Lab Assignment-7

Name – Md Junaid Mahmood

Enrolment Number – 19116040

Department – Computer Science & Engineering

Year – 2nd Year

Part 1 (ReLU Function)

(i) Now the number is given in 2's complement form and we need to detect whether it is positive or not. Let A = A_3 A_2 A_1 A_0, be the input 4-bit number. Here, A_3 is the most significant bit of A. Boolean logic for achieving this functionality is: **$Y = A_3'$**.

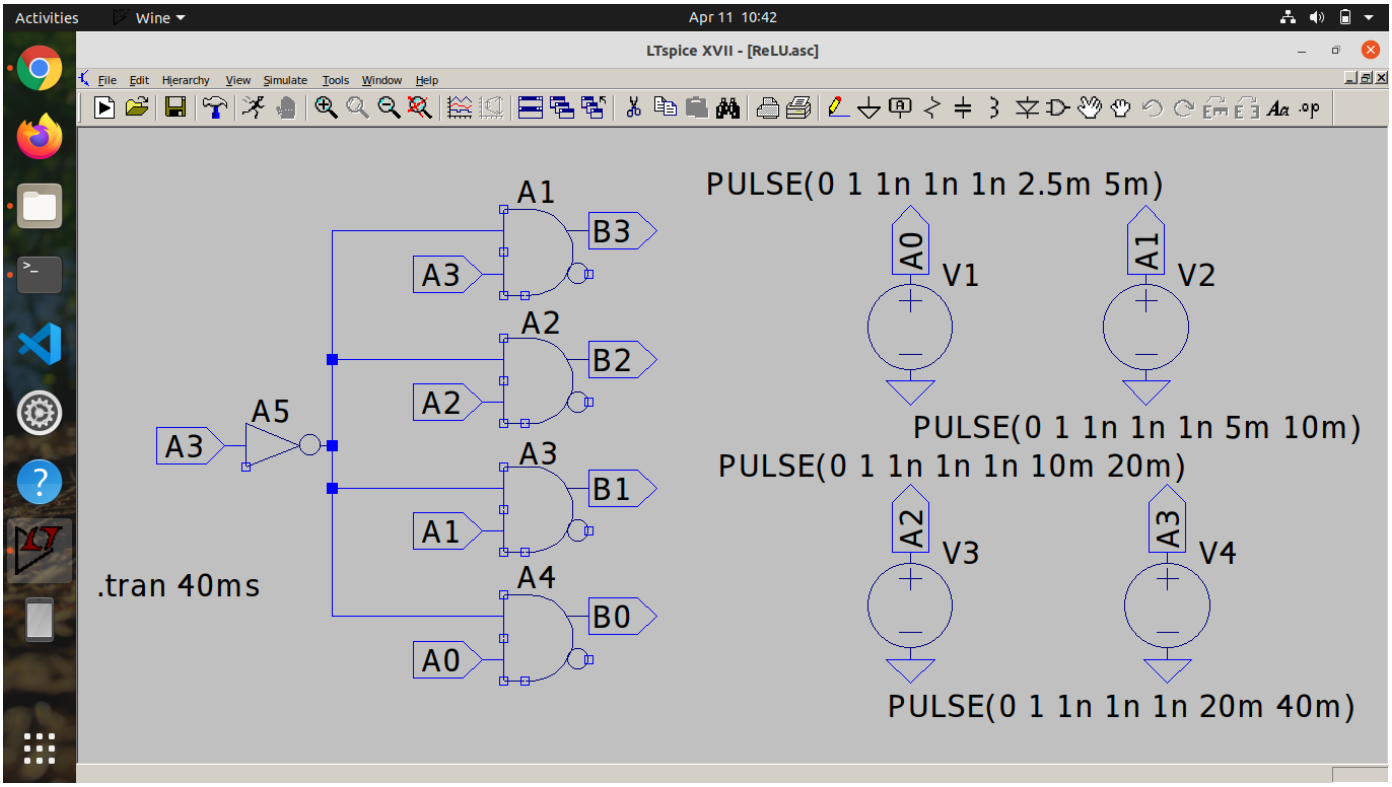
When A is negative, A_3 = 1. Thus, Y = 0, showing that the number isn't positive. When A is positive, A_3 = 0. Thus, Y = 1, showing that the number is positive.

