****

Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Computer Networks Laboratory ENCS4130

Digital system (Comp2340)

Prepared by

Yousef sharbeh-1202057

BIRZEIT

August – 2022

a)

MUX (2x1):

Truth table for 2-to-1 Multiplexer for A

|  |  |  |  |
| --- | --- | --- | --- |
| S | A | ~A | out |
| 0 | 0 | X | 0 |
| 0 | 1 | X | 1 |
| 1 | X | 0 | 0 |
| 1 | X | 1 | 1 |

A MUX2x1\_A consists of two inputs A and ~A, one select input S and one output out. 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.

MUX 2x1\_A code:

module MUX2x1\_A(input1,input2,s,out);

input input1,input2,s;

output out;

wire w1, w2, w3;

and gate1(w1, input1, s);

and gate2(w3, input2, w2);

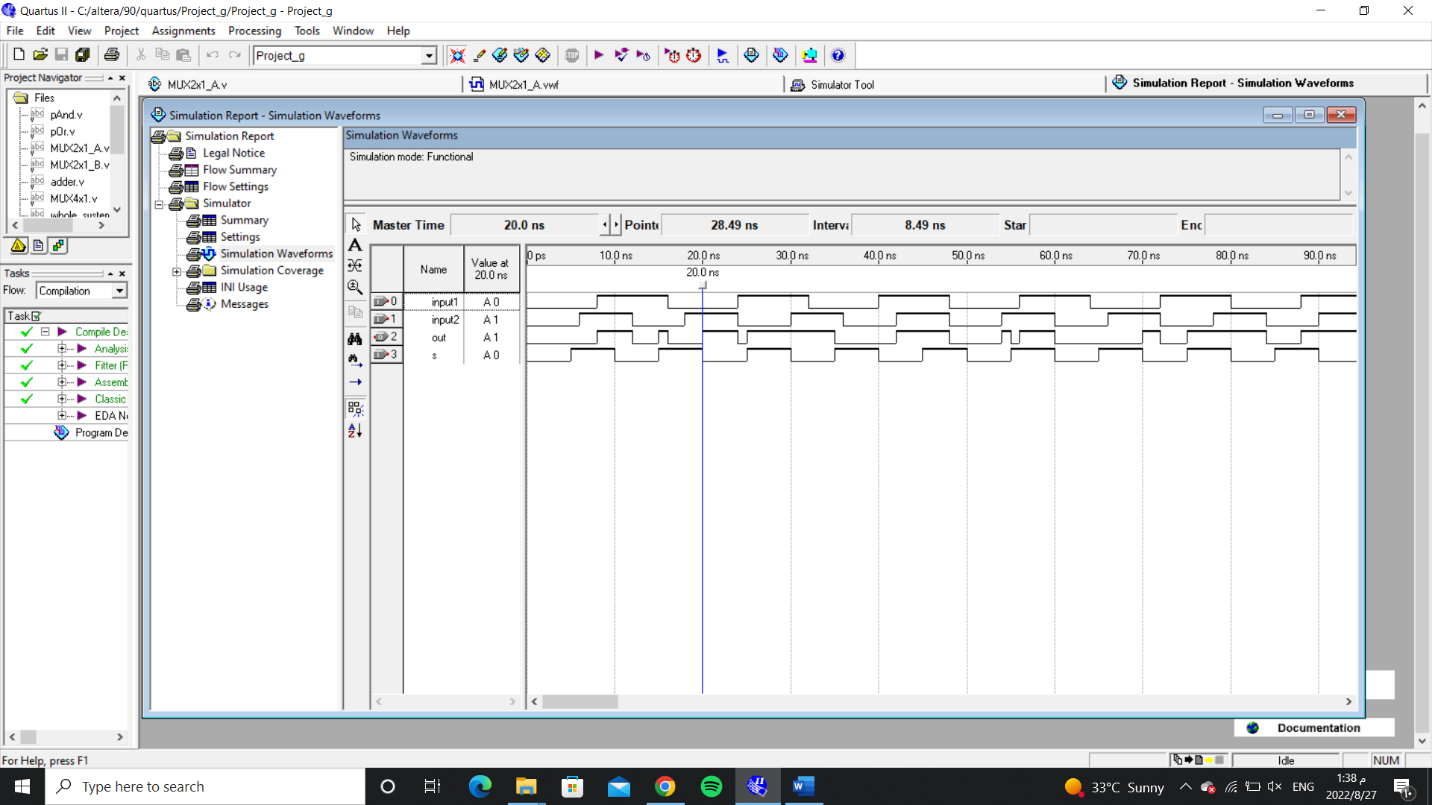
not gate3(w2, s);

or gate4(out, w1,w3);

