

2022 年度 3 回生前期学生実験 HW
team02 機能設計仕様書 植田健斗担当分

機能設計仕様書作成者: 植田健斗

グループメンバー：

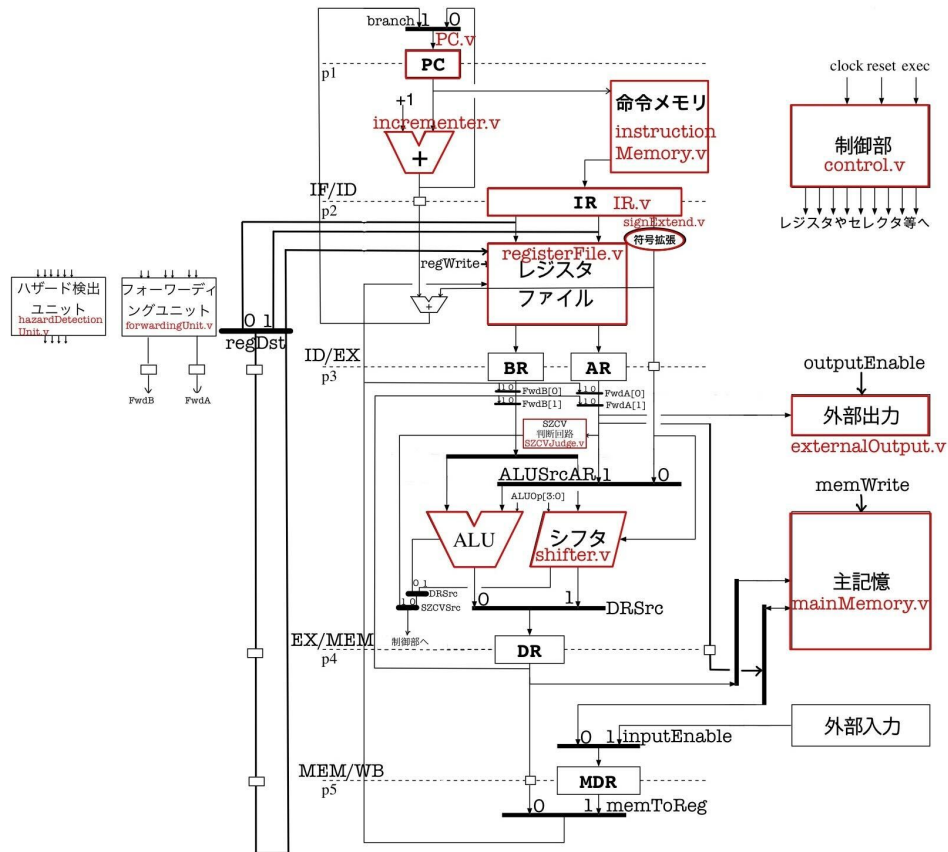
伊藤舜一郎 (学籍番号:1029-32-7548)

植田健斗 (学籍番号:1029-32-6498)

提出期限：6 月 9 日 13 時 提出日: 2022 年 6 月 3 日

1 全体のコンポーネント分割方法

それぞれの機能を持った回路ごとにモジュール分割した。各回路につけたファイル名は図1に示した通りである。



各レジスタ:register16.v,register4.v,register3.v

各マルチプレクサ:multiplexer16.v,multiplexer4.v,multiplexer3.v

全体:processor.bdf

図 1: コンポーネント分割の方法の図

追加・変更したコンポーネントの役割分担の表を以下の表 1 に示す。

表 1: 役割分担（最終報告時点）

コンポーネント	追加か変更か	担当
フォーワーディングユニット	追加	伊藤
クロックカウンター	追加	伊藤
アセンブリコード (基数ソート)	(応用プログラム)	伊藤
ハザード検出ユニット	追加	植田
パイプラインレジスタ	追加	植田
命令メモリ	追加	植田
制御部	変更	植田
レジスタファイル	変更	植田
全体	変更	植田
アセンブリコード (度数ソート)	(ソートコンテスト用)	植田 (レポートは伊藤)

2 変更を加えたコンポーネント

2.1 レジスタファイル (registerFile.v)

2.1.1 外部仕様 入力

この回路の入力は以下の表 2 のようになる。

表 2: レジスタファイル (registerFile.v) の入力 (=input)

input	input の説明
Rs[2:0]	命令中の Rs・Ra フィールドを受け取る。
Rd[2:0]	命令中の Rd・Rb フィールドを受け取る。
regWrite	内部のレジスタが書き換え可能かを表す信号 regWrite と changeEnable がともに 1 のときのみ書き換え可能
writeData[15:0]	レジスタに書き込むデータ
writeRegister[2:0]	writeData[15:0] を書きこむレジスタの番号
clock	クロック信号。clock の立ち上がりで、レジスタ書きこみが可能な場合、書きこみが行われる。
reset	初期化信号。reset が 1 のときにクロックが立ち上がると、レジスタの中身を 0 で初期化する。
changeEnable	内部のレジスタが書き換え可能かを表す信号。regWrite と changeEnable がともに 1 のときのみ書き換え可能

