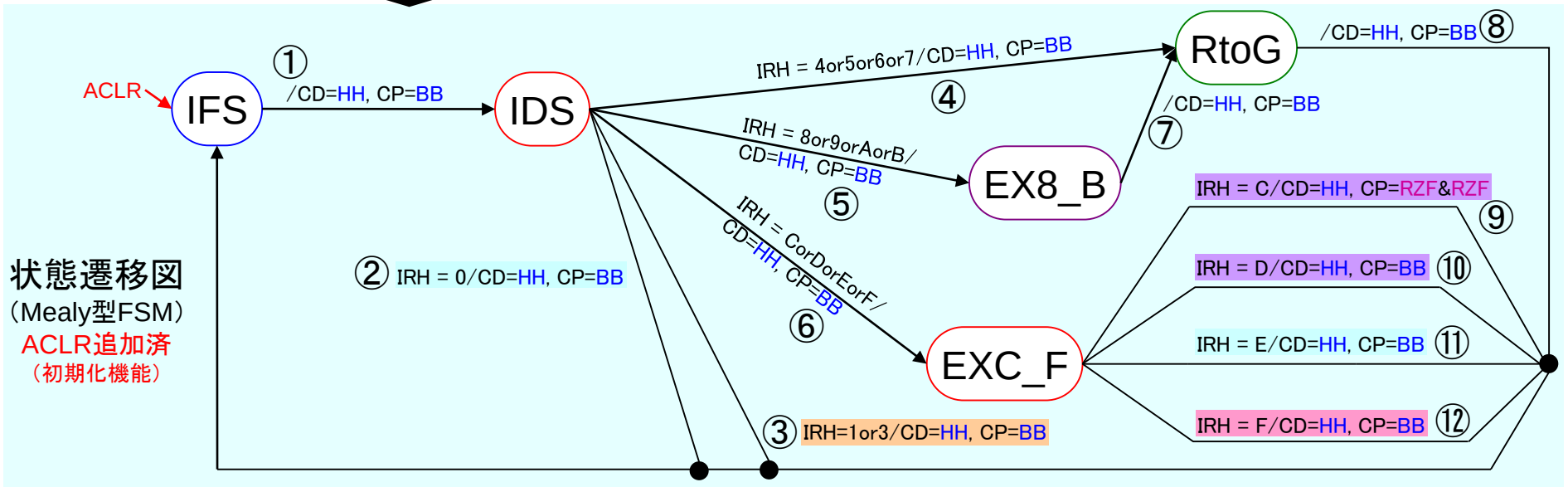
✓制御信号名の付け替え
✓値の表現の変更

		CD								CP	
		7	6	5	4	3	2	1	0	1	0
M[PC]→IR, PC+1→PC	①	HH								BB	
M[PC]→GR, PC+1→PC	②	HH								BB	
GR △ M[PC], regist ZR, PC+1→PC	③	HH								BB	
GR △ M[PC] → RR, regist ZR, PC+1→PC	④	HH								BB	
M[PC]→TR, PC+1→PC	⑤	HH								BB	
M[PC]→TR, GR→RR, PC+1→PC	⑥	HH								BB	
GR △ M[IRL&TR] → RR, regist ZR	⑦	HH								BB	
RR→GR	⑧	HH								BB	
if ZR=1 then IRL&TR→PC	⑨	HH								RZF&RZF	
IRL&TR→PC	⑩	HH								BB	
M[IRL&TR]→GR	⑪	HH								BB	
RR→M[IRL&TR]	⑫	HH								BB	

