機能設計仕様書

19 班 佐竹誠 (1029283837)

2018/5/10 提出

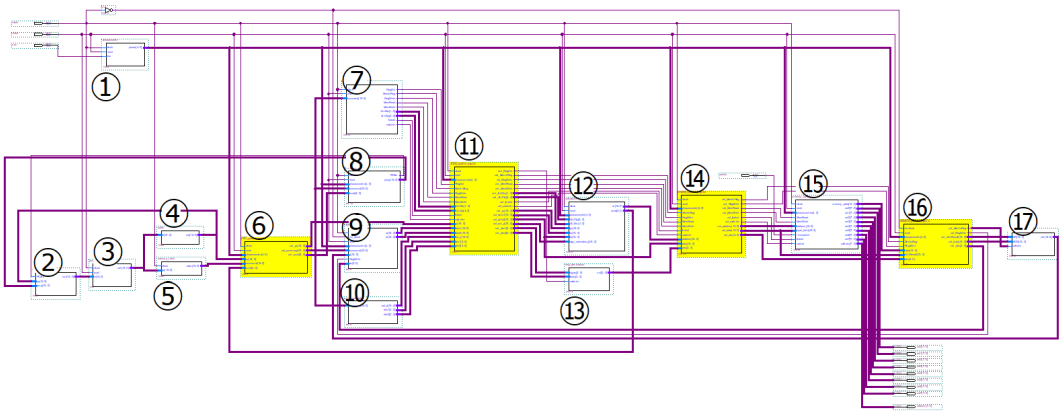

図 1　ブロック図

1　コンポーネントへの分割

プロセッサを制作するにあたって、全体を図 1 のようにコンポーネントに分割した。

2　設計を担当したコンポーネントの外部仕様

設計を担当したコンポーネントの外部仕様について以下で述べる。

特徴の項目ではコンポーネントの役割などに触れ、方針設計の項目では他のコンポーネントとの関わり、すなわちこのコンポーネントの計算機全体における位置づけや役割及び外部とのインターフェース、外部から見た動作についても述べる。特に動作の項目ではフェーズ・フロー・チャートとデータ・フロー・チャートのそれぞれでの位置づけについて言及する。

1. IFID_pipeline_register
   （a）特徴

   これはフェーズ 1 からフェーズ 2 に移行する際に必要なデータを保持しておくパイプラインレジスタである。具体的にはプログラムカウンタ、メモリーから取り出してきた命令、そしてひとつ前の命令で発生した条件フラグを保持している。

   （b）方針設計

   クロック、リセットとフェーズカウンターで管理する。フェーズ 2 にクロックが立ち上がったときレジスタの値を更新し、出力する。

   i. 構造

   図 1 の 6 番がこのコンポーネントにあたる。入力は clock,reset,phasecounter[4:0],pc[15:0],command[15:0],cond
   出力は out_pc[15:0],out_command[15:0],out_cond[3:0] である。

   ii. 動作

   A. フェーズ・フロー・チャートにおける位置づけ

   フェーズ 1 で得られたデータを保持し、フェーズ 2 に渡している。

B. データ・フロー・チャートにおける位置づけ

フェーズ 1 で adder からプログラムカウンタを、memory_fetch から命令を、calculation から条件フラグを受け取る。フェーズ 2 で branch_ALU に命令と条件フラグを、fetch_or_write_register に命令を、command_data に命令を、IDEX_pipeline_register にプログラムカウンタを渡している。

2. IDEX_pipeline_register

（a）特徴

これはフェーズ 2 からフェーズ 3 に移行する際に必要なデータを保持しておくパイプラインレジスタである。具体的にはコントローラーからの制御信号、プログラムカウンタ、レジスタから読み取ったデータをふたつ、符号拡張した即値あるいは変位、デスティネーションレジスタ番号をふたつを保持している。

（b）方針設計

クロック、リセットとフェーズカウンターで管理する。フェーズ 3 にクロックが立ち上がったときレジスタの値を更新し、出力する。

i. 構造

図 1 の 11 番がこのコンポーネントにあたる。入力は clock,reset,RegDst, MemtoReg, RegWrite, MemRead, MemWrite, ALUSrc[1:0], ALUOp[3:0], ledout, switchin, pc[15:0], data1[15:0],data2[15:0], ext_d[15:0], des1[2:0], des2[2:0]、出力は out_RegDst, out_MemtoReg, out_RegWrite, out_MemRead, out_MemWrite, out_ALUSrc[1:0], out_ALUOp[3:0], out_ledout, out_switchin, out_pc[15:0], out_data1[15:0],out_data2[15:0], out_ext_d[15:0], out_des1[2:0], out_des2[2:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

フェーズ 2 で得られたデータを保持し、フェーズ 3 に渡している。

B. データ・フロー・チャートにおける位置づけ

フェーズ 2 で controller から各種制御信号を、fetch_or_write_register からレジスタから読み取ったデータをふたつ、command_data から符号拡張した即値あるいは変位とデスティネーションレジスタ番号をふたつ、そして IFID_pipeline_register からプログラムカウンタを受け取る。フェーズ 3 で ALU に関する制御信号とレジスタから読み取ったデータ、符号拡張した即値あるいは変位を calculation に、ふたつデスティネーションレジスタ番号を mux_destination に、その他の制御信号を EXMEM_pipeline_register に渡している。

3. EXMEM_pipeline_register

（a）特徴

これはフェーズ 3 からフェーズ 4 に移行する際に必要なデータを保持しておくパイプラインレジスタである。具体的には ID/EX 間のパイプラインレジスタから受け取った制御信号、演算機の計算結果、レジスタから読み取ったデータ、デスティネーションレジスタ番号を保持している。