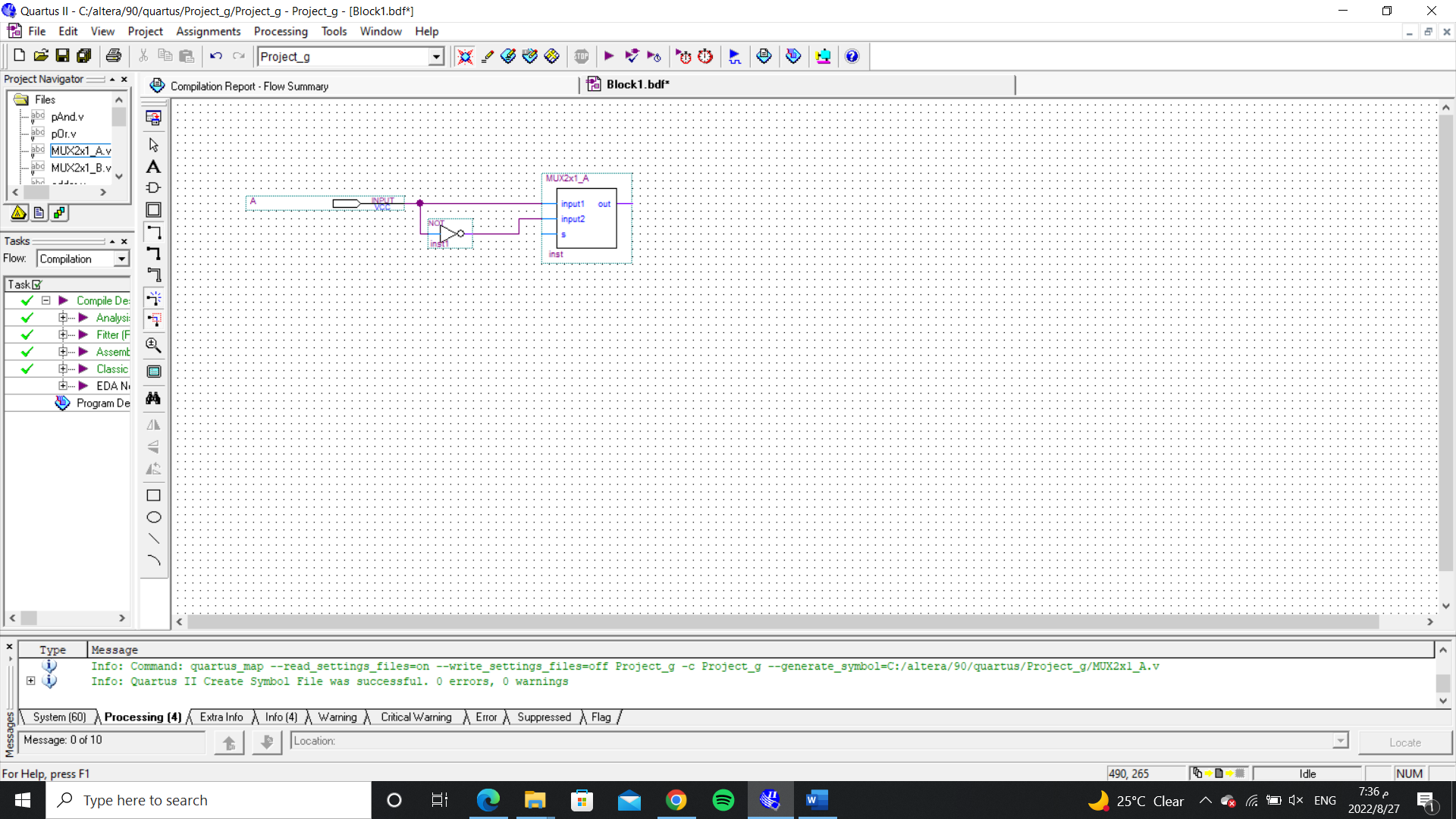
Endmodule

The simulated of result:

MUX2x1\_A waveform simulation-



MUX2x1\_A block diagram-



Truth table for 2-to-1 Multiplexer for B

|  |  |  |  |
| --- | --- | --- | --- |
| S | B | ~B | out |
| 0 | 0 | X | 0 |
| 0 | 1 | X | 1 |
| 1 | X | 0 | 0 |
| 1 | X | 1 | 1 |

A MUX2x1\_B consists of two inputs B and ~B, one select input S and one output out. 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.

MUX 2x1\_B code:

module MUX2x1\_B(input1,input2,s,out);

input input1,input2,s;

output out;

wire w1, w2, w3;

and gate1(w1, input1, s);

and gate2(w3, input2, w2);

not gate3(w2, s);

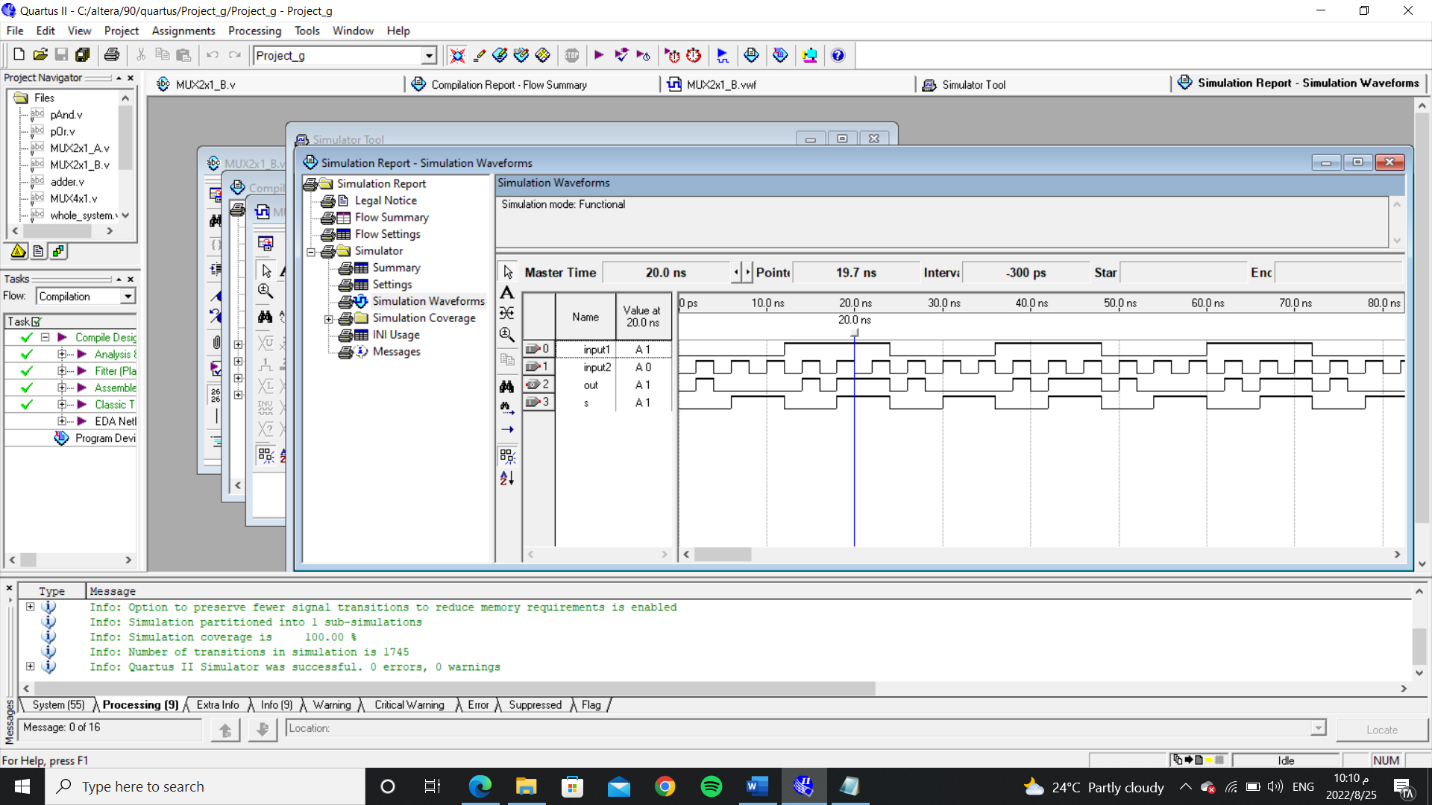
or gate4(out, w1,w3);

