8-Bit Arithmetic Logic Unit

Samuel Winchenbach, Department of Electrical and Computer Engineering
Mohammed Driss, Department of Electrical and Computer Engineering
University of Maine, Orono

Contents

Abstract

Digital design is an amazing and very broad field. The applications of digital design are present in our daily life, including Computers, calculators, video cameras etc. In fact, there will be always need for high speed and low power digital products which makes digital design a future growing business. ALU (Arithmetic logic unit) is a critical component of a microprocessor and is the core component of central processing unit. Furthermore, it is the heart of the instruction execution portion of every computer. ALU's comprise the combinational logic that implements logic operations, such as AND and OR, and arithmetic operations, such as ADD and SUBTRACT. We designed and had fabricated an 8-bit ALU (Arithmetic logic unit) that is formed by combining three 74S181 [1], 4-bit ALUs, and five multiplexers.

I. Introduction

A. Project Overview

THE ECE 547 VLSI design project described in this paper is an 8-bit Arithmetic Logic Unit (ALU). We used the 74S181 [1] 4-bit ALU design, which was manufactured by Texas Instruments, as the base of the 8-bit design. Our ALU takes two 8-bits inputs busses (A and B) and performs 32 arithmetic functions and 16 logic functions. There is a 4-bit function select bus (S) to choose the specific operation to perform on the inputs. Also the ALU has one carry input (Cin). The function select M is called the Mode selector. When M is high, the operation is a logic function; when M is low, an arithmetic operation is indicated. The carry input can affect the output produced for arithmetic functions. The output is one 8-bit bus (F) and one carry out (Cout). The carry output Cout tells us whether a carry has occurred by operation performed. The speed of the ALU was not a design consideration. However, after simulation with extracted parasitics we realized that it could run up to approximately 50MHz. The ALU ran at low power equal to 78.9 mW RMS at 5 volts (at 50 MHz). The final layout was implemented through MOSIS, with AMI's C5N process. We used the 0.6 m minimum gate length technology. The device is packaged in a 40-pin ceramic DIP.

B. Objectives

Because ALUs can be built in so many ways with wide specifications and since the objective of the class project is to learn the basic of VLSI design, the specifications of the ALU were relaxed. The main objective of the project is to have a working ALU that performs different arithmetic and logic functions for all possible combinations of the inputs. The speed of ALU was not an issue and we wanted it to run at low power.

II. Circuit Design

This chapter gives an overview of the Hierarchy of the 8-bit ALU and its design. First, we will introduce all the different types of logic gates that has been used in the design. Then, we will give an overview of the 4-bit ALU, 74S181 [1] and show how we broke down the original design of Texas Instruments into different blocks to facilitate the layout task later on. Finally, we will discuss the top level of the design and show how we combined three 4-bit ALU with five multiplexers in such manner to obtain our 8-bit ALU.

A. Logic Gates

We used the CMOS technology to build our gates. From the lowest level NMOS and PMOS, we designed the logic gates needed to form the different blocks of our 4-bit ALU. We used standard designs for logic gates with different possible pull-up and pull-down networks depending on the logic we want to perform. We used the minimum sizing for all the transistors with channel width and length are related as follow:

- Wp/Lp=3/0.6 for PMOS
- Wn/Lp=1.5/0.6 for PMOS

However, when the logic circuit had more than one possible pull-up (or pull-down) current path, we had to follow the general digital design rules for sizing based on the idea of ”worst case”. All the AND gates are NAND gates with inverters. Furthermore, we built a pass gate that we used for multiplexer design. All our logic gates were symmetrical with low power dissipation. The types of logic gates used in the design are as follow:

- Digital Output Buffer
- Inverter
- 2,3,4 and 5 input AND
- 2,3,4 and 5 input NAND
- 2,3,and 4 input NOR
- 2 input XOR
- PASS Transistor

See Table I for a listing of the various logic gates used in implementing the ALU.

Macro Name	Macro Description
SW_1X_INVERTER	Inverter
SW_1X_PASS_TRANS	Pass Transistor
SW_1X_2_AND	2 input AND
SW_1X_3_AND	3 input AND
SW_1X_4_AND	4 input AND
SW_1X_5_AND	5 input AND
MD_1X_2_NAND	2 input NAND
SW_1X_3_NAND	3 input NAND
MD_1X_4_NAND	4 input NAND
SW_1X_5_NAND	5 input NAND
MD_1X_2_NOR	2 input NOR
MD_1X_3_NOR	3 input NOR
MD_1X_4_NOR	4 input NOR
SW_1X_2_XOR	2 input XOR
AD_OUTPUT_BUFFER	Digital Output Buffer [1]

TABLE I
LOGIC GATES

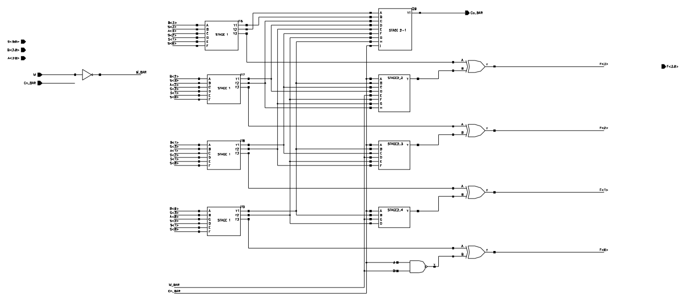

Fig. 1. 4 Bit ALU

B. 4 Bit ALU Circuit

After we built all the needed logic gates, we started putting all the different blocks together to form a 4-bit ALU. The design of the 4-bit ALU is basically the same circuit of the 74S181 [1] of Texas Instruments. This idea of using the 74S181 [1] was introduced to us by Dr. Segee since it can perform a large number of arithmetic and logic operations.