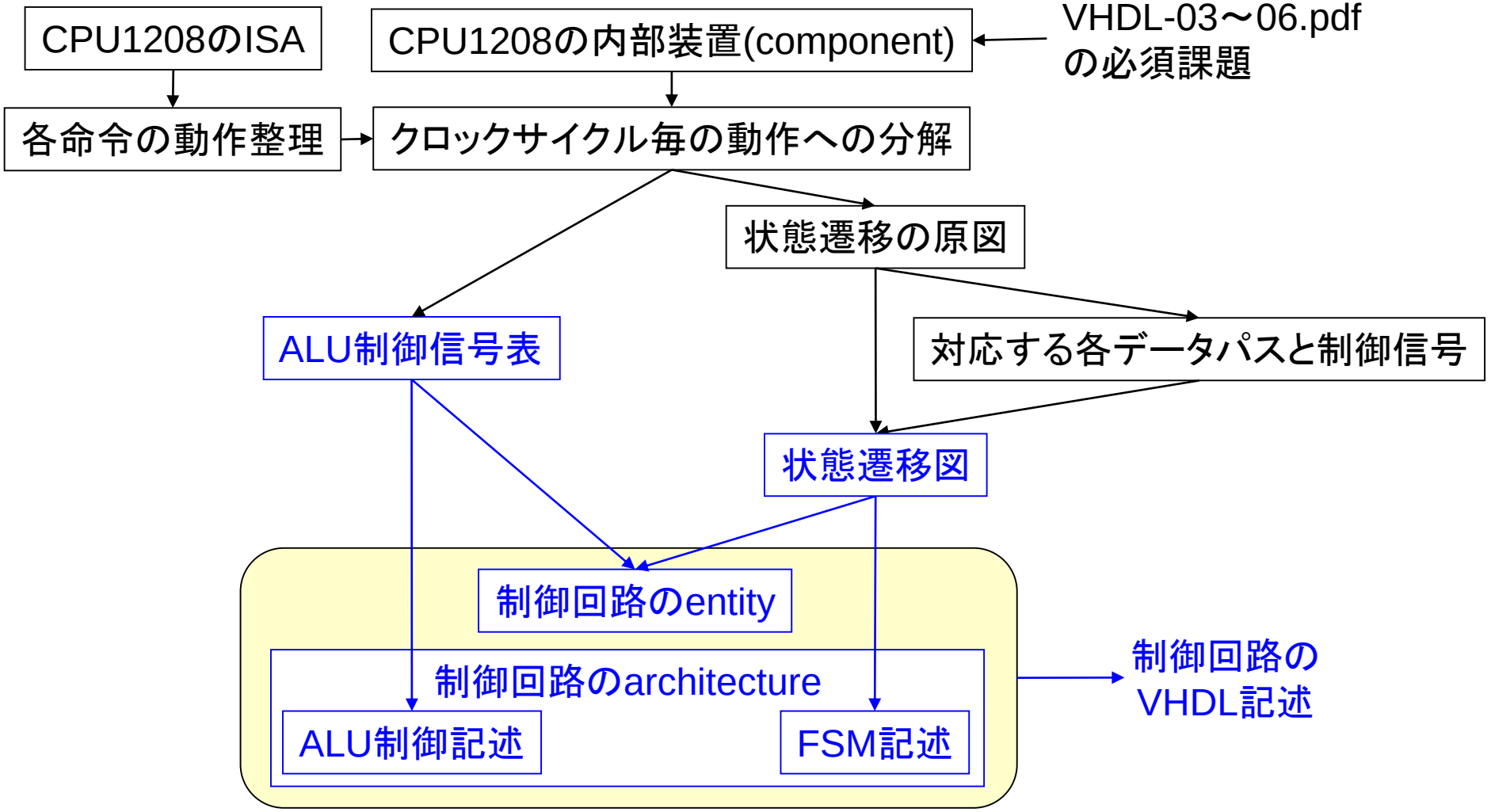
状態遷移の原図＋制御信号＋描図法→状態遷移図



Think2 ＝制御信号 ⇒ Mealy型FSMの描図法



手順


CPU1208のISA
CPU1208の内部装置(component)
VHDL-03～06.pdf
の必須課題
各命令の動作整理
クロックサイクル毎の動作への分解
状態遷移の原図
ALU制御信号表
対応する各データパスと制御信号
状態遷移図
制御回路のentity
制御回路のarchitecture
ALU制御記述
FSM記述
制御回路の
VHDL記述

ALU制御信号表＋状態遷移図→制御回路のVHDL記述

入力 IRH	出力 AF2～AF0	備考(△)	備考(基本命令命)
4	Think1	+	ADD
8			
1		and	BIT
5			AND
9			
6		or	OR
A			
3		eor	EQU
7			EOR
B			
0		thr	LD
C			JZE
D			JMP
E			LD
F			ST

