コンピュータ援用論理設計中間課題

4 年 情報工学科 2番

提出者： 4班 池内 隆一郎

提出締め切り： 平成28年6月24日（金）
提出日： 平成28年 6月24日（金）

第1章 課題内容

課題内容は、以下に指定する 8bit の CPU を作成し考察することである。 CPU の仕様について表 1 に CPU の命令表を示す。この仕様通りに 8bitCPU の内部には、ALU、デコーダ、マルチプレクサ 1、マルチプレクサ 2 を作成しそれを一つの回路にまとめ、テストを行い正しく動作しているか確認する。最終的な入出力仕様は以下の通りである。

入力

- Ain, Bin, Cin, Din (8bit) : 各レジスタの出力から与えられる入力
- Carryin (1bit) : Carry フラグから与えられる入力
- op (4bit) : プログラムカウンタから与えられる機械語

出力

- Aout, Bout, Cout, Dout (8bit) : 各レジスタへの出力
- Carryout (1bit) : 加算器の桁上げの有無

表 1: CPU 命令表

機械語	命令	op	ニーモック
0000	ADD	A, B	A <= A + B
0001	ADC	A, B	A <= A + B + Carry
0010	SUB	A, B	A <= A - B
0011	MUL	A, B	A <= A * B
0100	AND	A, B	A <= A and B
0101	OR	A, B	A <= A or B
0110	NOT	A	A <= not A
0111	XOR	A, B	A <= A xor B
1000	LD	A, B	A <= B
1001	LD	A, C	A <= C
1010	LD	A, D	A <= D
1011	LD	B, A	B <= A
1100	LD	B, C	B <= C
1101	LD	B, D	B <= D
1110	LD	C, A	C <= A
1111	LD	D, A	D <= A

第2章 ALU

ALU の verilog モジュールとテストのソースコード、alu_p44.v、alu_44_tb.v と、実行結果のタイミングチャートをソースコード 1, ソースコード 2、図 1 に示す。

```
module alu_pp44(Ain, Bin, Carryin, op, alu_enabled, Carryout, alu_out);
   input [7:0] Ain, Bin;
   input Carryin;
   input [2:0] op;
   input alu_enabled;
   output Carryout;
   output [7:0] alu_out;

   function [7:0] operate;
      input [7:0] Ain;
      input [7:0] Bin;
      input [2:0] op;
      input Carryin;
      input alu_enabled;
      begin
         if (alu_enabled == 1) begin
            case(op)
               3'b000: operate = Ain + Bin;
               3'b001: operate = Ain + Bin + Carryin;
               3'b010: operate = Ain - Bin;
               3'b011: operate = Ain * Bin;
               3'b100: operate = Ain & Bin;
               3'b101: operate = Ain | Bin;
               3'b110: operate = ~Ain;
               3'b111: operate = Ain ^ Bin;
            endcase
         end else operate = Ain;
      end
   endfunction
   function carry;
      input [7:0] Ain;
      input [7:0] Bin;
      input Carryin;
      input [2:0] op;
      begin
         if (alu_enabled == 1 && op == 3'b001) carry = (Ain + Bin
            + Carryin > 8'h80) ? 0 : 1;
      end
   endfunction
   assign alu_out = operate(Ain, Bin, op, Carryin, alu_enabled);
   assign Carryout = carry(Ain, Bin, Carryin, op);
endmodule
```