（b）方針設計

クロック、リセットとフェーズカウンターで管理する。フェーズ 4 にクロックが立ち上がったときレジスタの値を更新し、出力する。

i. 構造

図 1 の 14 番がこのコンポーネントにあたる。入力は clock,reset,MemtoReg, RegWrite, MemRead, MemWrite, ledout, switchin, address[15:0], data[15:0], des[2:0]、出力は out_MemtoReg, out_RegWrite, out_MemRead, out_MemWrite, out_ledout, out_switchin, out_address[15:0], out_data[15:0], out_des[2:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

フェーズ 3 で得られたデータを保持し、フェーズ 4 に渡している。

B. データ・フロー・チャートにおける位置づけ

フェーズ 3 で IDEX_pipeline_register から制御信号とレジスタから読み取ったデータのひとつを、calculation から演算結果を、mux_destination からデスティネーションレジスタ番号を受け取る。フェーズ 4 で memory_access にメモリーあるいはインプットアウトプットに関する制御信号、演算結果、レジスタから読み取ったデータを、MEMWB_pipeline_register にその他の制御信号と演算結果、デスティネーションレジスタ番号を渡している。

4. MEMWB_pipeline_register

（a）特徴

これはフェーズ 4 からフェーズ 5 に移行する際に必要なデータを保持しておくパイプラインレジスタである。具体的には EX/MEM 間のパイプラインレジスタから受け取った制御信号、メモリーから読みだしてきたデータ、演算結果、デスティネーションレジスタ番号を保持している。

（b）方針設計

クロック、リセットとフェーズカウンターで管理する。フェーズ 5 にクロックが立ち上がったときレジスタの値を更新し、出力する。

i. 構造

図 1 の 16 番がこのコンポーネントにあたる。入力は clock,reset,MemtoReg, RegWrite, ALUResult[15:0], data[15:0], des[2:0]、出力は out_MemtoReg, out_RegWrite, out_ALUResult[15:0], out_data[15:0], out_des[2:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

フェーズ 4 で得られたデータを保持し、フェーズ 5 に渡している。

B. データ・フロー・チャートにおける位置づけ

フェーズ 4 で EXMEM_pipeline_register から制御信号と演算結果、デスティネーションレジスタ番号を、memory_access から読み取ったデータを受け取り、フェーズ 5 で mux_DR_MDR にレジスタに書き込むデータに関する制御信号とメモリーから読み取ったデータ、演算結果を、レジスタに書き込むかどうかの制御信号とデスティネーションレジスタ番を fetch_or_write_register に渡している。

5. controller

（a）特徴

これは命令ビット列から各種制御信号を生成するコンポーネントである。具体的には命令ビット列を受け取り、デスティネーションレジスタ番号を指定する RegDst、ALU のソースを指定する

ALUSrc、レジスタの書き込みデータを指定する MemtoReg、レジスタにデータを書き込むかどうかを指定する RegWrite、データメモリを読みだすかどうかを指定する MemRead、データメモリに書き込むかどうかを指定する MemWrite、ALU の演算を指定する ALUOp、出力するかどうかを指定する ledout、入力を受け付けるかどうかを指定する switchin を出力する。また、各制御信号の意味は表 1、表 2 の通りである。ただし、ALUop には ALU の操作コードをそのまま割り当てる。

表 1　制御信号 (1 ビット)

制御信号の名称	制御対象	0	1
RegDst	デスティネーションレジスタ番号	command[13:11]	command[10:8]
MemtoReg	レジスタの書き込みデータ	ALU から得られる	データメモリから得られる
RegWrite	レジスタにデータを書き込むかどうか	書き込まない	書き込む
MemRead	データメモリを読みだすかどうか	読み出さない	読み出す
MemWrite	データメモリに書き込むかどうか	書き込まない	書き込む
ledout	出力するかどうか	出力しない	出力する
switchin	入力を受け付けるかどうか	受け付けない	受け付ける

表 2　制御信号 (2 ビット)

制御信号の名称	制御対象	00	01	10	11
ALUSrc	ALU のソース	ar & br	0 & ext_d	pc & ext_d	br & ext_d

（b）方針設計

クロック、リセットとフェーズカウンターで管理する。

i. 構造

図 1 の 7 番がこのコンポーネントにあたる。入力は clock,reset,command[15:0]、出力は RegDst, MemtoReg, RegWrite, MemRead, MemWrite, ALUSrc[1:0], ALUOp[3:0], ledout, switchin である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

制御部であるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャートにおける位置づけ

制御部であるのでデータ・フロー・チャートには登場しない。

6. branch_ALU

（a）特徴

これは分岐命令を処理するコンポーネントである。具体的には命令ビット列と条件フラグを受け取り、ALU で計算し、条件分岐するかどうかを指定する PCSrc と分岐先のアドレスを出力する。

（b）方針設計

クロック、リセットとフェーズカウンターで管理する。命令形式などは SIMPLE/B を基本とし、

op2 が 001,010,101,110 のときは以下のような命令形式と取る。

- $I_{15:14}$ (op1) 操作コード (10)
- $I_{13:11}$ (op2) 操作コード (001:BE,010:BLT,101:BLE,110:BNE)
- $I_{10:8}$ (Rs1) ソース・レジスタ番号 1
- $I_{7:5}$ (Rs2) ソース・レジスタ番号 2
- $I_{4:0}$ (d) 変位

この時の動作は、Rs1 と Rs2 の値を用いて条件フラグを作成し op2 に基づいて PCSrc を出力、プログラムカウンタに変位 d を足した値を jamp に出力する。条件分岐命令は op2 が 001,010,101,110 にそれぞれ BE,BLT,BLE,BNE の 4 命令を実装する。

i. 構造

図 1 の 8 番がこのコンポーネントにあたる。入力は clock,reset,phasecounter[4:0],command[15:0],pc[15:0]、出力は PCSrc,jamp[15:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

