計算機科学実験及演習 3
機能設計仕様書

提出期限:2019/5/9
提出日: 2019 年 5 月 9 日

グループ番号: 9
1029293806　大山 偉永

1　コンポーネント分割

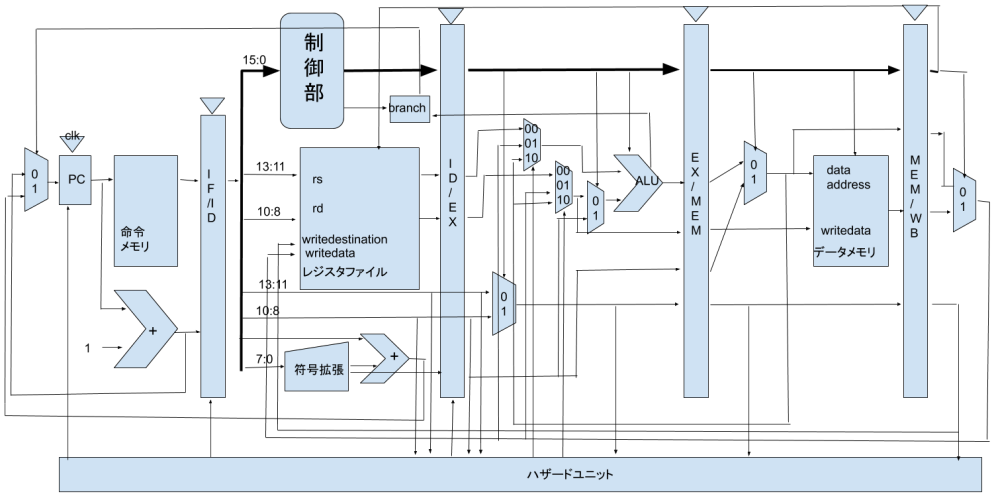

図 1　全体のブロック図

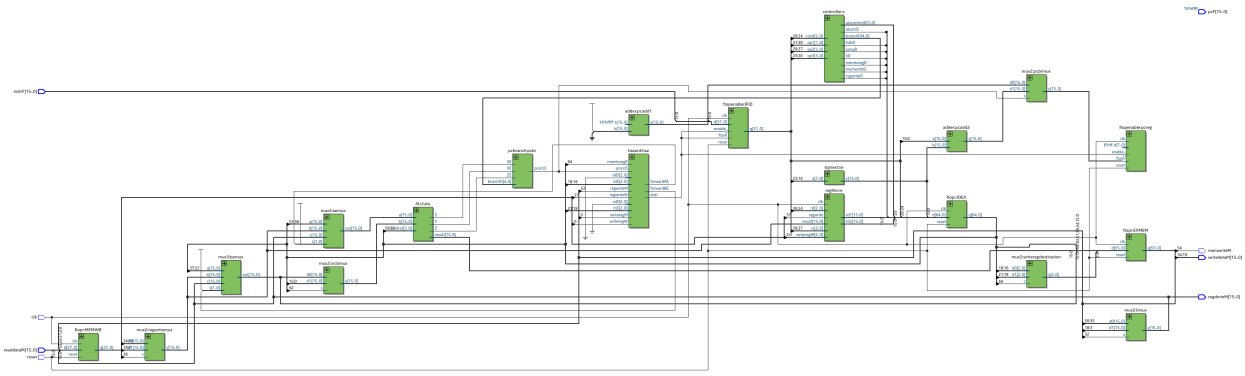

図 2　QuartusRTLviewer のブロック図

まず図 1 のように全体をデータメモリモジュール DM、命令メモリモジュール IM、制御モジュール controller、ALU モジュール ALU、マルチプレクサモジュール mux2、レジスタモジュール regfile、加算モジュール adder、プログラムカウンタ · パイプラインレジスタモジュール flopr、符号拡張モジュール signext、ブランチ制御モジュール branch、ハザード検出ユニット hazard、フォワーディングユニット fowarding に分割している。更に制御モジュール controller については図 3 のようにメインデコーダーモジュール maindec と ALU デコーダーモジュール aludec の 2 つのモジュールから成り立っている。各レジスタ等は bit 幅が違うものの図 4 のように記述すること可変幅のインスタンスを作ることができ各パイプラインレジスタやマルチプレクサを別々のモジュール

