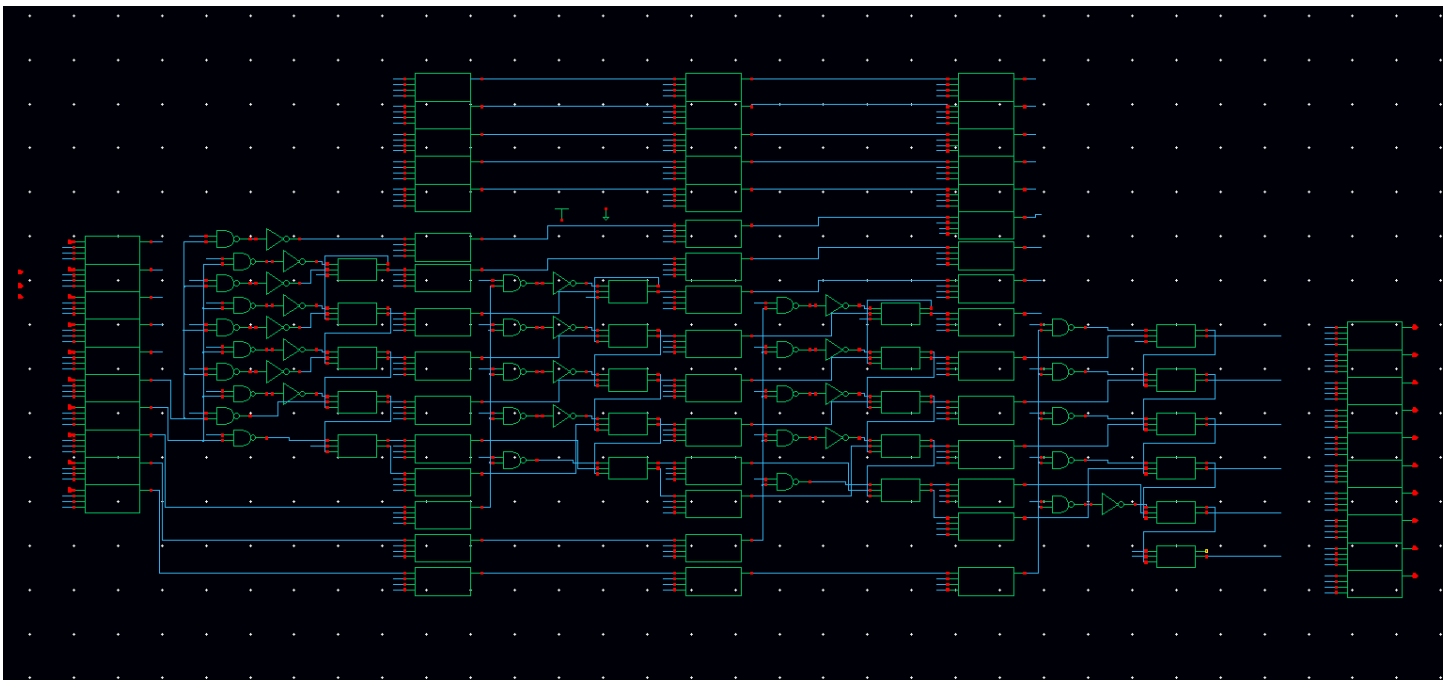
EE 577a Lab4

Spring 2017

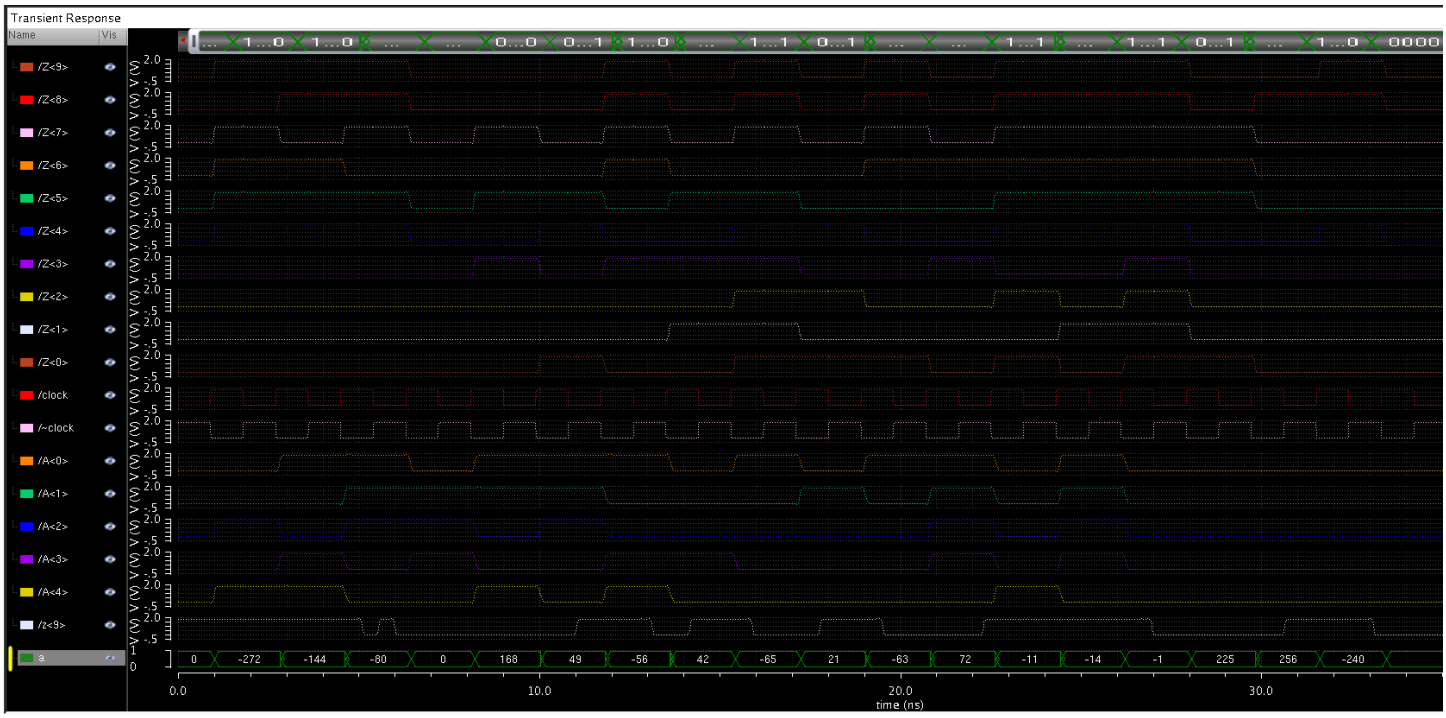
Huayu Fu
4745159848
3/24/2017

Part A: Pipelined 5 bit 2's complement Multiplier Design

a) Schematic of the 5-bit Pipelined 2's Complement Multiplier.



b) Functional test waveforms and results of the 5-bit Pipelined 2's Complement Multiplier for 10 random test cases and 5 directed test cases.