(ii) For implementing the ReLU Function, we have assumed that A = A_3 A_2 A_1 A_0 is the input 4-bit number and B = B_3 B_2 B_1 B_0 is the output 4-bit number. Boolean Logic for achieving ReLU Function is:

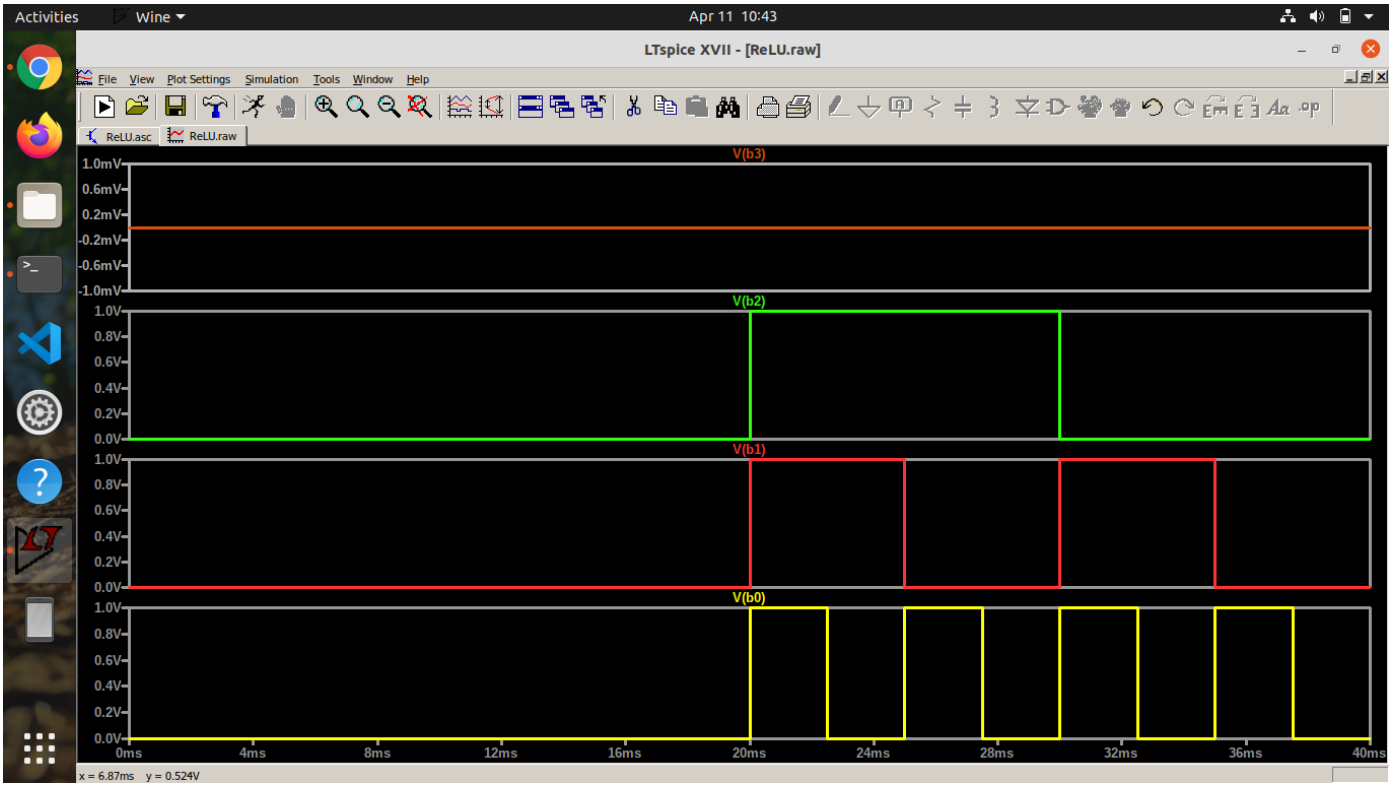
$B_3 = A_3' \,\&\, A_3$
$B_2 = A_3' \,\&\, A_2$
$B_1 = A_3' \,\&\, A_1$
$B_0 = A_3' \,\&\, A_0$

Here & represents and gate and A_3' represents complement of the bit A_3. Using this Boolean logic, circuit was created. For simulation, for the first 20ms, value of A was kept between -1 to -8 starting from -1 and ending at -8. For next 20ms, value of A was kept between 7 to 0 starting from 7 and ending at 0. Result of the simulation along with the circuit are attached below.

Circuit Diagram for the ReLU Function is:



Corresponding Simulation Result are:



Part 2 (Training an overly Simplistic Perceptron)