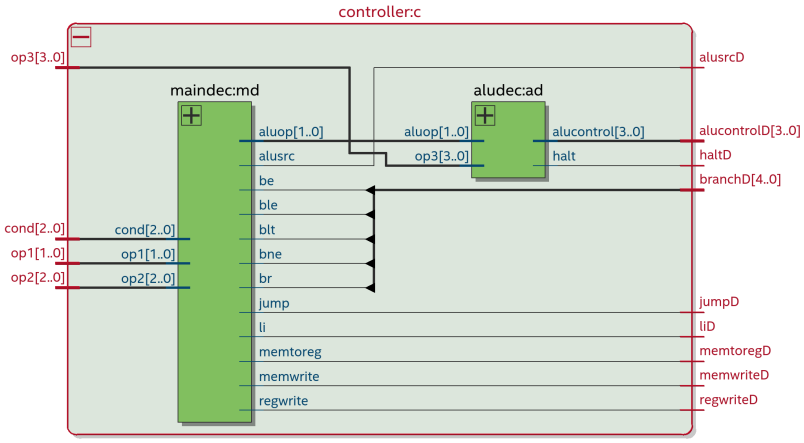

図 3　制御部のブロック図

で作る必要がないように工夫した。分岐予測の実装については現段階では時間があれば行うことにしているので設計等はしていない。

```
module flopr #(parameter WIDTH=8)
        ( input regp,
          input clk, reset,
          input [WIDTH-1:0] d,
          output reg [WIDTH-1:0] q );

always @ (posedge clk, posedge reset) begin
    if (reset) q <= 0;
    else  if (regp) begin
     q <= d;
  end
  end
  endmodule
```

図 4　可変長を指定できるインスタンス作成のコード例

2　設計を担当するコンポーネントの外部仕様

制御部以外の細かいモジュールの設計実装を担当したのでその周りの外部仕様を以下に示す。

- プログラムカウンタ前段のマルチプレクサはブランチ制御部から出力される信号 pcsrcM が 1 のときブランチ命令で指定されたアドレスが出力されている信号 pcbranchM を、pcsrcM が 0 のとき通常の PC に 1 足した値が出力されている信号 pcplus1E を信号 pc に出力し、レジスタがフェッチしてくる次の命令のアドレスを指定する役割を持つ。
- プログラムカウンタ PC は上述のマルチプレクサで指定されたアドレスが pc から入力され次クロックでそれを信号 pcnext に出力する。制御信号 pcwrite が 0 のときはストールさせて値を読み出さない。
- 命令メモリ IM はプログラムカウンタから出力されたアドレスを pcnext から受け取りそのアドレスに書かれているバイナリの命令 instrF[15:0] を出力する。この命令メモリにプログラムの命令列が入っていると考えて良い。
- プログラムカウンタから出力されたアドレス pcnext は 1 加算され信号 pcplus1F に入力され前述のマルチプレクサとパイプラインレジスタ IFID に出力される。
- パイプラインレジスタ IFID は命令メモリ IM からの出力 instrF と 1 加算されたアドレス pcplus1F を受け取り次クロックでそれをそのままそれらを instrD と pcplus1D に出力する。プログラムカウンタと同様、IFIDwrite が 0 のときはストールさせて値を読み出さない。
- レジスタ regfile は instrD[10:8] と instrD[13:11] を受け取りそのアドレスのレジスタに書き込まれている値を srcaD と writedataD に出力する。また制御部から入力される信号 RegwiteD が 1 のとき入力 writeregD が指定してるアドレスのレジスタに resultW から出力されている値を書き込む。
- 符号拡張 signext には instrD[7:0] が入力され符号拡張された値が signimmD に出力される。
- 制御部 controller は instrD[15:14]、instrD[13:11]、instrD[10:8]、instrD[7:4] を受け取り memtoregD、memwriteD、branchD、alucontrolD、alusrcD、regwriteD、liD を出力する。memtoregD は命令がロードのとき 1 に、それ以外のときは 0 に設定される。memwriteD はストア命令のときに 1 に、それ以外のときは 0 に設定される。branchD は 5bit の信号でそれぞれ命令が be、blt、ble、bne、br のときに各桁が 1 に設定される。alucontrolD は 4bit でその各値で ALU で行う演算を指定する。alusrcD はロード命令とストア命令のときに 1 それ以外のときは 0 に設定される。regwriteD は ALU で行う基本演算もしくは即値ロード含むロード命令のときに 1 に、それ以外のときには 0 に設定される。liD は即値ロードのとき 1 に、それ以外のときには 0 に設定される。これらの出力値はマルチプレクサに入力されストール時以外はこの入力がそのままマルチプレクサから出力される。