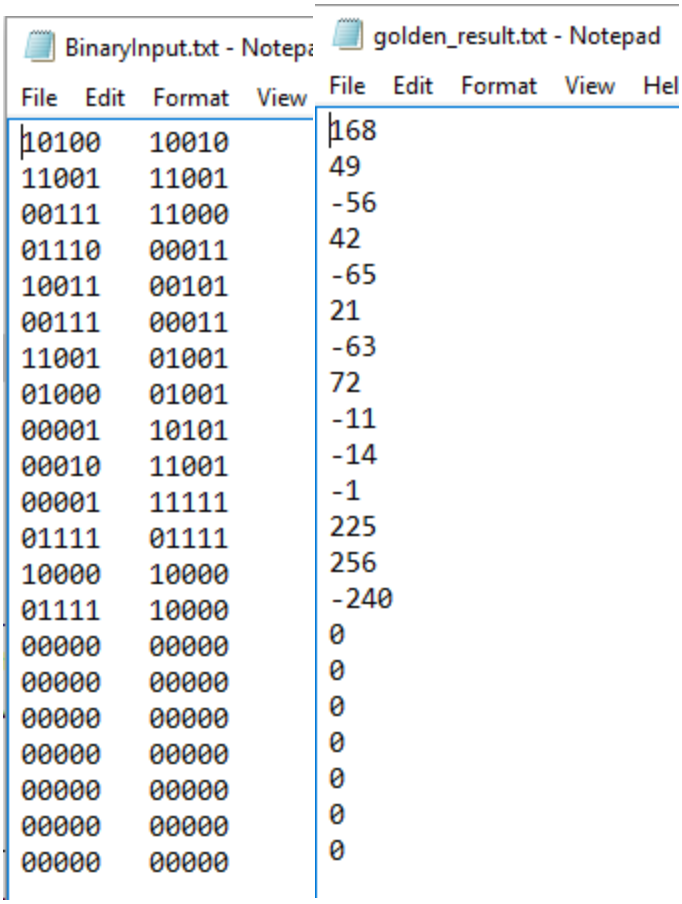


c) Write a front-end perl/python script to generate random test-cases and directed test cases and golden_results.txt that contains golden results for the 15 test cases. Also write a backend script to write the obtained results from cadence (.csv file) to actual_results.txt and compare golden results to actual results.

```
import random
def LFSR(start_state):
    start_state=start_state%32
    lfsr=start_state
    l=[]
    while True:
        bit=((lfsr>>0)^(lfsr>>2))&1
        lfsr=((lfsr)>>1)|((bit)<<4)
        l.append(lfsr)
        if(lfsr==start_state):
            break
    return l
def rawGen(number1,number2,out):

    if(number1>=16):
        number1=(number1%32)-32
    if(number2>=16):
        number2=(number2%32)-32
    number=number1*number2
    out.write(str(number)+"\n")


def main():
    out=open('golden_result.txt','w')
    out1=open('testcase.vec','w')
    out2=open('BinaryInput.txt','w')
    out1.write("radix\t1\t4\t1\t4\t1\t1\n")
    out1.write("io\ti\ti\ti\ti\ti\ti\n")
    out1.write("vname\ta<4>\ta<[3:0]>\tb<4>\tb<[3:0]>\tclock\t~clock\n")
    out1.write("slope\t0.01\n")
    out1.write("vih\t1.8\n")
    out1.write("tunit\tns\n")
    l=LFSR(47)
    period=1.8
    for i in range(10):
        o1a=str(i*period)+'\t'
        o1b=str((i+0.5)*period)+'\t'
        number1=l[random.randint(0,len(l)-1)]
        o1a+=str(number1/16)+'\t'
        o1a+=hex(number1%16)[2]+'\t'
        o1b+=str(number1/16)+'\t'
        o1b+=hex(number1%16)[2]+'\t'
        for x in range(4,-1,-1):
            bit=(number1%(pow(2,x+1)))/pow(2,x)
            if(x==0):
                out2.write(str(bit)+'\t')
            else:
                out2.write(str(bit))
        number2=l[random.randint(0,len(l)-1)]
        o1a+=str(number2/16)+'\t'
        o1a+=hex(number2%16)[2]+'\t'
        o1b+=str(number2/16)+'\t'
        o1b+=hex(number2%16)[2]+'\t'
        o1a+="0\t1\n"
        o1b+="1\t0\n"
        out1.write(o1a)
        out1.write(o1b)
        for x in range(4,-1,-1):
            bit=(number2%(pow(2,x+1)))/pow(2,x)
            if(x==0):
                out2.write(str(bit)+'\n')
            else:
                out2.write(str(bit))
        rawGen(number1,number2,out)
    a=[1,15,16,15,0,0,0,0,0,0,0]
    b=[31,15,16,16,0,0,0,0,0,0,0]
    for i in range(len(a)):
        o1a=str((i+10)*period)+'\t'
        o1b=str((i+10.5)*period)+'\t'
        number1=a[i]
        o1a+=str(number1/16)+'\t'
        o1a+=hex(number1%16)[2]+'\t'
        o1b+=str(number1/16)+'\t'
        o1b+=hex(number1%16)[2]+'\t'
        for x in range(4,-1,-1):
            bit=(number1%(pow(2,x+1)))/pow(2,x)
            if(x==0):
                out2.write(str(bit)+'\t')
            else:
                out2.write(str(bit))
        number2=b[i]
        o1a+=str(number2/16)+'\t'
        o1a+=hex(number2%16)[2]+'\t'
        o1b+=str(number2/16)+'\t'
        o1b+=hex(number2%16)[2]+'\t'
        o1a+="0\t1\n"
        o1b+="1\t0\n"
        out1.write(o1a)
        out1.write(o1b)
        for x in range(4,-1,-1):
            bit=(number2%(pow(2,x+1)))/pow(2,x)
            if(x==0):
                out2.write(str(bit)+'\n')
            else:
                out2.write(str(bit))
        rawGen(number1,number2,out)
    out.close()
    out1.close()
    out2.close()
if __name__ == "__main__": main()
```

