****

Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Computer Networks Laboratory ENCS4130

Digital system (Comp2340)

Prepared by

Yousef sharbeh-1202057

BIRZEIT

June – 2022

Q1: Given the following Combinational circuit, Use Verilog HDL on Quartus tool to a. Implement the 1-bit adder and use it to build 4-bit adder structurally b. Implement the MUX2x1 and then use it to build the Quad MUX 2x1 structurally. c. Implement the 4-bit OR Array d. Implement the 4-bit AND Array e. Use the blocks you implemented in the parts above to build the final system shown in the figure below. f. You should show simulation results for each of the above parts.

Solution:

a)

1-bit adder truth table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Carry in | sum | Carry out |
| 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 |

using the above table to try to implement the adder subtractor the subtractor idea is just to put xor gate on the B input with the original carry in, the over flow is just to xor the carryn-1 with carryn

Adder code:

module adder(a,b,cin,s,cout);

input a,b,cin;

output s,cout;

wire w0,w1,w2,w3;

xor gate1(w0,a,b);

xor gate2(s,w0,cin);

and gate3(c1,w0,cin);

and gate4(w2,a,b);

or gate5(cout,w1,w2);

endmodule

the adder subtractor with overflow code:

module sharbeh(a,b,sum,cin,o\_f);

input [3:0] a,b;

input cin;

output[3:0] sum;

output o\_f;

wire [3:0] p;

xor (p[0],b[0],cin);

xor (p[1],b[1],cin);

xor (p[2],b[3],cin);

xor (p[3],b[3],cin);

adder f\_a1(a[0],p[0],cin,cout1,sum[0]);

adder f\_a2(a[1],p[1],cout1,cout2,sum[1]);

adder f\_a3(a[2],p[3],cout2,cout3,sum[2]);

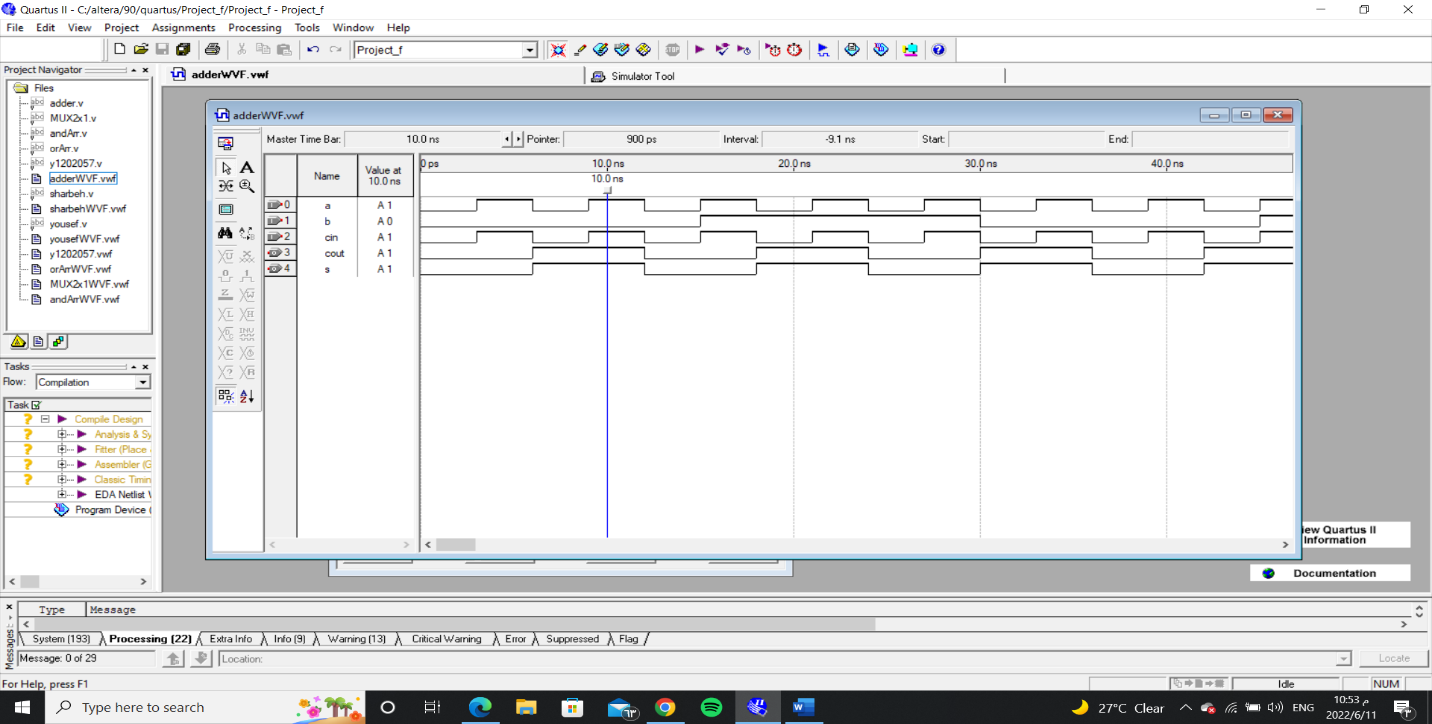
adder f\_a4(a[3],p[3],cout3,cout,sum[3]);