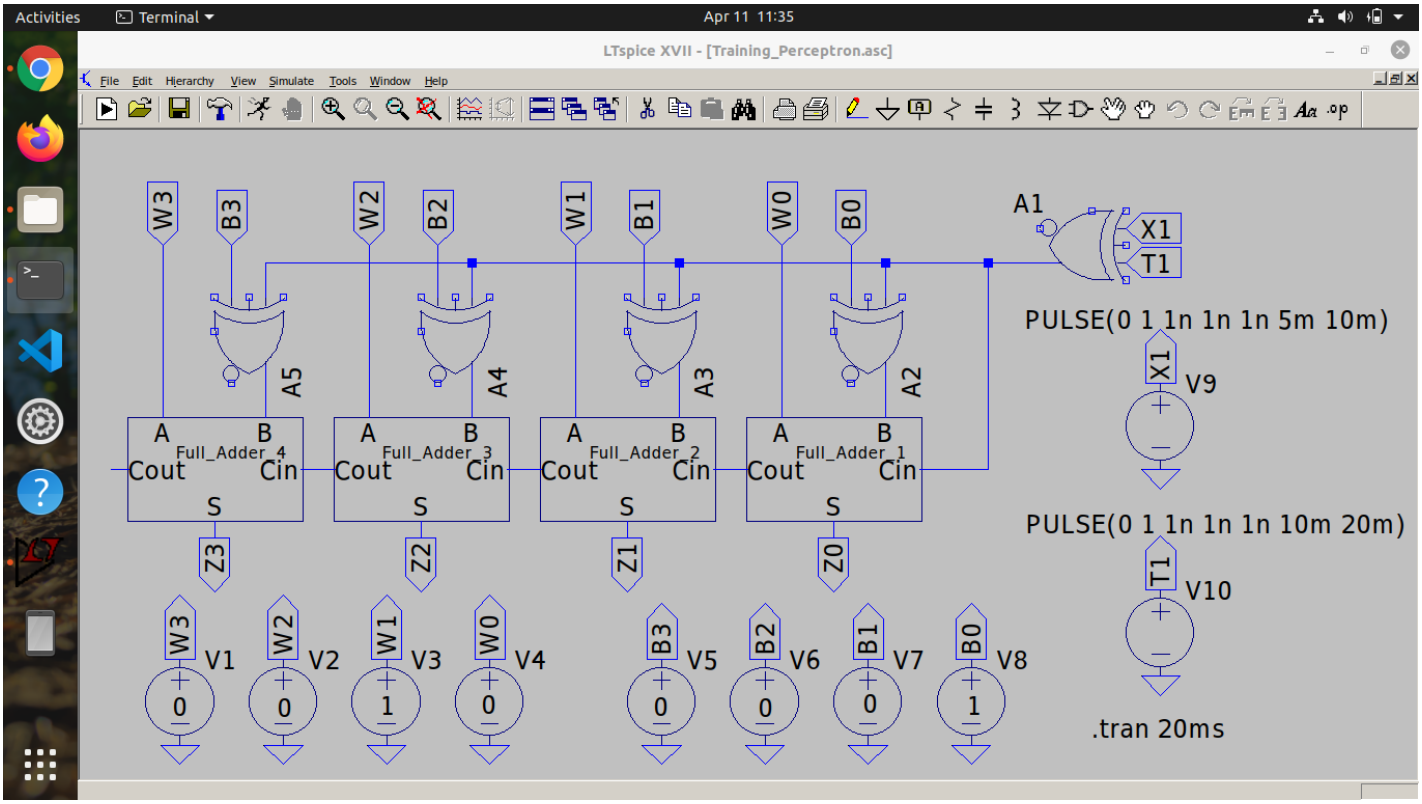
In this section, we have adopted following convention, $X = X_1\ X_0$, $T = T_1\ T_0$ and $W = W_3\ W_2\ W_1\ W_0$. In addition, an additional 4-bit number B was used for representing $(1)_{10}$, such that $B = B_3\ B_2\ B_1\ B_0 = (0001)_2$. The output for the circuit was shown using a 4-bit number $Z = Z_3\ Z_2\ Z_1\ Z_0$. The value of W was kept constant at 2. Now, $(1)_{10} = (01)_2$ and $(-1)_{10} = (11)_2$. Thus, only Most Significant Bit of X and T would vary. Boolean logic for achieving this functionality is:

At first let Y_1 be an intermediate value such that **$Y_1 = X_1$ xor T_1**.