2.1.2 外部仕様 出力

この回路の出力は以下の表 3 のようになる。

表 3: レジスタファイル (registerFile.v) の出力 (=output)

output	output の説明
AR[15:0]	Rs(Ra) フィールドで指定したレジスタに格納されているデータを読み込む。
BR[15:0]	Rd(Rb) フィールドで指定したレジスタに格納されているデータを読み込む。

2.1.3 内部仕様

内部に 8 個の 16bit レジスタ reg [15:0] r [7:0] を持ち、入力 Rs[2:0],Rd[2:0] の値に応じたレジスタに格納されたデータをそれぞれ AR[15:0],BR[15:0] として出力する。

また、reset が 0 で changeEnable と regWrite がともに 1 のときのみ書き換え可能状態になり、この状態で clock が立ち上がると、入力 writeData[15:0] で受け取ったデータを writeRegister[2:0] に対応する番号の内部レジスタ r にデータを書きこむ。

regWrite が 1 のときに、writeRegister[2:0] と Rs[2:0] もしくは Rd[2:0] の値が等しいときは AR[15:0] もしくは BR[15:0] には、writeData[15:0] が出力される。

reset が 1 のときに clock が立ち上がると、すべての内部レジスタ r の値を 0 で初期化する。

このレジスタファイルはレジスタへの書き込みとそのレジスタからのデータの読み出しを同じクロック周期の中で行ったときに、中間報告時点で作成したレジスタファイルでは、書き込みを行う前のデータが読みだされてしまっていたが、最終報告の段階ではあるレジスタへの書き込みとそのレジスタからのデータの読み出しを同じクロック周期の中で行ったときに、同じクロックの中で書き込みするデータを読みだせるように改良をした。

2.2 制御部 (control.v)

2.2.1 外部仕様 入力

この回路の入力は以下の表 4 のようになる。

表 4: 制御部 (control.v) の入力 (=input)

input	input の説明
clock	クロック信号。clock 立ち上がりで、レジスタ書きこみが可能な場合、書きこみが行われる。
instruction[15:0]	命令の格納された IR の出力を受け取る。
reset	初期化信号。reset が 1 のときにクロックが立ち上がると、出力 systemRunning 信号を 0 にする。
exec	全体の実行・停止信号 exec を受け取る。
SZCV[3:0]	前回の命令での SZCV フラグを受け取る。LD,IN 命令のときは SZCV の値は未定義

2.2.2 外部仕様 出力

この回路の出力は以下の表 5 のようになる。

表 5: 制御部 (control.v) の出力 (=output)

output	output の説明
regDst	書き込みをするレジスタを、入力 Instruction[10:8](Rd,Rb フィールド) で指定する (=0) か、Instruction[13:11](Ra,Rs フィールド) で指定する (=1) かを決める。
ALUSrcAR	ALU の入力 inB[15:0] の受け取る値が、命令中の即値 (=0) か、AR の値 (=1) かを決める。
ALUOp[3:0]	ALU の入力 op[3:0] に入力する値で、ALU で行う演算を決める。
DRSrc	DR に入力する値を、ALU の出力 out[15:0] にする (=0) か、シフタの出力 out[15:0] にする (=1) かを決める。また、SZCV に入力する値を、ALU の出力 SZCV[3:0] にする (=0) か、シフタの出力 SZCV[3:0] にする (=1) かを決める。
outputEnable	外部出力に値を受け渡す (=1) か受け渡さない (=0) かを決める。正確には OUT 命令のときのみ 1 となる。
SZCVSrc	SZCV フラグを ALU・シフタの値をもとに決めるか (=0)、AR の値をもとに決める (=1) かを決める。
memRead	LD 命令と IN 命令の時に 1 になる。そのほかは 0 になる。
inputEnable	外部入力から値を受け取る (=1) か受け取らない (=0) かを決める。正確には IN 命令のときのみ 1 となる。
memWrite	メモリに書きこむ (=1) か書きこまない (=0) かを決める。正確には ST 命令のときのみ 1 となる。
branch	入力の SZCV[3:0] と instruction[15:0] から、分岐をする (=1) か、分岐をしない (=0) かを決める。
regWrite	レジスタファイルに書き込みを行う (=1) か、レジスタファイルに書き込みを行わない (=0) かを決める。
memToReg	プログラムカウンタや、レジスタファイルに渡すデータを、DR の値にする (=0) か MDR の値にする (=1) かを決める。
notReadRsRd	ID フェーズでデコードした命令が NOP,HLT,LI,B,BE,BLT,BLE,BNE のときに 1 になり、パイプラインストールを起こさない。それ以外のときは 0
systemRunning	すべてのレジスタの書き換えを不可能にする (=0) か可能にする (=1) かを決める。つまり、機械が実行待機状態 (=0) か動作状態 (=1) かを決める。
IFFlush	通常は 0 で、HLT 命令が ID フェーズで処理された時から、その HLT 命令が WB フェーズにある時まで、1 になる。
PCWrite	通常は 1 で、HLT 命令が ID フェーズで処理された時から、その HLT 命令が WB フェーズにある時まで、0 になる。

2.2.3 内部仕様

systemRunning という 1bit の内部レジスタを持ち、その値を変えることで機械を実行待機状態 (systemRunning=0) か実行状態 (systemRunning=1) かを切り替えている。exec 入力については

1 クロック前の exec 入力を保持する 1bit レジスタ exec_pre とカウンタを表す 16bit の内部レジスタ counter を持つ。これを用いて以下の表 6 のように exec と exec_pre の値に応じて、counter と systemRunning を変更する。これにより exec ボタンのチャタリングを除去しながら、exec ボタンが押されるたびに機器が実行状態かどうかを決める systemRunning を反転させることができる。

表 6: exec と exec_pre と cnt(=counter) の値に応じて、更新する cnt(=counter) と systemRunning の値

exec_pre exec	00	01	10	11
cnt=0	cnt<=0	cnt<=1	cnt<=1	cnt<=0
0<cnt<15	cnt<=cnt+1	cnt<=0	cnt<=0	cnt<=cnt+1
cnt=15	cnt<=0	cnt<=0	cnt<=0	cnt<=0, systemRunning<=~systemRunning

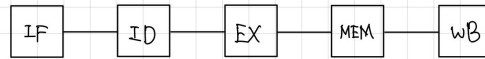
reset が呼ばれると、

```
counter <= 16'h 0000;
systemRunning <= 1'b 0;
exec\_pre <= exec;
```

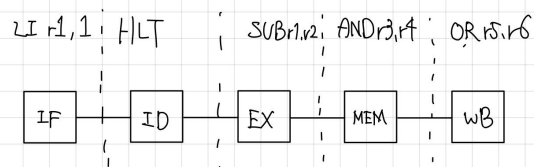
により初期化を行う。reset が押されると systemRunning は 0 になるので、実行が停止し、実行待機状態になる。その後 exec ボタンが押されることで systemRunning が 1 になり実行が行われるようになる。また、インプット命令の EX フェーズと halt 命令の WB フェーズで systemRunning を 0 にし実行待機状態になるようにしている。この場合も exec ボタンを押すことで実行を再開できる。

HLT 命令の処理は以下の図 2 ように行っている。

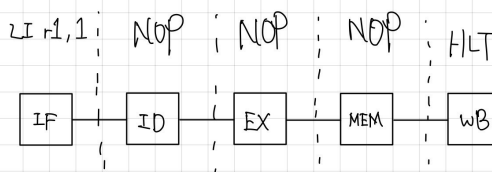
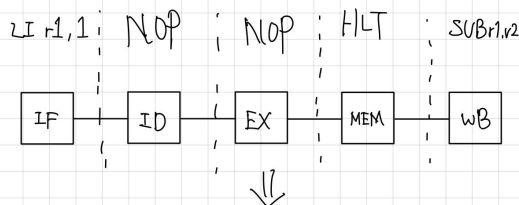
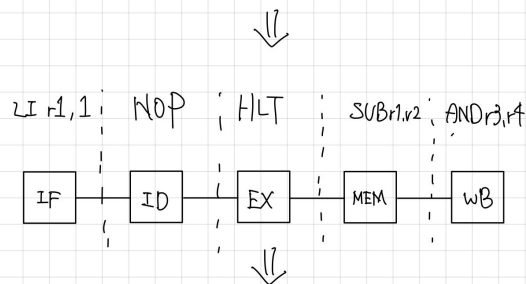
以下ではそれぞれのフェーズの処理を行うハードウェアを以下の図形で表すとする。
それぞれのハードウェアが処理している命令をそのハードウェアの図形の上に記す。



HLT命令がIDフェーズで処理されたからの図を以下に示す。



左のようにHLT命令がIDフェーズで処理されると、PCWriteを0に12, PCを書き換え不可にし、IFFlushを1に12, IDフェーズに次のクロックの立ち上がりにはIRにNOP命令が入るようになる。



左の図のようになったとき、SystemRunningが0になり、すべてのレジスタが書き換え不可になり、実行待機状態になる。この状態でexecボタンを押すと、SystemRunningが1になり、HLT命令の次の番地の命令から実行が再開されるようになる。

図 2: HLT 命令が呼ばれたときの制御

各制御信号は組み合わせ回路を用いて実現している。以下の表 7 でそれぞれの制御信号の出力の仕様をまとめた。

表 7: 制御信号の仕様

制御信号名	制御信号の仕様
regDst	LD 命令の時のみ 1
ALUSrcAR	演算, シフト, IN, OUT, NOP, HALT 命令の時に 1 になる。(MOV の下の reserved の時も 1)
ALUOp[3:0]	演算命令 (MOV の下の reserved も含む) のときは instruction[7:4](命令で指定した演算) になる。LI 命令の時は 4'b 0110(MOV 命令) になる。その他のときは 4'b 0000(アドレス計算のための足し算) になる。
DRSrc	シフト, IN, OUT, NOP, HALT 命令の時のみ 1
outputEnable	OUT 命令の時のみ 1
SZCVSrc	演算、シフト、IN, OUT, NOP, HALT, LD, ST 命令の時に 1 になる。(MOV の下の reserved の時も 1)
memRead	IN, LD 命令の時のみ 1 になる。LD, IN 命令のときに入力されるデータを次の命令で読み出す際におこるデータハザードが起こるかどうか判定する。
inputEnable	IN 命令の時のみ 1
memWrite	ST 命令の時のみ 1
branch	B 命令または (BE 命令かつ Z=1) または (BLT 命令かつ $S^{\wedge}V=1$) または (BLE 命令かつ $Z (S^{\wedge}V)=1$) または (BNE かつ $\sim Z=1$) の時のみ 1
regWrite	ADD, SUB, AND, OR, XOR, MOV, シフト, IN, LD, LI 命令の時のみ 1
memToReg	LD, IN 命令の時のみ 1
notReadRsRd	NOP, HLT, B, BE, BLT, BLE, BNE のとき 1 となる。これが 1 のとき、パイプラインストールストールを起こさない。
IFFlush	通常は 0 で HLT 命令が ID フェーズで読み込まれたときから、4 サイクルの間、IFFlush は 1 になる。HLT 命令以降の命令で読み込まれたものを実行しないように NOP 命令に置き換える。
PCWrite	通常は 1 で HLT 命令が ID フェーズで読み込まれたときから、4 サイクルの間、PCWrite は 0 になる。HLT 命令が ID フェーズに来た時に PC を書き換え不可にし、HLT 命令がある次の番地に PC の値を固定する。

3 追加したコンポーネント

3.1 ハザード検出ユニット (hazardDetectionUnit.v)

パイプラインハザードとは、ここでは LD,IN 命令の直後の命令で LD,IN 命令により保存したデータを参照することによるデータハザードを指している。

3.1.1 外部仕様 入力

この回路の入力は以下の表 8 のようになる。

表 8: ハザード検出ユニット (hazardDetectionUnit.v) の入力 (=input)

input	input の説明
IDEX_memRead	EX フェーズで処理をしている命令の memRead 信号を受け取る。
IDEX_RegRd[2:0]	EX フェーズで処理をしている命令の書き込みレジスタの番号を受け取る。
IFID_Rs[2:0]	ID フェーズで処理している命令中の Rs フィールドの値を受け取る。
IFID_Rd[2:0]	ID フェーズで処理している命令中の Rd フィールドの値を受け取る。
notReadRsRd	制御部 (control.v) の出力 notReadRsRd を受け取る。
branch	制御部 (control.v) の出力 branch を受け取る。

3.1.2 外部仕様 出力

この回路の出力は以下の表 9 のようになる。パイプラインハザードとは、ここでは LD,IN 命令の直後の命令で LD,IN 命令により保存したデータを参照することによるデータハザードを指している。

表 9: ハザード検出ユニット (hazardDetectionUnit.v) の出力 (=output)

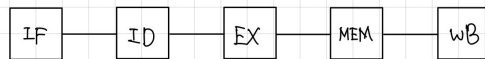
output	output の説明
PCWrite	パイプラインハザードが起きた時に 0 になり、PC の書き換えをやめる。
IFIDWrite	パイプラインハザードが起きた時に 0 になり、IF フェーズと ID フェーズの間のパイプラインレジスタ (IR) の書き換えをやめる。
IFFlush	分岐が成立したときに 1 になり (=branch)、IR に PC のアドレスで命令メモリから読み出した命令ではなく、NOP 命令を入力する。
IDFlush	パイプラインハザードが起きた時に 1 になり、ID フェーズと EX フェーズの間にある制御信号が格納されたパイプラインレジスタの値を初期化する。

3.1.3 内部仕様

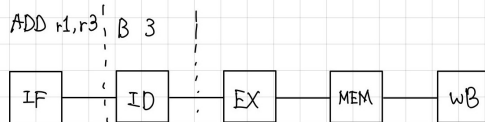
組み合わせ回路を用いて実現されている。LD,IN 命令で読み込んだ命令を直後の命令で参照する際におこるデータハザードと、分岐命令の際の制御ハザードに対処するために、制御信号を出す。

LD,IN 命令で読み込んだ命令を直後の命令で参照する際におこるデータハザードへの対処と分岐命令の際の制御ハザードへの対処は以下の図 3 ように行っている。

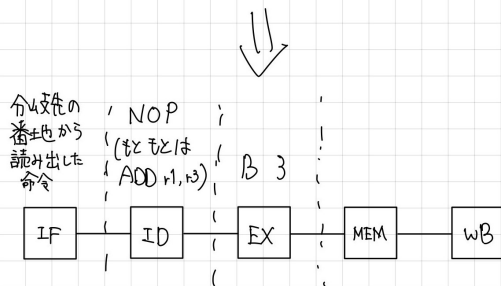
以下ではそれぞれのフェーズの処理を行うハードウェアを以下の図形で表すとする。
それぞれのハードウェアが処理している命令とそのハードウェアの図形の上に記す。



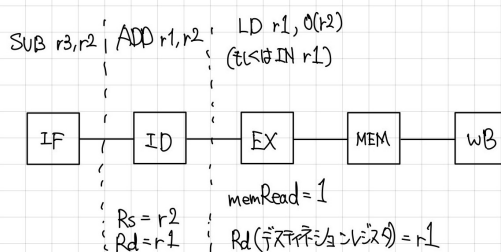
1. 分岐命令が成立するときの処理を以下で示す。



左のようにIDフェーズで分岐命令が処理され分岐が成立したときに、IDFlush信号が1になる。これにより、分岐命令の次の命令は次のクローラの立ち上がりでNOP命令におきかわり、PCには分岐先のアドレスがセットされる。



2. LD, IN 命令で読み込んだデータを、直後の命令で参照する場合の処理を以下で示す。



左図のように memRead 信号が1 (つまり、LD, IN 命令のいずれか) かつ、LD, IN 命令の書き込みレジスタと、次の命令 (例では ADD r1, r2) の読み出しレジスタが一致するとき、PC と IR の書き換え信号を0にし、さらに ID フェーズと EX フェーズの間にある制御信号の保存したパイプラインレジスタを初期化する。これにより、次のクローラでは、IF と ID フェーズにある命令はそのまま、EX フェーズに NOP 命令が挿入され、LD 命令は MEM フェーズに処理が始まる。LD, IN 命令は、MEM フェーズの終わりにならないと、読み込みのデータが命令からないので、この処理により、LD, IN 命令の次の命令でそのデータを読み出したとしても、フォークディングを用いて、データの受け渡しをすることができる。これにより、データハザードを防いだ。

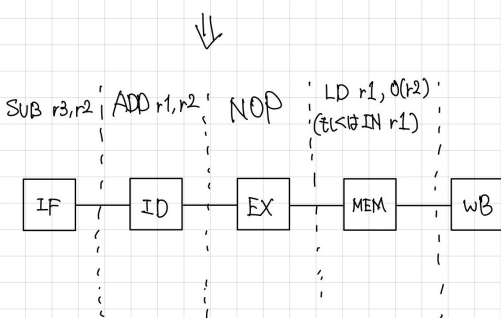


図 3: 分岐命令の制御ハザードと、LD, IN 命令で読み込んだ命令を直後の命令で参照する場合におけるデータハザードへの対処

3.2 命令メモリ (instructionMemory.v)

3.2.1 外部仕様 入力

この回路の入力は以下の表 10 のようになる。

表 10: 命令メモリ (instructionMemory.v) の入力 (=input)

input	input の説明
address[11:0]	PC の値を受け取る。メモリの読み書きを行う番地のアドレス。
clock	クロック信号を反転させて入力する。clock の立ち上がり (=クロック信号の立ち下り) で、メモリ書きこみが可能な場合書きこみが行われ、また、address[11:0] により指定した番地のデータの読み出しが行われる。
data[15:0]	メモリに書きこむためのデータ。このメモリには書き込みは行われない。
wren	書き込み可能信号。常に 0 が入力される。

3.2.2 外部仕様 出力

この回路の出力は以下の表 11 のようになる。

表 11: 命令メモリ (instructionMemory.v) の出力 (=output)

output	output の説明
q[15:0]	メモリから読み出しされたデータ

3.2.3 内部仕様

RAM を用いて実装した。PC の下位 12bit である address[11:0] で指定した番地に対して、clock が立ち上がる度にデータの読み出しを行う。書き込みは行われない。ROM での実装でも良いが拡張する可能性を考えて RAM での実装にしている。

命令を格納するためのメモリである。dosuSort.mif をここにセットし、ソートするデータが 0x400 から 0x7ff 番地に書かれた mif ファイル (sorted.mif, random.mif, r_sorted.mif) を mainMemory.v に入れることで、度数ソートを実行できる。

3.3 パイプラインレジスタ (IDEXRegister.v,EXMEMRegister.v,MEMWBRegister.v)

3.3.1 外部仕様 入力

この回路の入力は以下の表 12 のようになる。

表 12: パイプラインレジスタ (IDEXRegister.v,EXMEMRegister.v,MEMWBRegister.v) の入力 (=input)

input	input の説明
各制御信号	クロックの立ち上がりによりレジスタ書き込み可能ならば、内部のレジスタに書きこむ値
clock	クロック信号。clock の立ち上がりで、レジスタ書き込みが可能な場合、書きこみが行われる。
changeEnable	内部のレジスタが書き換え可能かを表す信号 (イネーブル信号)。この信号が 1 のときに clock が立ち上がると内部のレジスタの値が更新される。
reset	初期化信号。reset が 1 のときにクロックが立ち上がると、レジスタの中身を 0 で初期化する。
IDFlush	IDEXRegister.v のみ存在。1 のときにクロックが立ち上がると制御信号をすべて 0 にする。

3.3.2 外部仕様 出力

この回路の出力は以下の表 13 のようになる。

表 13: パイプラインレジスタ (IDEXRegister.v,EXMEMRegister.v,MEMWBRegister.v) の出力 (=output)

output	output の説明
各制御信号	内部のレジスタに格納されている制御信号の値を出力する。

3.3.3 内部仕様

レジスタを表している。clock の立ち上がりごとに以下のことを行う。reset 信号が 1 のときレジスタを 0 で初期化する。reset 信号が 0 のとき、changeEnable が 1 なら、入力の制御信号の値を内部のレジスタに格納する。それ以外ならば値の変更は何もしない。内部レジスタの値を常に出力する。IDEXRegister.v,EXMEMRegister.v,MEMWBRegister.v はそれぞれ IDEX フェーズ間、EXMEM フェーズ間、MEMWB フェーズ間の制御信号を格納するパイプラインレジスタを表している。

3.4 全体 (processor.bdf)

3.4.1 外部仕様 入力

この回路の入力は以下の表 14 のようになる。

表 14: 全体 (processor.bdf) の入力 (=input)

input	input の説明
clock	クロック信号の入力を受け取る。
reset	リセットボタンの入力を受け取り、機械を初期状態にし、実行待機状態にする。(この後に exec ボタンを押すことで実行が始まる。)
exec	exec ボタンの入力を受け取る。機械を実行停止・再開する。reset ボタンを押した後や、IN 命令が呼ばれた後、exec ボタンを押すことで実行が再開される。
externalInput[15:0]	IN 命令のときに入力されるデータ

3.4.2 外部仕様 出力

この回路の出力は以下の表 15 のようになる。外部出力の詳細はペアの伊藤くんの中間報告時の機能設計仕様書の外部出力 (externalOutput.v) と、ペアの伊藤くんの最終報告時の機能設計仕様書のクロックカウンタ (clockCounter.v) に書かれている。

表 15: 全体 (processor.bdf) の出力 (=output)

output	output の説明
SEG_A[7:0]	7SEG-LED に表示する数字の一つ
SEG_B[7:0]	7SEG-LED に表示する数字の一つ
SEG_C[7:0]	7SEG-LED に表示する数字の一つ
SEG_D[7:0]	7SEG-LED に表示する数字の一つ
SEG_E[7:0]	7SEG-LED に表示する数字の一つ
SEG_F[7:0]	7SEG-LED に表示する数字の一つ
SEG_G[7:0]	7SEG-LED に表示する数字の一つ
SEG_H[7:0]	7SEG-LED に表示する数字の一つ
select[7:0]	7SEG-LED の表示を行うためのセクタ信号
SEG_X[7:0]	7SEG-LED に表示する数字の一つ
SEG_Y[7:0]	7SEG-LED に表示する数字の一つ
selectX[3:0]	7SEG-LED の表示を行うためのセクタ信号
selectY[3:0]	7SEG-LED の表示を行うためのセクタ信号

3.4.3 内部仕様

全体の配線は図 1 のように行った。フォーワーディングユニットとハザード検出ユニットと制御部の配線は図 1 では省略されているので以下で特に詳しく説明する。

コントロールからの制御信号のうち、regDst,ALUSrcAR,DRSrc,SZCVSrc,inputEnable,branch,memToReg についてはパイプラインレジスタを通して図 1 のマルチプレクサ (図 1 では太い横線がマルチプレクサを表す) の制御信号として入力されている。

ALUOp[3:0] は ALU とシフタのオペコードを受け取る入力 op[3:0] につないだ。

outputEnable,memWrite,regWrite,systemRunning についてはパイプラインレジスタを通したあと、それらを用いてそれぞれのレジスタやメモリのイネーブル信号を構成し入力した。

制御信号の出力 IFFlush とハザード検出ユニットの出力 IFFlush の OR をとり、IR(命令レジスタ) の reset 信号とした。

制御信号の出力 PCWrite とハザード検出ユニットの出力 PCWrite の OR をとり、PC(プログラムカウンタ) のイネーブル信号とした。

入力 clock はすべてのモジュールの clock 入力につなぎ、さらに clock を反転させた信号を二つのメモリの入力 clock につないだ。

入力 reset はメタステーブルを防止するため dff を 2 つ直列につないだ後、負論理を正論理にするため反転させ、PC, レジスタファイル,IR, 制御部, 制御信号を格納するパイプラインレジスタ、externalOutput, 各レジスタ、クロックカウンタの入力 reset につないだ。

入力 exec はメタステーブルを防止するため dff を 2 つ直列につないだ後、負論理を正論理にするため反転させ、制御部につないだ。

制御部の出力 ID フェーズの notReadRsRd,ID フェーズの branch,EX フェーズの memRead をハザード検出ユニットの入力 notReadRsRd,branch,IDEX_memRead につないだ。

IR の Rs,Rd フィールドをハザード検出ユニットの入力 IFID_Rs,IFID_memRead につないだ。

regDst の制御信号が入力されているマルチプレクサからの出力の EX フェーズの値をハザード検出ユニットの入力 IDEX_RegRd とフォーワーディングユニットの入力 IDEX_RegRd につなぐ。

ハザード検出ユニットの出力 IDFlush を、IDEX フェーズ間の制御信号を格納するパイプラインレジスタ (IDEXRegister.v) の入力 IDFlush につなぐ。

ハザード検出ユニットの出力 IFIDWrite を、IR のイネーブル信号につなぐ。

regDst の制御信号が入力されているマルチプレクサからの出力の MEM フェーズの値をフォーワーディングユニットの入力 EXMEM_RegRd につなぐ。

IR の Rs,Rd フィールドの値をフォーワーディングユニットの入力 IFID_RegRs,IFID_RegRd にそれぞれつなぐ。

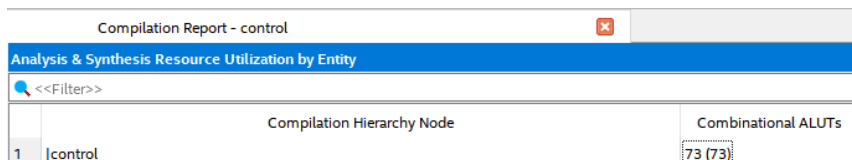
EX フェーズの制御信号 regWrite をフォーワーディングユニットの IDEX_RegWrite に、EX フェーズの制御信号 regWrite をフォーワーディングユニットの EXMEM_RegWrite につなぐ。

フォーワーディングユニットの出力 FwdA,FwdB はパイプラインレジスタを一つ通った後、図 1 のマルチプレクサの制御信号に入力される。

4 単体での性能評価 (制御部、ハザード検出ユニット)

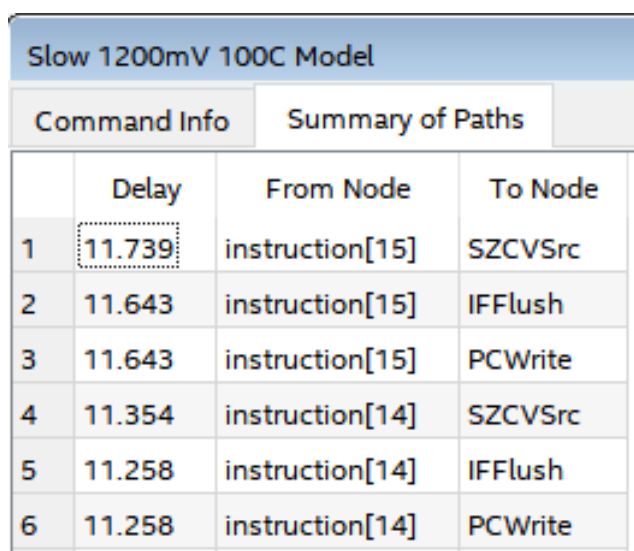
4.1 制御部

total logic element は 73(図 4)、遅延時間は 11.739ns(図 5)、クリティカルパスは instruction[15] から SZCVSrc(図 5) のパスである。



Compilation Report - control	
Analysis & Synthesis Resource Utilization by Entity	
<<Filter>>	
Compilation Hierarchy Node	Combinational ALUTs
1 control	73 (73)

図 4: Resource Utilization by Entity

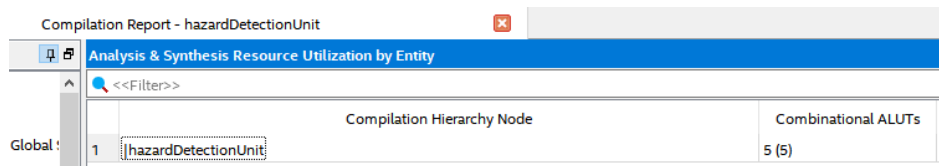


Slow 1200mV 100C Model			
Command Info		Summary of Paths	
	Delay	From Node	To Node
1	11.739	instruction[15]	SZCVSrc
2	11.643	instruction[15]	IFFlush
3	11.643	instruction[15]	PCWrite
4	11.354	instruction[14]	SZCVSrc
5	11.258	instruction[14]	IFFlush
6	11.258	instruction[14]	PCWrite

図 5: Report Path... の結果

4.2 ハザード検出ユニット

total logic element は 5(図 6)、遅延時間は 9.108ns(図 7)、クリティカルパスは IDEX_RegRd[0] から IFIDWrite(図 7) のパスである。



Compilation Report - hazardDetectionUnit	
Analysis & Synthesis Resource Utilization by Entity	
<<Filter>>	
Compilation Hierarchy Node	Combinational ALUTs
1 hazardDetectionUnit	5 (5)

図 6: Resource Utilization by Entity の結果

Slow 1200mV 100C Model			
Command Info		Summary of Paths	
	Delay	From Node	To Node
1	9.108	IDEX_RegRd[0]	IFIDWrite
2	9.098	IDEX_RegRd[0]	IDFlush
3	9.078	IDEX_RegRd[0]	PCWrite
4	9.018	IFID_Rs[2]	IFIDWrite
5	9.012	IDEX_memRead	IFIDWrite
6	9.008	IFID Rs[2]	IDFlush

図 7: Report Path... の結果

5 考察・感想

5.1 考察

このプロセッサにおいて、実行速度を上げるための考察を記す。このプロセッサは、フォーワーディングにより、WB フェーズから EX フェーズにデータが送られ、そのデータを ALU のオペランドとして使い、SZCV フラグを計算し、その SZCV フラグをもとに制御部で branch 信号を出力し、それがハザード検出ユニットを通して、IDFlush に入力するパスが、最も長いパスの一つである。SZCV フラグを計算するまでと計算してからで二つのフェーズにわけること、分岐予測が失敗したときのペナルティは大きくなるかもしれないが、実行可能周波数を大きくでき、実行速度を上げることができるとおもわれる。

5.2 感想

五段パイプライン化の構造の設計を主に担当したが、制御部の構造が複雑で特に大変だった。マルチサイクル方式から、パイプライン化を行うのはそこまで難しくはなかった。SIMPLE/B の構造を作るのが一番大変だった。この演習は、各コンポーネントの結合度が強く、分担ができない箇所が多いので、僕のほうが設計の大部分を担い、さらに全体の組み立て、デバックやペアの指導 (ペアの理解を助けたり、設計内容や書く内容を指示したり) などやることになり大変だった。

ただ、プロセッサを作ったことで、プロセッサの構造や、ハザードなどに対する理解が深まり良かったと思う。

6 実験環境

実験で使用したボードや CAD ツールを以下に記す。

- ボード
Rapid Prototyping Kit PowerMedusa
MU500-RXSET01(MU500-RX, MU500-RK, MU500-7SEG)
- CAD ツール
Quartus Prime Version 17.1.0 Build 590 10/25/2017 SJ Lite Edition