xor gate1(o\_f,cout,cout3);

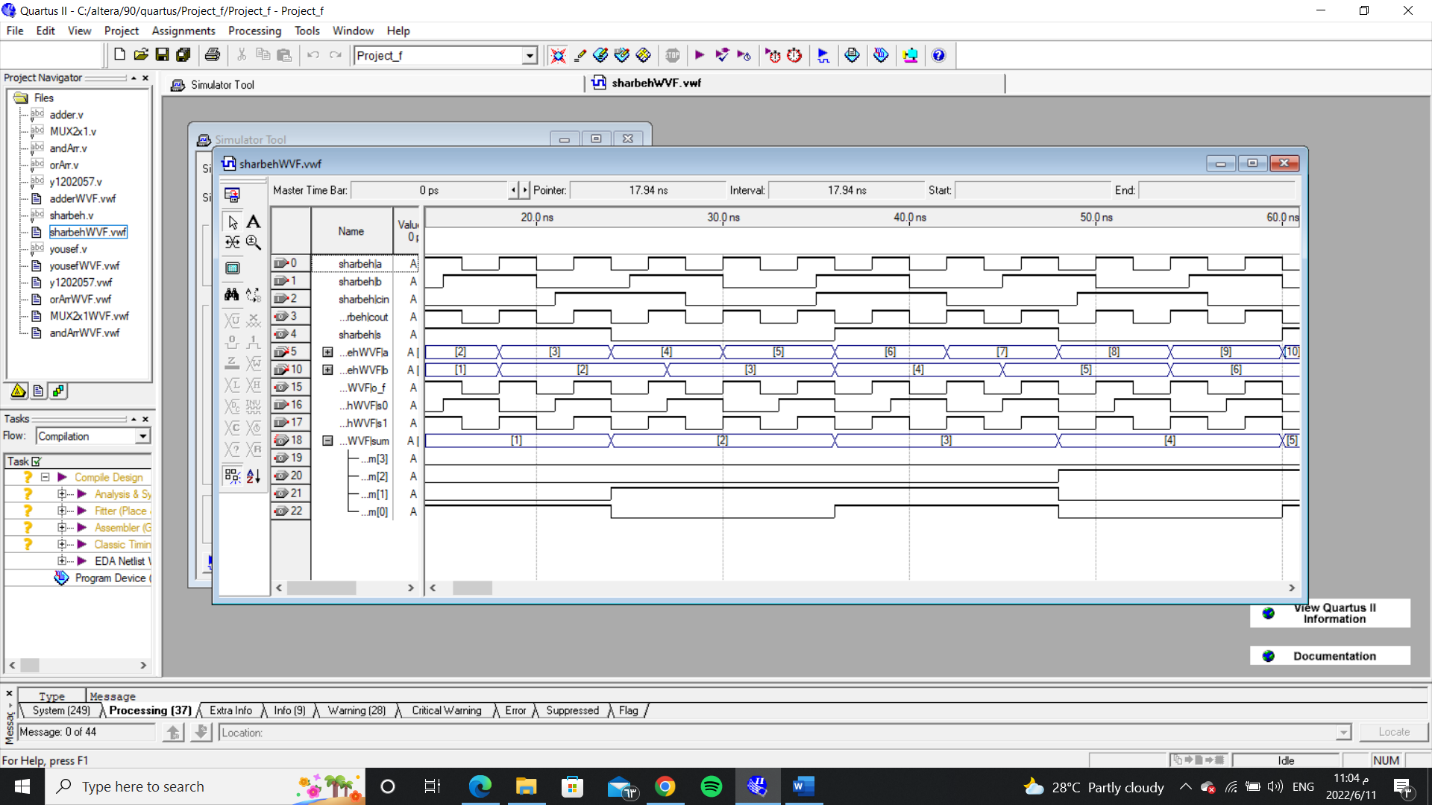
endmodule

the simulated of result:

adder simulation-



Adder subtractor with overflow simulation:



b)

MUX (2x1) truth table:

Truth table for 2-to-1 Multiplexer

|  |  |  |  |
| --- | --- | --- | --- |
| S | A | B | Y |
| 0 | 0 | X | 0 |
| 0 | 1 | X | 1 |
| 1 | X | 0 | 0 |
| 1 | x | 1 | 1 |

A MUX2x1 consists of two inputs A and B, one select input S and one output Y. Depending on the select signal, the output is connected to either of the inputs. Since there are two input signals, only two ways are possible to connect the inputs to the outputs, so one select is needed to do these operations.

Quad MUX code:

module MUX2x1(i0,i1,s,out);

input i0,i1,s;

output out;

wire w1, w2, w3;

and gate1(w1, b, s);

not gate2(w2, s);

and gate3(w3, a, w2);

or gate4(out, w1,w3);

endmodule

module yousef(in0,in1,s,out);

input [3:0] in0,in1;

input s;

output [3:0] out;

MUX2x1 m0(in0[0],in1[0],s,out[0]);