endmodule

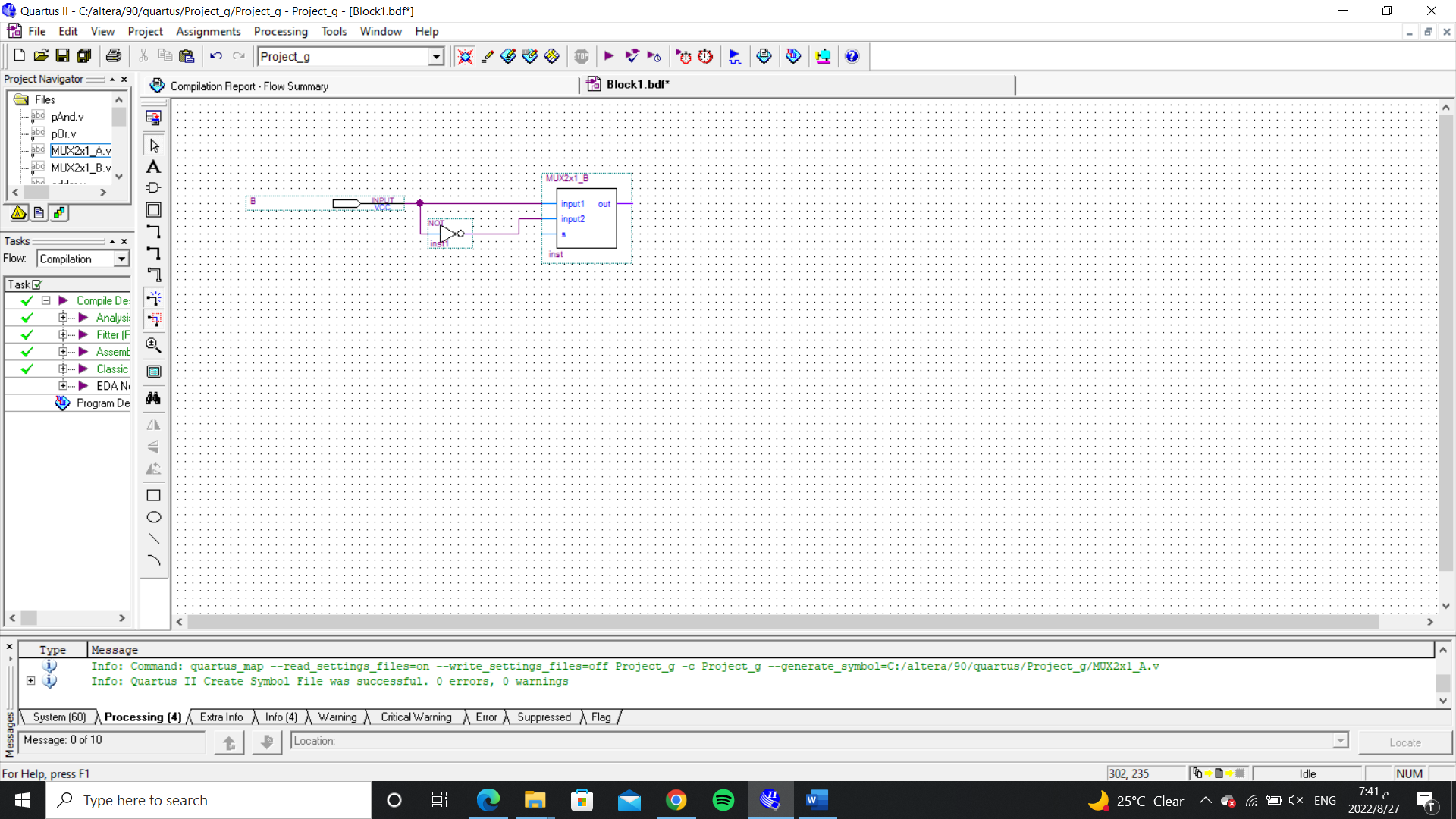
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The simulated of result:

MUX2x1\_B waveform simulation-



MUX2x1\_B block diagram-



b)

AND

AND truth table:

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

A pAnd (And Gate) consists of two inputs A and B and one output F, if all input (1) then the output will be (1) or the output will be (0) in all other cases.

AND code:

module pAnd(x, y, z);

input x,y;

