Mark Mitri 02/19/2022
ECE201 Project 1

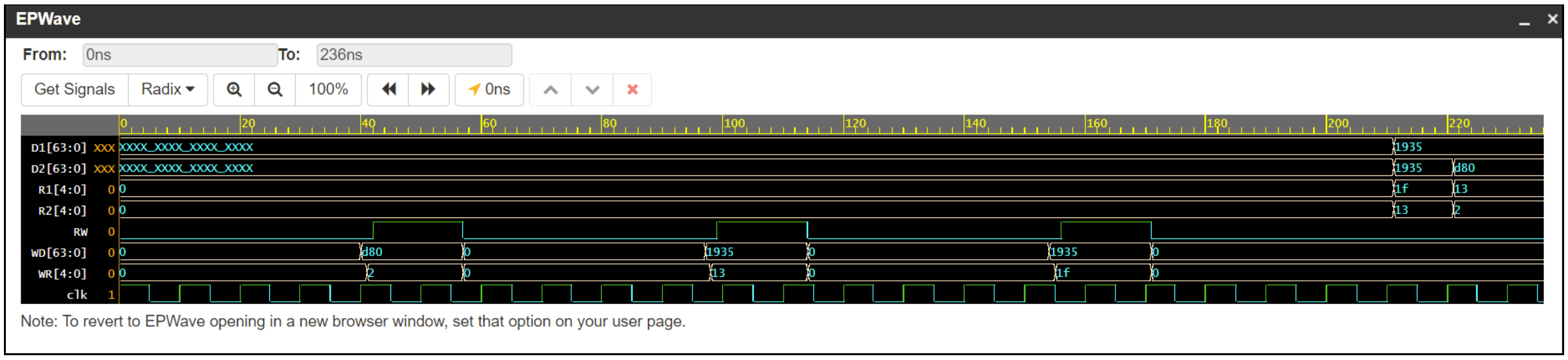
Part 1: Two-read, One-write Register file

32bit Register File

```
// Mark Mitri
// ECE 201
// Project 2
// Part 1: two-read one-write Register

module registerfile (Read1, Read2, WriteReg, WriteData, RegWrite, Data1, Data2, clk);

  input [4:0] Read1, Read2, WriteReg; // typo was making the read/write 6 wide instead
of 5 wide
  input [63:0] WriteData;
  input RegWrite, clk;
  output [63:0] Data1, Data2;

  reg [63:0] RF [31:0];

  assign Data1 = RF[Read1];
  assign Data2 = RF[Read2];

  initial $readmemh("file.txt",RF);

  always @(posedge clk)
  begin
    if(RegWrite)
      RF[WriteReg] <= WriteData;
  end
Endmodule
```

32bit Register File Testbench

```
// Mark Mitri
// ECE 201
// Project 2
// Part 1 Testbench

// inputs -> registers
// outputs -> wires

`timescale 1ns / 1ns

module twoRead_oneWrite_testbench();
  reg[4:0] R1, R2, WR;
  reg[63:0] WD;
  reg RW, clk;
  wire[63:0] D1, D2;

  registerfile u_dut(.Read1(R1), .Read2(R2), .WriteReg(WR), .WriteData(WD),
.RegWrite(RW), .Data1(D1), .Data2(D2), .clk(clk));

  always #5 clk = ~clk;

  initial
    begin
        $dumpfile("dump.vcd");
                $dumpvars;
    end

  initial
    begin
      clk = 1;
      RW = 0;
      R1 = 5'd0;
      R2 = 5'd0;
      WR = 5'd0;
      WD = 64'd0;

      #40 WD = 64'd3456;
      #1  WR = 5'd2;
      #1  RW = 1;
      #15 RW = 0;
          WD = 64'd0;
          WR = 5'd0;

      #40 WD = 64'd6453;
      #1  WR = 5'd19;
      #1  RW = 1;
      #15 RW = 0;
          WD = 64'd0;
          WR = 5'd0;

      #40 WD = 64'd6453;
      #1  WR = 5'd31;
      #1  RW = 1;
      #15 RW = 0;
          WD = 64'd0;
          WR = 5'd0;

      #40 R1 = 5'd31;
          R2 = 5'd19;
      #10 R1 = 5'd19;
          R2 = 5'd2;
      #15
    $finish;
    end
endmodule
```


EPWave

From: 0ns To: 236ns

Get Signals
Radix
100%
0ns
0
20
40
60
80
100
120
140
160
180
200
220
D1[63:0] xxx xxxx_xxxx_xxxx_xxxx 1935
D2[63:0] xxx xxxx_xxxx_xxxx_xxxx 1935 d80
R1[4:0] 0 0 1f 13
R2[4:0] 0 0 13 2
RW 0
WD[63:0] 0 0 d80 0 1935 0 1935 0
WR[4:0] 0 0 2 0 13 0 1f 0
clk 1

Note: To revert to EPWave opening in a new browser window, set that option on your user page.

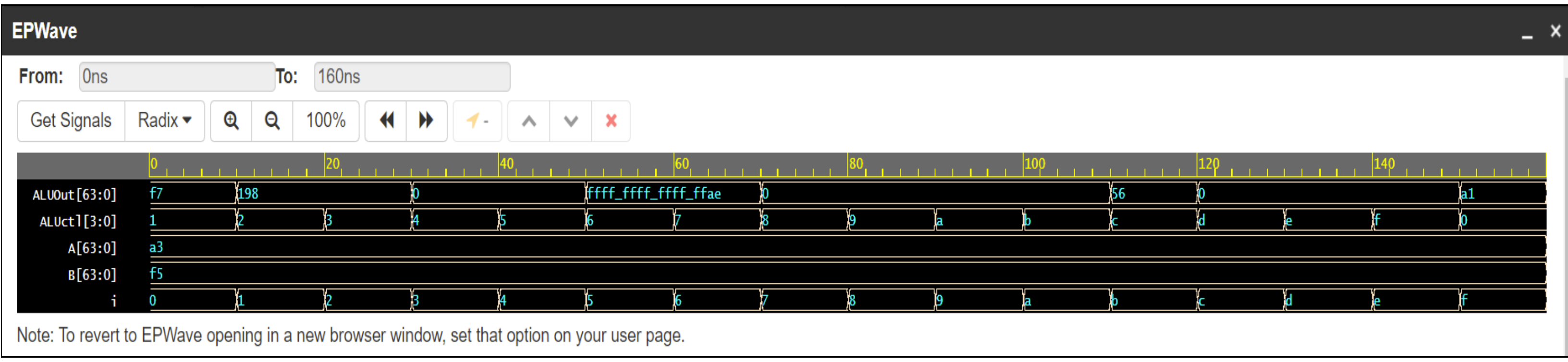
Part 2: 64bit ALU

ALU Design Code

```
// Mark Mitri
// ECE 201
// Project 2
// Part 2

module alu_behav(ALUctl, A, B, ALUOut, carryout, zero, overflow, negative);
  input [3:0] ALUctl;
  input [63:0] A,B;
  output reg [63:0] ALUOut;
  output carryout;
  output zero;
  output overflow;
  output negative;
  wire [8:0] tmp;
  assign tmp = {1'b0,A} + {1'b0,B};
  assign carryout = tmp[8];

  always @(ALUctl, A, B) begin
    case (ALUctl)
      4'b0000: ALUOut <= A & B; // AND
      4'b0001: ALUOut <= A | B; // ORR
      4'b0010: ALUOut <= A + B; // ADD
      4'b0011: ALUOut <= A + B; // ADDS (Add with flags
      4'b0110: ALUOut <= A - B; // SUB
      4'b0111: ALUOut <= A - B; // SUBS (Sub with flags
      4'b1001: ALUOut <= A << B; // LSL
      4'b1100: ALUOut <= A ^ B; // EOR
      default: ALUOut <= 0;
    endcase
  end

  assign overflow = ALUOut[63] ^ ALUOut[62];
  assign zero = (ALUOut == 0)? 1'b1 : 1'b0;
endmodule
```

ALU Testbench Code

```
// Mark Mitri
// ECE 201
// Project 2
// Part 2
// Testbench

`timescale 1ns / 1ns

module alu_behav_testbench;

 reg[63:0] A,B;
 reg[3:0] ALUctl;
 wire[63:0] ALUOut;
 wire carryout;
 wire zero;
 wire overflow;
 wire negative;
 integer i;
  alu_behav dut(.ALUctl(ALUctl),.A(A),.B(B),
.ALUOut(ALUOut),.carryout(carryout),.zero(zero),.overflow(overflow),.negative(negative));

 initial
   begin
     $dumpfile("dump.vcd");
     $dumpvars;
   end

 initial begin
   A = 63'b10100011;
   B = 63'b11110101;
   ALUctl = 4'b0000;
   for (i=0;i<=15;i=i+1) begin
     ALUctl = ALUctl + 4'b0001;
     #10;
   end;
   A = 63'b10100100;
   B = 63'b01011100;
 end
endmodule
```


EPWave
From: 0ns To: 160ns
Get Signals
Radix
100%
0
20
40
60
80
100
120
140
ALUOut[63:0] f7 198 0 ffff_ffff_ffff_ffae 0 56 0 a1
ALUctl[3:0] 1 2 3 4 5 6 7 8 9 a b c d e f 0
A[63:0] a3
B[63:0] f5
i 0 1 2 3 4 5 6 7 8 9 a b c d e f

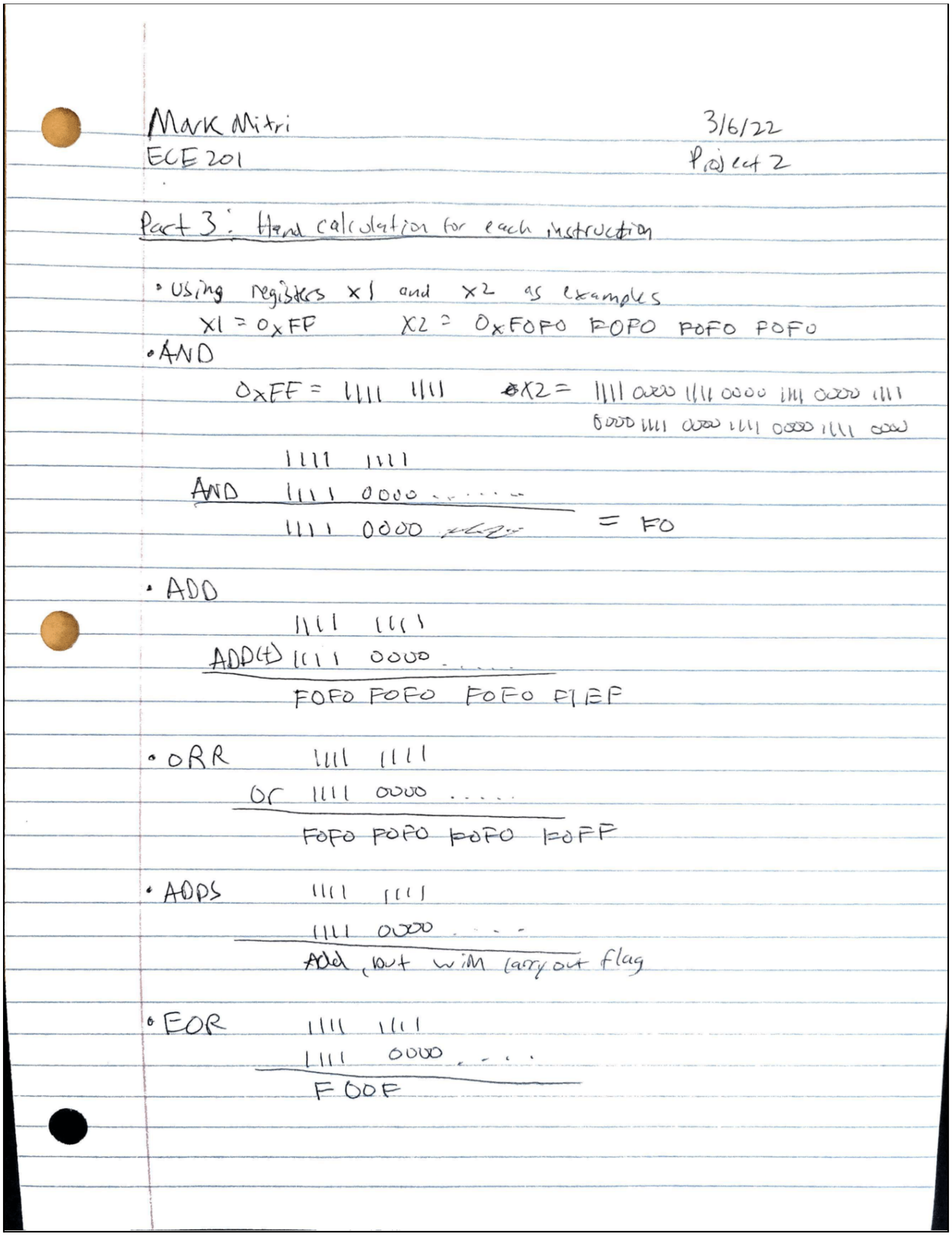
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Part 3: Datapath File