フェーズ 2 において分岐命令のときのみ動く。

B. データ・フロー・チャートにおける位置づけ

IFID_pipeline_register から命令ビット列と条件フラグを受け取り、mux_pc に PCSrc と分岐先のアドレスを渡す。

7. command_data

（a）特徴

これは命令ビット列を受け取り、ふたつのデスティネーションレジスタ番号と即値あるいは変位を符号拡張したものを出力するコンポーネントである。

（b）方針設計

組み合わせ回路である。

i. 構造

図 1 の 10 番がこのコンポーネントにあたる。入力は command[15:0]、出力は ext_d[15:0],des1[2:0],des2[2:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

組み合わせ回路であるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャートにおける位置づけ

IFID_pipeline_register から命令ビット列を受け取り、IDEX_pipeline_register にふたつのデスティネーションレジスタ番号と即値あるいは変位を符号拡張したものを渡す。

8. phaseconter

（a）特徴

これはクロックによってフェーズを管理するコンポーネントである。フェーズを管理するための 5 ビット列を出力し、フェーズによって動きを変える他のコンポーネントの入力とする。

（b）方針設計

クロックとリセット、停止/再開ボタンによって管理する。

i. 構造

図 1 の 1 番がこのコンポーネントにあたる。入力は clock,reset,exc、出力は phase[4:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

制御部なのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャートにおける位置づけ

制御部なのでデータ・フロー・チャートには登場しない。

9. memory_access

（a）特徴

これは制御信号によって主記憶にアクセスし、データを読み取ったり書き込んだりするコンポーネントである。また入出力もこのコンポーネントで行う。

（b）方針設計

クロックとリセット、フェーズカウンターによって管理する。MemRead がハイのときメモリの address からデータを読み出し memory_data に値を格納し、MemWrite がハイのときメモリの address に input_data を書き込む。switchin がハイのとき fromswitch から入力を受け付け memory_data に値を格納し、ledout がハイのとき addresss と input_data を出力する。

i. 構造

図 1 の 16 番がこのコンポーネントにあたる。入力は clock,reset,phasecounter[3:0],address[15:0],input_data[15
MemWrite, ledout, switchin、出力は memory_data[15:0],8 つの out[7:0],selector[7:0] である。

ii. 動作

A. フェーズ・フロー・チャートにおける位置づけ

フェーズ 4 においてロードストア命令または入出力命令のときに動く。ロード命令では address のアドレスのデータを memory_data に格納し、ストア命令では address のアドレスに input_data の値を格納する。入力命令ではスイッチからの入力を memory_data に格納し、出力命令では address と input_data の値を 7SEG LED などに出力する。

B. データ・フロー・チャートにおける位置づけ

IDMEM_pipeline_register からメモリに関する制御信号と入出力に関する制御信号、address と input_data を、INPUT から入力を受け取り、IDMEM_pipeline_register に memory_data を、OUTPUT に 8 つの 8 ビット列とセレクターを出力する。

3　実装を担当したコンポーネントの内部仕様

実装を担当したコンポーネントのそれぞれについて以下で述べる。また各コンポーネントの動作を説明するために、図 2 にフェーズ・フロー・チャートを、図 3 から図 7 にフェーズ 1 からフェーズ 5 までのデータ・フロー・チャートを掲載しておく。

1. IFID_pipeline_register

（a）実装上の特徴

クロックとリセット、フェーズカウンターで管理する。

命令	P1	P2	P3	P4	P5
ADD	・programconter→access_memory ・programconter→adder ・adder→pc ・memory_fetch→command	・レジスタRs→ra ・レジスタRd→rb	・ALUで(ra+rb)→dr ・条件コードを設定		・dr→レジスタRd
SUB			・ALUで(ra−rb)→dr ・条件コードを設定		
AND			・ALUで(ra&rb)→dr ・条件コードを設定		
OR			・ALUで(ra|rb)→dr ・条件コードを設定		
XOR			・ALUで(ra^rb)→dr ・条件コードを設定		
ADDi		・レジスタRs→ra ・d→ext_d(符号拡張)	・ALUで(ra+ext_d)→dr ・条件コードを設定		
SUBi			・ALUで(ra−ext_d)→dr ・条件コードを設定		
SLL		・レジスタRd→ra ・d→ext_d(符号拡張)	・ALUで(ra << d)→dr ・空いたビットに0を挿入 ・条件コードを設定		
SLR			・ALUで(ra << d)→dr ・空いたビットにシフトアウトされたビット列を挿入 ・条件コードを設定		
SRL			・ALUで(ra >> d)→dr ・空いたビットに0を挿入 ・条件コードを設定		
SRA			・ALUで(ra >> d)→dr ・空いたビットにシフトアウトされたビット列を挿入 ・条件コードを設定		
IN				・スイッチからの入力→dr	
OUT		・レジスタRs→ra ・レジスタRd→rb		・ra,rbを7SEG LEDに表示	
HALT		HALT			
LD		・レジスタRa→ra ・レジスタRb→rb ・Ra→des1 ・d→ext_d(符号拡張)	・ALUで(rb+ext_d)→dr	・アドレスがdrの値のメモリーデータ→MDR	
ST				・raの値をアドレスがdrの値のメモリーに書き込む	
LI		・d→ext_d(符号拡張)			・enx_d→レジスタRb
B		・pc→branch_ALU ・ext_d→branch_ALU ・branch_ALUで(pc+ext_d)→programcounter			
BE		・d→branch_ALU ・branch_ALUで条件分岐→programcounter			
BLT					
BLE					
BNE					
BE'		・レジスタRa→branch_ALU ・レジスタRb→branch_ALU ・d→branch_ALU ・branch_ALUで条件分岐→programcounter			
BLT'					
BLE'					
BNE'					