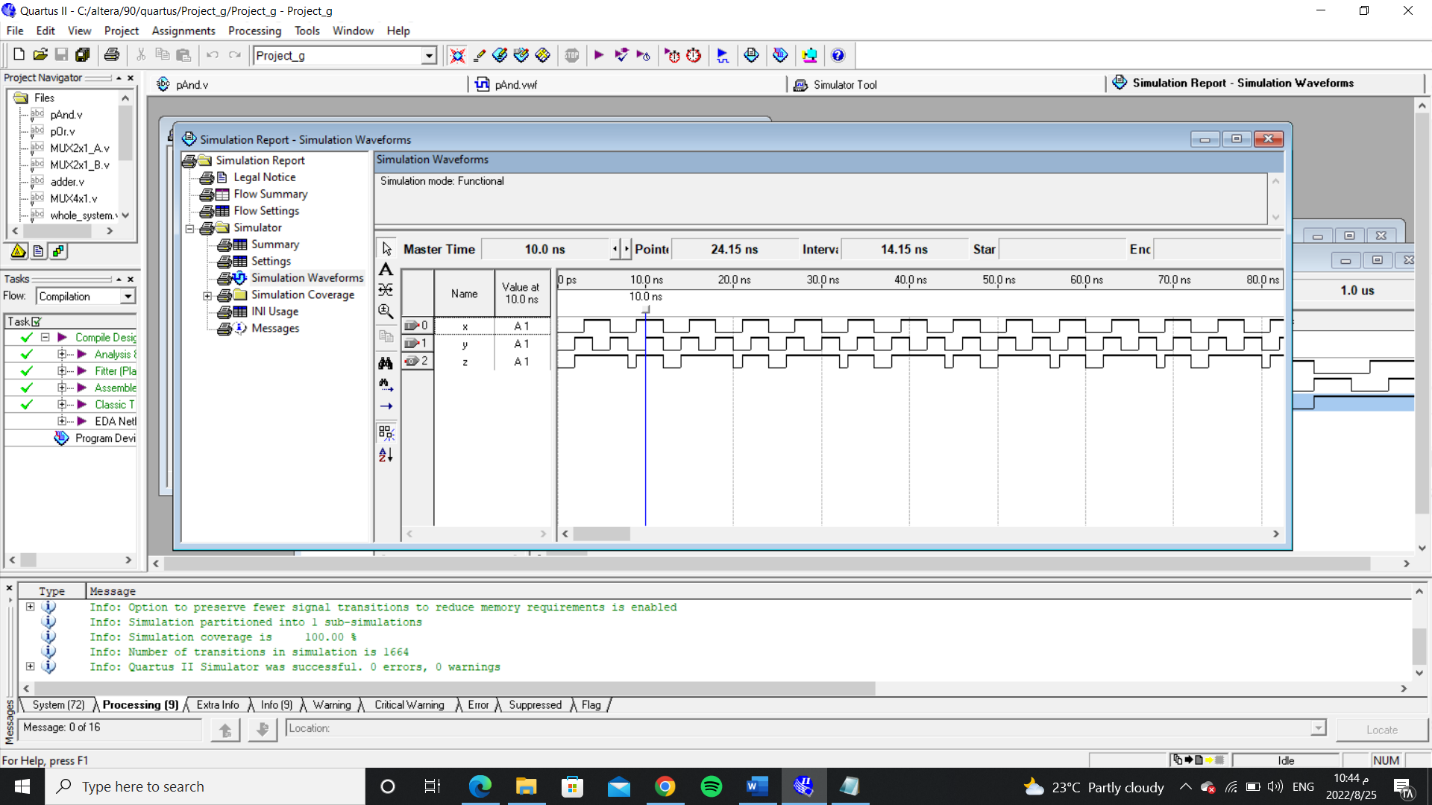
output z;

and and\_gate(z,x,y);

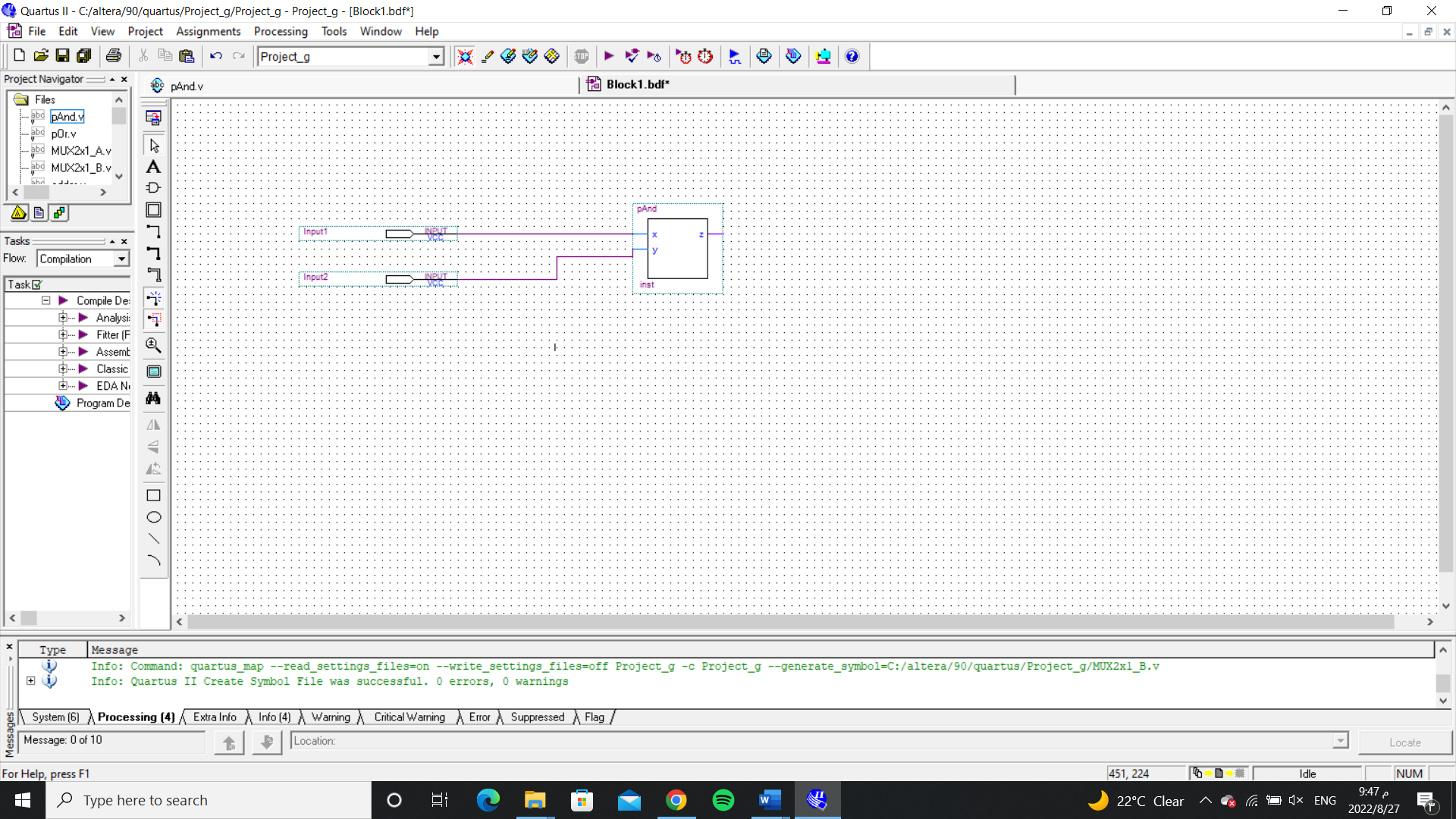
endmodule

The simulated of result:

The AND waveform simulation-



AND block diagram-



c)

OR

OR truth table:

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

A pOr (OR Gate) consists of two inputs A and B and one output F, if all input (0) then the output will be (0) or the output will be (1) in all other cases.