<u>Datapath Verilog Design Code</u>

```
// Mark Mitri
// ECE 201
// Project 2
// Part 3

module datapath(C, V, N, Z, RF_out,A,B,out,ALUSel,RF_add,Mux,RF_w,ALU_out,Mux_out,
clk);
  output C,V,N,Z // Flags
  output [7:0] RF_out;
  input [7:0] A,B,out;
  input [4:0] ALUSel, RF_add;
  input Mux, RF_w, clk;
  wire [7:0] ALU_out, Mux_out;

  aluBehav
alu_behav(.ALUctl(ALUSel),.A(A),.B(B),.ALUOut(ALU_out).carryout(C),.zero(Z),.overflow
(V),.negative(N));

  register_file
registerfile(.Read1(RF_out),.Read2(),.WriteReg(),.WriteData(),.RegWrite(),.Data1(),.Data2(),.
clk(clk));
endmodule
```

Hand Calculations for Datapath



Mark Mitri 3/6/22
ECE 201 Project 2

Part 3: Hand calculation for each instruction

• Using registers x1 and x2 as examples
X1 = 0xFF X2 = 0xF0F0 F0F0 F0F0 F0F0

• AND
0xFF = 1111 1111 X2 = 1111 0000 1111 0000 1111 0000 1111
0000 1111 0000 1111 0000 1111 0000

```
      1111 1111
AND   1111 0000 ......
      1111 0000          = F0
```

• ADD

```
        1111 1111
ADD(+)  1111 0000 .....
        F0F0 F0F0 F0F0 F1EF
```

• ORR

```
     1111 1111
or   1111 0000 .....
     F0F0 F0F0 F0F0 F0FF
```

• ADDS

```
     1111 1111
     1111 0000 ....
     Add, but with carryout flag
```

• EOR

```
     1111 1111
     1111 0000 ....
     F00F
```

```
SUB     1111 1111
        1111 0000 . . . .
        ------------------
        FOF OFOF OFDF 100F

LSL      1111 1111
         1111 0000 . . . .
        ------------------
         FF
```