Northeastern University

Department of Electrical and Computer Engineering

EECE2323: **Digital Systems Design Lab**

Lecturer: **Dr. Emad Aboelela**

TAs:
Ke Chen
Linbin Chen

Lab # 1-3: **8-Bit Adder, Partial Arithmetic Logic Unit, and Arithmetic Logic Unit**

Group # **???**:
Jin Hyeong Kim,
Timothy VanSlyke

Semester: Spring 2018
Date: February 10, 2018
Lab Session: Tuesday, 1:00PM
Lab Location: 9 Hayden Hall, Northeastern University,
Boston, MA 02115

Contents

1 Introduction

2 Design Approach

3 Results and Analysis

3.1 Design Simulation

3.1.1 8-Bit Adder

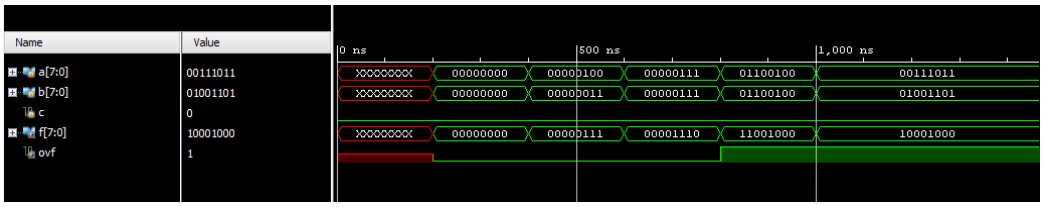

Figure 1: Simulation results for lab 1, **8-Bit Adder**.

3.1.2 Partial Arithmetic and Logic Unit

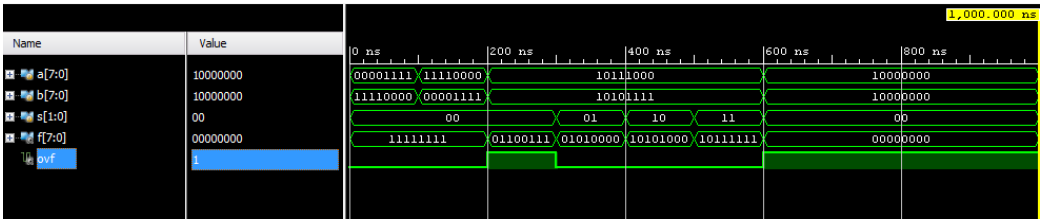

Figure 2: Simulation results for lab 2, **Partial Arithmetic and Logic Unit**.

3.1.3 Arithmetic and Logic Unit

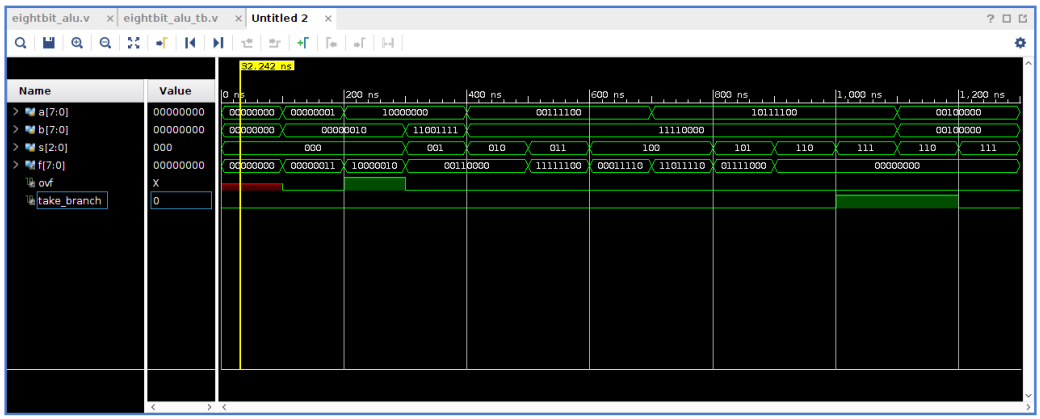

Figure 3: Simulation results for lab 3, **Arithmetic Logic Unit**.