OR code:

module pOr(x, y, z);

input x,y;

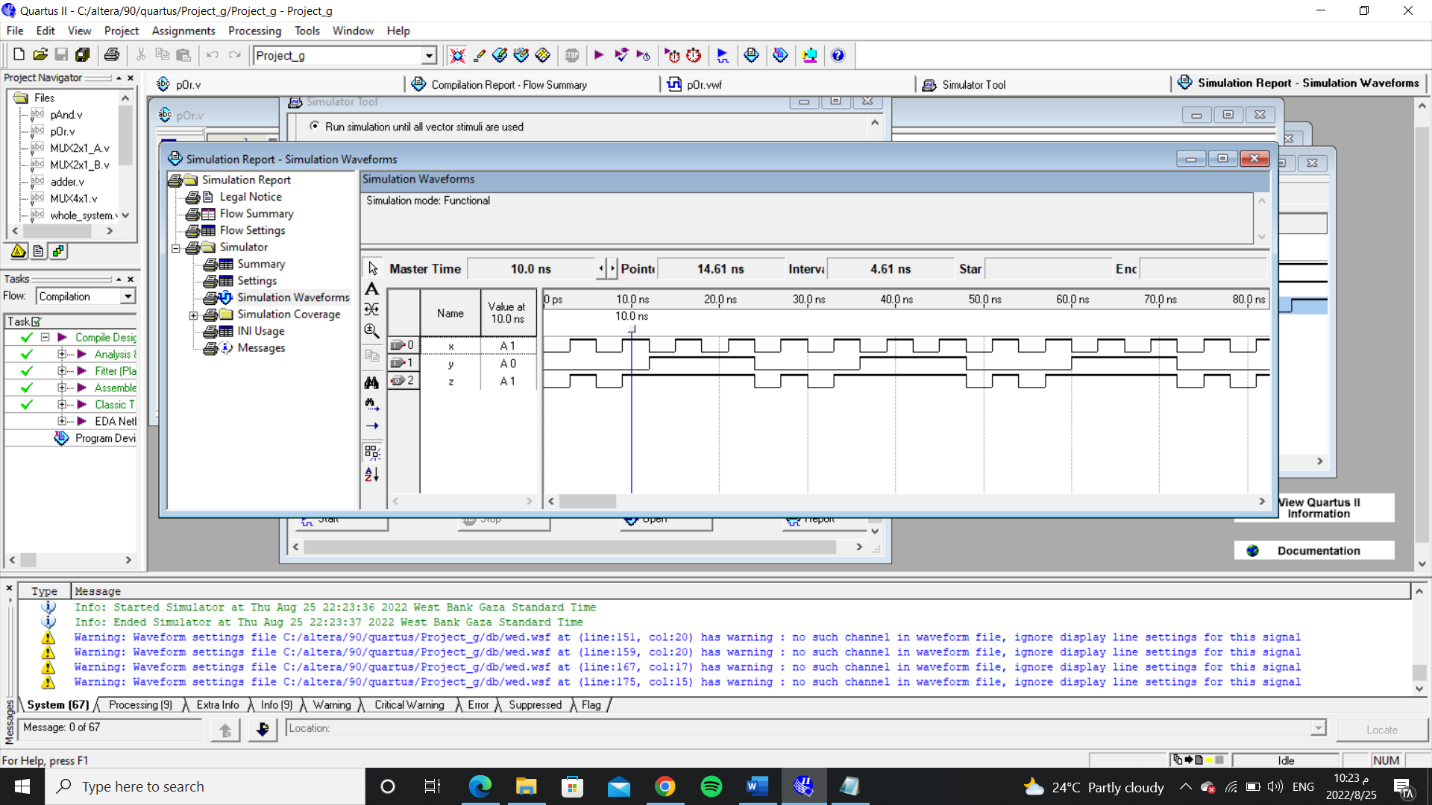
output z;

or or\_gate(z,x,y);

