EC413

October 12, 2022

ALUtopModule

EC413 Lab4

This module instantiates all other modules and has a mux to perform the various output the correct values depending on the input ALUOp. It performs the following functions:

1. MOV
2. NOT
3. Addition
4. Subtraction
5. Logical OR
6. Logical AND

The modules will all run and upon entering an ALUOp, it enters a MUX generated through case statements and outputs the value of R1. R1 then passes through the NbitReg to store the value of R1 into R0.

Nbit_NOT

This performs a bitwise NOT operation for each bit on the input and puts it into the output. Uses a loop to perform the action.

NbitAdder

Creates an N-bit ripple carry adder and adds the the two inputs. It takes in inputs a, b, and c_in and its outputs are c_out and sum.

NbitSub

This module takes two inputs a and b, and input c_in = 1. It performs the bitwise NOT operation on b by calling Nbit_NOT. It then will use the Full adder to create a ripple carry to generate the proper values. It returns c_out and sum.

Nbit_OR

This performs a bitwise OR operation between inputs inval1 and inval2. It does this in a loop using the generate function and an or gate and stores the output in outval.

Nbit_AND

This performs a bitwise AND operation between inputs inval1 and inval2. It does this in a loop using the generate function and an and gate and stores the output in outval.

NbitReg

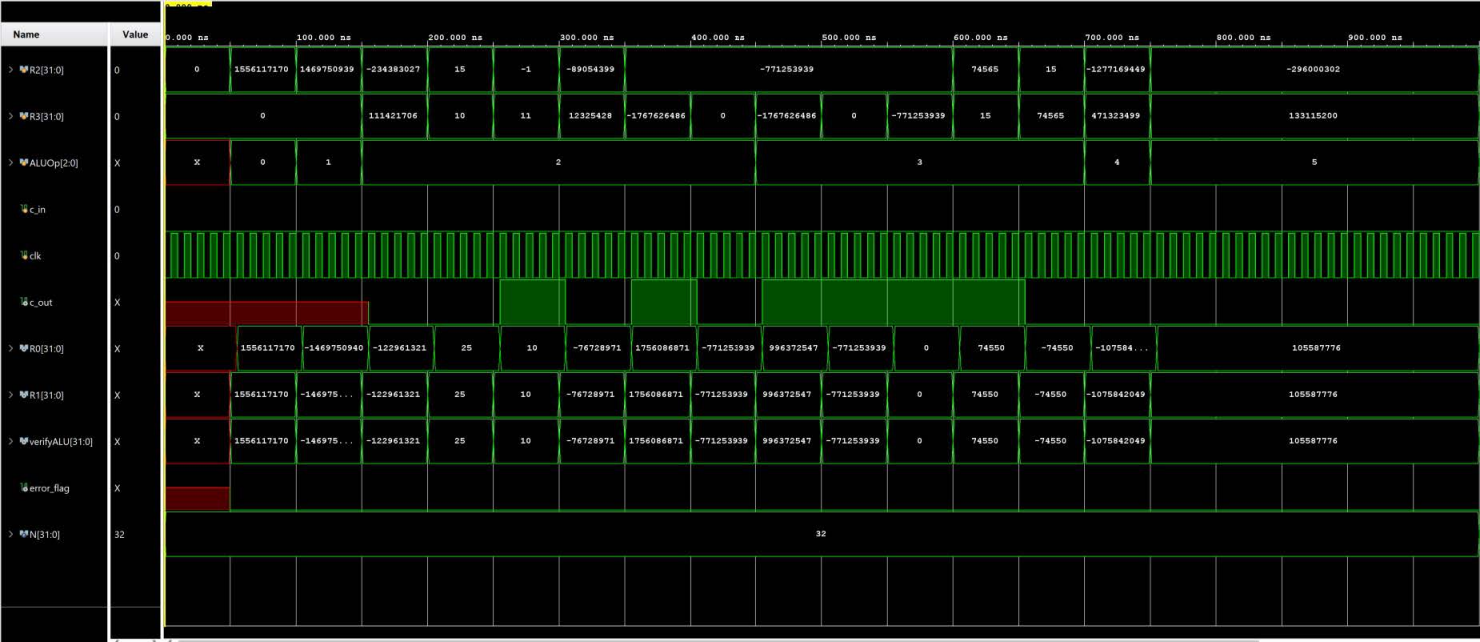
Uses a D Flip-Flop to store the value of the input inval into the output outval. The ALU uses this to store the value of R1 into the register R0.

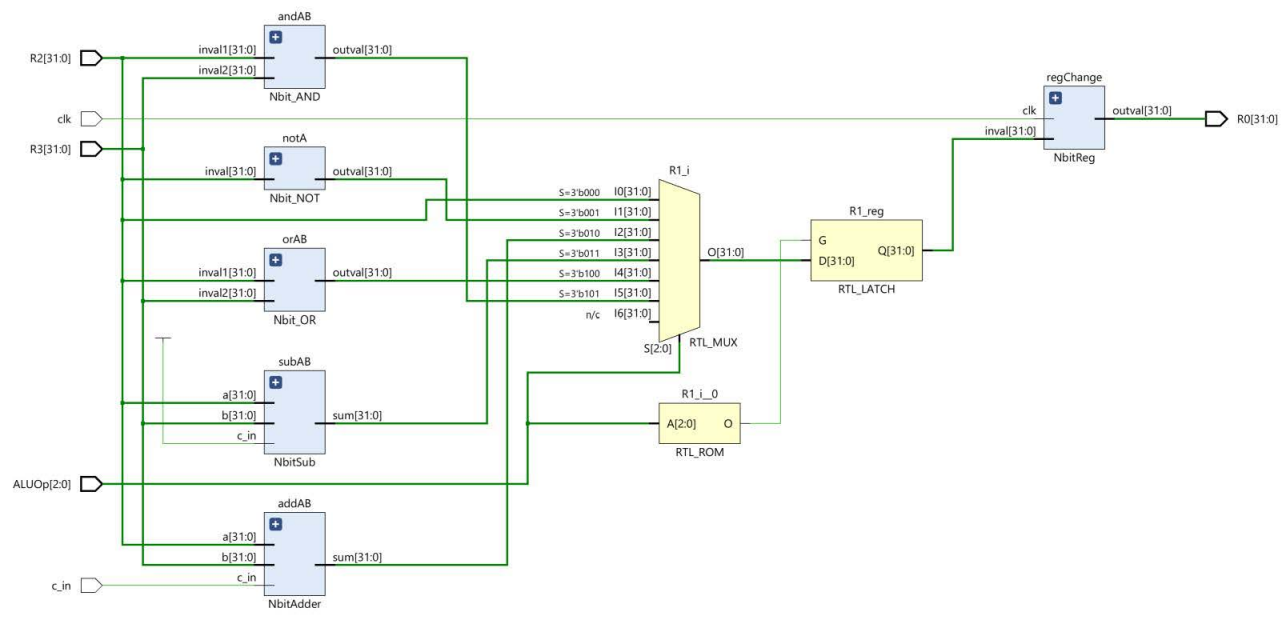
Overall Design Hierarchy:

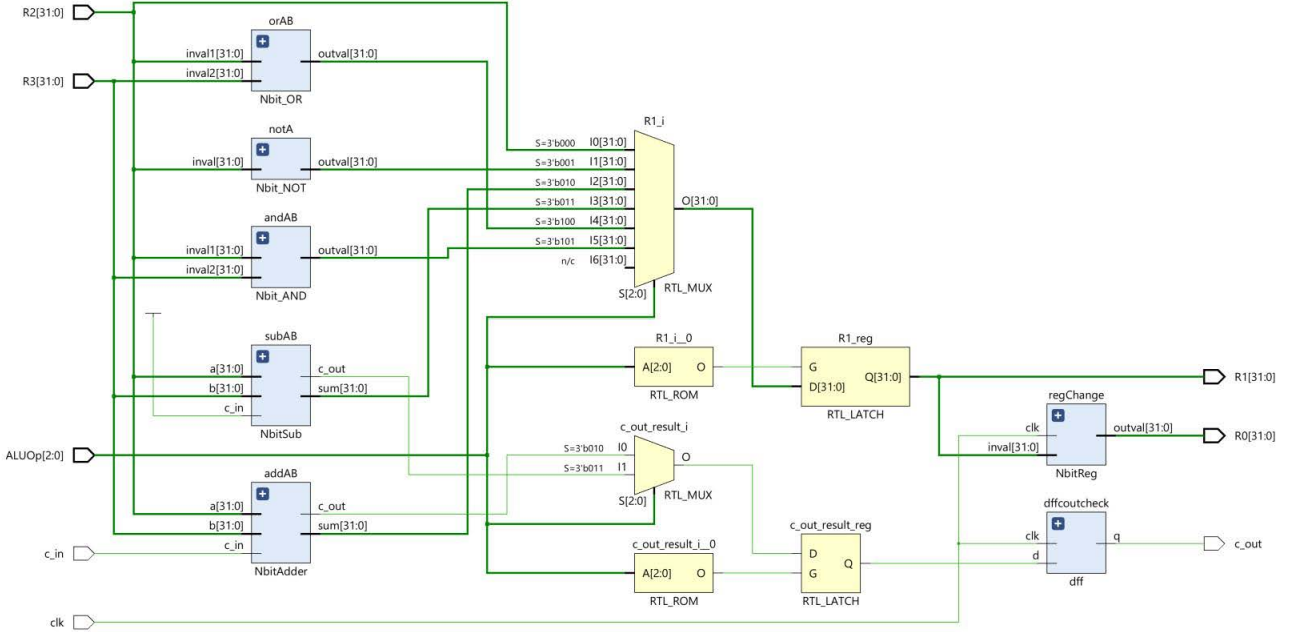
1. ALUtopModule
    a. Nbit_NOT
    b. NbitAdder
        i. FA_str
    c. NbitSub
        i. Nbit_NOT
        ii. FA_str
    d. Nbit_OR
    e. Nbit_AND
    f. NbitReg
        i. Dff

ALU Operations:

ALUOp	Function
3'b000	MOV
3'b001	NOT
3'b010	ADD
3'b011	SUB
3'b100	OR
3'b101	AND
3'b110	SLT


Name
Value
0.000 ns
100.000 ns
200.000 ns
300.000 ns
400.000 ns
500.000 ns
600.000 ns
700.000 ns
800.000 ns
900.000 ns
R2[31:0]
0
0
1556117170
1469750939
-234383027
15
-1
-89054399
-771253939
74565
15
-1277169449
-296000302
R3[31:0]
0
0
111421706
10
11
12325428
-1767626486
0
-1767626486
0
-771253939
15
74565
471323499
133115200
ALUOp[2:0]
X
X
0
1
2
3
4
5
c_in
0
clk
0
c_out
X
R0[31:0]
X
X
1556117170
-1469750940
-122961321
25
10
-76728971
1756086871
-771253939
996372547
-771253939
0
74550
-74550
-107584...
105587776
R1[31:0]
X
X
1556117170
-146975...
-122961321
25
10
-76728971
1756086871
-771253939
996372547
-771253939
0
74550
-74550
-1075842049
105587776
verifyALU[31:0]
X
X
1556117170
-146975...
-122961321
25
10
-76728971
1756086871
-771253939
996372547
-771253939
0
74550
-74550
-1075842049
105587776
error_flag
X
N[31:0]
32
32


andAB
inval1[31:0]
inval2[31:0]
outval[31:0]
Nbit_AND
notA
inval[31:0]
outval[31:0]
Nbit_NOT
orAB
inval1[31:0]
inval2[31:0]
outval[31:0]
Nbit_OR
subAB
a[31:0]
b[31:0]
c_in
sum[31:0]
NbitSub
addAB
a[31:0]
b[31:0]
c_in
sum[31:0]
NbitAdder
R2[31:0]
clk
R3[31:0]
ALUOp[2:0]
c_in
R1_i
S=3'b000 I0[31:0]
S=3'b001 I1[31:0]
S=3'b010 I2[31:0]
S=3'b011 I3[31:0]
S=3'b100 I4[31:0]
S=3'b101 I5[31:0]
n/c I6[31:0]
S[2:0]
O[31:0]
RTL_MUX
R1_i_0
A[2:0]
O
RTL_ROM
R1_reg
G
D[31:0]
Q[31:0]
RTL_LATCH
regChange
clk
inval[31:0]
outval[31:0]
NbitReg
R0[31:0]


R2[31:0]
R3[31:0]
orAB
inval1[31:0]
inval2[31:0]
outval[31:0]
Nbit_OR
notA
inval[31:0]
outval[31:0]
Nbit_NOT
andAB
inval1[31:0]
inval2[31:0]
outval[31:0]
Nbit_AND
subAB
a[31:0]
b[31:0]
c_in
c_out
sum[31:0]
NbitSub
ALUOp[2:0]
addAB
a[31:0]
b[31:0]
c_in
c_out
sum[31:0]
NbitAdder
c_in
clk
R1_i
S=3'b000 I0[31:0]
S=3'b001 I1[31:0]
S=3'b010 I2[31:0]
S=3'b011 I3[31:0]
S=3'b100 I4[31:0]
S=3'b101 I5[31:0]
n/c I6[31:0]
O[31:0]
S[2:0]
RTL_MUX
R1_i_0
A[2:0]
O
RTL_ROM
R1_reg
G
D[31:0]
Q[31:0]
RTL_LATCH
R1[31:0]
regChange
clk
inval[31:0]
outval[31:0]
NbitReg
R0[31:0]
c_out_result_i
S=3'b010 I0
S=3'b011 I1
O
S[2:0]
RTL_MUX
c_out_result_reg
D
G
Q
RTL_LATCH
dffcoutcheck
clk
d
q
dff
c_out
c_out_result_i_0
A[2:0]
O
RTL_ROM