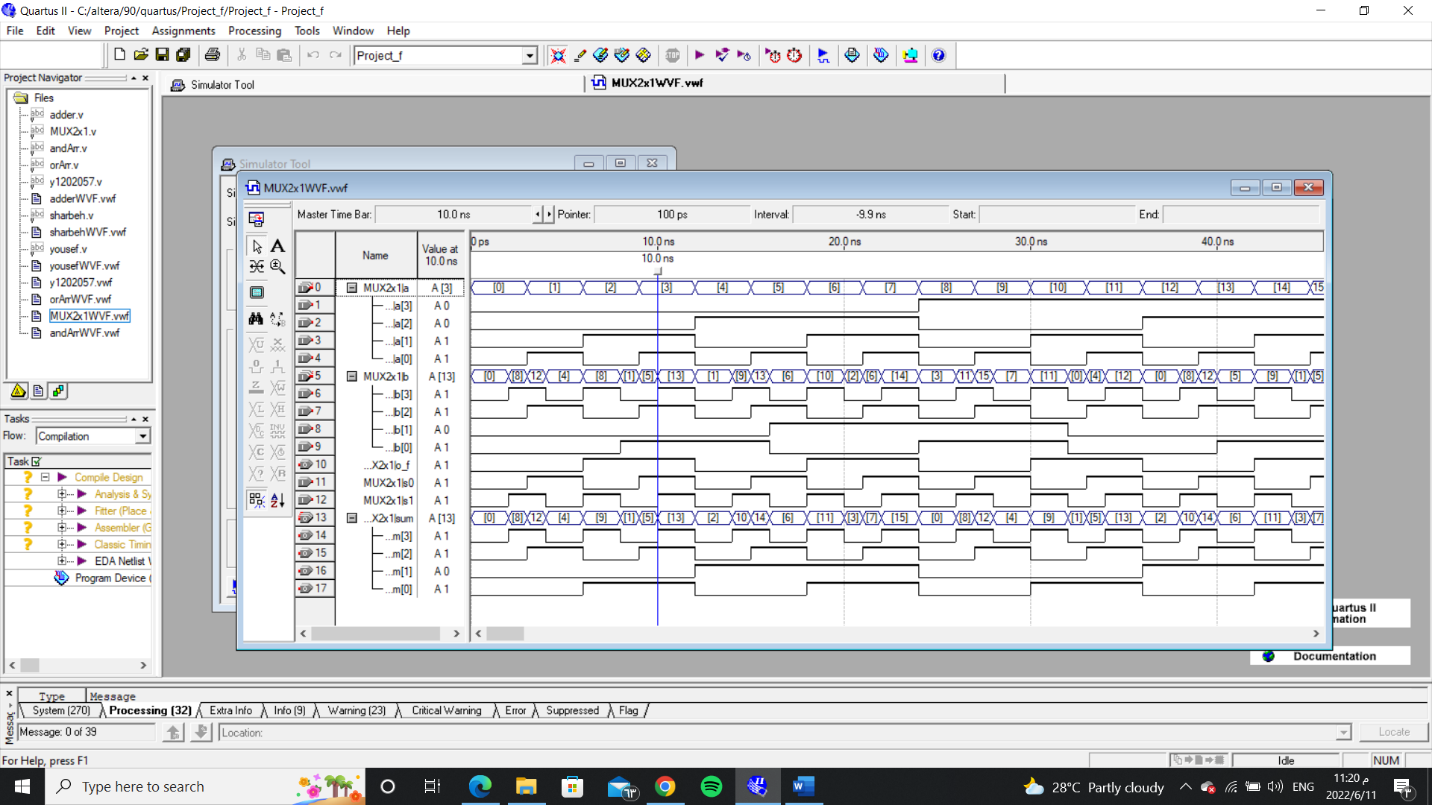
MUX2x1 m1(in0[1],in1[1],s,out[1]);

MUX2x1 m2(in0[2],in1[2],s,out[2]);