図２　フェーズ・フロー・チャート

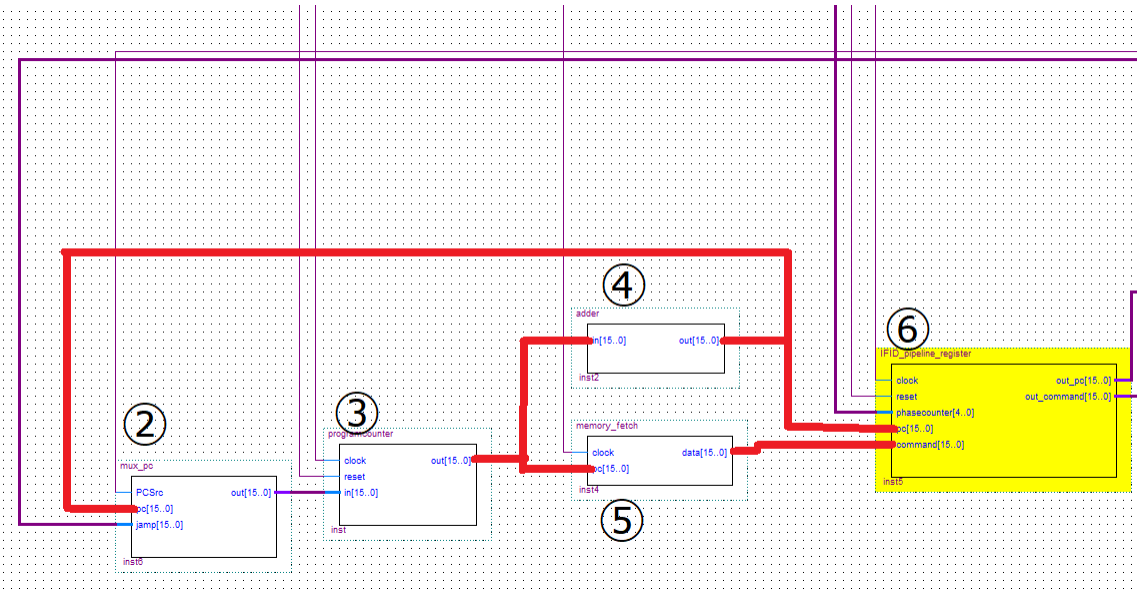

図 3　フェーズ 1 のデータ・フロー・チャート

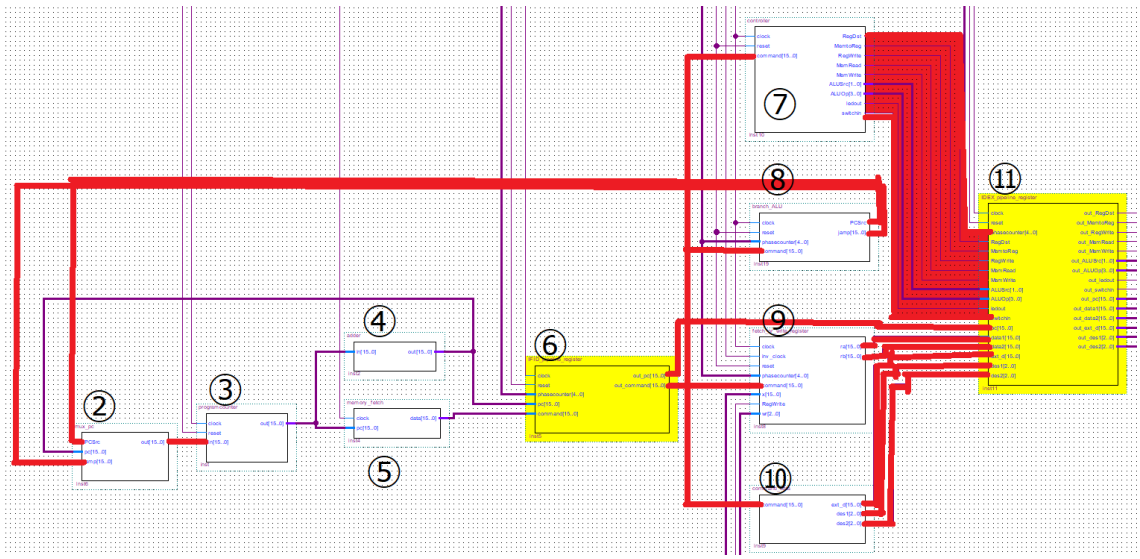

図 4　フェーズ 2 のデータ・フロー・チャート

（b）コンポーネントの内部構造と動作

i. 構造

コンポーネントのアルゴリズムの疑似コードを Listing1 に示す。クロックの立ち上がりあるいはリセットの立ち下がりが発生するたびにこのアルゴリズムを動かす。入力は clock,reset,phasecounter[4:0],pc[15:0],command[15:0],cond[3:0]、出力は out_pc[15:0],out_command[15:0],out_cond[3:0] である。

