2021/4/26

コンピュータアーキテクチャ論 Ex04

S1260027
Shunsuke Onuki

課題 4-1: ALU のシミュレーション
課題 4-2: レジスタファイルのシミュレーション

課題 4-1: ALU のシミュレーション

ALU 全体が正しく動作しているのかを確認するため、シミュレーション結果と期待値を比べる。以下の表を参考にシミュレーションを行う。

AND: ALU制御入力: 0000

A	B	ゼロ判定期待値	結果期待値
0000 0000	0000 0000	1	0
0F0F 0F0F	F0F0 F0F0	1	0
F0F0 F0F0	0F0F 0F0F	1	0
FFFF FFFF	FFFF FFFF	0	FFFF FFFF
D8A3 B8D4	63B9 D6F2	0	40A1 90D0

加算: ALU制御入力: 0010

A	B	ゼロ判定期待値	結果期待値
0000 0000	0000 0000	1	0
0F0F 0F0F	F0F0 F0F0	0	FFFF FFFF
7FFF FFFF	0000 0001	0	8000 0000
FFFF FFFF	0000 0001	1	0
D8A3 B8D4	63B9 D6F2	0	3C5D 8FC6

OR: ALU制御入力: 0001

A	B	ゼロ判定期待値	結果期待値
0000 0000	0000 0000	1	0
0F0F 0F0F	F0F0 F0F0	0	FFFF FFFF
F0F0 F0F0	0F0F 0F0F	0	FFFF FFFF
FFFF FFFF	FFFF FFFF	0	FFFF FFFF
D8A3 B8D4	63B9 D6F2	0	FBBB FEF6

減算: ALU制御入力: 0110

A	B	ゼロ判定期待値	結果期待値
0000 0000	0000 0000	1	0
0000 0000	0000 0001	0	FFFF FFFF
0000 0001	8000 0001	0	8000 0000
0000 0001	0000 0001	1	0
D8A3 B8D4	63B9 D6F2	0	74E9 E1E2

Set on less than: ALU制御入力: 0111

A	B	ゼロ判定期待値	結果期待値
0000 0000	0000 0000	1	0
0000 0000	0000 0001	0	1
0000 0001	8000 0001	0	1
0000 0001	0000 0001	1	0
D8A3 B8D4	63B9 D6F2	1	0