The Texas Instruments ALU design has X and Y outputs that can be used in carry look-ahead circuitry. However, in our design we eliminated those two outputs from the general 74S181 [1] circuit. We also, excluded the open collector output A=B due to it's limited use. The 74S181 can accommodate either active high or low for inputs. For subtraction, it uses the 1's complement. The 74S181 [1] supports as many as 16 logic and 32 arithmetic functions. All these functions are detailed in the Texas Instruments Datasheet [1]. The 4-bit ALU is shown in Figure 1.

C. Multiplexer/Path Selector

Pass gates were used to implement the multiplexer/path selector circuit. Figure 2 represents an 8-bit ALU using carry select which is implemented with multiplexors. Logically the output of the circuit can be written as ($F = AS + BS\_bar$) In other words, when the selector signal S is high, A is selected to be the output. On the other hand, when S is low, the output will be B [2].

D. 8 Bit ALU Circuit

The 8-bit ALU was formed by combining three 4-bit ALU's with 5 multiplexers as shown in Figure 2. The design of the 8-bit ALU is based on the use of a carry select line. The four lowest bits of the input are fed into one of the 4 bit ALU's. The carry out line from this ALU is used to select the outputs from one of the two remaining ALUs. If carry out is asserted then the ALU with carry in tied true is selected. If carry out is not asserted then the ALU with carry in tied false is selected. The outputs of the selectable ALUs are multiplexed together forming the upper and lower 4 bits, and carry out for the 8 Bit ALU.

[1]Design and Layout courtesy of Alma Delic-Ibukic

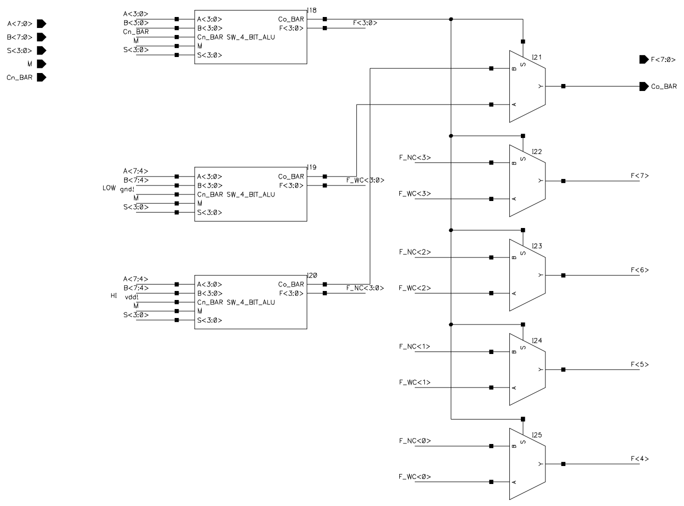

Fig. 2. 8 Bit ALU

III. Design Verification

To verify that the 4 bit ALU was working as expected we created a Verilog script (shown in Appendix VII) to test the logical function of the schematic. Verilog was extracted from the schematic and was not used to design the ALU. The script functions by testing every possible combination of inputs and comparing the Verilog output of the ALU with a expected result. If the output of the ALU does not match the solution a debug message is printed to the console indicating the input, output, and difference. Once the 4 Bit ALU was verified we extended the script to test the 8 Bit ALU. The 8 bit Verilog test script is shown in Appendix VIII. Both the 4-bit and 8-bit ALUs passed the Verilog simulation without errors.

IV. Physical Design

The design of the ALU was implemented using AMI's C5N process. With this process we had three layers of metal. All gates were custom designed. Since the design is based heavily on a bus-style architecture we ran metal 1 east-west and metal 2 north-south. Metal 3 was used for miscellaneous connections and as bridges to prevent antenna design rule violations. Each cell was built around a template which consisted of a grounded metal 1 guard ring with metal 2 tabs used for both I/O and voltage supply connections. Using this construction method reduces the possibility of latch up and also creates an easy to use component that connects to neighboring cells. Figure 3 shows the layout of the major components used to create the 8 Bit ALU. The completed layout passed both the *Design Rule Check* (DRC) and *Layout vs. Schematic* (LVS).