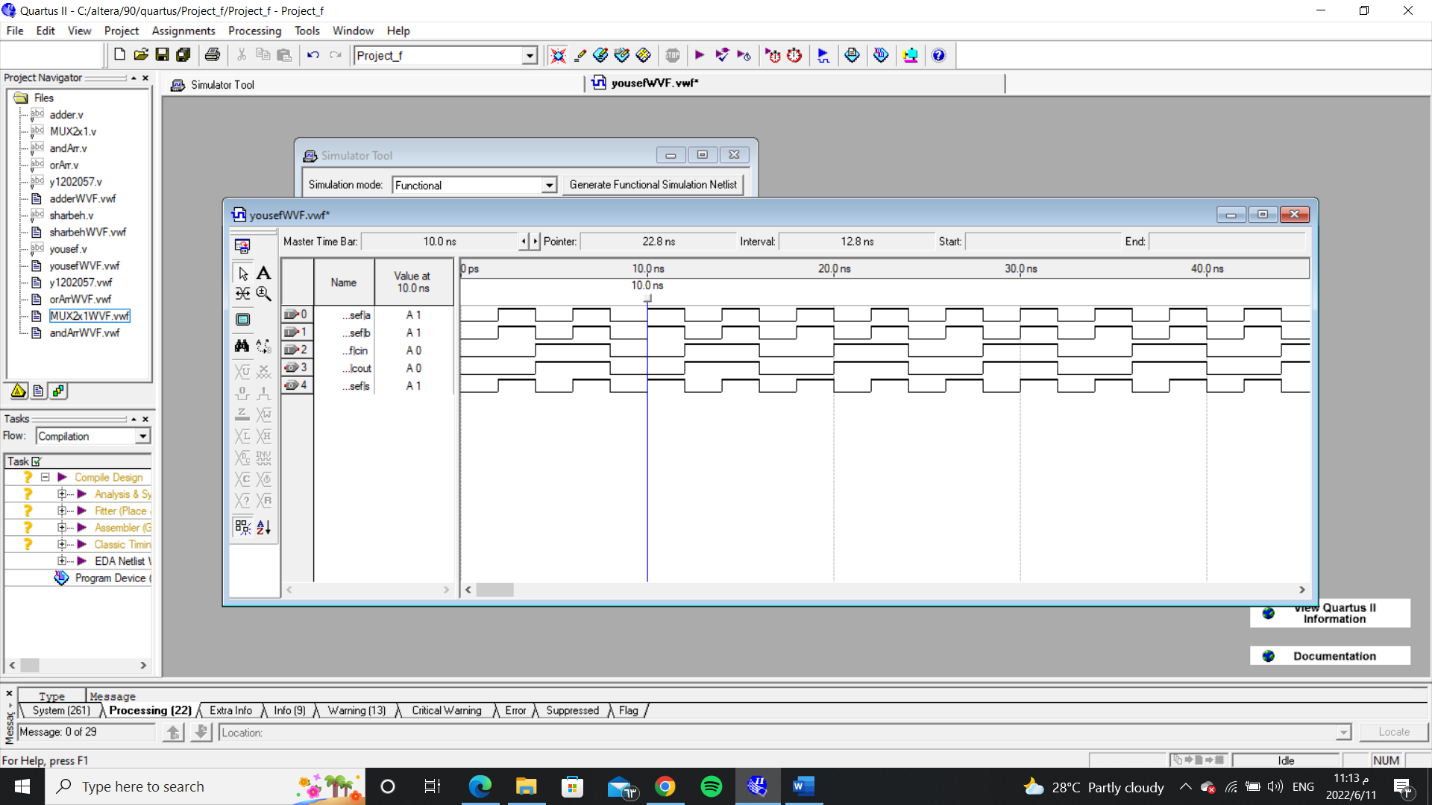
MUX2x1 m3(in0[3],in1[3],s,out[3]);

Endmodule

The MUX2x1 simulation:



The Quad MUX2x1



c)+d): Those two questions has a simple solution its just an array of (ands) and array of (or’s).

c)

OR truth table:

|  |  |  |
| --- | --- | --- |
| A | B | F |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

OR code:

module orArr(x, y, z);

input [3:0] x,y;

output [3:0] z;

or g1(z[0], x[0], y[0]);