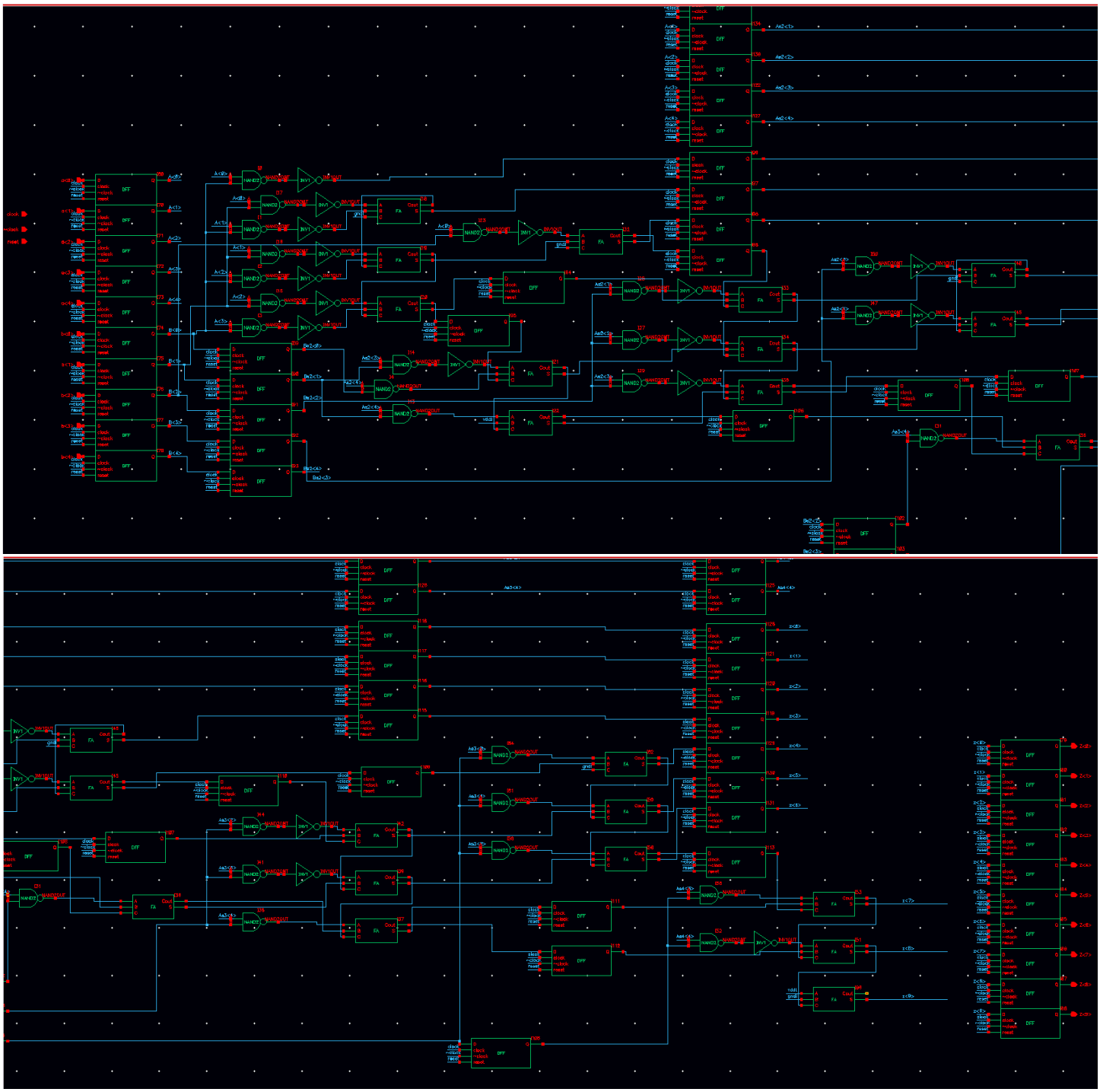


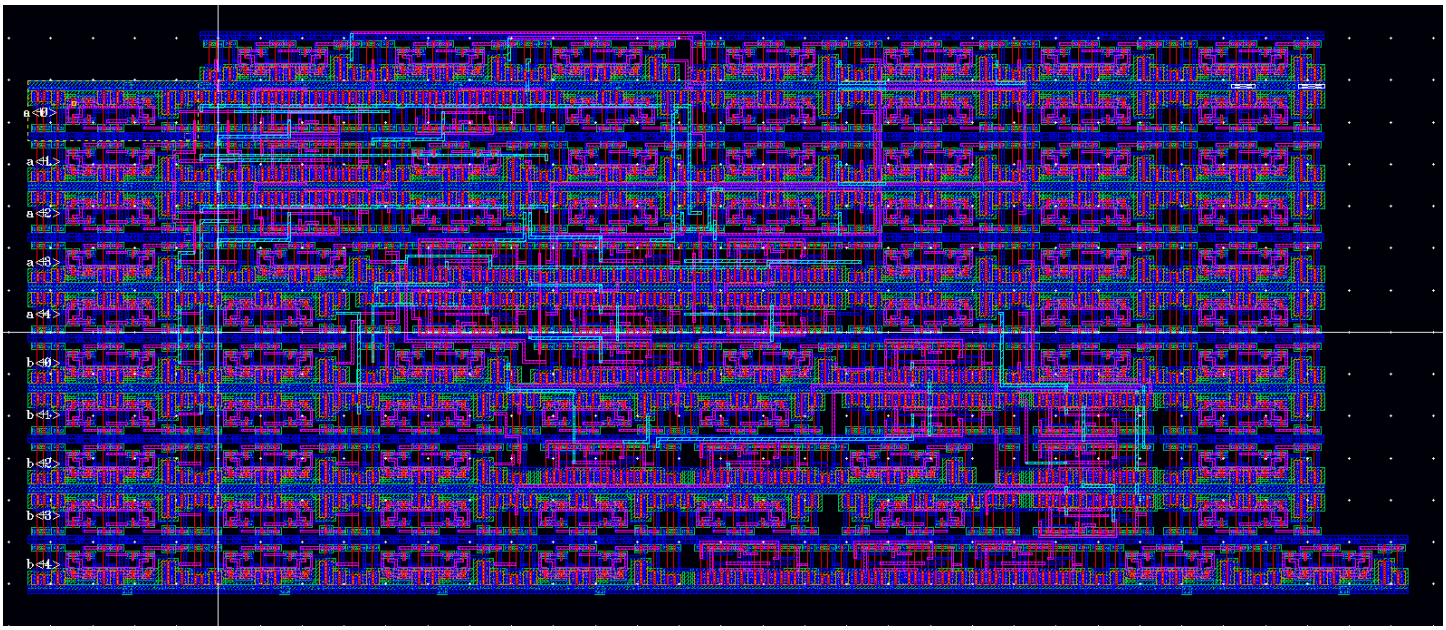
d) Report the minimal clock period that you can achieve.