- パイプラインレジスタ IDEX は srcaD、writedataD、signimmD、pcplus1D、memtoregD、memwriteD、branchD、alucontrolD、alusrcD、regwriteD、liD、instrD[10:8]、instrD[13:11] を受け取り次クロックでこれらをそのまま srcaE、writedataE、signimmE、pcplus1E、memtoregE、memwriteE、branchE、alucontrolE、alusrcE、regwriteE、liE、instrE[10:8]、instrE[13:11] に出力する。
- 次の命令メモリのアドレスを指す pcplus1E は符号拡張された signimmE と加算され pcbranchE に出力される。
- writedataE と signimmE が入力されるマルチプレクサはその制御信号 alusrcE が 1 のとき signimmE を、0 のとき writedataE を wakaran1 に出力する。
- srcaE、resultW、regdminM が入力されるマルチプレクサは制御信号 fowardA が 00 のとき srcaE を、01 のとき resultW、10 のとき regdminM を wakaran2 に出力する。
- wakaran1、resultW、regdminM が入力されるマルチプレクサは制御信号 fowardB が 00 のとき wakaran1 を、01 のとき resultW、10 のとき regdminM を wakaran3 に出力する。
- instrE[10:8] と instrE[13:11] が入力されているマルチプレクサは制御信号 memtoregE が 1 のとき instrE[13:11] を、0 のとき instrE[10:8] を writeregE に出力する。
- ALU は wakaran2 と wakara3 を制御線 alucontrolE の値に応じて各演算をしその結果を aluoutE に出力する。alucontrolE とその演算内容の対応については次頁図 5 に示す。また SIMPLE 仕様書に定めれているとおりに SE、ZE、CE、VE を出力する。

```
alucontrol <= 4'b0000; //add
alucontrol <= 4'b0001; //sub
alucontrol <= 4'b0010; //and
alucontrol <= 4'b0011; //or
alucontrol <= 4'b0100; //xor
alucontrol <= 4'b0101; //cmp
alucontrol <= 4'b0110; //mov
alucontrol <= 4'bxxxx; //reserved
alucontrol <= 4'b1000; //sll shift left logical
alucontrol <= 4'b1001; //slr shift left rotate
alucontrol <= 4'b1010; //srl shift right logical
alucontrol <= 4'b1011; //sra shift right arithmetic
alucontrol <= 4'b1100; //IN rd
alucontrol <= 4'b1101; //OUT rs
alucontrol <= 4'bxxxx; //reserved
 halt<= 1; //halt() stop meirei
alucontrol <= 4'bxxxx; //change
```

図 5　alucontrolE とその演算内容の対応

- aluoutE と signimmE が入力されるマルチプレクサはその制御信号 liE が 1 のとき signimmE を、0 のとき aluoutE を regdminE に出力する。
- パイプラインレジスタ EXMEM は writedataE、memtoregE、memwriteE、branchE、regwriteE、writeregE、pcbranchE、SE、ZE、CE、VE、regdminE を受け取り次クロックでこれらをそのまま writedataM、memtoregD、memwriteM、branchM、regwriteM、writeregM、pcbranchM、SM、ZM、CM、VM、regdminM に出力する。
- データメモリ DM はその制御線 memwriteM が 0 のとき入力 regdminM から入力されるアドレスの値を readdataM に出力する。また制御線 memwriteM が 1 のときは writedataM に入力された値を入力 regdminM から入力されるアドレスに保存する。
- ブランチ制御モジュール branch は制御信号 branchM、SM、ZM、CM、VM を受け取り pcsrcM を出力する。この pcsrcM は、branchM が示すどのブランチ命令を行うかと分岐を行う際の条件判定によりその出力値が 0 か 1 か決まる。例えば branchM が今回の命令が BNE であることを示していてなおかつ Z が 0 の場合 pcsrcM は 1 に設定される。
- パイプラインレジスタ MEMWB は memtoregM、regwriteM、writeregM、readdataM、regdminM を受け取り次クロックでこれらをそのまま memtoregW、regwriteW、writeregW、readdataW、regdminW に出力する。
- readdataW と regdminW が入力されているマルチプレクサはその制御信号 memtoregW が 1 のとき readdataW を、0 のとき regdminW を resultW に出力する。

以上で自分が担当するコンポーネント周りの外部仕様の説明は終わる。フォワーディングユニット、ハザード検出、制御部、全てのコンポーネントを繋ぐ作業はペアの分担である。

3　実装を担当するコンポーネントの内部仕様

自分が担当したコンポーネントの内部仕様を記す。ただし我々の班は各コンポーネントをそれぞれひとつのモジュールに作ってそれらを別の一つのモジュールでまとめただけなので、深い階層構造になっておらず各コンポーネントは Quartus 上で一つの論理回路で記述されるためブロック図を掲載することはできない。

3.1　マルチプレクサ

マルチプレクサはどの使用用途でも同じモジュールからインスタンス化出来るように汎用性の高い設計にした。具体的には入力の bit 幅は可変で指定できるように工夫した。その記述方法は図 4 の一行目と同様で parameter のあとの値を自由に設定できる。こうすることで様々な処理段階で使用できるマルチプレクサをひとつのモジュールで作ることが出来る。実装については三項演算子?:で一行で行える。

3.2 加算器

ALU で行わない加算をする際に用いる加算器。現在のプログラムカウンタに 1 足すときと、現在のプログラムカウンタ +1 に分岐先の相対アドレスを足すときに用いる。こちらも verilog では二項演算子 + を用いて記述するだけなので特筆することはない。

3.3 レジスタ

クロックの立ち上がり時に書き込みを行い、クロックの逆位相を用いることで元のクロックの立ち下がりのときに読み出しを行う。こうすることでパイプライン化の際同じフェーズで読み出しと書き込みが同時に行える。書き込みについては if 文を用いて regwrite の制御線が 1 のときのみ書き込みを行うようにする。これも verilog の記述的には容易で前者はクロック clk の入力ごとに書き込みデータをレジスタにノンブロッキング代入し、後者は逆位相のクロック c̄lk の入力ごとに指定されたレジスタ内の内容を読み出す。

3.4 符号拡張

符号拡張への元の入力の最上位 bit で符号拡張する部分を埋める。これも verilog の記述では入力 a、符号拡張した出力 y に対して
y = { {8{a[7]}}, a} ;
のように簡単にかける。具体的には a[7] が a の最上位 bit を表しその値で上位 8bit を埋める操作をしている。

3.5 ALU

制御線 alucontrolE の値によって演算を行う。その対応表は図 5 の通り。各演算についてどのような計算を行ったかは図 6 に示す。この図での各行のオレンジ色の 4bit 値が alucontrolE の値に対応する。ADD、SUB、AND、OR XOR、CMP、MOV、については演算子と C の定義のままである。

SLL は verilog の << の演算子によりこの演算子の右のオペランドの値だけこの演算子の左側の値を符号なしで左シフトするのでこのように実現できる。SLR についてはまず 16-b[3:0] が b を 16 で割ったあまりであることから説明する。まず 2 進数 b の各 bit を下位 bit から順に $a_0, a_1, \cdots, a_{15}$ とすると b の 10 進表現は

$$b_{10} = 2^{15} \times a_{15} + 2^{14} \times a_{14} + \cdots + 2 \times a_1 + 1 \times a_0$$
$$= 16 \times (2^{11} \times a_{15} + 2^{10} \times a_{14} + \cdots + 2 \times a_5 + 1 \times a_4) + 2^3 \times a_3 + 2^2 \times a_2 + 2 \times a_1 + 1 \times a_0$$

と表せる。したがって

$$b_{10}\, mod(16) \equiv 2^3 \times a_3 + 2^2 \times a_2 + 2 \times a_1 + 1 \times a_0$$

つまり b[3:0] は b の 16 で割ったあまりを示している。例えば $b_{10} = 18$ のときこれを 16 で割ったあまりは 2 であり実際 $b[3:0] = 0010$ となっている。よって $(16 - b[3:0])_{10}$は 14 を表しこれに対して $\{a, a\} >> (16 - b[3:0])$ という演算を行うと a を 2 つつなげた 32bit の値 {a,a} を 14 だけ右にずらすということなので結果的にこれは左に 2 シフトしてあまりを右に補う操作と同値になる。以上が SLR の説明である。SRL は SLL と同様でシフトの向きが逆なだけである。SRA は verilog の >>> が符号有りでシフトする演算なのでこのように記述できる。

```
4'b0000:
  { C, result } <= a + b;
4'b0001, 4'b0101:
  { C, result } <= a - b;
4'b0010:
  result <= a & b;
4'b0011:
  result <= a | b;
4'b0100:
  result <= a ^ b;
4'b0110:
  result <=  b;
4'b1000:
  { C, result } <= a << b;
4'b1001:
  result <= { a, a } >> (16-b[3:0]);
4'b1010:
  { result, C } <= { a , 1'b0 } >> b;
4'b1011:
  { result, C } <= { a , 1'b0 } >>> b;
4'b1100:
  result <= a + b;
```

図 6　alucontrolE に対応する演算内容

3.6　パイプラインレジスタ

これもマルチプレクサと同様、可変の parameter を用意する。特にパイプラインレジスタは改良の際にそのレジスタへの入力値の変更が起きやすいのでこのように設計することで改良がしやすかった。コンポーネントの役割は簡単でストール・フラッシュの制御線が 0 のときはクロックごとに入力を出力にあたえるだけである。ストールの制御線が 1 のときは入力を出力に渡さないようにする。フラッシュの制御線が 1 のときは何も行わない信号を出力する。SIMPLE であれば 1100_0000_0111_0000 などが reserved で何もしない命令となっている。

3.7　メモリ

データメモリ DM、命令メモリ IM ともに枚クロックごとに読み込んだアドレスに格納されている値を出力する。データメモリについては memwriteM の値が 1 のときのみ入力 writedata の値を入力アドレスに書き込む。

3.8　ブランチ制御ユニット

このユニットは ALU で出力された S、Z、C、V と制御部からの branch 信号で pcsrsM を定める。branch 信号はまず制御部から branchD = {be,blt,ble,bne,br} として verilog で記述され出力される。例えば branchD = 00010 の場合、今回の命令は BNE であることを示し、branchD = 00000 なら今回の命令は branch 命令でないことを示す。これと S、Z、C、V を用いることで例えば branchD[0] & ZE=1 のときこれは BE の条件が成り立ち pcsrcM に 1 が出力される。このように (branchD の各 bit)&(その bit が示す分岐が成立する条件コード) を羅列することでこれが pcsrcM が 1 になる条件となる。もちろんそれ以外のときは pcsrcM には 0 が設定される。