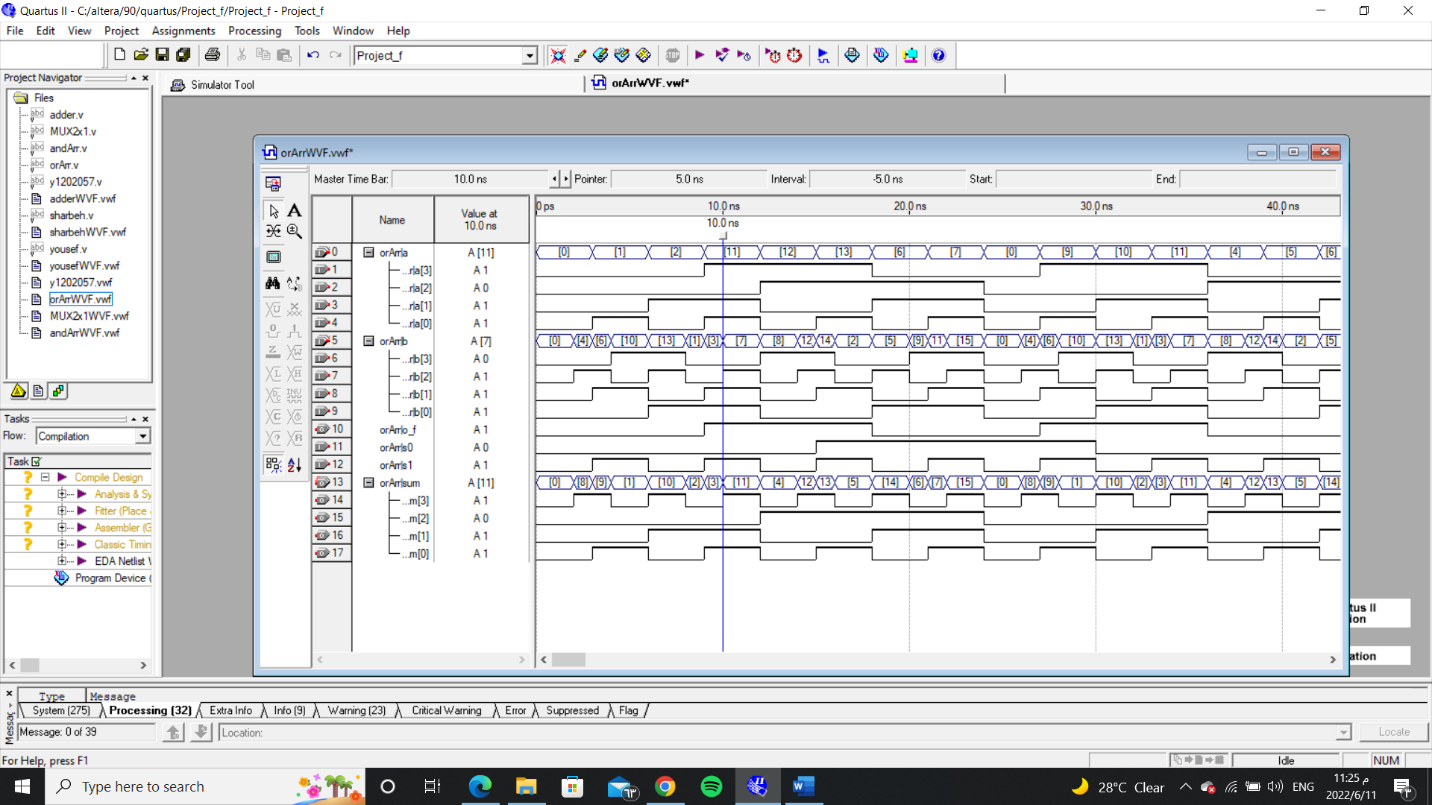
or g2(z[1], x[1], y[1]);

or g3(z[2], x[2], y[2]);

or g4(z[3], x[3], y[3]);

endmodule

OR simulation:



d)

AND truth table:

|  |  |  |
| --- | --- | --- |
| A | B | F |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

AND code:

module andArr(x, y, z);

input [3:0] x,y;

output [3:0] z;

and gate1(z[0],x[0],y[0]);