Listing 1　IFID_pipeline_register

```
if(reset == 1'b0)
```

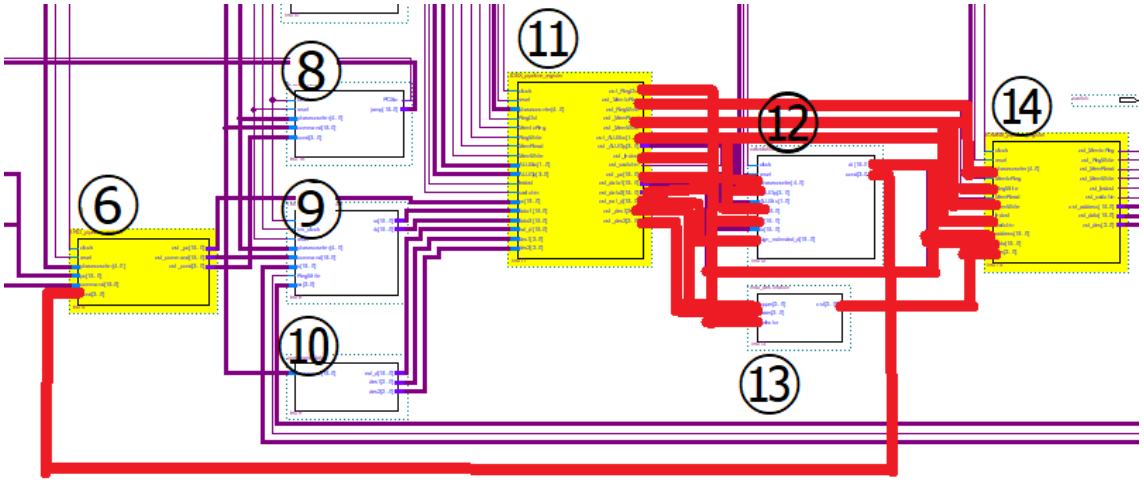

図５　フェーズ３のデータ・フロー・チャート

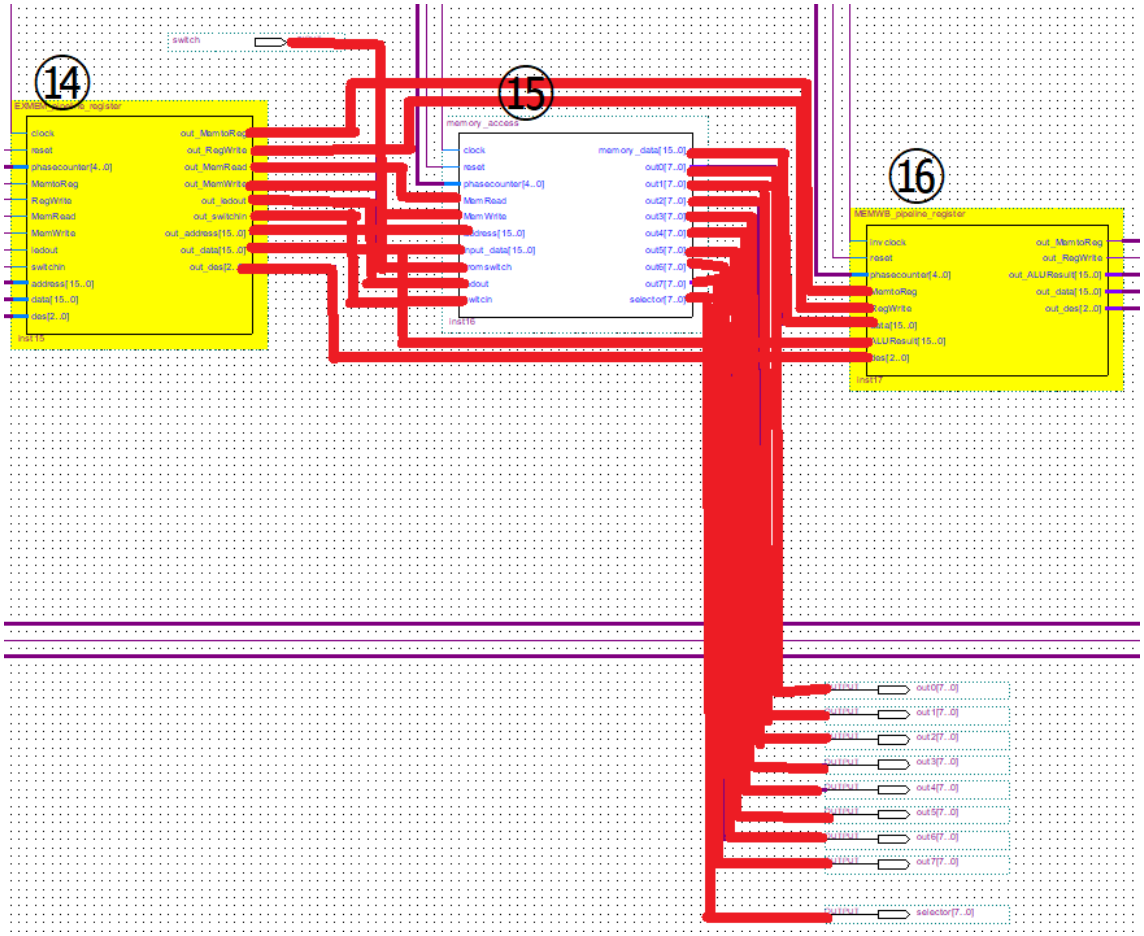

図６　フェーズ４のデータ・フロー・チャート

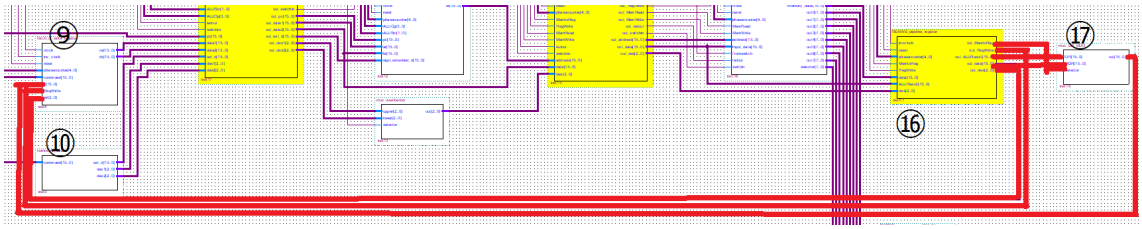

図 7　フェーズ 5 のデータ・フロー・チャート

```
        begin
                out_pc <= 16'b0;
                out_command <= 16'b0;
                out_cond <= 4'b0;
        end
        else if(phasecounter == 5'b00010)
        begin
                out_pc <= pc;
                out_command <= command;
                out_cond <= cond;
        end
```

ii. 動作

A. フェーズ・フロー・チャート

データをフェーズ間で受け渡しするだけのパイプラインレジスタであるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャート

図 3、図 4、図 5 に 6 番として登場している。

C. タイミング・チャート

データをフェーズ間で受け渡しするだけのパイプラインレジスタであるので動作の説明にタイミング・チャートは不必要である。

（c）論理設計にあたって特に留意すべき点

特になし。

2. IDEX_pipeline_register

（a）実装上の特徴

クロックとリセット、フェーズカウンターで管理する。

（b）コンポーネントの内部構造と動作

i. 構造

コンポーネントのアルゴリズムの疑似コードを Listing2 に示す。クロックの立ち上がりあるいはリセットの立ち下がりが発生するたびにこのアルゴリズムを動かす。入力は clock,reset,RegDst, MemtoReg, RegWrite, MemRead, MemWrite, ALUSrc[1:0], ALUOp[3:0], ledout, switchin, pc[15:0], data1[15:0],data2[15:0], ext_d[15:0], des1[2:0], des2[2:0]、出力は out_RegDst, out_MemtoReg, out_RegWrite, out_MemRead, out_MemWrite, out_ALUSrc[1:0], out_ALUOp[3:0], out_ledout, out_switchin, out_pc[15:0], out_data1[15:0],out_data2[15:0], out_ext_d[15:0], out_des1[2:0], out_des2[2:0] である。

Listing 2　IDEX_pipeline_register

```
if(reset == 1'b0)
        begin
                {out_RegDst, out_MemtoReg, out_RegWrite, out_MemRead,
                    out_MemWrite} <= 5'b00000;
                out_ALUSrc <= 2'b0;
                out_ALUOp <= 4'b0;
                {out_ledout, out_switchin} <= 2'b00;
                out_pc <= 16'b0;
                out_data1 <= 16'b0;
                out_data2 <= 16'b0;
                out_ext_d <= 16'b0;
                out_des1 <= 3'b0;
                out_des2 <= 3'b0;
        end
else if(phasecounter == 5'b00010)
        begin
                {out_RegDst, out_MemtoReg, out_RegWrite, out_MemRead,
                    out_MemWrite} <= {RegDst, MemtoReg, RegWrite, MemRead,
                    MemWrite};
                {out_ledout, out_switchin} <= {ledout, switchin};
                out_ALUSrc <= ALUSrc;
                out_ALUOp <= ALUOp;
                out_pc <= pc;
                out_data1 <= data1;
                out_data2 <= data2;
                out_ext_d <= ext_d;
                out_des1 <= des1;
                out_des2 <= des2;
        end
```

ii. 動作

A. フェーズ・フロー・チャート
データをフェーズ間で受け渡しするだけのパイプラインレジスタであるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャート
図 4、図 5 に 11 番として登場している。

C. タイミング・チャート
データをフェーズ間で受け渡しするだけのパイプラインレジスタであるので動作の説明にタイミング・チャートは不必要である。

（c）論理設計にあたって特に留意すべき点
特になし。

3. EXMEM_pipeline_register

（a）実装上の特徴
クロックとリセット、フェーズカウンターで管理する。

（b）コンポーネントの内部構造と動作

i. 構造
コンポーネントのアルゴリズムの疑似コードを Listing3 に示す。クロックの立ち上が

りあるいはリセットの立ち下がりが発生するたびにこのアルゴリズムを動かす。入力は clock,reset,MemtoReg, RegWrite, MemRead, MemWrite, ledout, switchin, address[15:0], data[15:0], des[2:0]、出力は out_MemtoReg, out_RegWrite, out_MemRead, out_MemWrite, out_ledout, out_switchin, out_address[15:0], out_data[15:0], out_des[2:0] である。

Listing 3　EXMEM_pipeline_register

```
if(reset == 1'b0)
                begin
                        {out_MemtoReg, out_RegWrite, out_MemRead,
                            out_MemWrite} <= 4'b0000;
                        {out_ledout, out_switchin} <= 2'b00;
                        out_address <= 16'b0;
                        out_data <= 16'b0;
                        out_des <= 3'b0;
                end
        else if(phasecounter == 5'b01000)
                begin
                        {out_MemtoReg, out_RegWrite, out_MemRead,
                            out_MemWrite} <= {MemtoReg, RegWrite, MemRead,
                            MemWrite};
                        {out_ledout, out_switchin} <= {ledout, switchin};
                        out_address <= address;
                        out_data <= data;
                        out_des <= des;
                end
```

ii. 動作

A. フェーズ・フロー・チャート
データをフェーズ間で受け渡しするだけのパイプラインレジスタであるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャート
図 5、図 6 に 14 番として登場している。

C. タイミング・チャート
データをフェーズ間で受け渡しするだけのパイプラインレジスタであるので動作の説明にタイミング・チャートは不必要である。

（c）論理設計にあたって特に留意すべき点
特になし。

4. MEMWB_pipeline_register

（a）実装上の特徴
クロックとリセット、フェーズカウンターで管理する。

（b）コンポーネントの内部構造と動作

i. 構造
コンポーネントのアルゴリズムの疑似コードを Listing4 に示す。クロックの立ち上がりあるいはリセットの立ち下がりが発生するたびにこのアルゴリズムを動かす。入力は clock,reset,MemtoReg, RegWrite, ALUResult[15:0], data[15:0], des[2:0]、出力は out_MemtoReg, out_RegWrite, out_ALUResult[15:0], out_data[15:0], out_des[2:0] である。

Listing 4　EXMEM_pipeline_register

```
if(reset == 1'b0)
                begin
                        {out_MemtoReg, out_RegWrite} <= 2'b00;
                        out_ALUResult <= 16'b0;
                        out_data <= 16'b0;
                        out_des <= 3'b0;
                end
        else if(phasecounter == 5'b10000)
                begin
                        {out_MemtoReg, out_RegWrite} <= {MemtoReg, RegWrite
                            };
                        out_ALUResult <= ALUResult;
                        out_data <= data;
                        out_des <= des;
                end
```

ii. 動作

A. フェーズ・フロー・チャート
データをフェーズ間で受け渡しするだけのパイプラインレジスタであるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャート
図 6、図 7 に 16 番として登場している。