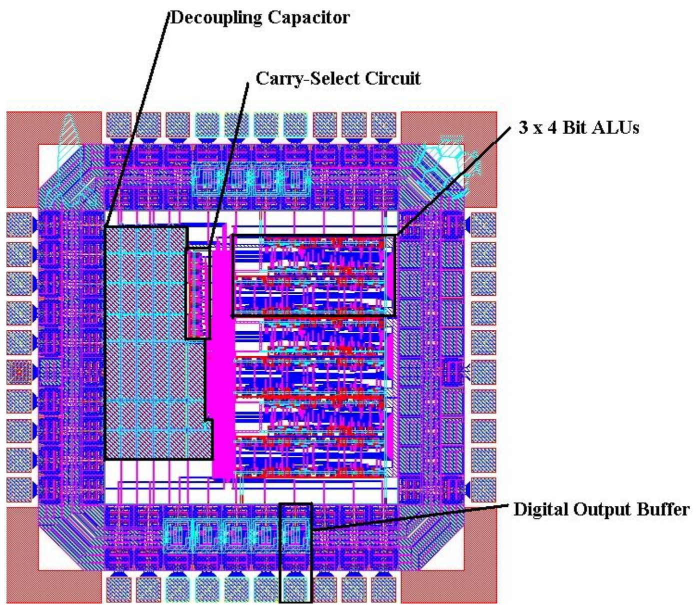

Fig. 3. Final 8 Bit ALU

```
n = 1e3;
f = 10e6;
NUMBER_INPUTS = 22;
T = 1/f;
trans = T*(1/10);
for j = 1:NUMBER_INPUTS
  t = 0;
  a = hardlim(rand(1,n)-0.5) .* 5;
  fid = fopen(['./inputs/input' num2str(j) '.txt'],'w');
  for i = 1:n
    for k = 1:2
      fprintf(fid,'%o\t%o\n',[t;a(i)]);
      t = t + (T/2 - trans);
    end
    t = t + trans - (T/2 - trans);
  end
  fclose(fid);
end
```

Fig. 4. makeinput.m

V. Circuit Performance

Once verification was complete parasitics were extracted. To create a test for finding the maximum operating frequency a pseudo-random input stream was applied to each input of the ALU and observed the current during the input transition. Seeing that this is a CMOS design the current drawn by the ALU will approach zero between states if operating correctly. If the switching of the transistors takes longer than the period of the input then the ALU may give a erroneous output. The Matlab code to create the pseudo-random inputs is shown in Figure 4. A random input stream similar to the one shown in Figure 5 was applied to each input of the ALU.

The current through the voltage supply pin for a 10 MHZ random input stream is shown in Figure 6. The average current for this case is 2.18 mA (8.51 mA RMS).

Figure 7 shows the current through the voltage supply pin for a 50 MHZ random input stream. This gives an average current of 7.15 mA (15.79 mA RMS). It is important to note that the current still returns to zero between inputs. At this frequency the current should be five times greater than the 10 MHz test case. It is possible that the ALU is not able to switch at 50 MHz.

Figure 8 shows the current through the voltage supply pin got a 100 MHZ random input stream. This gives an average current of 13.89 mA (22.45 mA RMS). Notice that the ALU is no longer functioning properly. At this frequency the transistors are still switching (on certain occasions) when the next input stream is applied.

VI. Test And Characterization

To test the fabricated 8-bit ALUs the circuit board shown in Figure 9 was designed and manufactured. The sockets labeled SW1, SW2 and SW3 are bussed tri-state switches. The three output states are ground, voltage supply, and float. The output of the switches are connected to the inputs of the ALU. The straight connectors are used to provide direct access to the input and output of the ALU. With this setup it is possible to set each input pin individually to logic 0,1 or float. Having the switches set to float allows external signals to be applied to the input of the ALU through the straight header.

Due to a lack of appropriate test equipment a limited number of tests were performed. To test the carry select circuitry and all outputs of the F bus the select lines were set, using the tri-state switches, to add A and B together. All lines on the B input bus were set to logic 1. Bits 1 through 7 of the A input bus were set to logic 0. Bit 0 of A was set to float and a square wave signal was applied to the pin. This sets the ALU to add 255 plus either 1 or 0 based on the square wave. This will cause the output of the ALU to alternate between two states. Every bit of the F bus will be logic 1 and Cout will be logic 0 when bit 0 of A is logic 0. Every bit of the F bus will be logic 0 and Cout will be logic 1 when bit 0 of A is logic 1. With this test setup the ALU performed as expected.

Figure 10 shows the supply current during a frequency sweep of the above test condition. The maximum current occurs when the switching frequency of bit 0 of the A input bus is slightly above 40 MHz. This suggests that the maximum operating frequency of the ALU is approximately 40 MHz.

A number of other ALU functions were testing including both arithmetic and logic functions. The ALU performed as expected for all test situations.

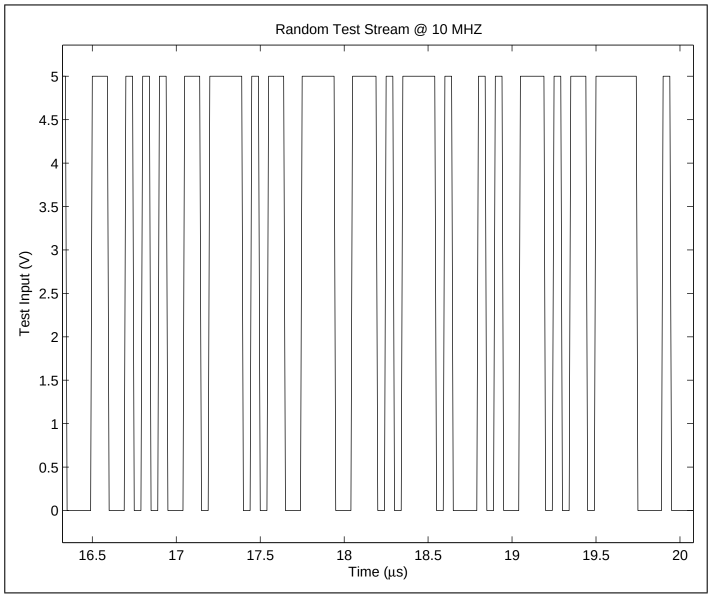

Fig. 5. Example of a Pseudo-Random Test Stream

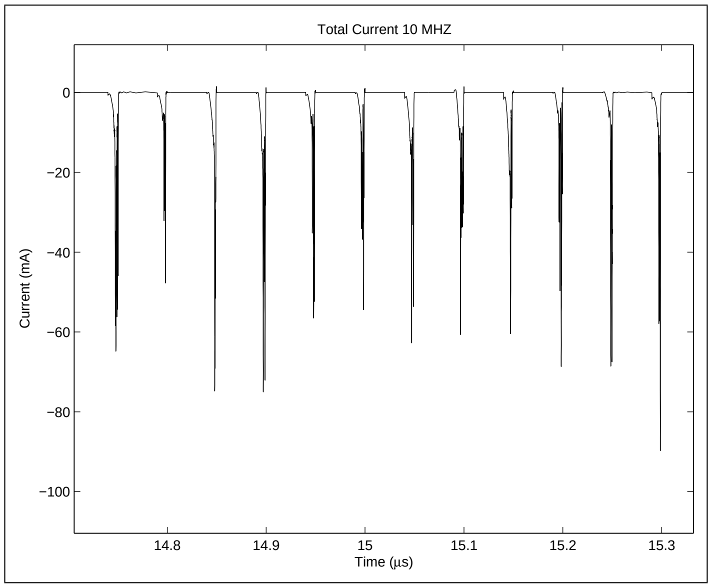

Fig. 6. 10 MHZ Random Input Extracted Simulation

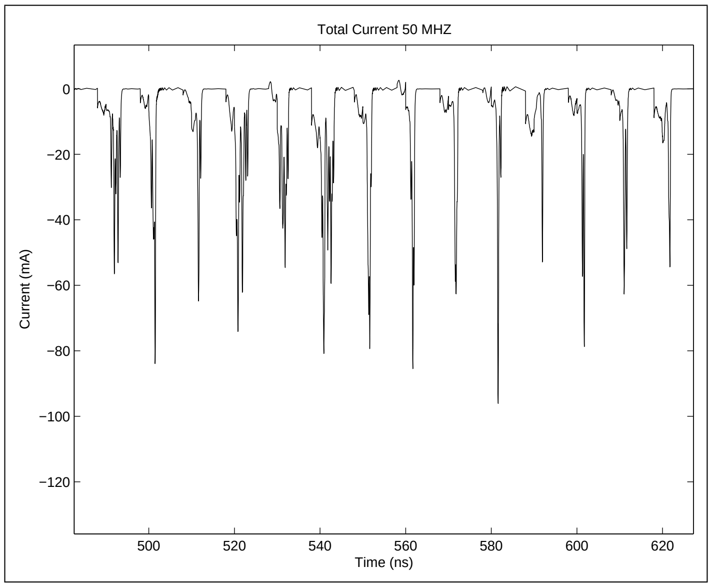

Fig. 7. 50 MHZ Random Input Extracted Simulation

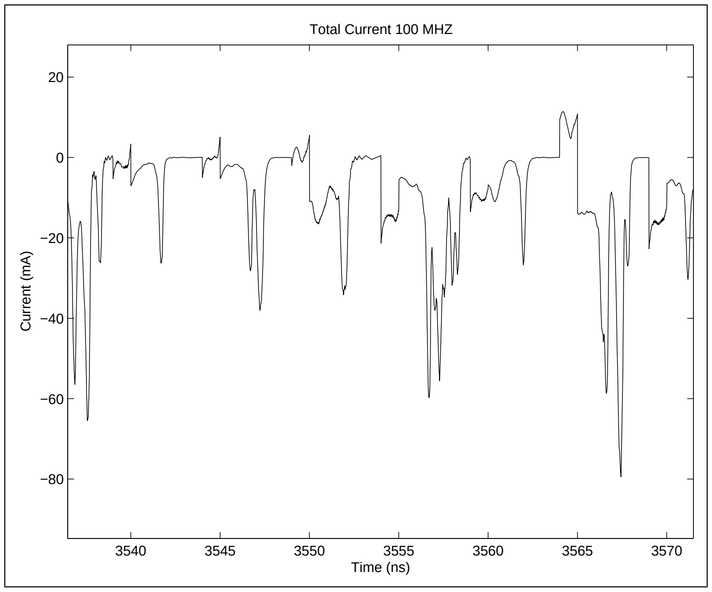

Fig. 8. 100 MHZ Random Input Extracted Simulation

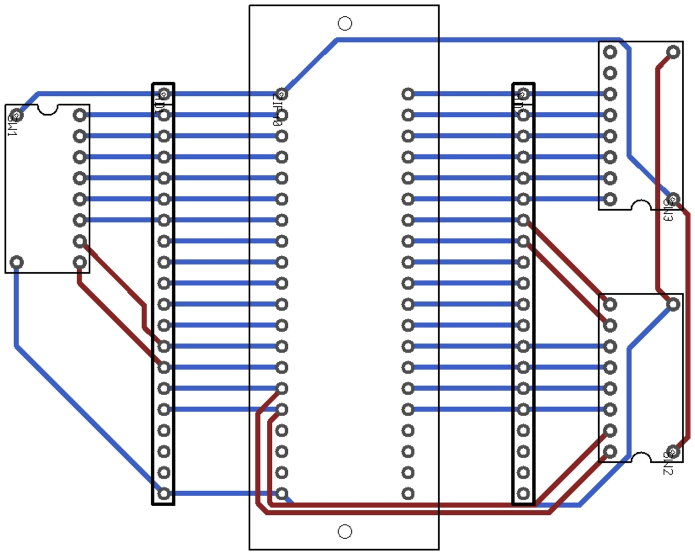

Fig. 9. PCB Test Board

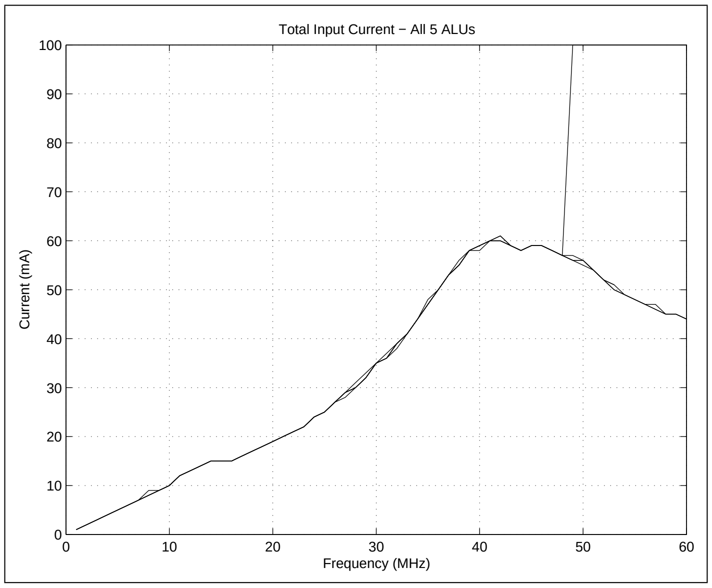

Fig. 10. Frequency Sweep of All 5 Packaged ALUs

VII. 4 Bit ALU Verilog Verification Code

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


// Default verilog stimulus.

integer Bx, Sx, ALU, Cx, Mx, FMASK, ALUMASK, Ax;

reg[4:0] GUESS;

initial
begin
   A[3:0] = 4’b0000;

   B[3:0] = 4’b0000;

   Cn_BAR = 1’b0;
   M = 1’b0;
   S[3:0] = 4’b0000;

  FMASK = 4’b1111;
  ALUMASK = 5’b11111;


/*
  S = 4’b0010;
  B = 4’b0000;
  #150

  for(Ax=0; Ax <= 15; Ax = Ax + 1’b1)
  begin
    A = Ax;
    #150
    GUESS = {1’b0, (A|˜B)} + !Cn_BAR;
    $display("˜B: %b, A | ˜B: %d, !Cn_BAR: %b, 1111 + 1: %b",˜B,(A | ˜B),!Cn_BAR, 4’b1111+1’b1);
    $display("%d",GUESS);
    ALU = {!Co_BAR, F[3:0]};
    $display("M: %b , S: %b , A: %d , B: %d , Cn_BAR: %d\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
  end
*/


  for(Mx = 0; Mx <= 1; Mx = Mx+1)
  begin
    M = Mx;
    #150

    for(Sx=0; Sx < 16; Sx = Sx + 1)
    begin
      S[3:0] = Sx;
      #150

      for(Cx = 0; Cx <= 1; Cx = Cx+1)
      begin
        Cn_BAR = Cx;
        #150

        for(Ax=0; Ax < 16; Ax = Ax+1)
        begin
          A[3:0] = Ax;
          #150

          for(Bx=0; Bx < 16; Bx = Bx+1)
          begin
            B[3:0] = Bx;
            #150
            ALU = (M == 0) ? {!Co_BAR, F[3:0]} & ALUMASK : F[3:0] & FMASK;

            if(S == 0)
            begin
              GUESS = (M == 0) ? (A + !Cn_BAR) & ALUMASK : (˜A) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 1)
            begin
              GUESS = (M == 0) ? ((A | B) + !Cn_BAR) & ALUMASK : (˜(A | B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 2)