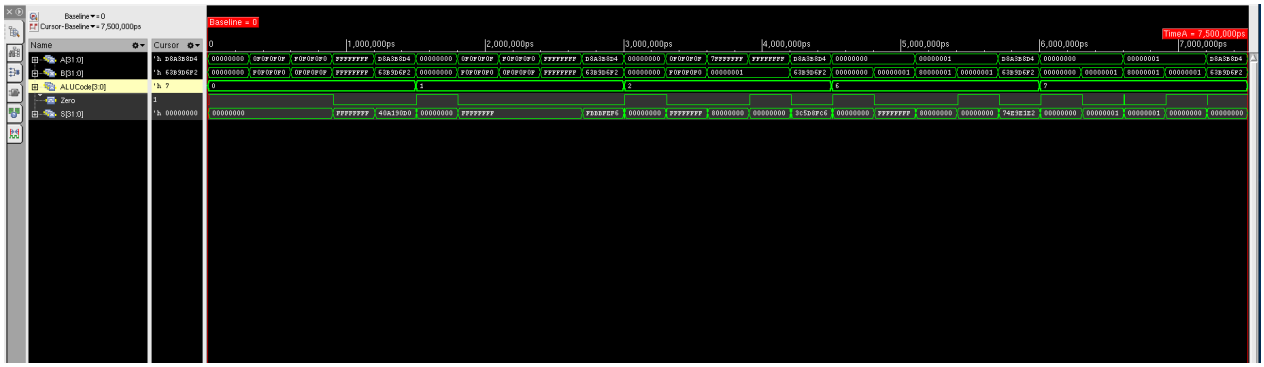
Testfixture.verilog

```
// Verilog stimulus file.
// Please do not create a mo

// Default verilog stimulus.

initial
begin

   A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000000;

   ALUCode[3:0] = 4'b0000;
#300
   A[31:0] = 32'h0F0F0F0F;

   B[31:0] = 32'hF0F0F0F0;

#300
   A[31:0] = 32'hF0F0F0F0;

   B[31:0] = 32'h0F0F0F0F;

#300
   A[31:0] = 32'hFFFFFFFF;

   B[31:0] = 32'hFFFFFFFF;

#300
   A[31:0] = 32'hD8A3B8D4;

   B[31:0] = 32'h63B9D6F2;

#300
    A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000000;

   ALUCode[3:0] = 4'b0001;
#300
   A[31:0] = 32'h0F0F0F0F;

   B[31:0] = 32'hF0F0F0F0;

#300
   A[31:0] = 32'hF0F0F0F0;

   B[31:0] = 32'h0F0F0F0F;

#300
   A[31:0] = 32'hFFFFFFFF;

   B[31:0] = 32'hFFFFFFFF;

#300
   A[31:0] = 32'hD8A3B8D4;

   B[31:0] = 32'h63B9D6F2;

#300
   A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000000;

   ALUCode[3:0] = 4'b0010;
#300
   A[31:0] = 32'h0F0F0F0F;

   B[31:0] = 32'hF0F0F0F0;

#300
   A[31:0] = 32'h7FFFFFFF;

   B[31:0] = 32'h00000001;

#300
   A[31:0] = 32'hFFFFFFFF;

   B[31:0] = 32'h00000001;

   #300
   A[31:0] = 32'hD8A3B8D4;

   B[31:0] = 32'h63B9D6F2;

#300
   A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000000;

   ALUCode[3:0] = 4'b0110;
#300
   A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000001;

#300
   A[31:0] = 32'h00000001;

   B[31:0] = 32'h80000001;

#300
   A[31:0] = 32'h00000001;

   B[31:0] = 32'h00000001;

   #300
   A[31:0] = 32'hD8A3B8D4;

   B[31:0] = 32'h63B9D6F2;

#300
```

```
   A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000000;

   ALUCode[3:0] = 4'b0111;
#300
   A[31:0] = 32'h00000000;

   B[31:0] = 32'h00000001;

#300
   A[31:0] = 32'h00000001;

   B[31:0] = 32'h80000001;

#300
   A[31:0] = 32'h00000001;

   B[31:0] = 32'h00000001;

#300
   A[31:0] = 32'hD8A3B8D4;

   B[31:0] = 32'h63B9D6F2;

#300
$finish;
end
```