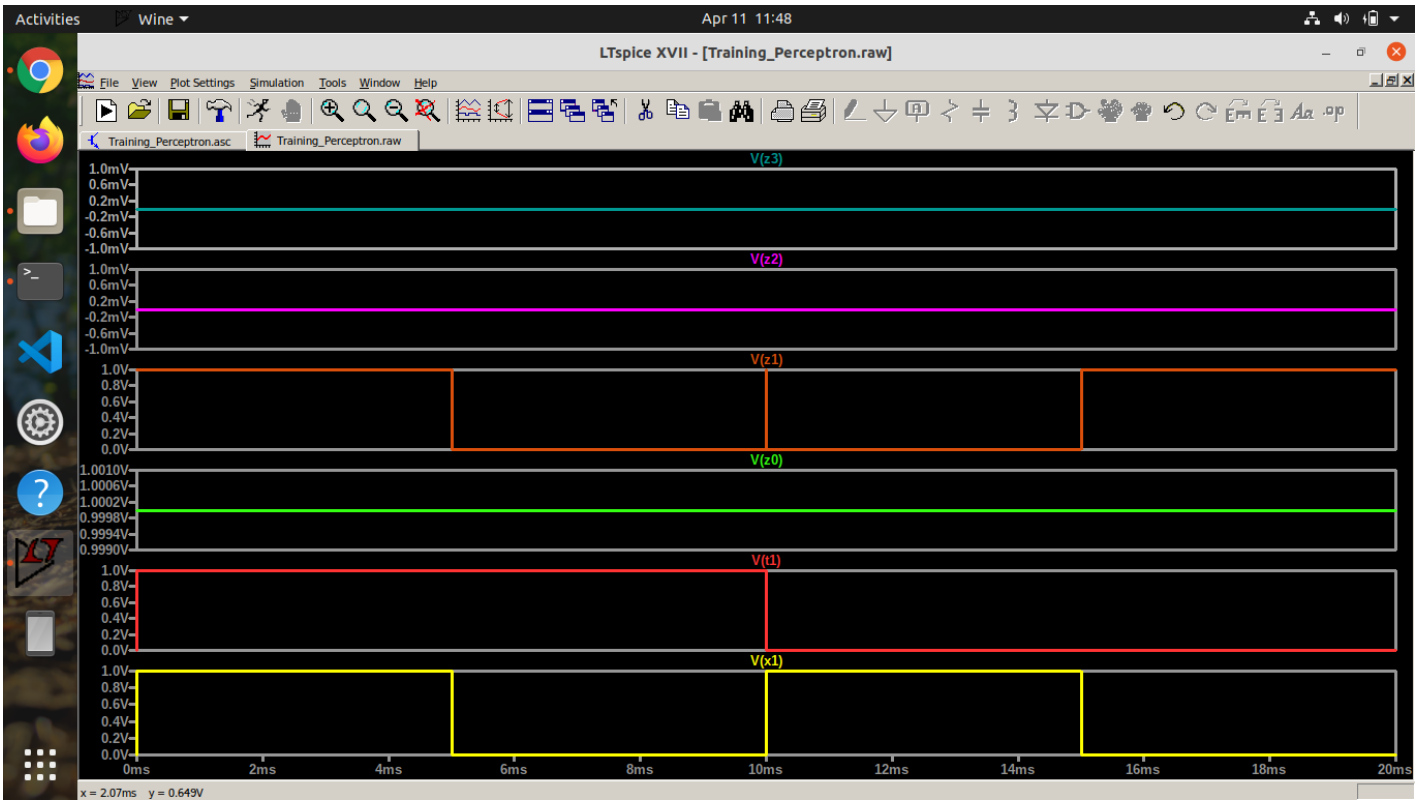
1. If $X_1 = T_1$, then $Y_1 = 0$. Thus, input carry to the first adder is 0. In addition, we have: **B_j xor $Y_1 = B_j$ xor 0 = B_j for all j = 0, 1, 2, 3**. Thus, we are simply doing the addition of W with B (where W = 2 and B = 1).

2. If $X_1 \neq T_1$, then $Y_1 = 1$. Thus, input carry to the first adder is 1. In addition, we have: **B_j xor $Y_1 = B_j$ xor 1 = B_j' for all j = 0, 1, 2, 3**, where B_j' represents the complement of bit B_j.Thus, we are simply subtracting B from W (where B = 1 and W = 2).

In this way, the required functionality was achieved. Circuit Diagram corresponding to above logic is:



Corresponding Simulation Result are:



In above simulation value of X_1 is represented by yellow line and that of T_1 is represented by red line. Following results were obtained:

1. During 5ms to 15ms value of X_1 and T_1 are different. Thus, output is given by $Z = (0001)_2$.
2. During 0ms to 5ms and 15ms to 20ms value of X_1 and T_1 are same. Thus, output is given by $Z = (0011)_2$.

Thus, above simulation results are correct.

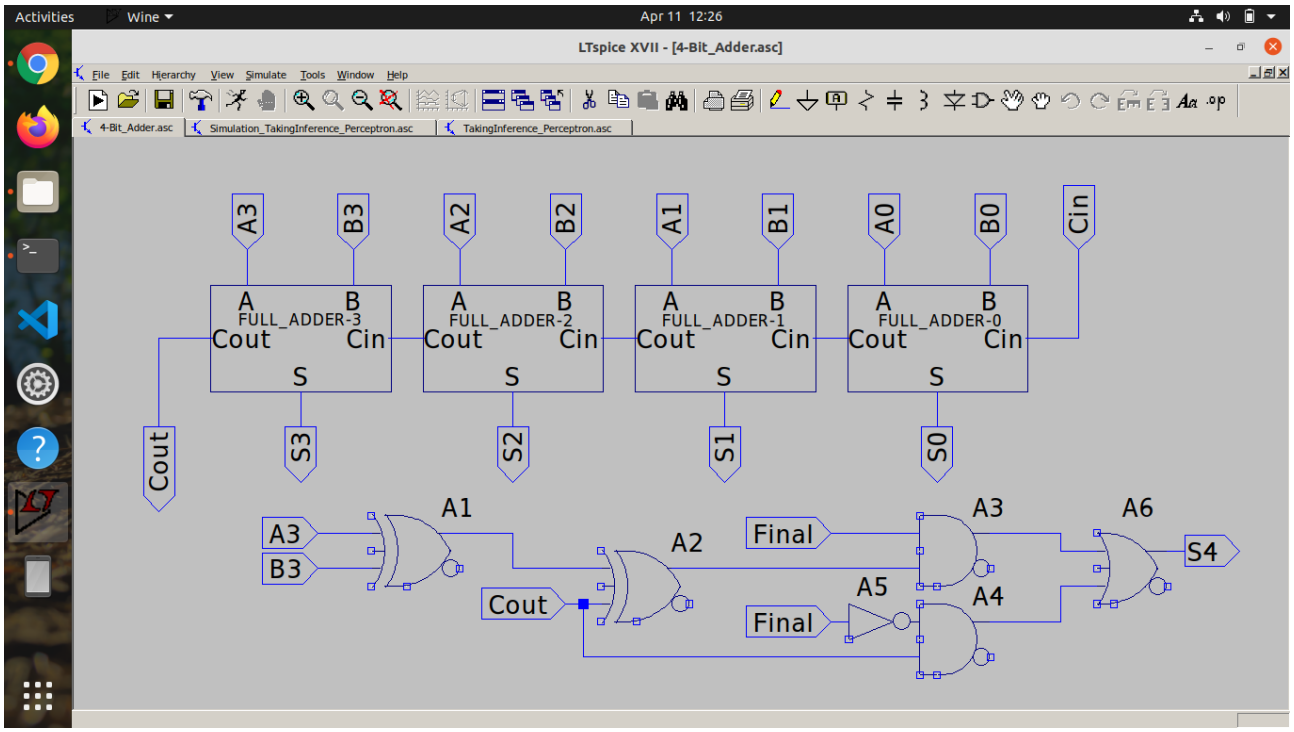
Part 3 (Doing inference with an overly Simplistic Perceptron)