制御回路のVHDL記述

制御回路のarchitecture

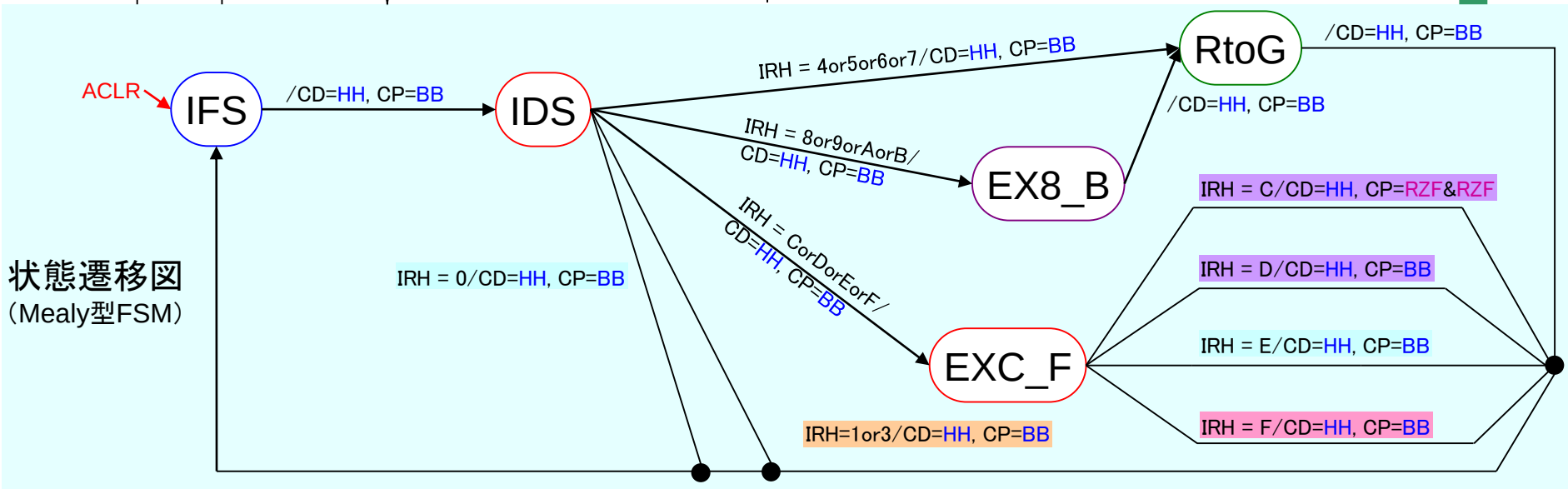
条件付信号代入文
または
process文・case文・if文
etc

Mealy型FSM
の記述

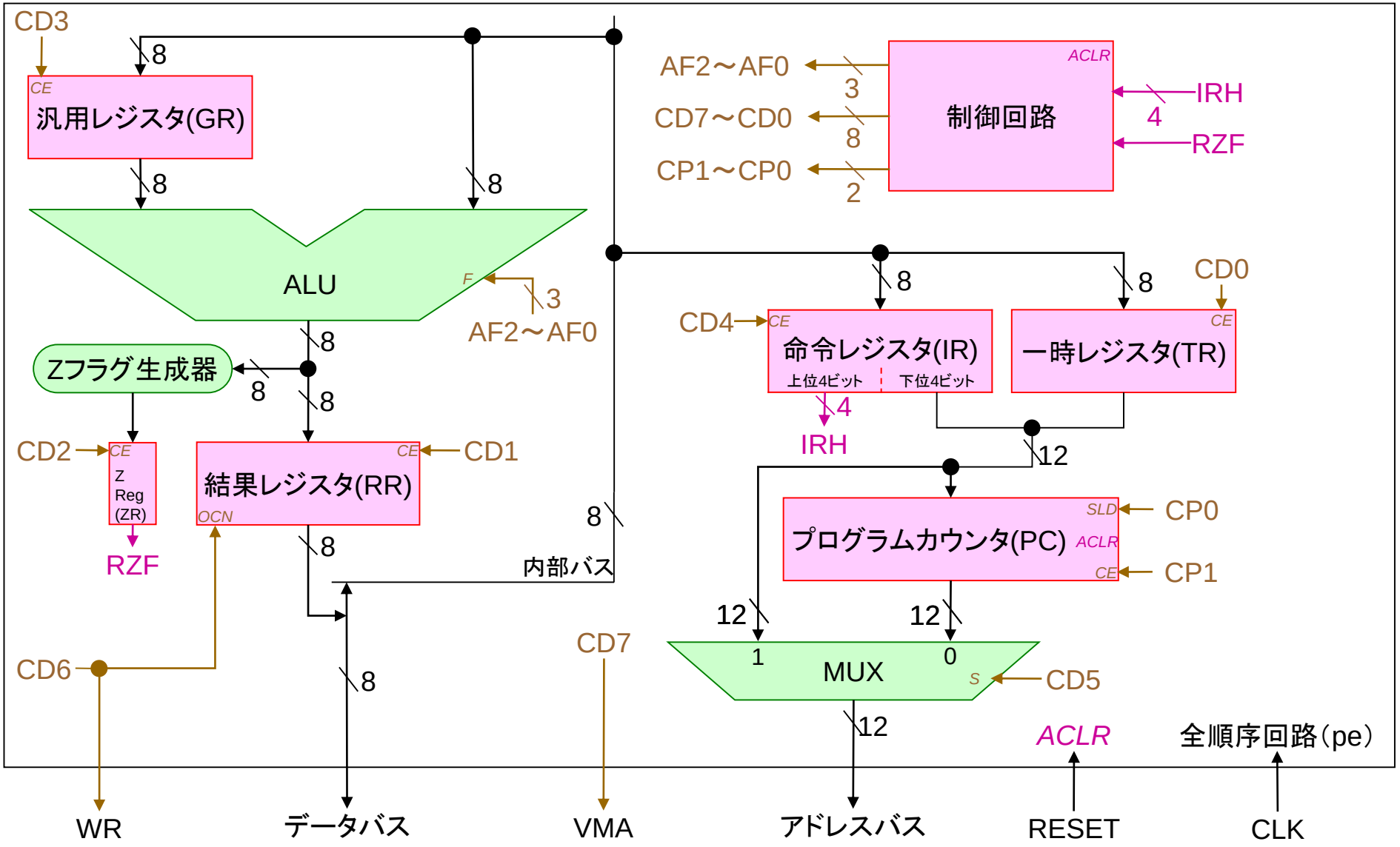
制御回路のentity

入力：IRH，RZF，CLK，ACLR　出力：AF，CD，CP

状態遷移図
(Mealy型FSM)



CPU1208　外部インタフェース　内部構成



組合せ回路　順序回路

VMA	WR	ASS	ICE	GCE	ZCE	RCE	TCE	PCCE	PCLD
CD								CP	
7	6	5	4	3	2	1	0	1	0

■必須課題■

■プロジェクト「myFSM_cpu1208ctrl」
先に説明した手順に沿って、マイクロプロセッサCPU1208の制御回路を設計せよ。

これまでの例の整理

Top-Level Entity名	スライド	論理回路種別	Process文	代入文	演算子	内部信号	generic宣言		モジュール化デザイン
mynand2	VHDL-01	組み合わせ	ー	信号代入	論理	ー	ー		ー
myor3	VHDL-02	組み合わせ	ー	信号代入	論理	○	ー		ー
mynand2xM	VHDL-02	組み合わせ	ー	信号代入	論理	ー	○	値渡しなし	ー
mynand2_cmp_inext	VHDL-02	組み合わせ	ー	信号代入 （下位Level ）	論理	ー	ー		○（その1）
mynand2xM_cmp_or3	VHDL-02	組み合わせ	ー	信号代入 （Top/下位Level ）	論理	○	○	値渡しあり ・Top-Level (signal宣言文) ・下位Level (generic map)	○（その2）