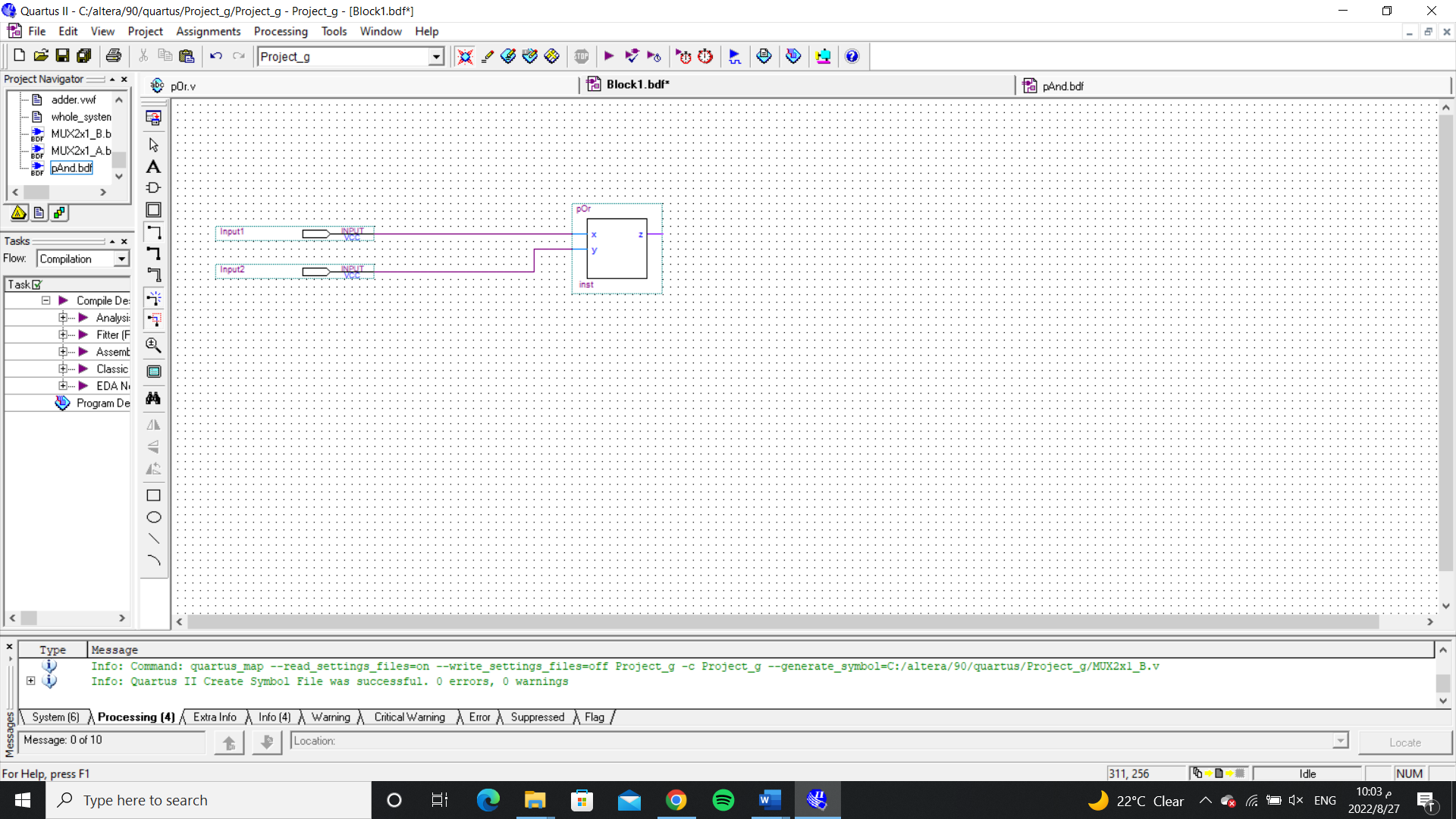
endmodule

The simulated of result:

The OR waveform simulation-



OR block diagram-



d)

Adder

1-bit adder truth table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| input1 | input2 | 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 |

A adder (Full\_Adder) consists of three inputs (input1, input2, Carry in) and two output sum , and

Carry out , the idea of the full adder to calculate the sum and carry.

Adder code:

module adder(input1,input2,c\_in,sum,c\_out);

input input1,input2,c\_in;

output sum,c\_out;

wire w0,w1,w2;

xor gate1(w0,input1,input2);

xor gate2(sum,w0,c\_in);

and gate3(c1,w0,c\_in);

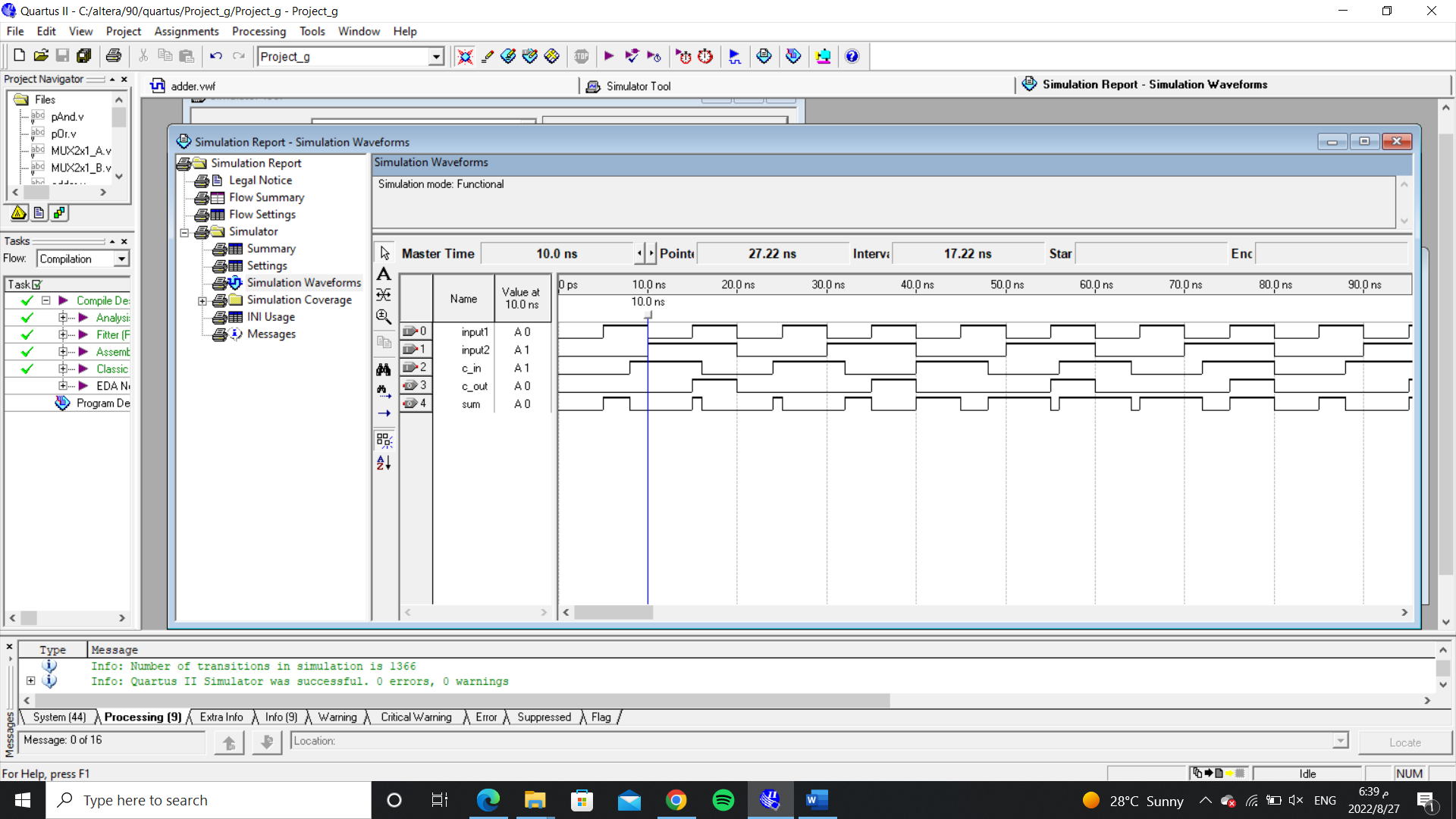
and gate4(w2,input1,input2);

or gate5(c\_out,w1,w2);