            begin
              GUESS = (M == 0) ? ({1’b0,((˜B) | A)} + !Cn_BAR) & ALUMASK : ((˜A) & B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 3)
            begin
              GUESS = (M == 0) ? (4’b1111 + !Cn_BAR) & ALUMASK : (0) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 4)
            begin
              GUESS = (M == 0) ? (A + {1’b0, (A & ˜B)} + !Cn_BAR) & ALUMASK : (˜(A & B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end
            if(S == 5)
            begin
              GUESS = (M == 0) ? ((A | B) + {1’b0, (A & ˜B)} + !Cn_BAR) & ALUMASK : (˜B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 6)
            begin
              GUESS = (M == 0) ? ( A + {1’b0,˜B} + !Cn_BAR) & ALUMASK : (AˆB) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 7)
            begin
              GUESS = (M == 0) ? ( { 1’b0,(A & ˜B) } + 4’b1111 + !Cn_BAR) & ALUMASK : (A & (˜B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 8)
            begin
              GUESS = (M == 0) ? (A + (A & B) + !Cn_BAR) & ALUMASK : (˜A | B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 9)
            begin
              GUESS = (M == 0) ? (A + B + !Cn_BAR) & ALUMASK : (˜(AˆB)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 10)
            begin
              GUESS = (M == 0) ? ( {1’b0, (A | ˜B)} + (A & B) + !Cn_BAR) & ALUMASK : (B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 11)
            begin
              GUESS = (M == 0) ? ({1’b0, (A & B)} + 4’b1111 + !Cn_BAR) & ALUMASK : (A & B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 12)
            begin
              GUESS = (M == 0) ? ((A + A) + !Cn_BAR) & ALUMASK : 15;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 13)
            begin
              GUESS = (M == 0) ? ( (A | B) + A + !Cn_BAR ) & ALUMASK : (A | ˜B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 14)
            begin
              GUESS = (M == 0) ? ( {1’b0, (A | ˜B)} + A + !Cn_BAR ) & ALUMASK : (A | B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end