mynand2_cmp_fg	VHDL-02	組み合わせ	ー	信号代入 （下位Level ）	論理	ー	○	値渡しあり ・Top-Level (for-generate文)	○（その3）
myadderM	VHDL-03	組み合わせ	ー	信号代入	算術(+)	ー	ー		ー
myshift1	VHDL-03	組み合わせ	ー	信号代入	連接(&)	ー	ー		ー
mydec1to2	VHDL-03	組み合わせ	ー	条件付信号代入	関係	ー	ー		ー
myenc4to2	VHDL-04	組み合わせ	○(if)	信号代入	関係	ー	ー		ー
mydec1to2	VHDL-04	組み合わせ	○(case)	信号代入	なし	ー	ー		ー
myandM	VHDL-04	組み合わせ	○(for-loop)	変数代入	論理	ー	○	値渡しあり (for-loop文)	ー
myreg_***	VHDL-05	順序(記憶要素)	○(if)	信号代入 条件付信号代入	論理	○	○	値渡しあり (signal宣言文、集合体)	ー
mycnt_***	VHDL-06	順序	○(if)	信号代入 条件付信号代入	算術(+)	○	○	値渡しあり (signal宣言文、集合体)	ー
mysr_***	VHDL-07	順序	○(if)	信号代入 条件付信号代入	連接(&)	○	○	値渡しあり (signal宣言文、集合体)	ー
myFSMmealy myFSMmoore	VHDL-08	順序	○(if, case)	信号代入 条件付信号代入	なし	○	ー		ー

さまざまなバリエーションで学習済 ➡ 適宜応用する

ただし、「type宣言文：新しいデータ型の宣言」追加