2ns

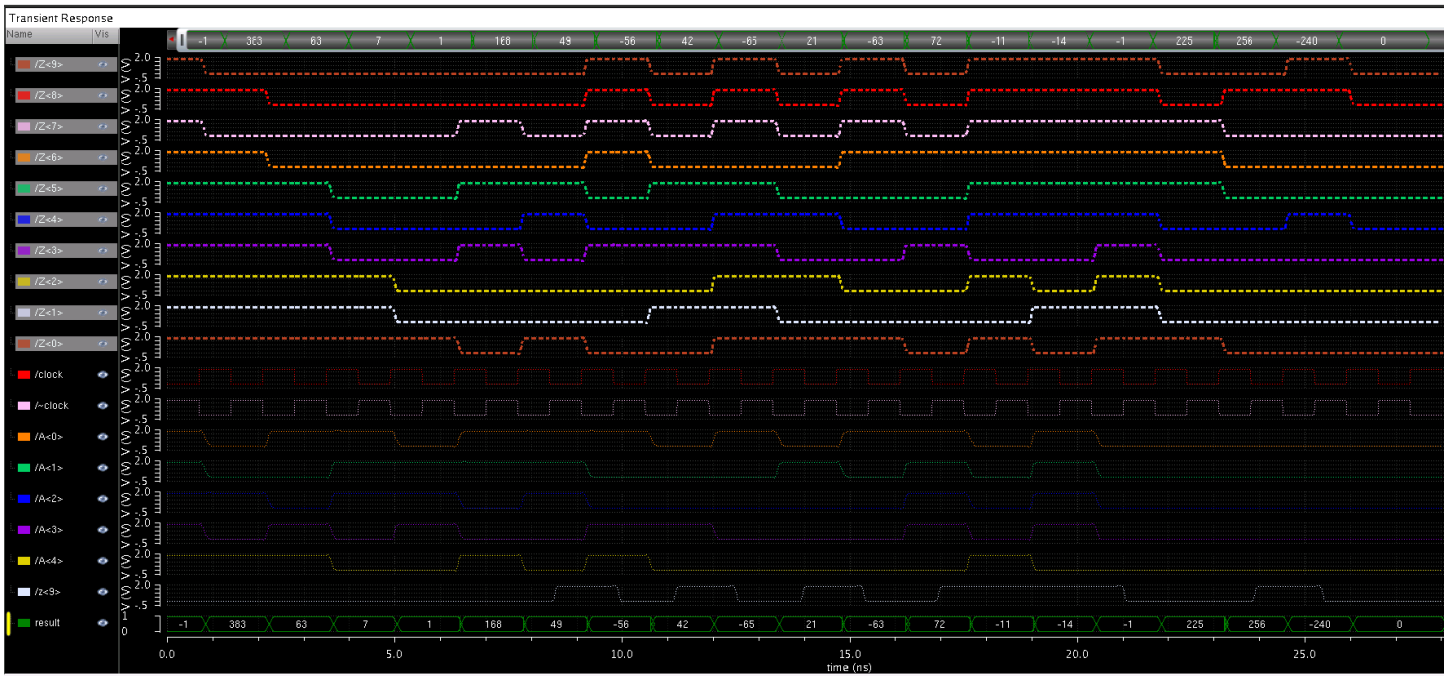
Part B. Pipelined 5-bit 2's Complement Multiplier Optimization

e) Schematicand layoutof the optimized 5-bit Pipelined 2's Complement Multiplier.





f) Functional testwaveforms and resultsof the 5-bit Pipelined 2's Complement Multiplierfor 10 random test casesand 5 directed test cases.The 15test cases should be the sameas what you have used in Part A.



g) Report the minimal clock period that you can achieveafter optimization.

1.4ns

h) Compare the minimal clock period you achieved in part A and B. Explain the reason why the clock period can be reducedafter optimization.
For part A 1.8ns for Part B 1.4ns,22%improvement, since the critical path is optimized. Each stage a signal goes to at most 3 full adder.

i) Compare the number of Flip-Flops in part A and B and explain the difference.
PA=142-89+1=53 PB=141-89+1=52 part B saves 1 flip flops