Design basic digital logic circuits like logic gates, adders, and multiplexers using Verilog within the VLSI software. Simulate the Verilog designs to ensure correct functionality. Use the waveform viewer in the VLSI software to analyze the simulation results.//giva code in python//

Input program

# AND Gate

def and\_gate(a, b):

return a & b # Logical AND

# OR Gate

def or\_gate(a, b):

return a | b # Logical OR

# NOT Gate

def not\_gate(a):

return ~a & 1 # Logical NOT (only taking least significant bit)

# XOR Gate

def xor\_gate(a, b):

return a ^ b # Logical XOR

# Test logic gates

if \_\_name\_\_ == "\_\_main\_\_":

a, b = 1, 0

print(f"AND({a}, {b}) = {and\_gate(a, b)}")

print(f"OR({a}, {b}) = {or\_gate(a, b)}")

print(f"NOT({a}) = {not\_gate(a)}")

print(f"XOR({a}, {b}) = {xor\_gate(a, b)}")

# Half Adder

def half\_adder(a, b):

sum = xor\_gate(a, b) # Sum is A XOR B

carry = and\_gate(a, b) # Carry is A AND B

return sum, carry

# Test half adder

if \_\_name\_\_ == "\_\_main\_\_":

a, b = 1, 1

sum, carry = half\_adder(a, b)

print(f"Half Adder: A = {a}, B = {b}, Sum = {sum}, Carry = {carry}")

# Full Adder

def full\_adder(a, b, cin):

sum1, carry1 = half\_adder(a, b)

sum2, carry2 = half\_adder(sum1, cin)

carry\_out = or\_gate(carry1, carry2)

return sum2, carry\_out

# Test full adder

if \_\_name\_\_ == "\_\_main\_\_":

a, b, cin = 1, 1, 1

sum, carry\_out = full\_adder(a, b, cin)

print(f"Full Adder: A = {a}, B = {b}, Cin = {cin}, Sum = {sum}, Carry Out = {carry\_out}")

# 4-to-1 Multiplexer

def mux\_4to1(d0, d1, d2, d3, s0, s1):

if s0 == 0 and s1 == 0:

return d0

elif s0 == 0 and s1 == 1:

return d1

elif s0 == 1 and s1 == 0:

return d2

else:

return d3

# Test 4-to-1 Multiplexer

if \_\_name\_\_ == "\_\_main\_\_":

d0, d1, d2, d3 = 0, 1, 0, 1

s0, s1 = 1, 0 # Select line

output = mux\_4to1(d0, d1, d2, d3, s0, s1)

print(f"4-to-1 MUX Output: D0 = {d0}, D1 = {d1}, D2 = {d2}, D3 = {d3}, S0 = {s0}, S1 = {s1}, Output = {output}")

def simulate\_full\_adder():

print(f"{'A':>2} {'B':>2} {'Cin':>3} {'Sum':>3} {'Cout':>5}")

for a in [0, 1]:

for b in [0, 1]:

for cin in [0, 1]:

sum, cout = full\_adder(a, b, cin)

print(f"{a:>2} {b:>2} {cin:>3} {sum:>3} {cout:>5}")

if \_\_name\_\_ == "\_\_main\_\_":

simulate\_full\_adder()

output program

AND(1, 0) = 0

OR(1, 0) = 1

NOT(1) = 0

XOR(1, 0) = 1

Half Adder: A = 1, B = 1, Sum = 0, Carry = 1

Full Adder: A = 1, B = 1, Cin = 1, Sum = 1, Carry Out = 1

4-to-1 MUX Output: D0 = 0, D1 = 1, D2 = 0, D3 = 1, S0 = 1, S1 = 0, Output = 0

A B Cin Sum Cout

0 0 0 0 0

0 0 1 1 0

0 1 0 1 0

0 1 1 0 1

1 0 0 1 0

1 0 1 0 1

1 1 0 0 1

1 1 1 1 1