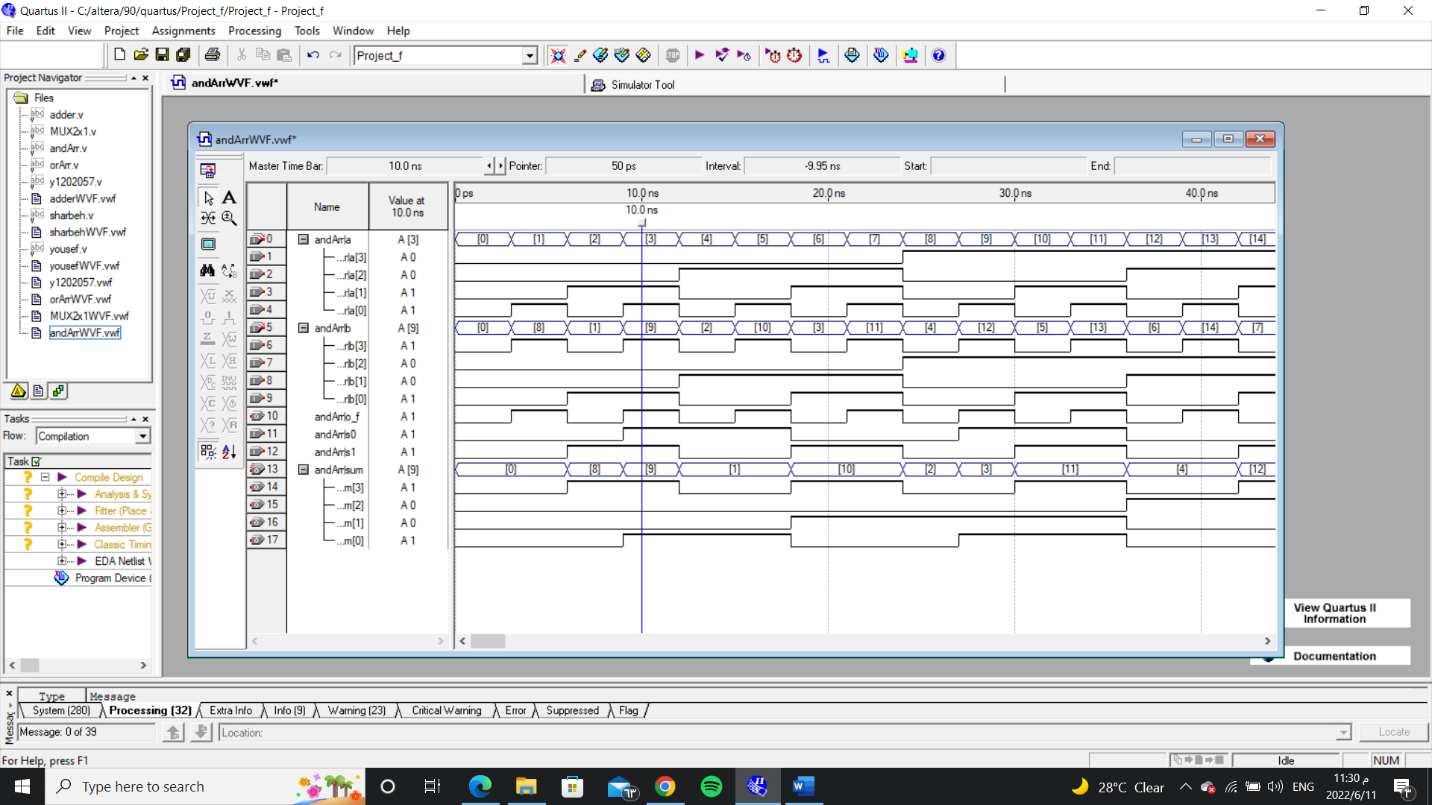
and gate2(z[1],x[1],y[1]);

and gate3(z[2],x[2],y[2]);

and gate4(z[3],x[3],y[3]);

endmodule

AND simulation:



The system:

the inputs with the overflow of the adder subtractor it choses the operation depending on the S0 input the output depends on the S1 input if S1 is zero then the output will be the adder subtractor otherwise it will be the other quad MUX between the (and) and (or) array.

System code:

module y1202057(a,b,s1,s0,o\_f,sum);

input [3:0] a,b;

input s0,s1;

output o\_f;

output [3:0]sum;

wire [3:0] w1,w2,w3,w4;

sharbeh(a,b,s0,w1,o\_f);

andArr gate1(a,b,w2);

orArr(a,b,w3);

yousef(w2,w3,s1,w4);

yousef(w1,w4,s1,sum);

endmodule

system simulation:

