Intro to Processor Architecture

Assignment 1

ALU

Arshini Govindu - 2020102009

Sankeerthana Venugopal – 2020102008

APPROACH

The assignment consists of 4 modules with one wrapper module that is used to access the other 4 modules. The four modules are:

ADD

The ***ADD*** function adds two 64-bit numbers. There is a ***fulladder*** module that acts like a normal full adder. It adds two numbers ***a*** and ***b*** to give ***sum*** and a ***carry*** as output. This module is used in the ***ADD*** module in a for loop to perform this operation 64 times – one for every bit in both the numbers. The 65^{th} bit, if any, is stored in the final carry output ***Cout.*** The testbench for this code is also given as ***ADDTb.v***

SUB

The ***SUB*** function subtracts 64-bit numbers from another. There is a ***fulladder*** module that acts like a normal full adder. It adds two numbers ***a*** and ***b*** to give ***sum*** and a ***carry*** as output. This module is used in the ***SUB*** module in a for loop to perform this operation 64 times – one for every bit in both the numbers. There is a ***not*** gate as well. The 65^{th} bit, if any, is stored in the final carry output ***Cout.*** The testbench for this code is also given as ***SUBTb.v***

AND

The ***and*** function calculates the result when we and two 64-bit numbers. This operation is usually done using bitwise operator '&'. There is an ***andgate*** module that finds out the and of two numbers. It ands two numbers ***x*** and ***y*** to give ***z***. This module is used in the ***andfunc*** module in a for loop to perform this operation 64 times – one for every bit in both the numbers. The testbench for this code is also given as ***AndTb***.v

OR

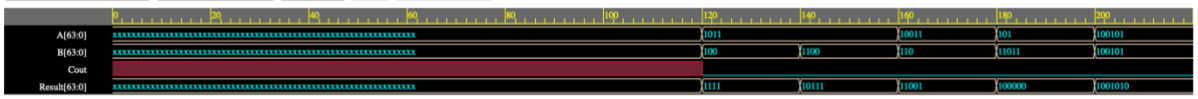
The ***or*** function calculates the result when we or two 64-bit numbers. This operation is usually done using bitwise operator '|'. There is an ***orgate*** module that finds out the or of two numbers. It ors two numbers ***x*** and ***y*** to give ***z***. This module is used in the ***orfunc*** module in a for loop to perform this operation 64 times – one for every bit in both the numbers. The testbench for this code is also given as ***OrTb.v***

WRAPPER ALU UNIT

The wrapper ALU unit consists of an ***ALU*** module which decides the operation to be done upon the two input 64-bit numbers. We used the case statements to decide which control input should be activated. The previous modules and functions are accessed using include statements.
The control input 1 allows the ***ADD*** function to run.
The control input 2 allows the ***SUB*** function to run.
The control input 3 allows the ***AND*** function to run.
The control input 4 allows the ***OR*** function to run.

RESULTS

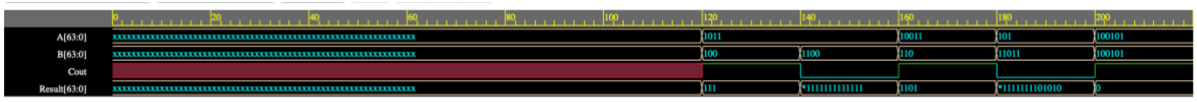
ADD



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

#0
A=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
B=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Cout=x
Result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#120
A=0000000000000000000000000000000000000000000000000000000000001011
B=0000000000000000000000000000000000000000000000000000000000000100
Cout=0
Result=0000000000000000000000000000000000000000000000000000000000001111
#140
A=0000000000000000000000000000000000000000000000000000000000001011
B=0000000000000000000000000000000000000000000000000000000000001100
Cout=0
Result=0000000000000000000000000000000000000000000000000000000000010111
#160
A=0000000000000000000000000000000000000000000000000000000000010011
B=0000000000000000000000000000000000000000000000000000000000000110
Cout=0
Result=0000000000000000000000000000000000000000000000000000000000011001
#180
A=0000000000000000000000000000000000000000000000000000000000000101
B=0000000000000000000000000000000000000000000000000000000000011011
Cout=0
Result=0000000000000000000000000000000000000000000000000000000000100000
#200
A=0000000000000000000000000000000000000000000000000000000000100101
B=0000000000000000000000000000000000000000000000000000000000100101
Cout=0
Result=0000000000000000000000000000000000000000000000000000000001001010

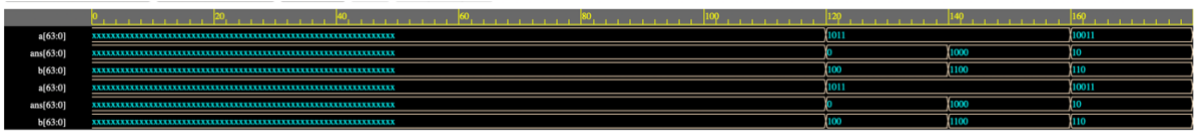
SUB



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

#0
A=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
B=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Cout=x
Result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#120
A=00000000000000000000000000000000000000000000000000000000000001011
B=00000000000000000000000000000000000000000000000000000000000000100
Cout=1
Result=00000000000000000000000000000000000000000000000000000000000000111
#140
A=00000000000000000000000000000000000000000000000000000000000001011
B=00000000000000000000000000000000000000000000000000000000000001100
Cout=0
Result=11111111111111111111111111111111111111111111111111111111111111111
#160
A=00000000000000000000000000000000000000000000000000000000000010011
B=00000000000000000000000000000000000000000000000000000000000000110
Cout=1
Result=00000000000000000000000000000000000000000000000000000000000001101
#180
A=00000000000000000000000000000000000000000000000000000000000000101
B=00000000000000000000000000000000000000000000000000000000000011011
Cout=0
Result=11111111111111111111111111111111111111111111111111111111111101010
#200
A=00000000000000000000000000000000000000000000000000000000000100101
B=00000000000000000000000000000000000000000000000000000000000100101
Cout=1
Result=00000000000000000000000000000000000000000000000000000000000000000

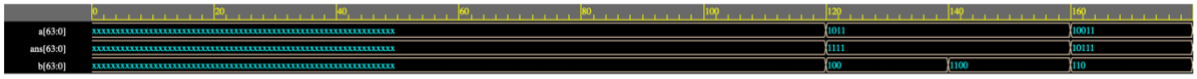
AND



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

#0
a=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
b=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ans=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#120
a=00000000000000000000000000000000000000000000000000000000000001011
b=00000000000000000000000000000000000000000000000000000000000000100
ans=00000000000000000000000000000000000000000000000000000000000000000
#140
a=00000000000000000000000000000000000000000000000000000000000001011
b=00000000000000000000000000000000000000000000000000000000000001100
ans=00000000000000000000000000000000000000000000000000000000000001000
#160
a=00000000000000000000000000000000000000000000000000000000000010011
b=00000000000000000000000000000000000000000000000000000000000000110
ans=00000000000000000000000000000000000000000000000000000000000000010
180
a=00000000000000000000000000000000000000000000000000000000000000101
b=00000000000000000000000000000000000000000000000000000000000011011
ans=00000000000000000000000000000000000000000000000000000000000000001

OR



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

#0
a=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
b=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ans=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#120
a=00000000000000000000000000000000000000000000000000000000000001011
b=00000000000000000000000000000000000000000000000000000000000000100
ans=00000000000000000000000000000000000000000000000000000000000001111
#140
a=00000000000000000000000000000000000000000000000000000000000001011
b=00000000000000000000000000000000000000000000000000000000000001100
ans=00000000000000000000000000000000000000000000000000000000000001111
#160
a=00000000000000000000000000000000000000000000000000000000000010011
b=00000000000000000000000000000000000000000000000000000000000000110
ans=00000000000000000000000000000000000000000000000000000000000010111
#180
a=00000000000000000000000000000000000000000000000000000000000000101
b=00000000000000000000000000000000000000000000000000000000000011011
ans=00000000000000000000000000000000000000000000000000000000000011111