Now, mod(19116040, 4) + 1 = 1. Thus, set 1 corresponding to CSE group was chosen. Thus, equation of straight line is X + 3Y – 3 = 0. Comparing this equation with Ax + BY + C = 0, we get: $A = (1)_{10} = (0001)_2$, $B = (3)_{10} = (0011)_2$ and constant term $C = (-3)_{10} = (1101)_2$. The two points, which were chosen for testing the circuit is (X, Y) = (-4, 1) and (4, 1). Following steps are involved in the construction of the above circuit:

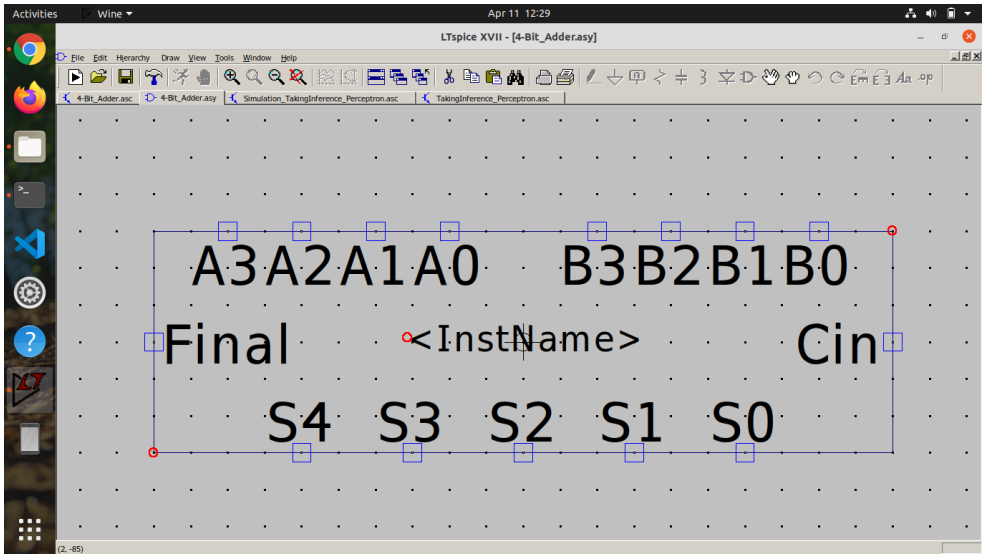
1. Using Bough-Wooley Multiplier, 8-bit product for A.X and B.Y was obtained.

2. To add two 8-bit numbers, a 8-bit adder was designed using 4-bit adder as the building block. The 4-bit adder was designed using full adder as the building block. Description of the 4-bit adder is mentioned below:
    - The four full adders were arranged with output carry of one adder as the input carry of another adder as shown in the circuit below.
    - There is an input named *'Final'*. *'Final'* is kept 1 if the respective 4-bit adder is the last or final block in the construction of any higher bit adder (such as an 8-bit adder). Otherwise, it is kept 0.
    - If '*Final*' is 0, then the output carry from the 4-bit adder is passed as output *'S4'* without any modification.
    - If '*Final*' is 1, then the output carry from the 4-bit adder is passed as output *'S4'* without any modification, if both the input numbers to the 4-bit adder are of same sign. If the two input numbers are of opposite sign, then the complement of the output carry is passed as the output *'S4'*.