3.2 Hardware Testing

3.2.1 8-Bit Adder

3.2.2 Partial Arithmetic and Logic Unit

3.2.3 Arithmetic and Logic Unit

4 Conclusions

Appendices

A 8-Bit Adder Module: `eightbit_adder.v`

```
module eightbit_palu(
    input signed [7:0] a,
    input signed [7:0] b,
    output [7:0] f,
    output ovf
  );
  // module-local registers
  reg [7:0] f_reg, ovf_reg;
  // continuous assignment to outputs from module-local registers
  assign f = f_reg;
  assign ovf = ovf_reg;
  always @(a, b) begin
    f_reg = a + b;
    // overflow check for addition
    ovf_reg = (s == 0) && (f[7] != a[7]) && (f[7] != b[7]);
  end
endmodule
```

Figure 4: `eightbit_adder.v` implementing an 8-bit adder in Verilog.

B Partual ALU Verilog Module: `eightbit_palu.v`

```
module eightbit_palu(
    input signed [7:0] a,
    input signed [7:0] b,
    input [1:0] s,
    output [7:0] f,
    output ovf
  );
  // module-local registers
  reg [7:0] f_reg, ovf_reg;
  // continuous assignment to outputs from module-local registers
  assign f = f_reg;
  assign ovf = ovf_reg;
  always @(a, b, s) begin
    // operations on 'f'
    case(s)
      0: f_reg = a + b;
      1: f_reg = ~b;
      2: f_reg = a & b;
      3: f_reg = a | b;
    endcase
    // overflow check for addition
    ovf_reg = (s == 0) && (f[7] != a[7]) && (f[7] != b[7]);
  end
endmodule
```

Figure 5: `eightbit_palu.v` implementing an 8-bit partial ALU in Verilog.

C ALU Verilog Module: `eightbit_alu.v`

```
module eightbit_alu(
    input signed [7:0] a,
    input signed [7:0] b,
    input [2:0] s,
    output [7:0] f,
    output ovf,
    output take_branch
  );
  // module-local registers
  reg [7:0] f_reg, ovf_reg, take_branch_reg;
  // continuous assignment to outputs from module-local registers
  assign f = f_reg;
  assign ovf = ovf_reg;
  assign take_branch = take_branch_reg;
  always @(a, b, s) begin
    // operations on 'f'
    case(s)
      0: f_reg = a + b;
      1: f_reg = ~b;
      2: f_reg = a & b;
      3: f_reg = a | b;
      4: f_reg = a >>> 1;
      5: f_reg = a << 1;
      6: f_reg = 0;
      7: f_reg = 0;
    endcase
    // overflow check for addition
    ovf_reg = (s == 0) && (f[7] != a[7]) && (f[7] != b[7]);
    // 'take_branch' special cases
    take_branch_reg = ((s == 6) && (a == b)) || ((s == 7) && (a != b));
  end
endmodule
```

Figure 6: `eightbit_alu.v` implementing an 8-bit ALU in Verilog.

D Hardware Test Results for Lab 1: 8-Bit Adder

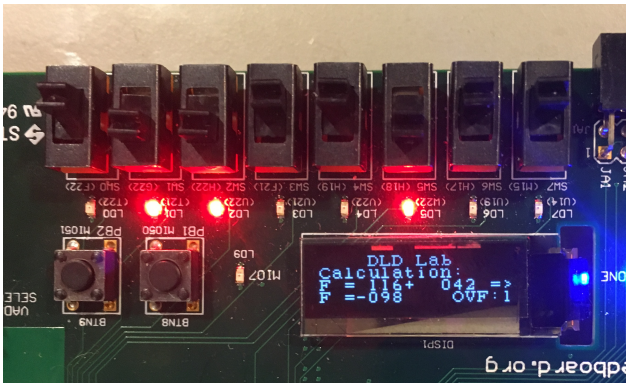

Figure 7: First test case.

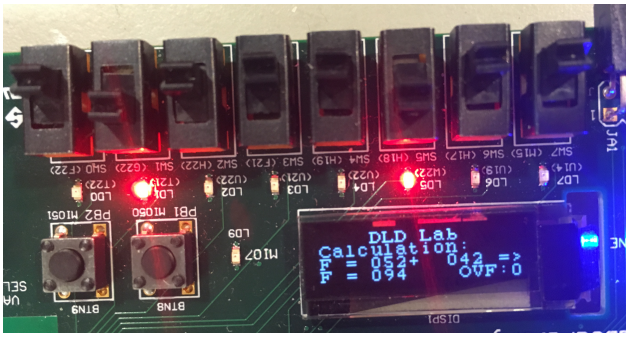

Figure 8: Second test case.

E Hardware Test Results for Lab 2: Partial Arithmetic Logic Unit

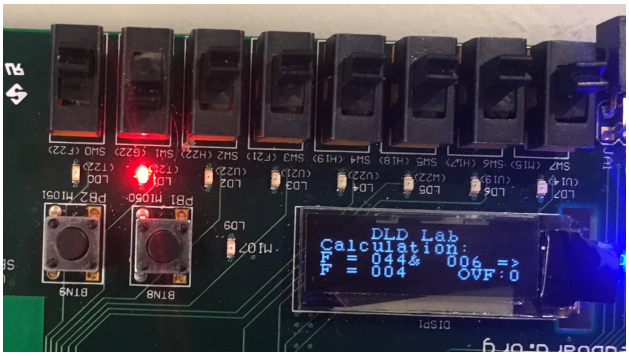

Figure 9: First test case.

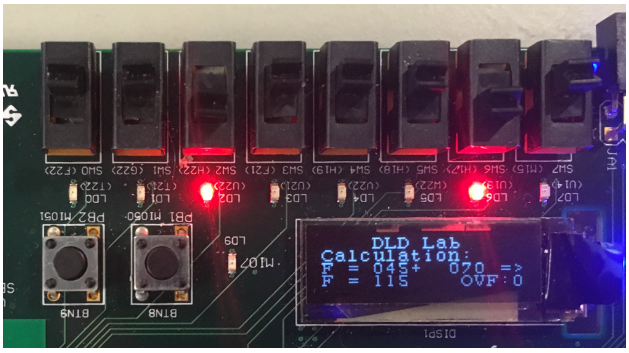

Figure 10: Second test case.

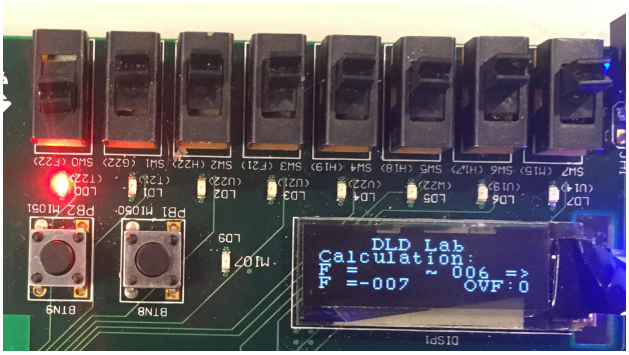

Figure 11: Third test case.

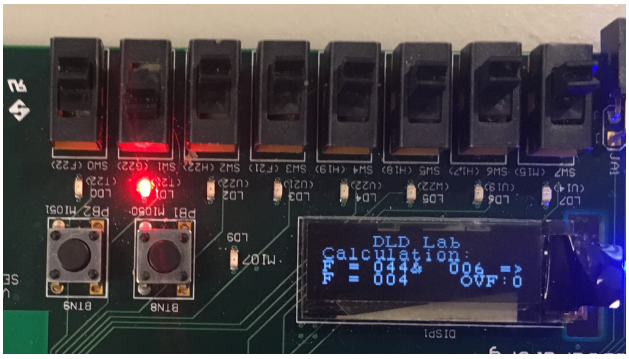

Figure 12: Fourth test case.

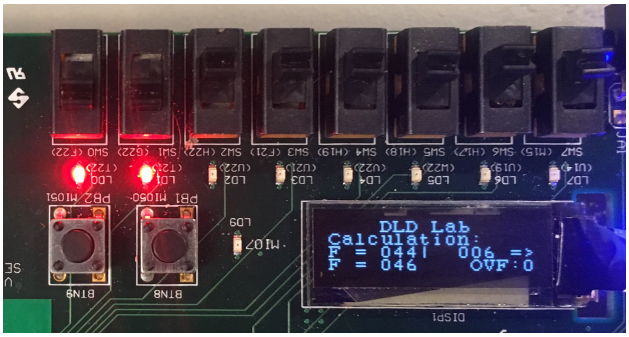

Figure 13: Fifth test case.

F Hardware Test Results for Lab 3: Arithmetic Logic Unit

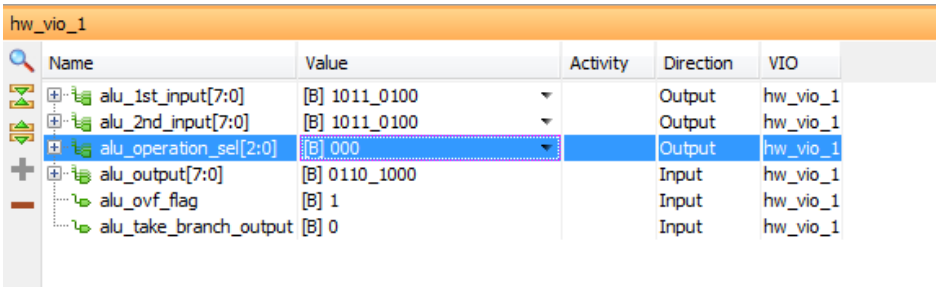

Figure 14: First test case – $a + b$.

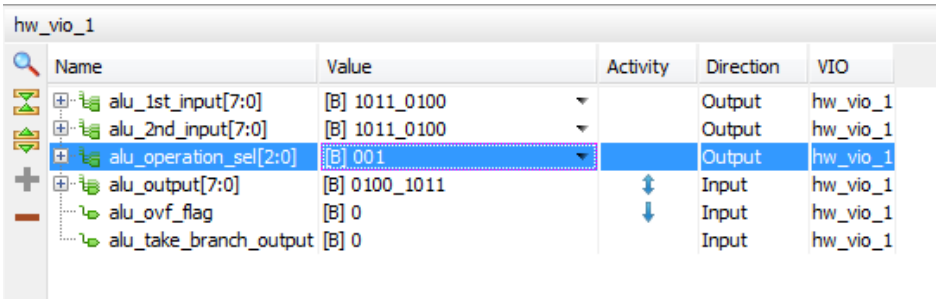

Figure 15: Second test case – $\tilde{}b$.

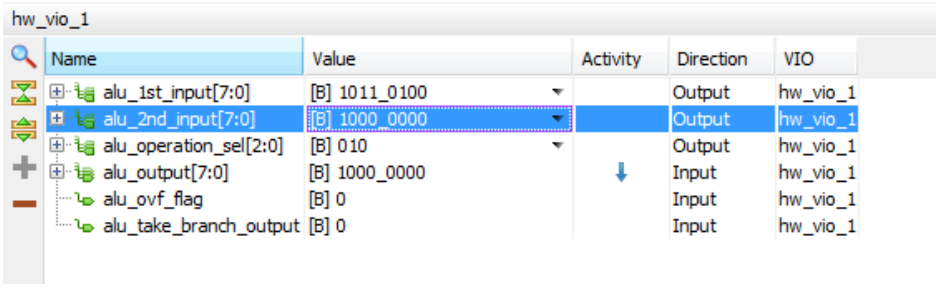

Figure 16: Third test case – $a \cdot b$.

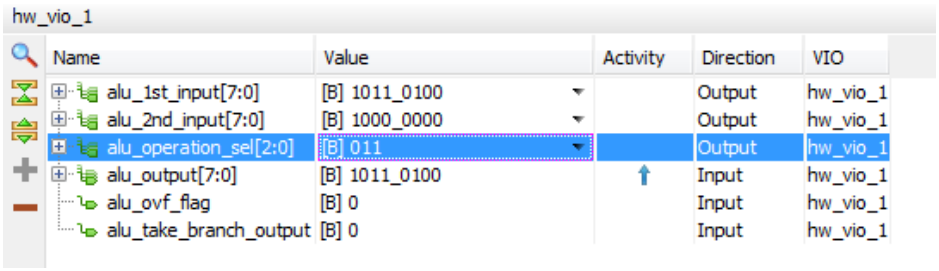

Figure 17: Fourth test case – $a|b$.

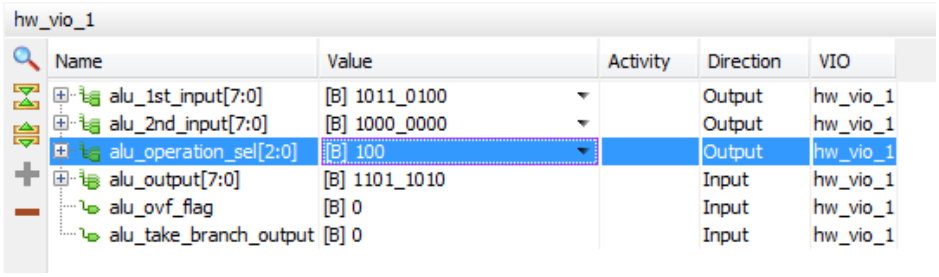

Figure 18: Fifth test case – $a >>> 1$.

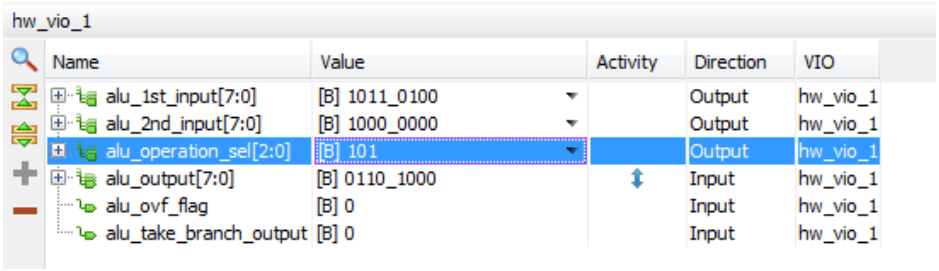

Figure 19: Sixth test case – $a << 1$.

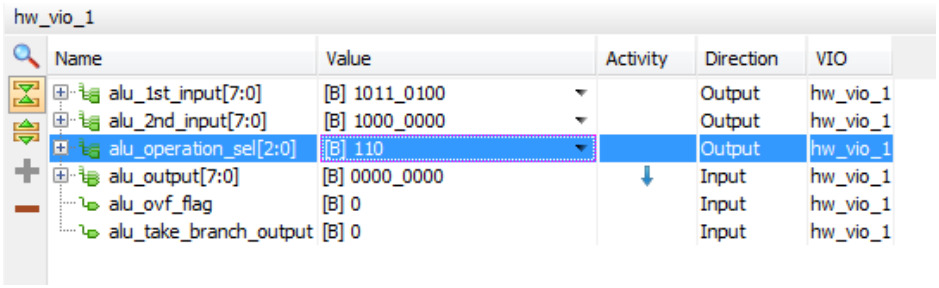

Figure 20: Seventh test case – $beq(a, b) \quad (a \neq b)$.

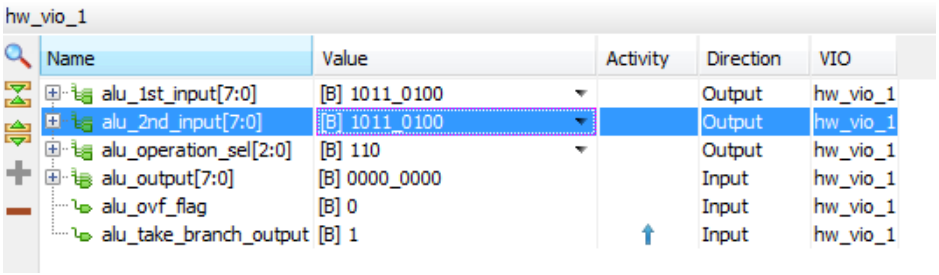

Figure 21: Eighth test case – $beq(a, b) \quad (a = b)$.

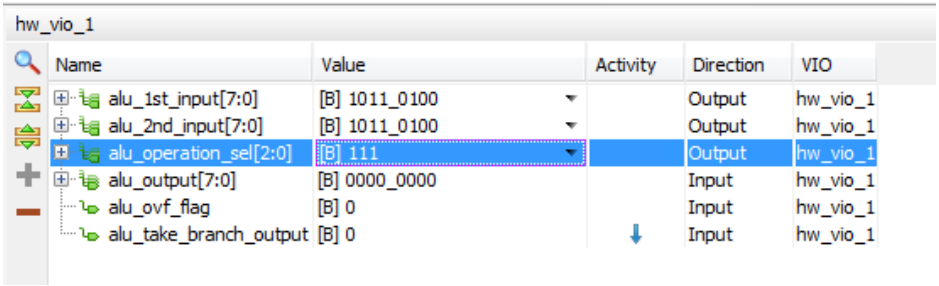

Figure 22: Ninth test case – $bne(a, b) \quad (a = b)$.