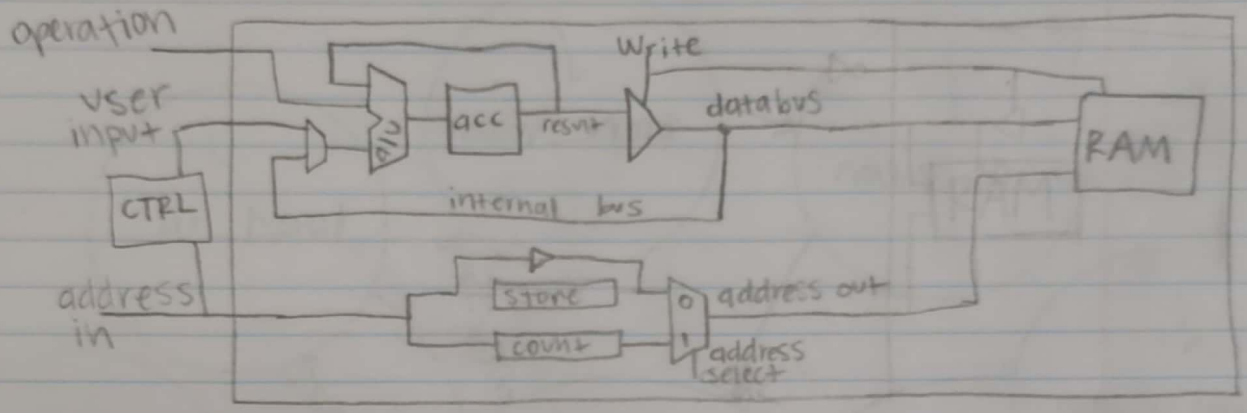
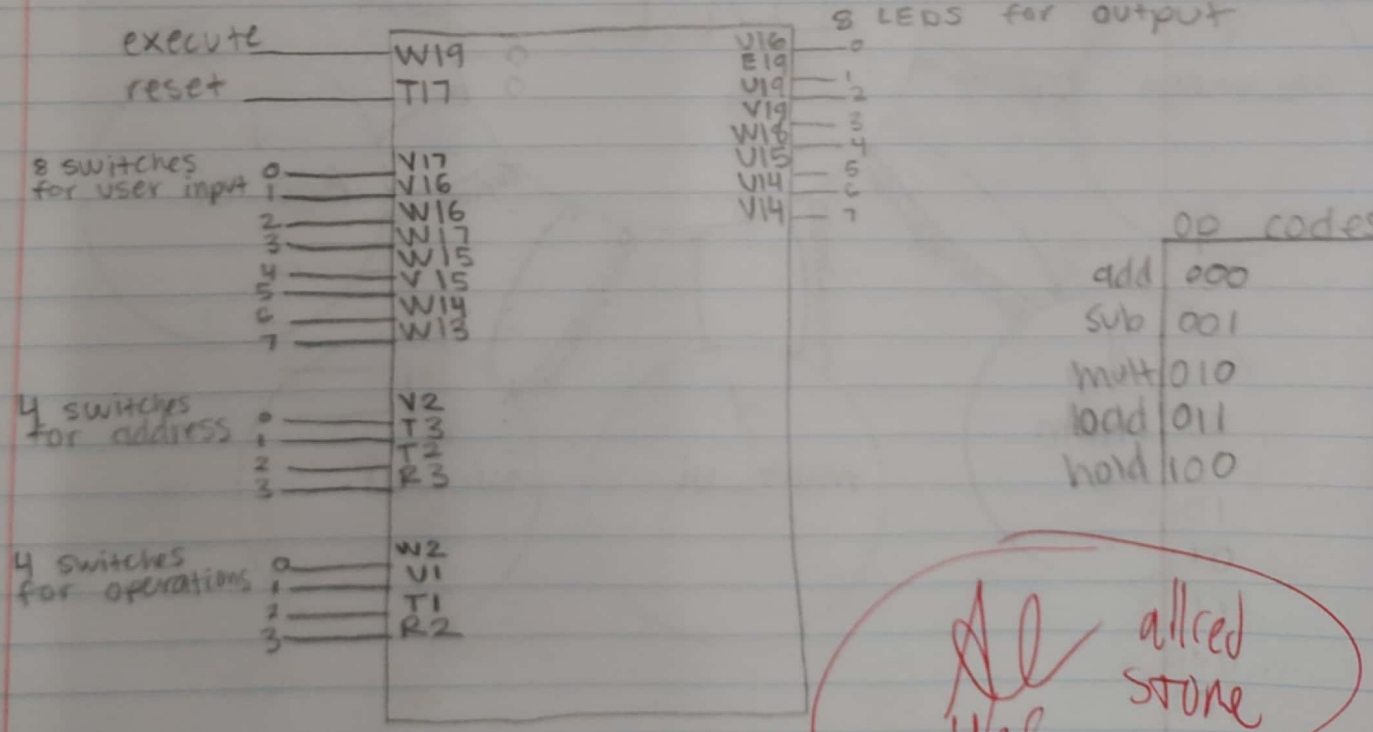
Project 3