Circuit Diagram for 4-bit adder is:

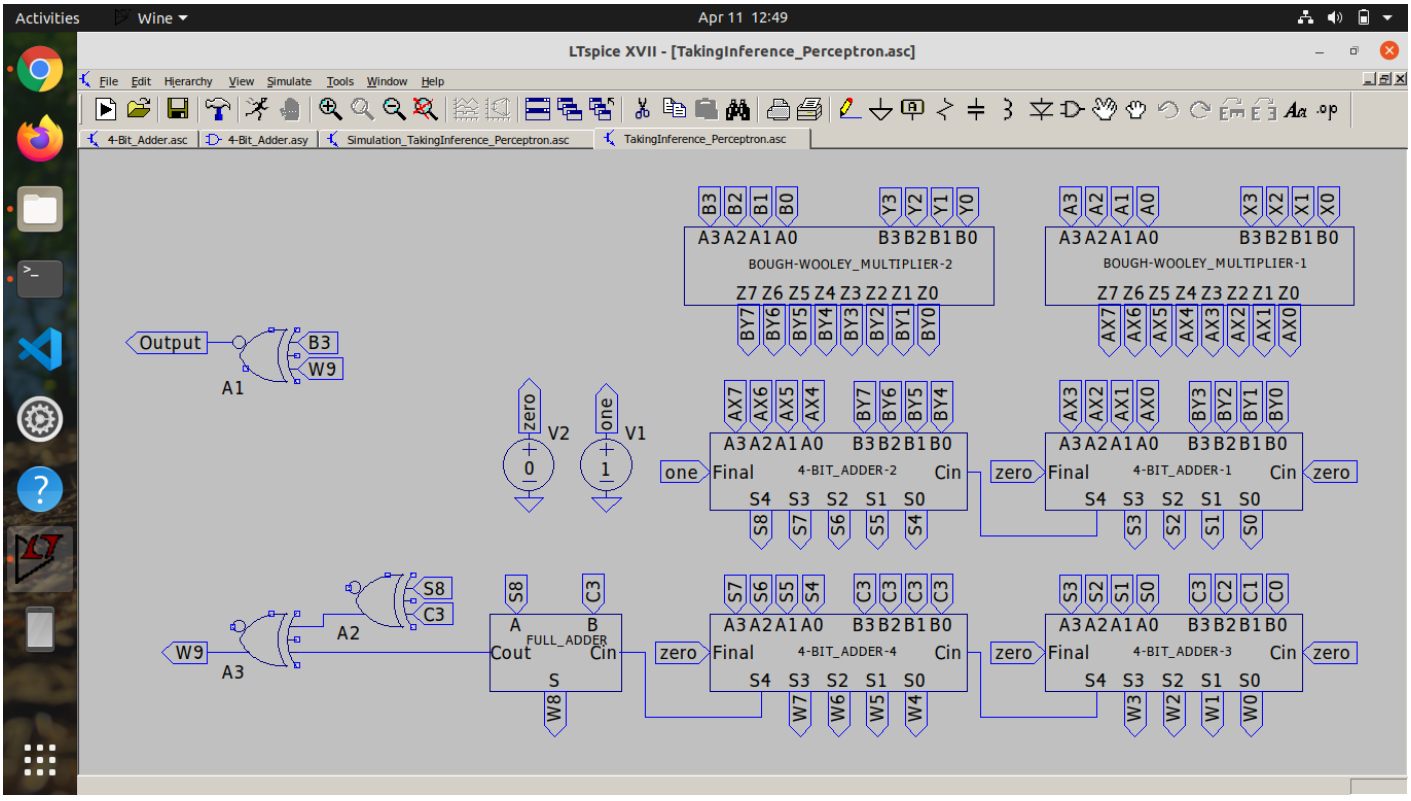


Symbol used for the above 4-bit adder is:

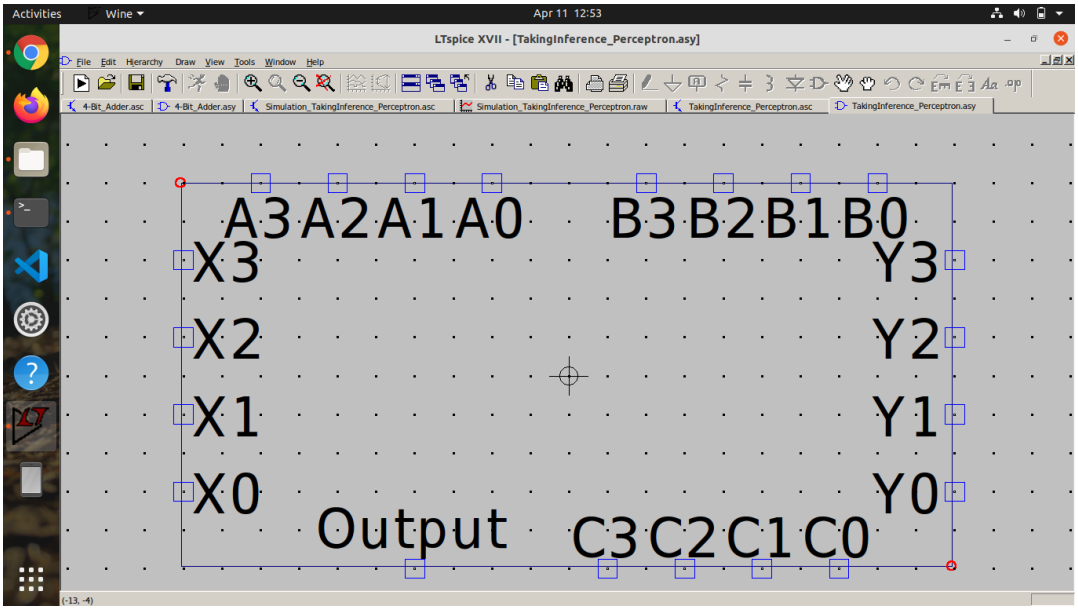


3. Thus, using two 4-bit adder S = A.X + B.Y was calculated. The output S was a 9-bit number. As a result, using two 4-bit adder and one full adder, the final output W = S + C was calculated. If S and C were of opposite sign, then the complement of the output carry of the full adder was taken as W_9 (Most Significant Bit of W). Otherwise, the output carry was passed as it is to W_9. Thus, W = A.X + B.Y + C and W_9 is the most significant bit of W.
4. Result is given by: **Output = W_9 xnor B_3**. If W and B are of same sign, then Output = 1. Otherwise, it is equal to 0.

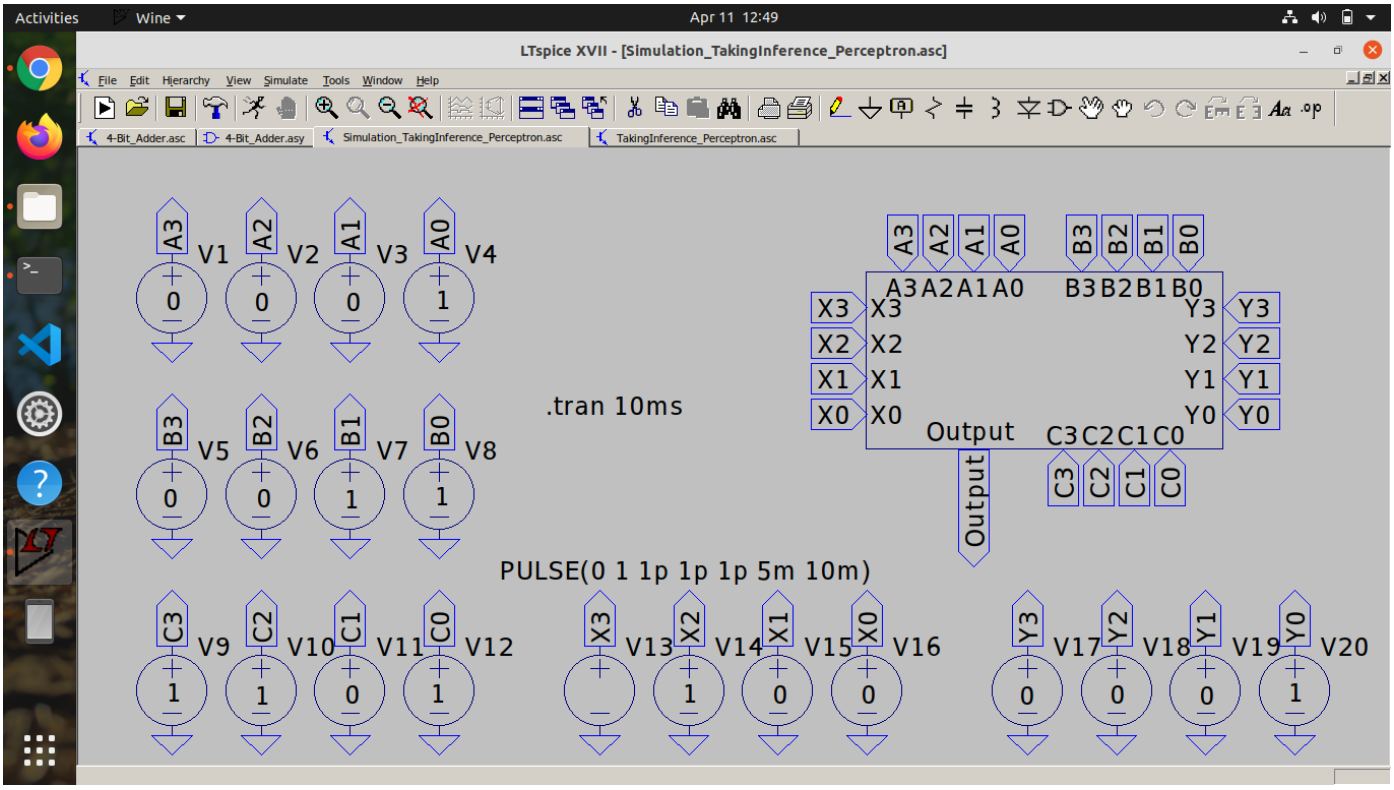
In this way, the required functionality was achieved. Circuit Diagram corresponding to above logic is:



A symbol was created for above circuit and it was used for simulation. Symbol for above circuit is:



Corresponding Simulation Circuit is:

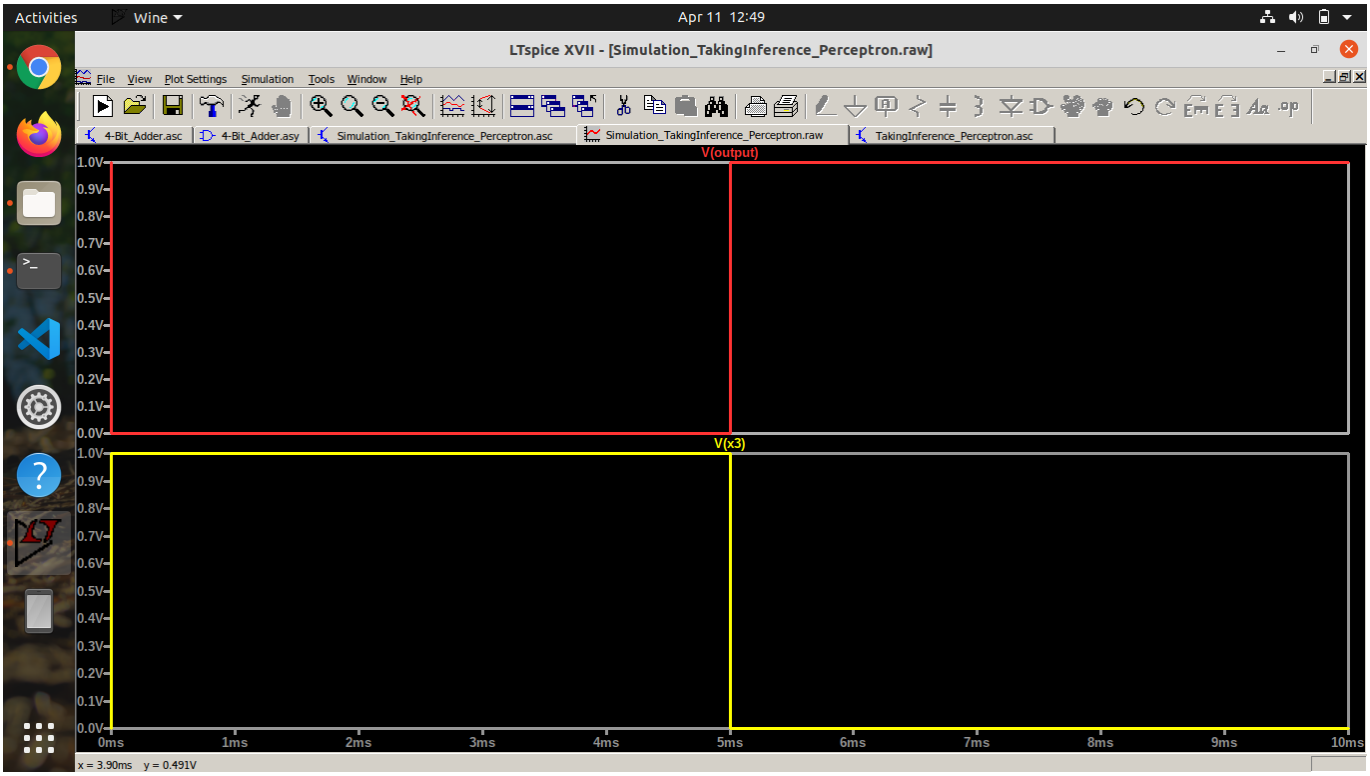


In above simulation circuit, value A, B, C and Y are kept constant at:

1. $A = A_3\ A_2\ A_1\ A_0 = (0001)_2$
2. $B = B_3\ B_2\ B_1\ B_0 = (0011)_2$
3. $C = C_3\ C_2\ C_1\ C_0 = (1101)_2$
4. $Y = Y_3\ Y_2\ Y_1\ Y_0 = (0001)_2$

Now, for $X = X_3\ X_2\ X_1\ X_0$, value is $(1100)_2$ for first 5ms. This corresponds to the situation $X = (-4)_{10}$ and (X, Y) = (-4, 1). For next 5ms, value of X is $(0100)_2$. This corresponds to the situation $X = (4)_{10}$ and (X, Y) = (4, 1). Thus, only bit X_3 is changing and all other bits of X are constant.

<u>Corresponding Simulation Result are attached below:</u>



Yellow line shows the bit X_3 and red line shows the Output. We have:

1. When $X_3 = 1$, then (X, Y) = (-4, 1). This point is below the line $X + 3Y - 3 = 0$. Hence, Output = 0.
2. When $X_3 = 0$, then (X, Y) = (4, 1). This point is above the line $X + 3Y - 3 = 0$. Hence, Output = 1.

Hence, above simulation results are correct for the given two points.