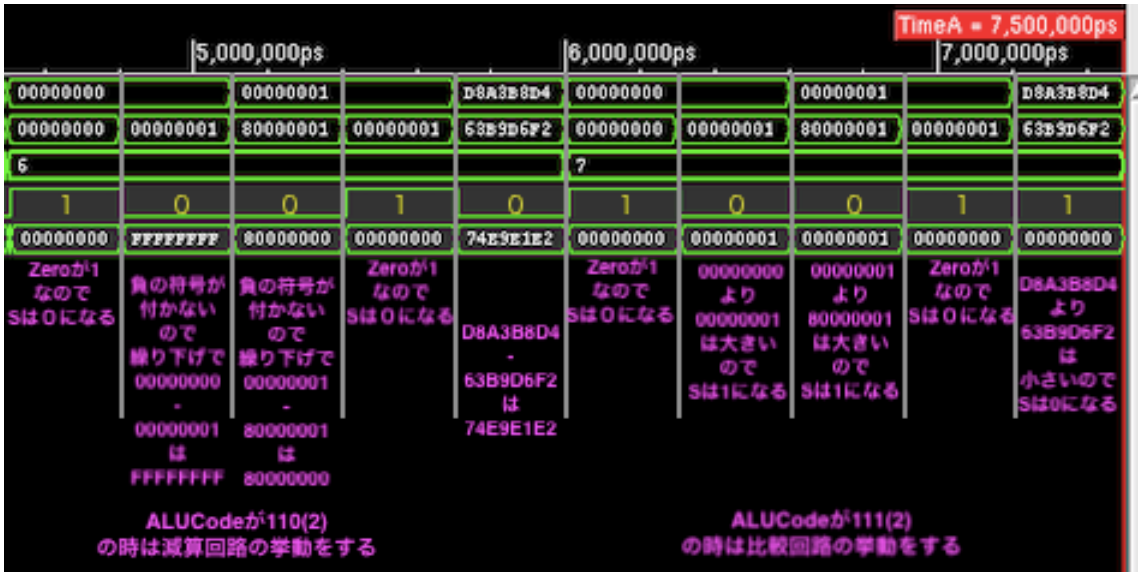
回路図

結果




Cursor-Baseline = 7,500,000ps
Name
Cursor
A[31:0]
B[31:0]
ALUCode[3:0]
Zero
S[31:0]
'h D8A3B8D4
'h 63B9D6F2
'h 7
1
'h 00000000
Baseline = 0
0
1,000,000ps
00000000
0F0F0F0F
F0F0F0F0
FFFFFFFF
D8A3B8D4
63B9D6F2
1
0
40A190D0
Zeroが0なのでSは0になる
Zeroが0なのでSは0になる
Zeroが0なのでSは0になる
FFFFFFFF and FFFFFFFF は FFFFFFFF
D8A3B8D4 and 63B9D6F2 は FBBBFEF6
ALUCodeが0の時はAND回路の挙動をする


2,000,000ps
3,000,000ps
4,000,000ps
00000000
0F0F0F0F
F0F0F0F0
FFFFFFFF
D8A3B8D4
63B9D6F2
1
2
7FFFFFFF
00000001
80000000
3c5D8FC6
FBBBFEF6
Zeroが1なのでSは0になる
0F0F0F0F or F0F0F0F0 は FFFFFFFF
F0F0F0F0 or 0F0F0F0F は FFFFFFFF
FFFFFFFF or FFFFFFFF は FFFFFFFF
D8A3B8D4 or 63B9D6F2 は FBBBFEF6
0F0F0F0F + F0F0F0F0 は FFFFFFFF
7FFFFFFF + 00000001 は 80000000
D8A3B8D4 + 63B9D6F2 は 3C5D8FC6
ALUCodeが1の時はOR回路の挙動をする
ALUCodeが10(2)の時は加算回路の挙動をする