            if(S == 15)
            begin
              GUESS = (M == 0) ? ({1’b0, A} + 4’b1111 + !Cn_BAR) & ALUMASK : A & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[4:0],GUESS[4:0],ALU[4:0]ˆGUESS[4:0]);
              end
            end
          end //B
        end //A
      end //S
    end //Cn_BAR
  end //M
end //Initial
```

VIII. 8 Bit ALU Verilog Verification Code

```
integer Bx, Sx, ALU, Cx, Mx, FMASK, ALUMASK, Ax;

reg[8:0] GUESS;

initial
begin
  HI         = 1'b1;
  LOW      = 1'b0;

   A[7:0] = 8'b00000000;

   B[7:0] = 8'b00000000;

   Cn_BAR = 1'b0;
   M = 1'b0;
   S[3:0] = 4'b0000;

  FMASK = 8'b11111111;
  ALUMASK = 9'b111111111;

  for(Mx = 0; Mx <= 1; Mx = Mx+1)
  begin
    M = Mx;
    #150

    for(Sx=0; Sx < 16; Sx = Sx + 1)
    begin
      S[3:0] = Sx;
      #150

      for(Cx = 0; Cx <= 1; Cx = Cx+1)
      begin
        Cn_BAR = Cx;
        #150

        for(Ax=0; Ax < 256; Ax = Ax+1)
        begin
          A[7:0] = Ax;
          #150

          for(Bx=0; Bx <256; Bx = Bx+1)
          begin
            B[7:0] = Bx;
            #150
            ALU = (M == 0) ? {!Co_BAR, F[7:0]} & ALUMASK : F[7:0] & FMASK;

            if(S == 0)
            begin
              GUESS = (M == 0) ? (A + !Cn_BAR) & ALUMASK : (~A) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 1)
            begin
              GUESS = (M == 0) ? ((A | B) + !Cn_BAR) & ALUMASK : (~(A | B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 2)
            begin
              GUESS = (M == 0) ? ({1'b0,((~B) | A)} + !Cn_BAR) & ALUMASK : ((~A) & B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 3)
            begin
              GUESS = (M == 0) ? (8'b11111111 + !Cn_BAR) & ALUMASK : (0) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 4)
            begin
              GUESS = (M == 0) ? (A + {1'b0, (A & ~B)} + !Cn_BAR) & ALUMASK : (~(A & B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 5)
            begin
              GUESS = (M == 0) ? ((A | B) + {1'b0, (A & ~B)} + !Cn_BAR) & ALUMASK : (~B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 6)
            begin
              GUESS = (M == 0) ? ( A + {1'b0,~B} + !Cn_BAR) & ALUMASK : (A^B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 7)
            begin
              GUESS = (M == 0) ? ( { 1'b0,(A & ~B) } + 8'b11111111 + !Cn_BAR) & ALUMASK : (A & (~B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 8)
            begin
              GUESS = (M == 0) ? (A + (A & B) + !Cn_BAR) & ALUMASK : (~A | B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 9)
            begin
              GUESS = (M == 0) ? (A + B + !Cn_BAR) & ALUMASK : (~(A^B)) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 10)
            begin
              GUESS = (M == 0) ? ( {1'b0, (A | ~B)} + (A & B) + !Cn_BAR) & ALUMASK : (B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 11)
            begin
              GUESS = (M == 0) ? ({1'b0, (A & B)} + 8'b11111111 + !Cn_BAR) & ALUMASK : (A & B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 12)
            begin
              GUESS = (M == 0) ? ((A + A) + !Cn_BAR) & ALUMASK : 255;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 13)
            begin
              GUESS = (M == 0) ? ( (A | B) + A + !Cn_BAR ) & ALUMASK : (A | ~B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 14)
            begin
              GUESS = (M == 0) ? ( {1'b0, (A | ~B)} + A + !Cn_BAR ) & ALUMASK : (A | B) & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

            if(S == 15)
            begin
              GUESS = (M == 0) ? ({1'b0, A} + 8'b11111111 + !Cn_BAR) & ALUMASK : A & FMASK;
              if(GUESS != ALU)
              begin
                $display("M: %b , S: %b , A: %b , B: %b , Cn_BAR: %b\nALU: %b , Guess: %b , Difference: %b\n",M,S,A,B,Cn_BAR,ALU[8:0],GUESS[8:0],ALU[8:0]^GUESS[8:0]);
              end
            end