C. タイミング・チャート
データをフェーズ間で受け渡しするだけのパイプラインレジスタであるので動作の説明にタイミング・チャートは不必要である。

（c）論理設計にあたって特に留意すべき点
特になし。

5. controller

（a）実装上の特徴
クロック、リセットを用いて管理する。

（b）コンポーネントの内部構造と動作

i. 構造
コンポーネントのアルゴリズムの疑似コードを Listing5 に示す。クロックの立ち上がりあるいはリセットの立ち下がりが発生するたびにこのアルゴリズムを動かす。入力は clock,reset,command[15:0]、出力は RegDst, MemtoReg, RegWrite, MemRead, MemWrite, ALUSrc[1:0], ALUOp[3:0], ledout, switchin である。

Listing 5　controller

```
if (reset == 1'b0) begin
                {RegDst, MemtoReg, RegWrite, MemRead, MemWrite} <= 5'b00000;
                switchin <= 1'b0;
                ledout <= 1'b0;
                ALUOp <= 4'b0;
        end else begin
                case (op1)
```

```
2'b00ロード://
begin
        {RegDst, MemtoReg, RegWrite, MemRead,
            MemWrite} <= 5'b01110;
        switchin <= 1'b0;
        ledout <= 1'b0;
        ALUOp <= 4'b0000;
        ALUSrc <= 2'b11;
end
2'b01ストア://
begin
        {RegDst, MemtoReg, RegWrite, MemRead,
            MemWrite} <= 5'bxx001;
        switchin <= 1'b0;
        ledout <= 1'b0;
        ALUOp <= 4'b0000;
        ALUSrc <= 2'b11;
end
2'b10即値ロード、分岐://
begin
        switchin <= 1'b0;
        ledout <= 1'b0;
        ALUOp <= 4'b0000;
        case (op2)
                3'b000: // load immediate
                begin
                        {RegDst, MemtoReg, RegWrite,
                             MemRead, MemWrite} <=
                            5'b10100;
                        ALUSrc <= 2'b01;
                end
                3'b111: // branch
                begin
                        {RegDst, MemtoReg, RegWrite,
                             MemRead, MemWrite} <=
                            5'bxx000;
                        ALUSrc <= 2'b10;
                end
                default:
                begin
                        {RegDst, MemtoReg, RegWrite,
                             MemRead, MemWrite} <=
                            5'b00000;
                        ALUSrc <= 2'b10;
                end
        endcase
end
2'b11演算://
begin
ALUOp <= op3;
ALUSrc <= 2'b00;
        case (op3)
                4'b1100: // input
```

```
                                        begin
                                                {RegDst, MemtoReg, RegWrite,
                                                     MemRead, MemWrite} <=
                                                    5'b00000;
                                                switchin <= 1'b1;
                                                ledout <= 1'b0;
                                        end
                                        4'b1101: // output
                                        begin
                                                {RegDst, MemtoReg, RegWrite,
                                                     MemRead, MemWrite} <=
                                                    5'b00000;
                                                switchin <= 1'b0;
                                                ledout <= 1'b1;
                                        end
                                        default: // culculation
                                        begin
                                                {RegDst, MemtoReg, RegWrite,
                                                     MemRead, MemWrite} <=
                                                    5'b10100;
                                                switchin <= 1'b0;
                                                ledout <= 1'b0;
                                        end
                                endcase
                        end
                        default:
                        begin
                                {RegDst, MemtoReg, RegWrite, MemRead,
                                    MemWrite} <= 5'b00000;
                                switchin <= 1'b0;
                                ledout <= 1'b0;
                                ALUOp <= 4'b0;
                                ALUSrc <= 2'b00;
                        end
                endcase
        end
```

ii. 動作

A. フェーズ・フロー・チャート
制御部のコンポーネントであるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャート
図 4 に 7 番として登場している。

C. タイミング・チャート
フェーズ 2 のクロック立ち上がり時にすべての制御信号を出力するのでタイミングチャートは不必要である。

(c) 論理設計にあたって特に留意すべき点
制御信号を正しく記述すること。

6. command_data

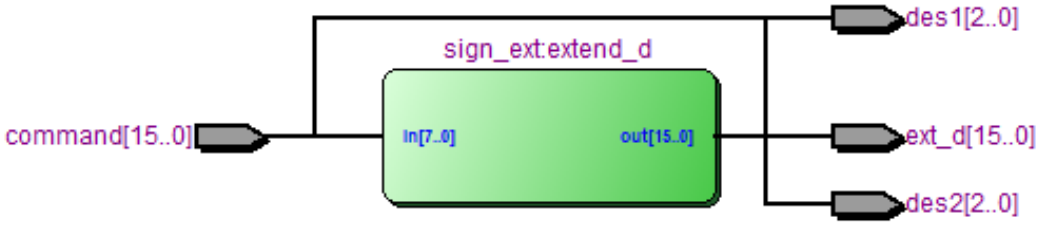

図 8　command_data のブロック図

（a）実装上の特徴

組み合わせ回路である。

（b）コンポーネントの内部構造と動作

i. 構造

入力は command[15:0]、出力は ext_d[15:0],des1[2:0],des2[2:0] である。図 8 にブロック図を示す。

ii. 動作

A. フェーズ・フロー・チャート

図 2 のフェーズ・フロー・チャートにおいて青色で色付けした箇所がこのコンポーネントの担当する部分である。

B. データ・フロー・チャート

図 4 に 10 番として登場している。

C. タイミング・チャート

フェーズ 2 のクロック立ち上がり時にすべての信号を出力するのでタイミングチャートは不必要である。

（c）論理設計にあたって特に留意すべき点

特になし。

7. phaseconter

（a）実装上の特徴

クロックとリセット、停止/再開ボタンで管理する。5 ビットのレジスタの値を操作することでフェーズを操作する。動作状態と停止状態を 1 ビットレジスタ onoff で管理し、停止状態のときはフェーズ 5 から次の命令 (フェーズ 1)1 に移行しないようにする。また 2 ビットシフトレジスタ shft を用いてチャタリング対策を行った。