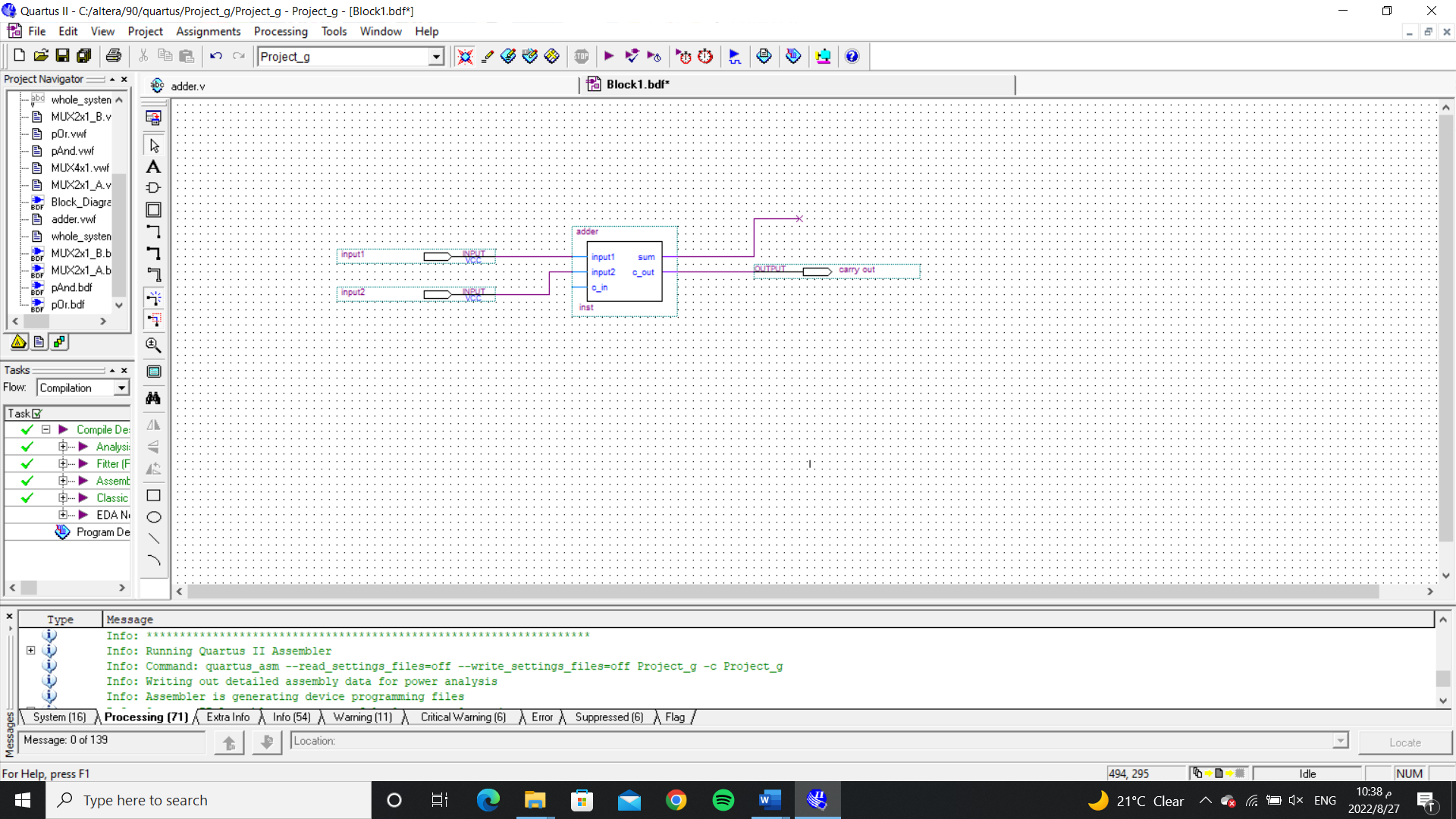
endmodule

The simulated of result:

Adder waveform simulation-



Adder block diagram-



e)

MUX4x1

Truth table for 4-to-1 Multiplexer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S0** | **S1** | **D0** | **D1** | **D2** | **D3** |
| 0 | 0 | 0 | X | X | X |
| 0 | 0 | 1 | X | X | X |
| 0 | 1 | X | 0 | X | X |
| 0 | 1 | X | 1 | X | X |
| 1 | 0 | X | X | 0 | X |
| 1 | 0 | X | X | 1 | X |
| 1 | 1 | X | X | X | 0 |
| 1 | 1 | X | X | X | 1 |

MUX 4x1 code:

module MUX4x1(input1,input2,input3,input4,res,s);

input input1,input2,input3,input4;

input [1:0]s;

output res;

wire w\_s1,w\_s0;

wire w1, w2, w3;

not n1(w\_s1,s[1]);

not n2(w\_s0,s[0]);

and a1(w1,input1,w\_s1,w\_s0);

and a2(w2,input2,w\_s1,s[0]);

and a3(w3,input3,s[1],w\_s0);

and a4(w4,input4,s[1],s[0]);

