機能設計仕様書

2018 年 4 月 23 日

1029-28-9483 勝田 峻太朗

p2

コンポーネントの外部仕様

概要

このコンポーネントの基本的な動作としては, 命令を解釈すると同時に, 後続の p3 が必要とする値をレジスタから読み出す.

入力

このコンポーネントは, 入力として, 以下の 2 つを受け取る.

clock(2bit) クロック
command(16bit) p1 によって読み出された命令を入力とする.

出力

alu1, alu2(16bit) 演算命令の場合,ALU が使用する値 2 つをレジスタや即値から取得する.
opcode(4bit) ALU で処理をする場合に, 行うべき演算を示している.

code	計算
0000	in1 + in2
0001	in1 - in2
1000	in1 & in2 (bitwise)
1001	in1 | in2 (bitwise)
1010	in1 « i2
1011	in1 » in2

writereg(1bit) 演算または, ロード命令の場合, 結果をレジスタに書き込む必要がある. 書き込む場合は 1, 書き込まない場合は 0 である.
regaddress(3bit) writereg が 1 の場合, 書き込む対象となるレジスタの番号を示す.
memwrite(2bit) メモリに行う操作をコードで表す. 何もしない場合は 00, 読み込み時は 01, 書き込み時は 10 を示す.
address(16bit) メモリに操作をを行う場合, どの番地に行うかを示している.
storedata(16bit) メモリに書き込みを行う場合, 書き込む内容を示す.

内部仕様

入力として命令とクロックをを受け取り, 出力レジスタに対して, クロックの立ち上がりとともに, 対応するデータを書き込む. 書き込む値は, 全て場合分け関数をもちいて出力される.

また, 内部モジュールとして, メインレジスタ Registers を含む.

```
// connect to register
Registers(.clock(clock), .rs(alu1address), .rd(alu2address), .readflag(1'b1), .value(16'b0), .read1(
```

以下では, 各出力を決定するために用いている関数を示す.

alu1,alu2,opcode

```
// connect to register
Registers(.clock(clock), .rs(alu1address), .rd(alu2address), .readflag(1'b1), .value(16'b0), .read1(

// omitted...

function [2:0] getaluaddress1;
input [15:0] command;
case (command[15:14])
    3: getaluaddress1 = command[13:11];
    0: getaluaddress1 = command[13:11];
    1: getaluaddress1 = command[13:11];
    default: getaluaddress1 = 3'b000;
endcase
endfunction

function [2:0] getaluaddress2;
input [15:0] command;
case (command[15:14])
    3: if (command[7:4] <= 4'd8) begin
            getaluaddress2 = command[10:8];
        end else begin
            getaluaddress2 = command[3:0];
        end
    0: getaluaddress2 = command[10:8];
    1: getaluaddress2 = command[10:8];
    default: getaluaddress2 = 3'b000;
endcase
endfunction

//..

always @(posedge clock) begin
    //..
    // get alu1 and 2
    alu1address = getaluaddress1(command);
    alu2address = getaluaddress2(command);
    alu1 = alu1val;
    alu2 = alu2val;
```

```
    opcode = command[7:4];
end
```

writereg, regaddress

```
// function to get writereg
function getwritereg;
input [15:0] command;
    case (command[15:14])
        3: getwritereg = 1'b1;
        0: getwritereg = 1'b1;
        1: getwritereg = 1'b0;
        2: getwritereg = 1'b1;
        default: getwritereg = 1'b0;
    endcase
endfunction

// function to get regaddress
function [2:0] getregaddress;
input [15:0] command;
    case (command[15:14])
        3: getregaddress = command[10:8];
        0: getregaddress = command[13:11];
        2: getregaddress = command[10:8];
        default: getregaddress = 2'b00;
    endcase
endfunction

always @(posedge clock) begin
    //..
    // get register things
    writereg = getwritereg(command);
    regaddress = getregaddress(command);
end
```

memwrite, address

```
// function to get memwrite
function [1:0] getmemwrite;
input [15:0] command;
    case (command[15:14])
        3: getmemwrite = 2'b00;
        0: getmemwrite = 2'b01;
        1: getmemwrite = 2'b10;
```

```
        2: getmemwrite = 2'b01;
        default: getmemwrite = 2'b00;
    endcase
endfunction

// function to get memory address
function [15:0] getaddress;
input [15:0] alu2;
input [15:0] command;
    case (command[15:14])
    0: getaddress = alu2 + signext8(command[7:0]);
    1: getaddress = alu2 + signext8(command[7:0]);
    2: getaddress = signext8(command[7:0]);
    endcase
endfunction

//..

always @(posedge clock) begin
    //..
    // get memory things
    memwrite = getmemwrite(command);
    address = getaddress(alu2val, command);
end
```

Registers.v

内部仕様

外部仕様