ソースコード 1 alu_pp44.v

```
module alu_pp44_tb;

    // Inputs
    reg [7:0] Ain;
    reg [7:0] Bin;
    reg Carryin;
    reg [2:0] op;
    reg alu_enabled;

    // Outputs
    wire Carryout;
    wire [7:0] alu_out;

    // Instantiate the Unit Under Test (UUT)
    alu_pp44 uut (
        .Ain(Ain),
        .Bin(Bin),
        .Carryin(Carryin),
        .op(op),
        .alu_enabled(alu_enabled),
        .Carryout(Carryout),
        .alu_out(alu_out)
    );

    initial begin
        // Initialize Inputs
        Ain = 0;
        Bin = 0;
        Carryin = 0;
        op = 0;
        alu_enabled = 0;

        // Wait 100 ns for global reset to finish
        #20 Ain = 25;  Bin = 145; Carryin = 0; op = 6; alu_enabled = 0;

        #20 Ain = 251; Bin = 203; Carryin = 0; op = 0; alu_enabled = 1;
        #20 Ain = 21;  Bin = 45;  Carryin = 1; op = 0; alu_enabled = 1;
        #20 Ain = 248; Bin = 146; Carryin = 0; op = 0; alu_enabled = 1;

        #20 Ain = 31;  Bin = 96;  Carryin = 1; op = 1; alu_enabled = 1;
        #20 Ain = 57;  Bin = 211; Carryin = 1; op = 1; alu_enabled = 1;
        #20 Ain = 78;  Bin = 160; Carryin = 0; op = 1; alu_enabled = 1;

        #20 Ain = 13;  Bin = 24;  Carryin = 0; op = 2; alu_enabled = 1;
        #20 Ain = 189; Bin = 50;  Carryin = 1; op = 2; alu_enabled = 1;
        #20 Ain = 123; Bin = 26;  Carryin = 0; op = 2; alu_enabled = 1;

        #20 Ain = 117; Bin = 108; Carryin = 1; op = 3; alu_enabled = 1;
        #20 Ain = 126; Bin = 4;   Carryin = 0; op = 3; alu_enabled = 1;
        #20 Ain = 39;  Bin = 226; Carryin = 0; op = 3; alu_enabled = 1;

        #20 Ain = 243; Bin = 87;  Carryin = 0; op = 4; alu_enabled = 1;
        #20 Ain = 84;  Bin = 16;  Carryin = 1; op = 4; alu_enabled = 1;
        #20 Ain = 156; Bin = 53;  Carryin = 1; op = 4; alu_enabled = 1;

        #20 Ain = 222; Bin = 237; Carryin = 0; op = 5; alu_enabled = 1;
        #20 Ain = 98;  Bin = 44;  Carryin = 1; op = 5; alu_enabled = 1;
        #20 Ain = 190; Bin = 24;  Carryin = 1; op = 5; alu_enabled = 1;
```

```
        #20 Ain = 55;  Bin = 200; Carryin = 0; op = 6; alu_enabled = 1;
        #20 Ain = 186; Bin = 32;  Carryin = 1; op = 6; alu_enabled = 1;
        #20 Ain = 101; Bin = 57;  Carryin = 1; op = 6; alu_enabled = 1;

        #20 Ain = 252; Bin = 234; Carryin = 0; op = 7; alu_enabled = 1;
        #20 Ain = 11;  Bin = 107; Carryin = 1; op = 7; alu_enabled = 1;
        #20 Ain = 188; Bin = 230; Carryin = 0; op = 7; alu_enabled = 1;

        #20 $finish;
    end
endmodule
```

