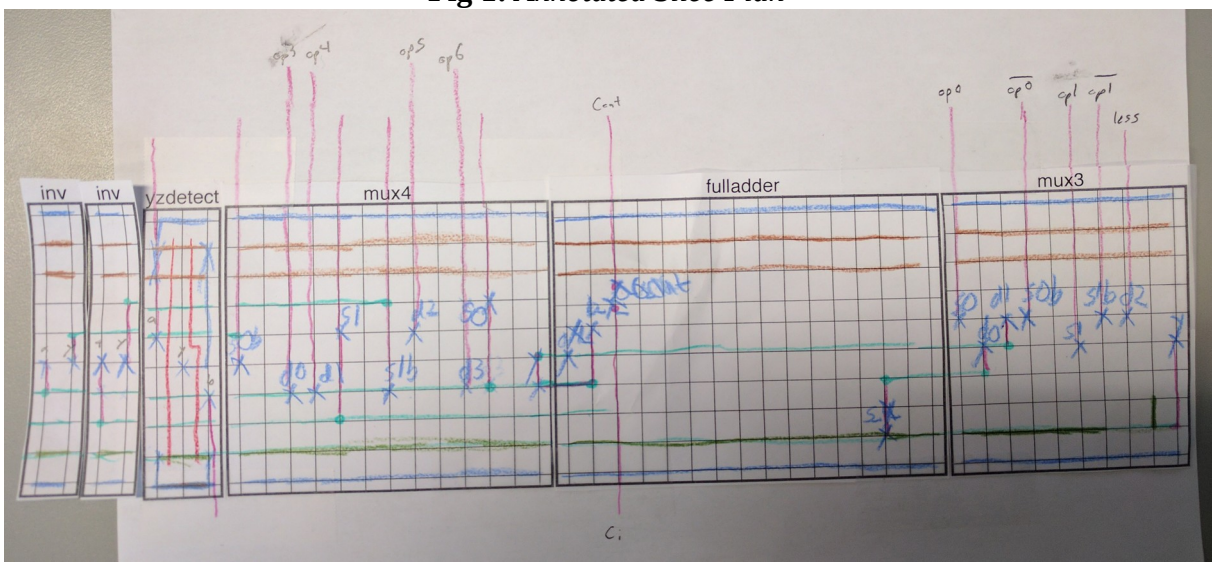
Ian D. Miller
ECE 425
Lab 4: Hierarchical Layout Extended ALU

Fig 1: *Annotated Slice Plan*



Note: In the final version, yzdetect has been swapped with the inverters.