          end //B
        end //A
      end //S
    end //Cn_BAR
  end //M
end //Initial
```

IX. Circuit Schematics

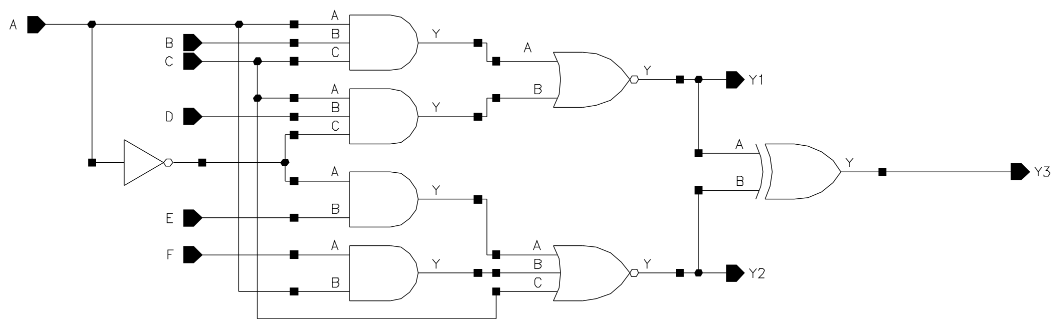

Fig. 11. Stage 1 Of 4 Bit ALU

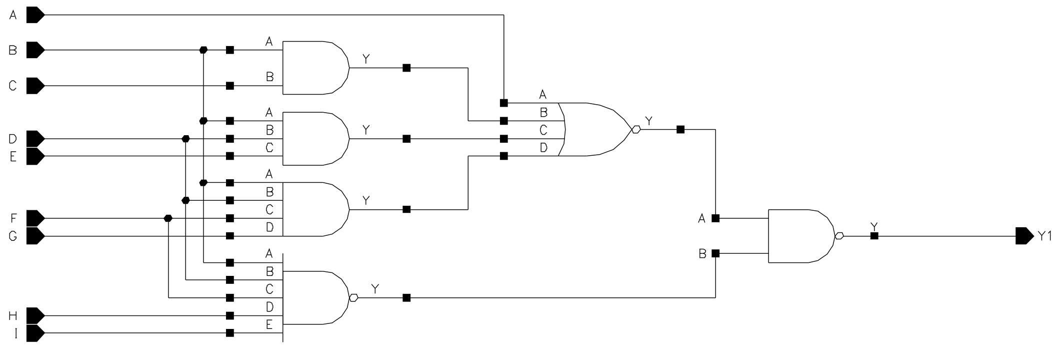

Fig. 12. Stage 2-1 Of 4 Bit ALU

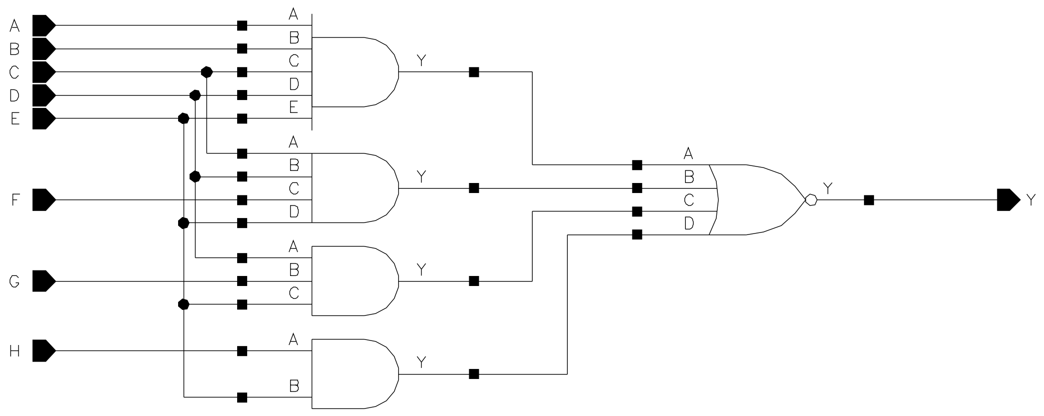

Fig. 13. Stage 2-2 Of 4 Bit ALU

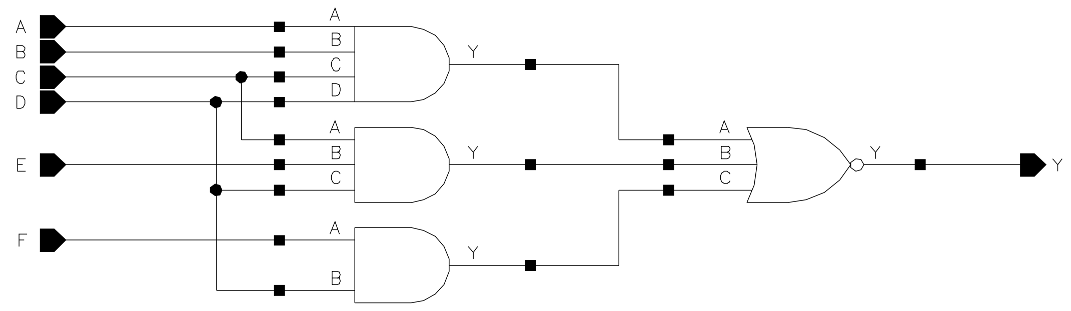

Fig. 14. Stage 2-3 Of 4 Bit ALU

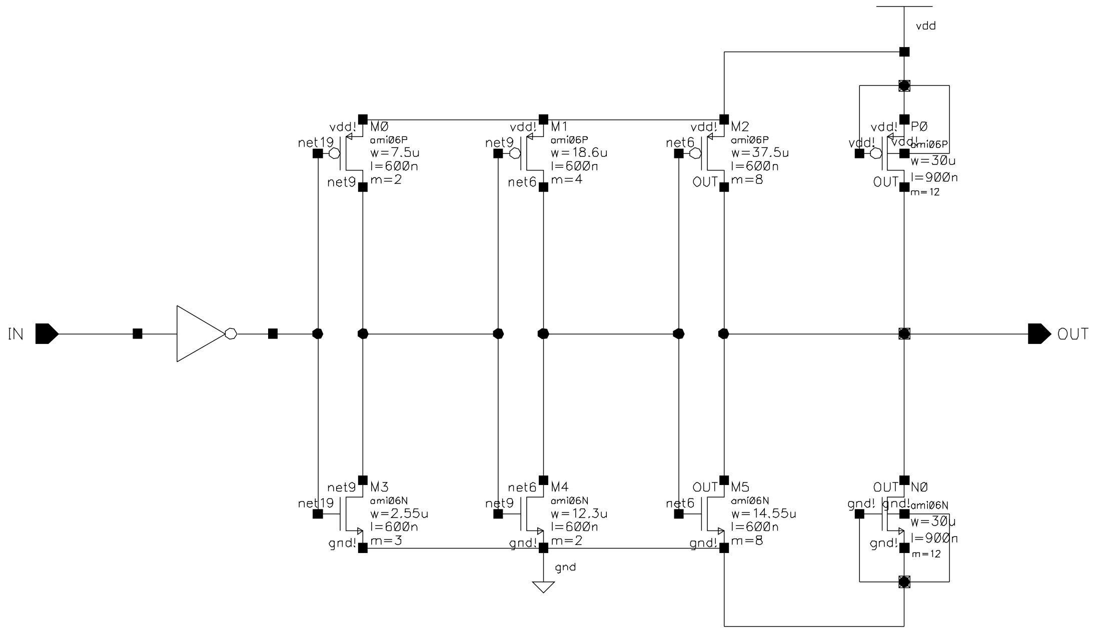
vdd
IN
OUT
gnd
M0 ami06P w=7.5u l=600n m=2
M1 ami06P w=18.6u l=600n m=4
M2 ami06P w=37.5u l=600n m=8
P0 ami06P w=30u l=900n m=12
M3 ami06N w=2.55u l=600n m=3
M4 ami06N w=12.3u l=600n m=2
M5 ami06N w=14.55u l=600n m=8
N0 ami06N w=30u l=900n m=12

Fig. 15. Digital Output Buffer

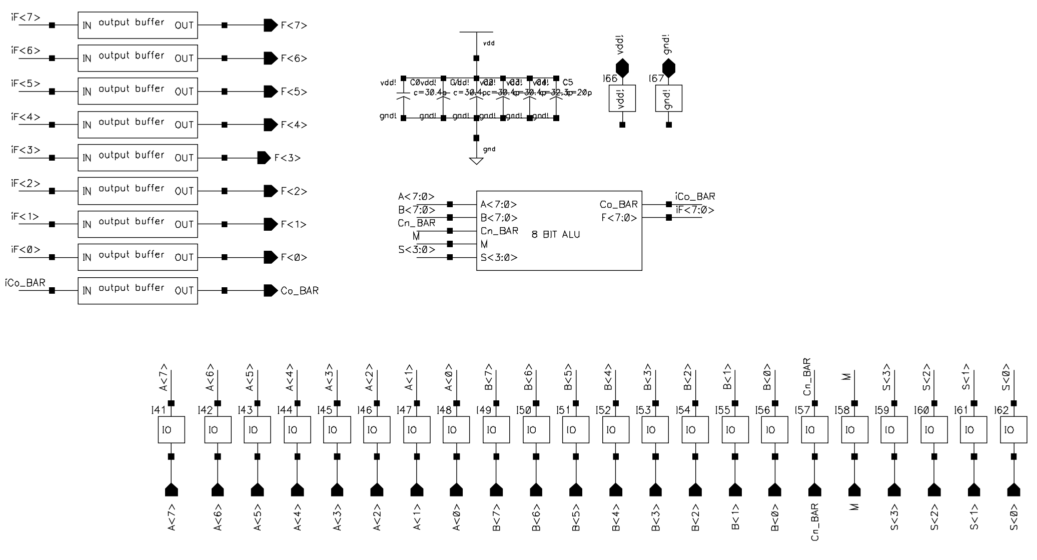

Fig. 16. Final 8 Bit ALU

References

[1] *Arithmetic Logic Units/Function Generators Datasheet*, Texas Instruments, Dallas, Texas, 1988. [Online]. Available: http://www-s.ti.com/sc/ds/sn54s181.pdf
[2] H. W. L. R. J. Baker and D. E. Boyce, *CMOS Circuit Design, Layout, and Simulation*. New York, NY: IEEE Press, 1998.

Samuel A Winchenbach recieved the B.S. in Electrical Engineering from the University of Maine in 2002, and is currently working towards his M.S. degree. His research interests include marine mammal tracking using underwater sonar, digital signal processing, time-frequency domain analysis and embedded systems design.

Mohammed Driss was born in Tunis, Tunisia. He received the B.S. in Electrical Engineering from the University of Maine in 2002,and is currently working towards his M.S. degree. He has worked as: a Laboratory Teaching Assistant for the Department of Electrical and Computer Engineering, a computer Consultant for Information Technologies and a Mathematics Department Tutor and Grading Assistant. He has also worked two consecutive summer (2001 and 2002) as a Product/Test Engineer (Intern) for Analog Devices Inc.