ソースコード 2 alu_pp44_tb.v

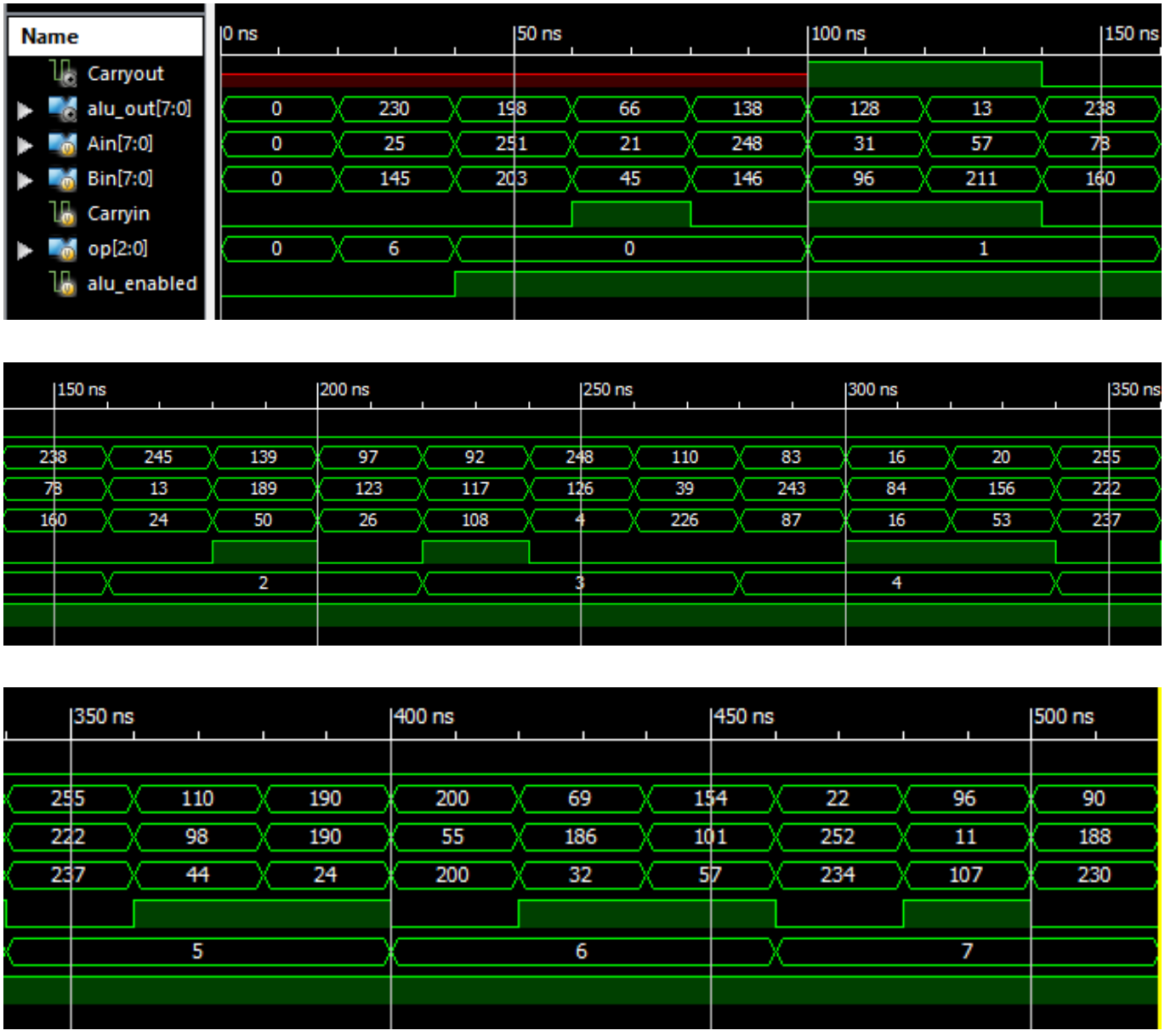


図 1 alu_pp44 のタイミングチャート

図 1 の実行結果をまとめ、ALU の実行結果の入出力表を表 2 に示す。

表 2: alu_p44 の入出力表

入力					出力	
Ain	Bin	Carryin	op	alu_enabled	alu_out	Carryout
25	145	0	6	0	230	不定
251	203	0	0	1	198	不定
21	45	1	0	1	66	不定
248	146	0	0	1	138	不定
31	96	1	1	1	128	1
57	211	1	1	1	13	1
78	160	0	1	1	238	0
13	24	0	2	1	245	0
189	50	1	2	1	139	0
123	26	0	2	1	97	0
117	108	1	3	1	92	0
126	4	0	3	1	248	0
39	226	0	3	1	110	0
243	87	0	4	1	83	0
84	16	1	4	1	16	0
156	53	1	4	1	20	0
222	237	0	5	1	255	0
98	44	1	5	1	110	0
190	24	1	5	1	190	0
55	200	0	6	1	200	0
186	32	1	6	1	69	0
101	57	1	6	1	154	0
252	234	0	7	1	22	0
11	107	1	7	1	96	0
188	230	0	7	1	90	0

ソースコード 1, 2, 図 1, 表 2 について、alu_enabled 端子と ALU が正しく動作していることについて説明する。

- alu_enabled 端子: 全体の回路で 1 章の表 1 の命令表の機械語 1001 が入力された場合、ALU の opsel には 001 が入力される。ここでもし alu_enabled 端子がなければ、Carryout を書き換えてしまい前の値を正常に受け継げない。機械語 0001 と 1001 を判別するために、ソースコード 1 のように alu_enabled 端子を作成しこれにより、Carryout 端子が正常に前の値をそのまま使うように設計した。また、これにより、ALU を使わない場合 (LD 命令を使う場合) に複雑な case 文を通らなくて済むので、性能の向上を期待することができる。

- ALU が正しく動作しているか: 表 2 の実行結果より、以下の式が成り立つ。また Carryout は前の値を正しく受け継いでいるので ALU は正しく動作している。

$$
\begin{aligned}
alu\_out &= 25 = \overline{00011001} = 11100110 = 230 \\
alu\_out &= 251 + 203 = 11111011 + 11001011 = (1)\,11000110 = 198 \\
alu\_out &= 21 + 45 = 66 \\
alu\_out &= 248 + 146 = 11111000 + 10010010 = (1)\,10001010 = 138 \\
alu\_out &= 31 + 96 + 1 == 128 \\
alu\_out &= 57 + 211 = 111001 + 11010011 = 00001100 = 100001100 = 12 \\
alu\_out &= 78 + 160 = 238 \\
alu\_out &= 13 - 24 = 100000000 - 00001011 = 11110101 = 245 \\
alu\_out &= 189 - 50 = 139 \\
alu\_out &= 123 - 26 = 97 \\
alu\_out &= 117 * 108 = 1110101 * 1101100 = (11000)\,101011100 = 92 \\
alu\_out &= 126 * 4 = 1111110 * 100 = (1))\,11111000 = 248 \\
alu\_out &= 39 * 226 = 100111 * 11100010 = (10001))\,001101110 = 110 \\
alu\_out &= 243and87 = 11110011and1010111 = 83 \\
alu\_out &= 84and16 = 1010100and10000 = 16 \\
alu\_out &= 156and53 = 100111004and110101 = 10100 = 20 \\
alu\_out &= 222or237 = 11011110or11101101 = 11111111 = 255 \\
alu\_out &= 98or44 = 1100010or101100 = 1101110 = 110 \\
alu\_out &= 190or24 = 10111110or11000 = 10111110 = 190 \\
alu\_out &= not55 = not110111 = 11001000 = 200 \\
alu\_out &= not186 = not10111010 = 1000101 = 69 \\
alu\_out &= not101 = not1100101 = 10011010 = 154 \\
alu\_out &= 252xor234 = 11111100xor11101010 = 10110 = 22 \\
alu\_out &= 1xor107 = 1xor1101011 = 1100000 = 96 \\
alu\_out &= 188xor230 = 10111100xor11100110 = 1011010 = 90
\end{aligned}
$$

第3章 デコーダ

デコーダの verilog モジュールとテストのソースコード、op_decode.v、op_decode_tb.v と、実行結果のタイミングチャートをソースコード 3, ソースコード 4、図 2 に示す。

```
module op_decode(op, op_sel, reg_sel1, reg_sel2, alu_enabled);
    input[3:0] op;

    output[2:0] op_sel;
    output[1:0] reg_sel1, reg_sel2;
    output alu_enabled;

    reg[2:0] op_sel;
    reg[1:0] reg_sel1, reg_sel2;
    reg alu_enabled;

    always @(op) begin
        if (op < 4'b1000) begin
                op_sel <= op;
                reg_sel1 <= 2'b11;
                reg_sel2 <= 2'b10;
                alu_enabled <= 1;
        end else begin
            op_sel <= 0;
            alu_enabled <= 0;
            if (op < 4'b1011) reg_sel1 <= op;
            else begin
                reg_sel1 <= 2'b11;
                if (op < 4'b1110) reg_sel2 <= op;
                else reg_sel2 <= 2'b10;
            end
        end
    end

endmodule
```

ソースコード 3 op_decode.v

```
module op_decode_tb;

    // Inputs
    reg [3:0] op;

    // Outputs
    wire [2:0] op_sel;
    wire [1:0] reg_sel1;
    wire [1:0] reg_sel2;
    wire alu_enabled;

    // Instantiate the Unit Under Test (UUT)
    op_decode uut (
        .op(op),
        .op_sel(op_sel),
        .reg_sel1(reg_sel1),
        .reg_sel2(reg_sel2),
        .alu_enabled(alu_enabled)
    );

    initial begin
        // Initialize Inputs
        op = 0;

        // Wait 100 ns for global reset to finish
        #50 op = 4'b0000;
        #50 op = 4'b0001;
        #50 op = 4'b0010;
        #50 op = 4'b0011;

        #50 op = 4'b0100;
        #50 op = 4'b0101;
        #50 op = 4'b0110;
        #50 op = 4'b0111;

        #50 op = 4'b1000;
        #50 op = 4'b1001;
        #50 op = 4'b1010;
        #50 op = 4'b1011;

        #50 op = 4'b1100;
        #50 op = 4'b1101;
        #50 op = 4'b1110;
        #50 op = 4'b1111;

        #50 $finish;

        // Add stimulus here

    end

endmodule
```

ソースコード 4 op_decode_tb.v

図 2 op_decode のタイミングチャート

図 2 の実行結果をまとめ、op_decode の実行結果の入出力表を表 2 に示す。

表 3: op_decode の入出力表

入力	出力			
op	op_sel	reg_sel1	reg_sel2	alu_enabled
0000	0	3	2	1
0001	1	3	2	1
0010	2	3	2	1
0011	3	3	2	1
0100	4	3	2	1
0101	5	3	2	1
0110	6	3	2	1
0111	7	3	2	1
1000	0	0	2	0
1001	1	1	2	0
1010	2	2	2	0
1011	3	3	3	0
1100	4	3	0	0
1101	5	3	1	0
1110	6	3	2	0
1111	7	3	2	0

ソースコード 3, 4, 図 2, 表 3 より、デコーダ (op_decode) が正しく動作していることについて説明する。第 2 章 ALU で説明した通り、ALU を有効かするかどうかを判定しなければ、Carryout が正しく動作しないため、alu_enabled 端子を出力端子として作成した。この端子は、命令 (op) の最上位 bit が 0 から始まるのみ 1 を出力し ALU を有効化する。op_sel には、下位 3bit を出力することにより、ALU にどの演算かを伝える。また、reg_sel1, 2 には、不定を出力させないために各マルチプレクサーを使わない命令の場合は、そのまま Ain, Bin を、Aout, Bout, へ出力できるようにした。マルチプレクサー 4(reg_sel1) では 3、マルチプレクサー 3(reg_sel2) では 2 を出力することにより、各マルチプレクサを使わない場合に入力をそのまま返すことができる。転送命令 (LD) の場合は 下位 2bit を出力することによりどこから転送するかを決定する。表 2 より命令 (op) が上位 1bit が 1 の場合は、演算命令なので alu_enabled 端子を High にし、下位 3it を op_sel へ出力し、命令 (op) がマルチプレクサを使う場合は使うマルチプレクサに reg_sel に下位 2bit を出力しているので、デコーダの実行結果は正しく動作していることが確認できる。

第4章 マルチプレクサー

マルチプレクサ mux3, mux4 の verilog モジュールとテストのソースコード、mux3.v、mux3_tb.v、mux4.v、mux4_tb.v、それぞれの実行結果のタイミングチャートをソースコード 5, ソースコード 6、ソースコード 7, ソースコード 8、図 3, 図 4 に示す。

```
module mux3(Ain, Bin, Cin, Din, reg_sel, Bout);
    input[7:0] Ain, Bin, Cin, Din;
    input[1:0] reg_sel;

    output[7:0] Bout;

    function [7:0] select;
        input [7:0] Ain, Bin, Cin, Din;
        input [1:0] reg_sel;
        begin
            case (reg_sel)
                2'b11: select = Ain;
                2'b00: select = Cin;
                2'b01: select = Din;
                2'b10: select = Bin;
            endcase
        end
    endfunction

    assign Bout = select(Ain, Bin, Cin, Din, reg_sel);
endmodule
```

ソースコード 5 mux3.v

```
module mux3_tb;

    // Inputs
    reg [7:0] Ain;
    reg [7:0] Bin;
    reg [7:0] Cin;
    reg [7:0] Din;
    reg [1:0] reg_sel;

    // Outputs
    wire [7:0] Bout;

    // Instantiate the Unit Under Test (UUT)
    mux3 uut (
        .Ain(Ain),
        .Bin(Bin),
        .Cin(Cin),
        .Din(Din),
        .reg_sel(reg_sel),
```

```
        .Bout(Bout)
    );

    initial begin
        // Initialize Inputs
        Ain = 0;
        Bin = 0;
        Cin = 0;
        Din = 0;
        reg_sel = 0;

        // Wait 100 ns for global reset to finish
        #20 Ain = 12;  Bin = 114; Cin = 18;  Din = 27;  reg_sel = 0;
        #20 Ain = 196; Bin = 152; Cin = 240; Din = 221; reg_sel = 0;
        #20 Ain = 253; Bin = 241; Cin = 149; Din = 171; reg_sel = 0;

        #20 Ain = 26;  Bin = 242; Cin = 75;  Din = 228; reg_sel = 1;
        #20 Ain = 108; Bin = 90;  Cin = 240; Din = 143; reg_sel = 1;
        #20 Ain = 4;   Bin = 195; Cin = 61;  Din = 148; reg_sel = 1;

        #20 Ain = 201; Bin = 182; Cin = 158; Din = 203; reg_sel = 2;
        #20 Ain = 26;  Bin = 196; Cin = 101; Din = 233; reg_sel = 2;
        #20 Ain = 244; Bin = 182; Cin = 24;  Din = 240; reg_sel = 2;

        #20 Ain = 178; Bin = 221; Cin = 111; Din = 31;  reg_sel = 3;
        #20 Ain = 209; Bin = 150; Cin = 173; Din = 137; reg_sel = 3;
        #20 Ain = 71;  Bin = 173; Cin = 160; Din = 86;  reg_sel = 3;

        #20 $finish;
        // Add stimulus here

    end

endmodule
```

ソースコード 6 mux3_tb.v

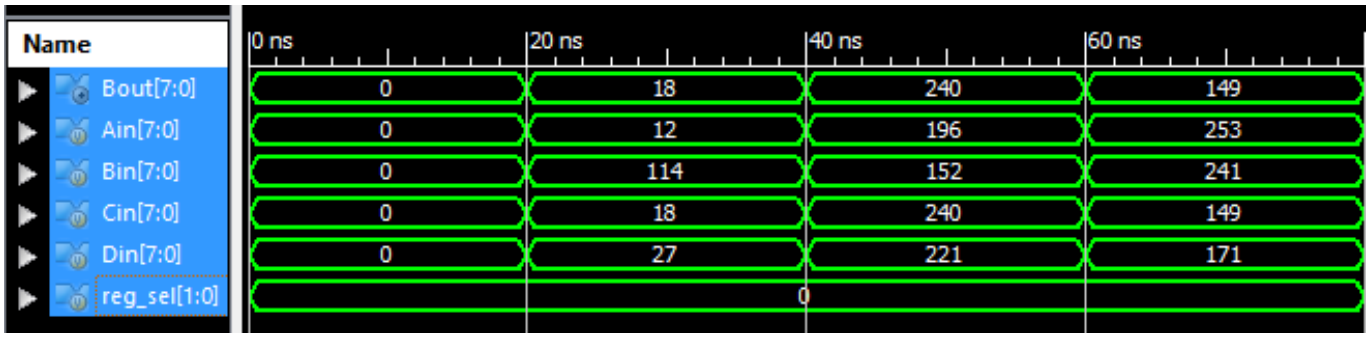
```
module mux4(Ain, Bin, Cin, Din, reg_sel, Aout);
    input[7:0] Ain, Bin, Cin, Din;
    input[1:0] reg_sel;

    output[7:0] Aout;

    function [7:0] select;
        input [7:0] Ain, Bin, Cin, Din;
        input [1:0] reg_sel;
        begin
            case (reg_sel)
                2'b00: select = Bin;
                2'b01: select = Cin;
                2'b10: select = Din;
                2'b11: select = Ain;
            endcase
        end
    endfunction

    assign Aout = select(Ain, Bin, Cin, Din, reg_sel);
endmodule
```

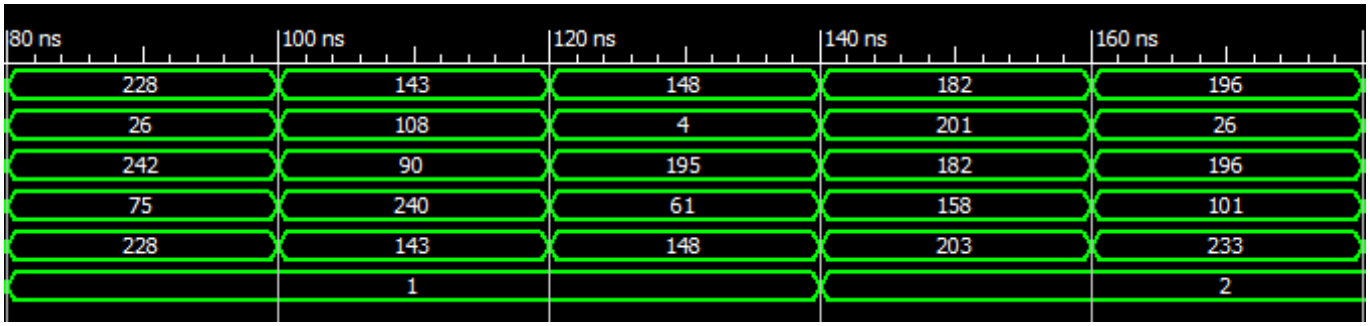
ソースコード 7 mux4.v

```
module mux4_tb;

    // Inputs
    reg [7:0] Ain;
    reg [7:0] Bin;
    reg [7:0] Cin;
    reg [7:0] Din;
    reg [1:0] reg_sel;

    // Outputs
    wire [7:0] Aout;

    // Instantiate the Unit Under Test (UUT)
    mux4 uut (
        .Ain(Ain),
        .Bin(Bin),
        .Cin(Cin),
        .Din(Din),
        .reg_sel(reg_sel),
        .Aout(Aout)
    );

    initial begin
        // Initialize Inputs
        Ain = 0;
        Bin = 0;
        Cin = 0;
        Din = 0;
        reg_sel = 0;

        // Wait 100 ns for global reset to finish
```

```
        #20 Ain = 12;  Bin = 114; Cin = 18;  Din = 27;  reg_sel = 0;
        #20 Ain = 196; Bin = 152; Cin = 240; Din = 221; reg_sel = 0;
        #20 Ain = 253; Bin = 241; Cin = 149; Din = 171; reg_sel = 0;

        #20 Ain = 26;  Bin = 242; Cin = 75;  Din = 228; reg_sel = 1;
        #20 Ain = 108; Bin = 90;  Cin = 240; Din = 143; reg_sel = 1;
        #20 Ain = 4;   Bin = 195; Cin = 61;  Din = 148; reg_sel = 1;

        #20 Ain = 201; Bin = 182; Cin = 158; Din = 203; reg_sel = 2;
        #20 Ain = 26;  Bin = 196; Cin = 101; Din = 233; reg_sel = 2;
        #20 Ain = 244; Bin = 182; Cin = 24;  Din = 240; reg_sel = 2;

        #20 Ain = 178; Bin = 221; Cin = 111; Din = 31;  reg_sel = 3;
        #20 Ain = 209; Bin = 150; Cin = 173; Din = 137; reg_sel = 3;
        #20 Ain = 71;  Bin = 173; Cin = 160; Din = 86;  reg_sel = 3;

        #20 $finish;
        // Add stimulus here

    end

endmodule
```

ソースコード 8 mux4_tb.v





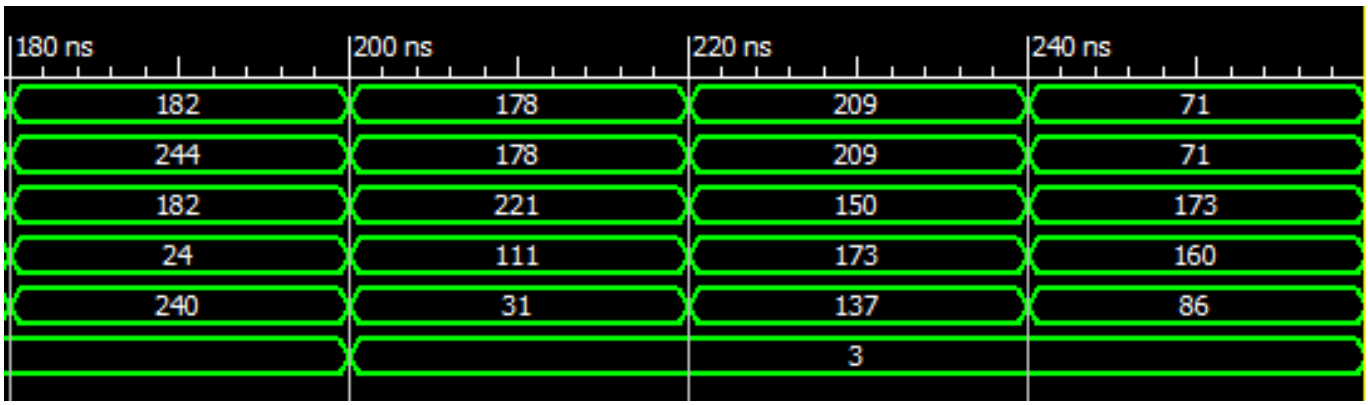


図 3 mux3 のタイミングチャート

図 4 mux4 のタイミングチャート

図 3 の実行結果をまとめ、mux3 の実行結果の入出力表を表 2 に示す。

表 4: mux3 実行結果の入出力表

入力					出力
Ain	Bin	Cin	Din	reg_sel	Bout
12	114	18	27	0	18
196	152	240	221	0	240
253	241	149	171	0	149
26	242	75	228	1	228
108	90	240	143	1	143
4	195	61	148	1	148
201	182	158	203	2	182
26	196	101	233	2	196
244	182	24	240	2	182
178	221	111	31	3	178
209	150	173	137	3	209
71	173	160	86	3	71

図 4 の実行結果をまとめ、mux4 の実行結果の入出力表を表 2 に示す。

表 5: mux4 実行結果の入出力表

入力					出力
Ain	Bin	Cin	Din	reg_sel	Aout
12	114	18	27	0	114
196	152	240	221	0	152
253	241	149	171	0	241
26	242	75	228	1	75
108	90	240	143	1	240
4	195	61	148	1	61
201	182	158	203	2	203
26	196	101	233	2	233
244	182	24	240	2	240
178	221	111	31	3	178
209	150	173	137	3	209
71	173	160	86	3	71

ソースコード 5, 6, 7, 8 , 図 3, 図 4, 表 4, 表 5 より、マルチプレクサー (mux3, mux4) が正しく動作していることについて説明する。mux4 は第 1 章表 1 の CPU 命令表の 1000 から 1010, mux3 は 1011 から 1100 の範囲で Aout, Bout に命令表通りに Ain, Bin, Cin, Din を出力する。ここで、Aout, Bout の出力が不定になってしまうのを避けるために mux4 では reg_sel が 11 のとき、mux3 では reg_sel が 10 のときに Ain, Bin を Aout, Bout にそのまま出力する。表 4 より、mux3 は reg_sel が、0 のときは Cin の値、1 のときは Din の値、2 のときは Bin の値、3 のときは Bin の値が Aout に出力されていることが確認することができる。表 5 より、mux4 は reg_sel が、0 のときは Bin の値、1 のときは Cin の値、2 のときは Din の値、3 のときは Ain の値が Aout に出力されていることが確認することができる。これよりマルチプレクサーは正しく動作していることが確認することができる。

第5章 8bitCPU