BLOCK DIAGRAM



FPGA PINOUT



OP codes

add	000
sub	001
mult	010
load	011
hold	100

All 4/26 allred store

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Calculator is
Port (input: in std_logic_vector(7 downto 0);
      address,operation: in std_logic_vector(3 downto 0);
      clk,rst,execute: in std_logic:= '0';
      result: out std_logic_vector( 7 downto 0));
end Calculator;

architecture Behavioral of Calculator is
component CTRL2 is
Port (clk,rst,enter: in std_logic;
      input_in: in std_logic_vector(7 downto 0);
      addr_in,user_op: in std_logic_vector(3 downto 0);
      addr_out: out std_logic_vector(3 downto 0);
      Input: out std_logic_vector(7 downto 0);
      in_select,mem_wr,addr_inc, addr_load,addr_sel: out std_logic;
      operation: out std_logic_vector(2 downto 0):="000");
end component;

component Calc_datapath is
Port (Input: in std_logic_vector( 7 downto 0);
      in_select: in std_logic;
      mem_wr: in std_logic;
      clk,rst: in std_logic;
      operation: in std_logic_vector(2 downto 0);
      addr_inc, addr_load: in std_logic;
      addr_sel : in std_logic;
      addr_in: in std_logic_vector(3 downto 0);
      addr_out: out std_logic_vector(3 downto 0);
      data_bus: inout std_logic_vector( 7 downto 0);
      Result_out: out std_logic_vector( 7 downto 0);
      en: in std_logic:='1');
end component;


component memory is
Port (clk,wr: in std_logic;
      address: in std_logic_vector( 3 downto 0);
      data: inout std_logic_vector(7 downto 0));
end component;

signal op_s: std_logic_vector(2 downto 0);
signal input_s,data_bus_s: std_logic_vector( 7 downto 0);
signal addr_s, addr_out_s: std_logic_vector(3 downto 0);
signal in_select_s,mem_wr_s,addr_inc_s, addr_load_s,addr_sel_s: std_logic;
begin
Controler: ctrl2 port map(clk => clk,
                          rst => rst,
                          enter => execute,
                          input_in => input,
                          addr_in => address,
                          user_op => operation,
                          input => input_s,
                          addr_out  => addr_s,
                          in_select => in_select_s,
                          mem_wr => mem_wr_s,
                          addr_inc => addr_inc_s,
                          addr_load => addr_load_s,
                          addr_sel => addr_sel_s,
                          operation => op_s);

calc: calc_datapath port map(Input => input_s,
                             in_select => in_select_s,
                             mem_wr => mem_wr_s,
                             clk => clk,
                             rst => rst,
```

```
                                operation => op_s,
                                addr_inc => addr_inc_s,
                                addr_load => addr_load_s,
                                addr_sel => addr_sel_s,
                                addr_in  => addr_s,
                                addr_out => addr_out_s,
                                Result_out => result,
                                data_bus => data_bus_s);

    Mem: memory port map(clk => clk,
                         wr => mem_wr_s,
                         address => addr_out_s,
                         data => data_bus_s);


    end Behavioral;

```

```vhdl
library ieee ;
    use ieee.std_logic_1164.all ;
    use ieee.numeric_std.all ;

entity Calculator_tb is
end Calculator_tb ;

architecture structural of Calculator_tb is
    component calculator is
        port(input: in std_logic_vector(7 downto 0);
             address, operation: in std_logic_vector(3 downto 0);
             clk, rst, execute: in std_logic;
             result: out std_logic_vector(7 downto 0));
    end component;