（b）コンポーネントの内部構造と動作

i. 構造

Listing6 に疑似コードを示す。入力は clock,reset,exc、出力は phase[4:0] である。

Listing 6　phasecounter

```
if(reset == 1'b0) begin
                                        phase <= 5'b00000;
                                        shift <= 2'b11;
                                        onoff <= 1'b0;
                end else begin

                        shift[1] <= shift[0];
                        shift[0] <= exc;


                        if(shift == 2'b10) begin
                                if(onoff == 1'b0) onoff <= 1'b1;
                                else onoff <= 1'b0;
                        end



                        if (onoff == 1'b1) begin
                                case(phase)
                                        5'b00001:phase <= 5'b00010;
                                        5'b00010:phase <= 5'b00100;
                                        5'b00100:phase <= 5'b01000;
                                        5'b01000:phase <= 5'b10000;
                                        default:phase <= 5'b00000;
                                endcase
                        end else begin
                                case(phase)
                                        5'b00001:phase <= 5'b00010;
                                        5'b00010:phase <= 5'b00100;
                                        5'b00100:phase <= 5'b01000;
                                        5'b01000:phase <= 5'b10000;
                                        5'b10000:phase <= 5'b00001;
                                        default:phase <= 5'b00001;
                                endcase
                        end

                end
```

ii. 動作

A. フェーズ・フロー・チャート

制御部のコンポーネントであるのでフェーズ・フロー・チャートには登場しない。

B. データ・フロー・チャート

制御部のコンポーネントであるのでデータ・フロー・チャートには登場しない。

C. タイミング・チャート

図 9 にこのコンポーネントのタイミング・チャートを示す。クロックのたびに phasecounter レジスタのビット列を操作する。

(c) 論理設計にあたって特に留意すべき点

特になし。

8. memory_access

(a) 実装上の特徴

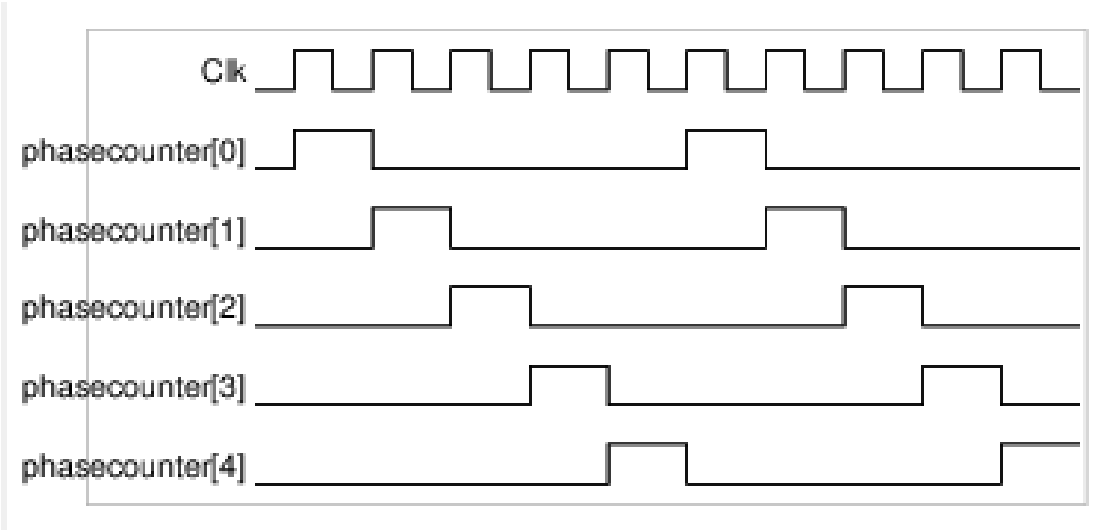

図 9　phasecounter のタイミング・チャート

クロックとリセット、フェーズカウンターで管理する。メモリと入出力に関する制御信号を受け取り、それによりメモリ及び入出力デバイスに接続する。

（b）コンポーネントの内部構造と動作

i. 構造

入力は clock,reset,phasecounter[3:0],address[15:0],input_data[15:0],fromswitch,MemRead, MemWrite, ledout, switchin、出力は memory_data[15:0],8 つの out[7:0],selector[7:0] である。Listing8 に疑似コードを示す。

Listing 7　memory_data

```
if(reset == 1'b0) begin
        end else if(phasecounter[4] == 1'b1) begin
                if(ledout == 1'b1) begin\\
                        input\を出力する_data
                end else if(switchin == 1'b1) begin\\から入力を受け取る
                        fromswitch
                end else begin\\入出力どちらもしない

                end

                if(MemWrite == 1'b1) begin\\メモリのに
                        addressinput\を書き込む_data
                end else if(MemRead == 1'b1) begin\\メモリから読み出し、
                        memory\に値を格納する_data
                end else begin\\メモリ書き込み読み出しどちらもしない

                end
        end
```

ii. 動作

A. フェーズ・フロー・チャート

図 3 のフェーズ・フロー・チャートの中で赤色で色付けした箇所がこのコンポーネントの担当する部分である。

B. データ・フロー・チャート
図 6 に 15 番として登場している。

C. タイミング・チャート
メモリや入出力デバイスに接続するだけのコンポーネントであるのでタイミング・チャートは不必要である。

（c）論理設計にあたって特に留意すべき点
特になし。