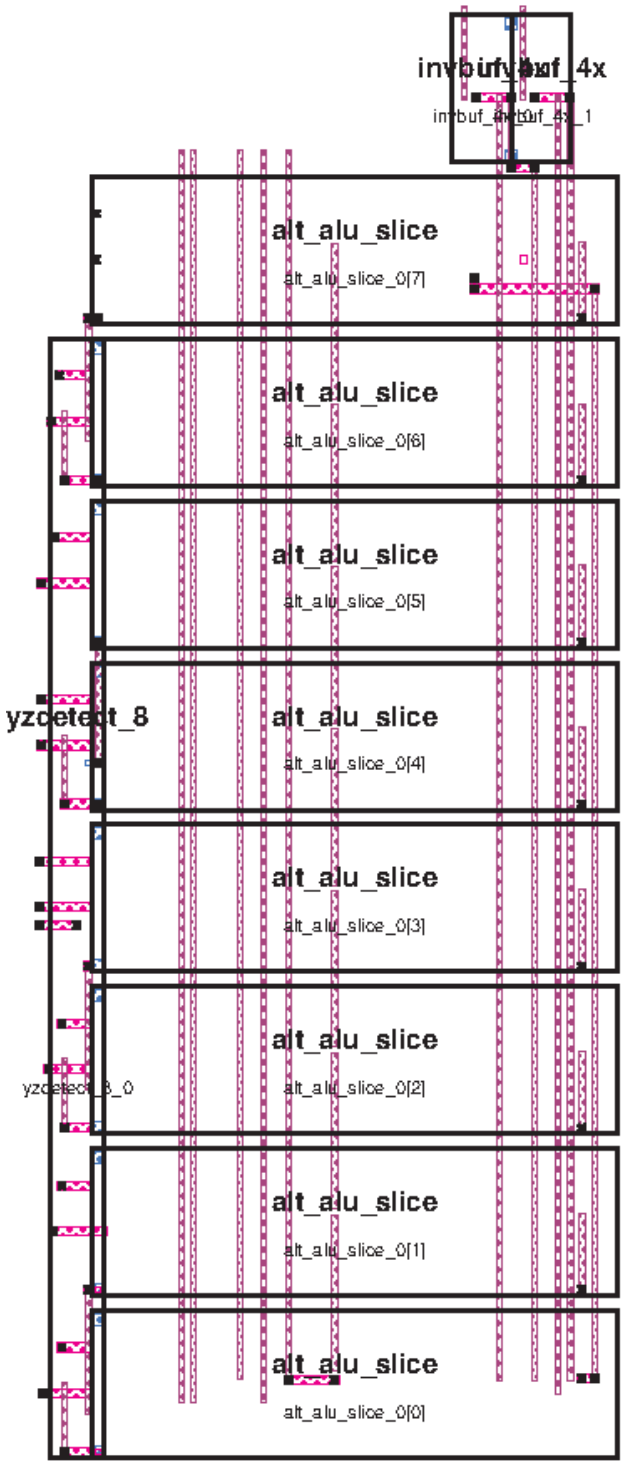


Fig. 2: *Unexpanded ALU Design*

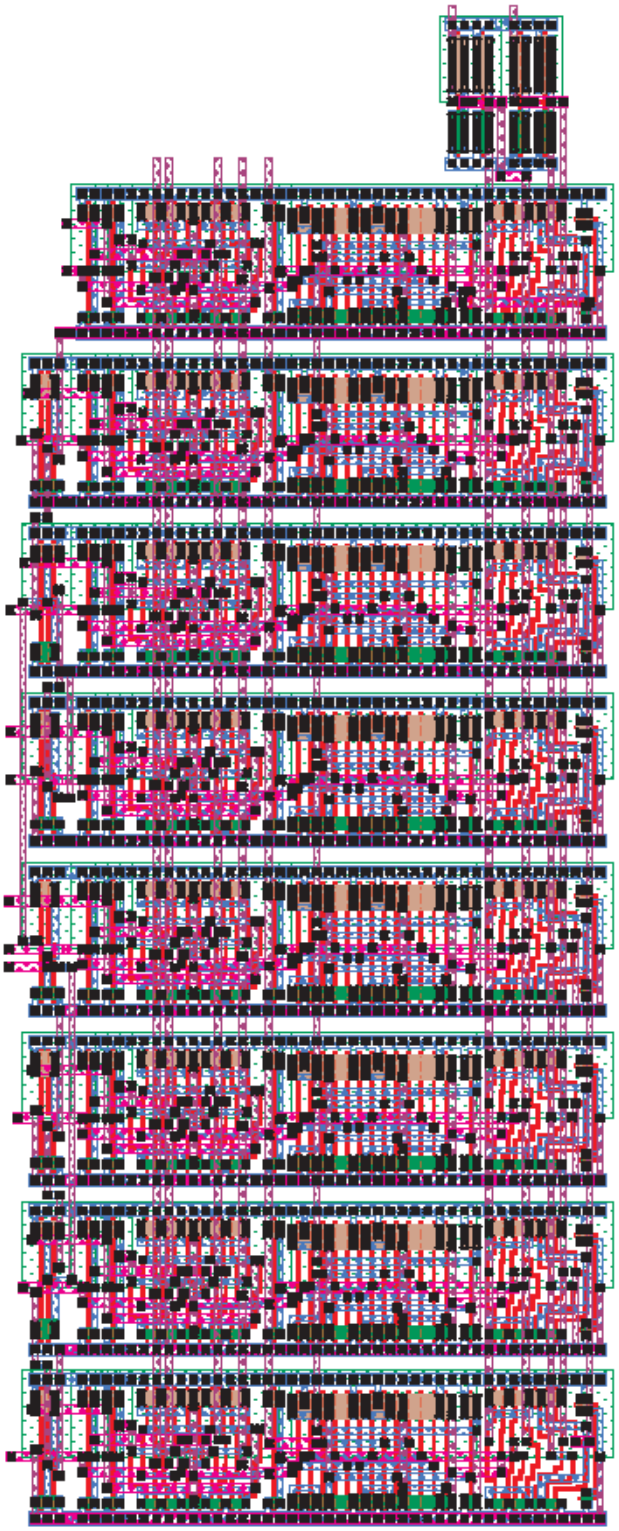


Fig. 3: *Expanded ALU Design*

The new ALU is 397x987 lambda, (or 119.1x296.1 um), while the old ALU was 410x986 lambda (123x295.9 um). Therefore, the new ALU has decreased in size by 12,421 labmda^2 (a 3% decrease). This isn't all that much, but the new ALU also introduced additional capabilities and opcodes, so it is impressive that we now have a device which has increased features at lower cost.

Testing

Our test python script (see Appendix A) generates a bunch of testvectors automatically. For every opcode, the 'a' input is run through every possible input (0x00 through 0xff). For each 'a' input, b is tested with values from 0x00 through 0xff, but increasing by 0x11 each time (a total of 16 tests). Therefore, each function was tested 3364 times. Increasing by 0x11 for b makes sure that at some point all bits will be one, which assures that all carry channels will be tested, and all functions will all be tested over a wide range of inputs. The test also only takes about a minute to run, which is much more efficient than running a completely exhaustive test.

Prepended to this file is some test code which simply sets up the testvector definitions and dumps things to a logfile.

Time to Complete Lab

This lab took about 8 hours to complete (2 lab periods and a couple hours outside of lab). Most of the time was taken up by dealing with Magic itself, as well as in debbugging.

Appendix A: *testgen.py*

```
oplist = [0b0001001, 0b0111001, 0b1000001, 0b0110001, 0b0101000,
0b1010100, 0b1010110]

f = open('tb.cmd', 'w')

index = 0
for op in oplist:
    f.write("setvector op "+format(op, '07b')+"\n")
    for a in range(0,0b11111111):
        f.write("setvector a "+format(a, '08b')+"\n")
        for b in range(0,0b11111111, 0b00010001):
            f.write("setvector b "+format(b, '08b')+"\n")
            if index == 0:
                result = a&b
            elif index == 1:
                result = a|b
            elif index == 2:
                #nor
                result = (a|b)^0b11111111
            elif index == 3:
                result = a^b
            elif index == 4:
                result = (a+b) % (1<<8)
            elif index == 5:
                result = (a-b) % (1<<8)
            elif index == 6:
                if (a<b):
                    result = 0b1
```

```
                else:
                    result = 0b0
            else:
                #this is bad
                print "ahhhhhhh"

            f.write("s\n")
            f.write("assert result "+format(result, '08b')+"\n")
            if (result == 0):
                f.write("assert zero 1\n")
            else:
                f.write("assert zero 0\n")

    index += 1

f.close()
```