理想値の表と見比べると正しい値が出力されていることがわかる。

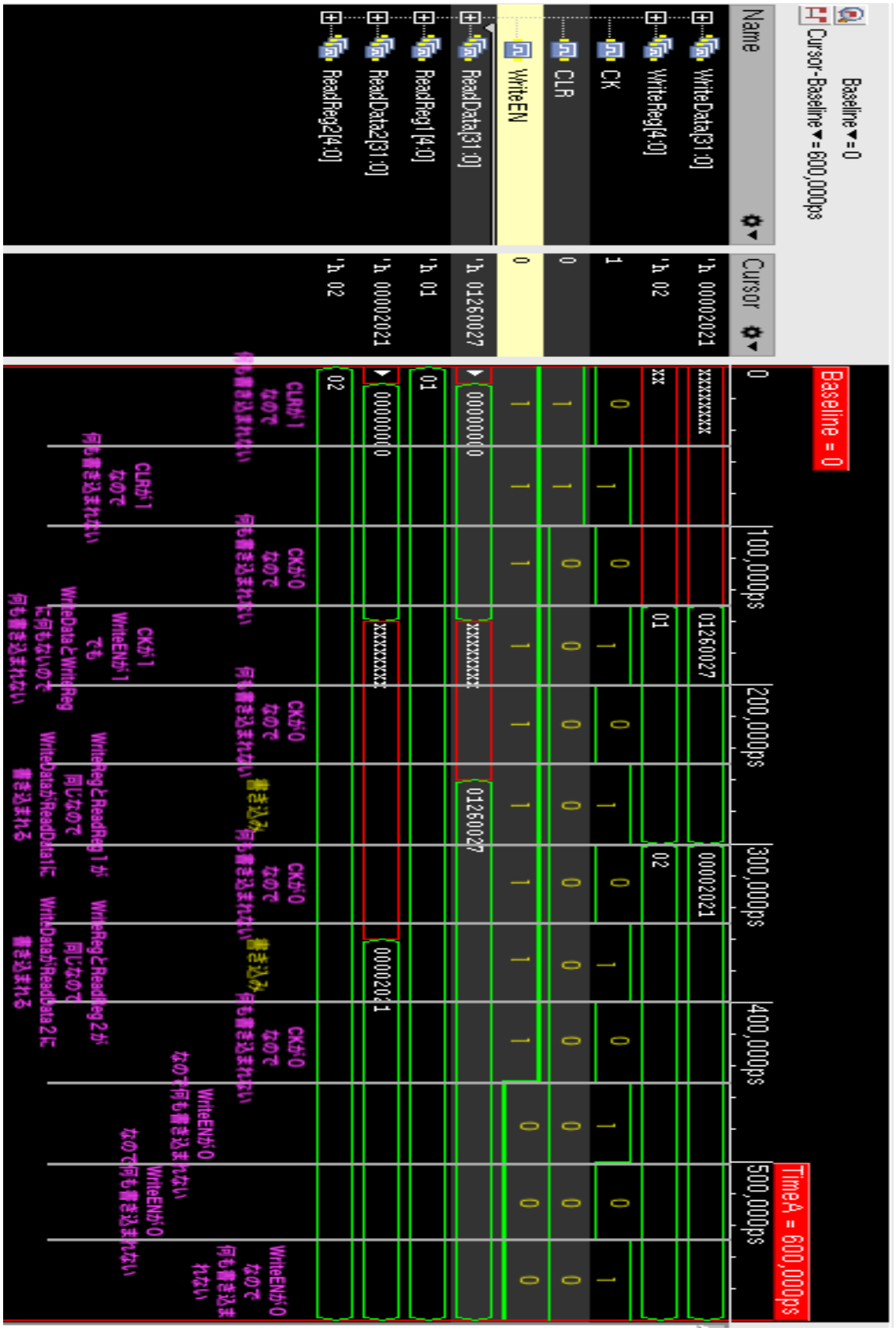
課題 4-2: レジスタファイルのシミュレーション
レジスタファイルの動作を確認するためテストベンチ(testfixture)を作成し、動作を確認する

Testfixture

```
// Verilog stimulus file.
// Please do not create a module in this file.


// Default verilog stimulus.

initial
begin

   CK = 1'b0;
   CLR = 1'b1;
   WriteEN = 1'b1;
   ReadReg1[4:0] = 5'b00001;

   ReadReg2[4:0] = 5'b00010;

   #100

   CLR = 1'b0;

   #50

   WriteData[31:0] = 32'b00000001001001100000000000100111;
   WriteReg[4:0] = 5'b00001;

   #150

   WriteReg[4:0] = 5'b00010;
   WriteData[31:0] = 32'b00000000000000000010000000100001;

   #150

   WriteEN = 1'b0;

   #150

   $finish;

end
   always #50 CK = ~CK;
```

回路図

結果


Baseline = 0
Cursor-Baseline = 600,000ps
Name
Cursor
WriteData[31:0]
WriteReg[4:0]
CK
CLR
WriteEN
ReadData[31:0]
ReadReg1[4:0]
ReadData2[31:0]
ReadReg2[4:0]
'h 00002021
'h 02
1
0
0
'h 01260027
'h 01
'h 00002021
'h 02
0
100,000ps
200,000ps
300,000ps
400,000ps
500,000ps
TimeA = 600,000ps
CLRが1なので何も書き込まれない
CKが0なので何も書き込まれない
CKが1でもWriteENが1でもWriteDataとWriteRegに何もないので何も書き込まれない
WriteRegとReadReg1が同じなのでWriteDataがReadData1に書き込まれる
WriteRegとReadReg2が同じなのでWriteDataがReadData2に書き込まれる
WriteENが0なので何も書き込まれない