    constant add_ui: std_logic_vector(3 downto 0):= "0000";
    constant add_m: std_logic_vector(3 downto 0):= "0001";
    constant sub_ui: std_logic_vector(3 downto 0):= "0010";
    constant sub_m: std_logic_vector(3 downto 0):= "0011";
    constant mult_ui: std_logic_vector(3 downto 0):= "0100";
    constant mult_m: std_logic_vector(3 downto 0):= "0101";
    constant store_m: std_logic_vector(3 downto 0):= "0110";
    constant inc_m: std_logic_vector(3 downto 0):= "0111";
    constant load_ui: std_logic_vector(3 downto 0):= "1000";
    constant load_m: std_logic_vector(3 downto 0):= "1001";

    signal input_tb,result_tb: std_logic_vector(7 downto 0):="00000000";
    signal address_tb, operation_tb: std_logic_vector(3 downto 0):="0000";
    signal clk_tb, rst_tb: std_logic := '0';
    signal execute_tb: std_logic:='1';

    begin
    Test: calculator port map (input => input_tb,
                               address => address_tb,
                               operation => operation_tb,
                               clk => clk_tb,
                               rst => rst_tb,
                               execute => execute_tb,
                               result => result_tb);
    clk_tb <= NOT clk_tb after 5 ns;
    execute_TB <= not execute_TB after 20 ns;
    -- This process will test our operations
    process
        begin
        input_tb <= "00000001";
        operation_tb <= load_ui;  --load user input
        rst_tb <= '1';
        wait for 7 ns;
        rst_tb <= '0';            --testing reset
        wait for 5 ns;
        wait for 20 ns;
        operation_tb <= add_ui;   --add user input
        input_tb <= "00000010";
        wait for 40 ns;
        address_tb <= "0000";
        operation_tb <= inc_m;   --increment memory
        input_tb <= "00001010";
        wait for 40 ns;
        address_tb <= "0000";
        operation_tb <= add_m;     --add from memory
        input_tb <= "00000111";
        wait for 40 ns;
        operation_tb <= sub_ui;    --subtract user input
        input_tb <= "00001010";
        wait for 40 ns;
        address_tb <= "0000";
        operation_tb <= inc_m;    --increment memory
        input_tb <= "00001010";
        wait for 40 ns;
```

```
70          address_tb <= "0000";
71          operation_tb <= mult_ui;   --multiply user input
72          input_tb <= "00001010";
73          wait for 40 ns;
74          address_tb <= "0000";
75          operation_tb <= sub_m;     --subtract memory
76          input_tb <= "00001010";
77          wait for 40 ns;
78          address_tb <= "0000";
79          operation_tb <= mult_m;    --multiply memory
80          input_tb <= "00001010";
81          wait for 40 ns;
82          address_tb <= "0000";
83          operation_tb <= load_m;   --load memory
84          input_tb <= "00001010";
85          wait for 40 ns;
86          wait for 40 ns;
87          address_tb <= "0000";
88          operation_tb <= add_ui;   --add user input
89          input_tb <= "00000010";
90          wait for 40 ns;
91          address_tb <= "0000";
92          operation_tb <= add_ui;  --add user input
93          input_tb <= "00111111";
94          wait for 40 ns;
95          address_tb <= "0000";
96          operation_tb <= inc_m;   --increment memory
97          input_tb <= "00001010";
98          wait for 40 ns;
99          address_tb <= "0000";
100         operation_tb <= add_m;   --add from memory
101         input_tb <= "00000111";
102         wait for 40 ns;
103         address_tb <= "0010";
104         operation_tb <= sub_ui;   --subtract user input
105         input_tb <= "00001010";
106         wait for 40 ns;
107         address_tb <= "0010";
108         operation_tb <= inc_m;   --increment memory
109         input_tb <= "00001010";
110         wait for 40 ns;
111         address_tb <= "0010";
112         operation_tb <= mult_ui;  --multiply user input
113         input_tb <= "00001010";
114         wait for 40 ns;
115         input_tb <= "00000001";
116         address_tb <= "0100";
117         operation_tb <= load_ui;   --load user input
118         wait for 40 ns;
119         address_tb <= "0010";
120         operation_tb <= sub_m;    --subtract memory
121         input_tb <= "00001010";
122         wait for 40 ns;
123         address_tb <= "0010";
124         operation_tb <= mult_m;   --multiply memory
125         input_tb <= "00001010";
126         wait for 40 ns;
127         address_tb <= "0000";
128         operation_tb <= load_m;   --load memory
129         input_tb <= "00001010";
130         wait for 40 ns;
131         wait;
132     end process;
133 end structural;
```