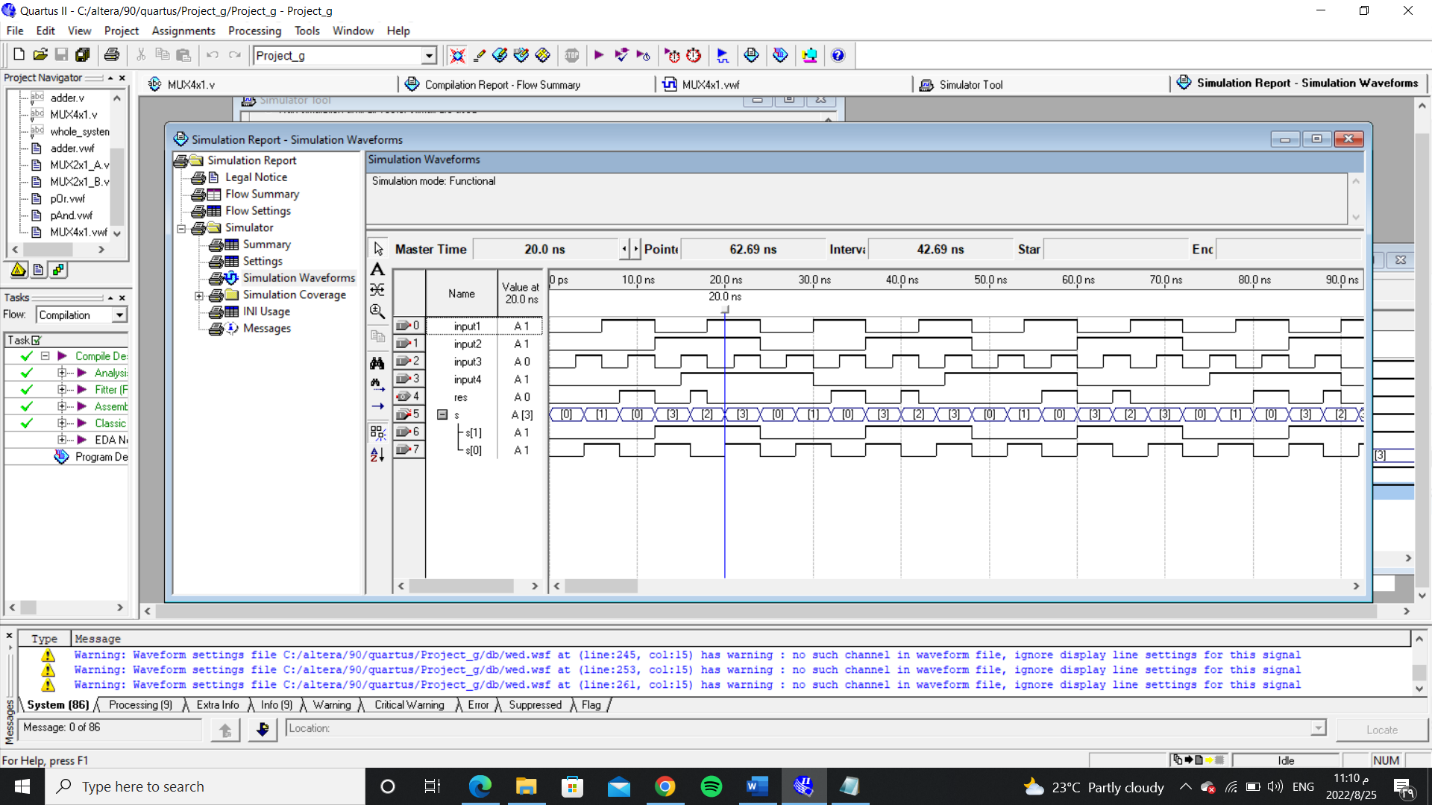
or a5(res,w1,w2,w3);

endmodule

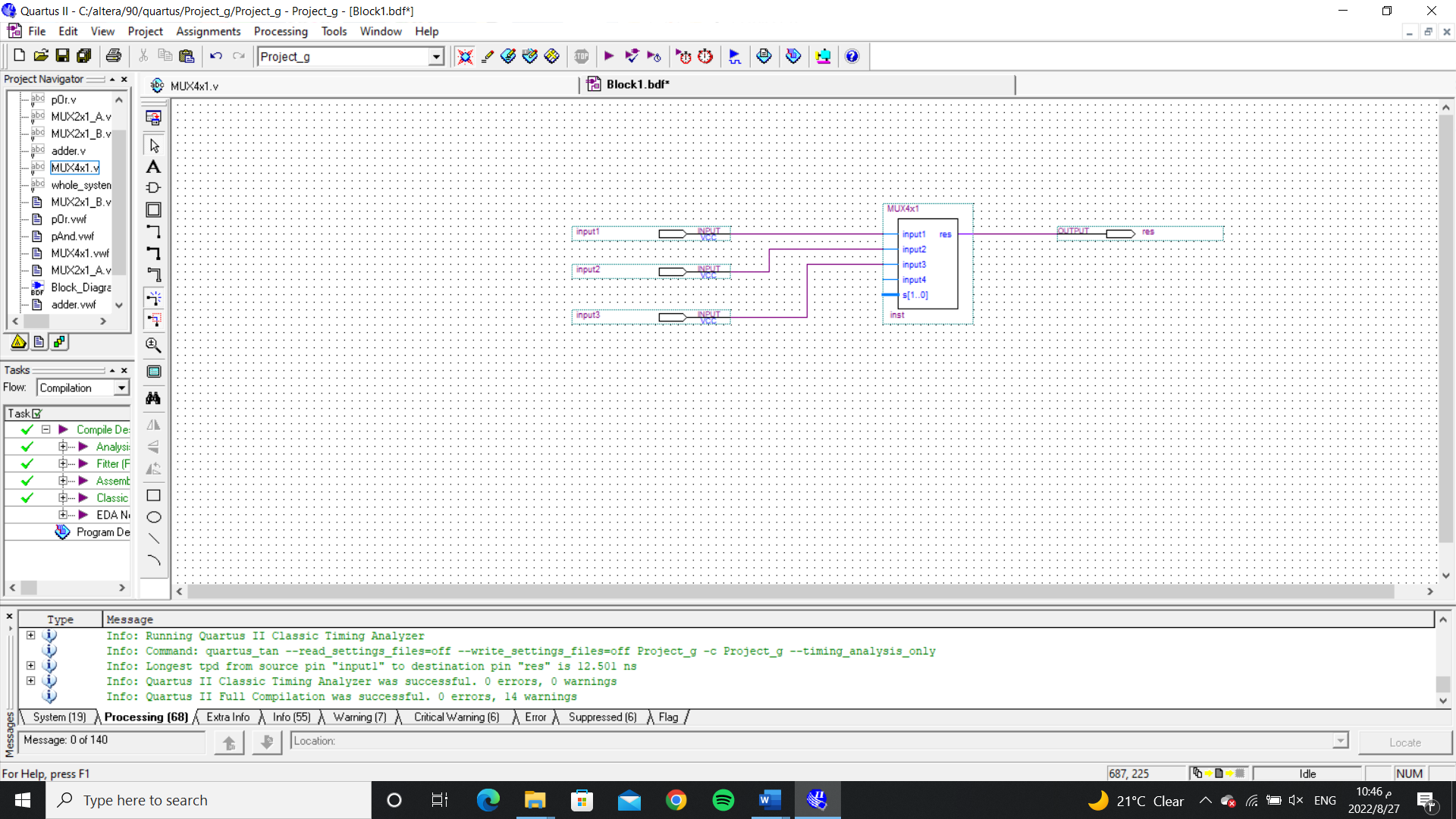
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The simulated of result:

The MUX4x1 waveform simulation-



MUX4x1 block diagram-



The Whole system:

Whole System code:

module whole\_system(a,b,invertA,invertB,c\_in,input4,s,c\_out,res);

input a,b,invertA,invertB,c\_in,input4;

input [1:0]s;

output c\_out,res;

wire w1,w2,w3,w4,w5;

MUX2x1\_A(a,~a,invertA,w1);

MUX2x1\_B(b,~b,invertB,w2);

pAnd(w3,w1,w2);

pOr(w4,w1,w2);

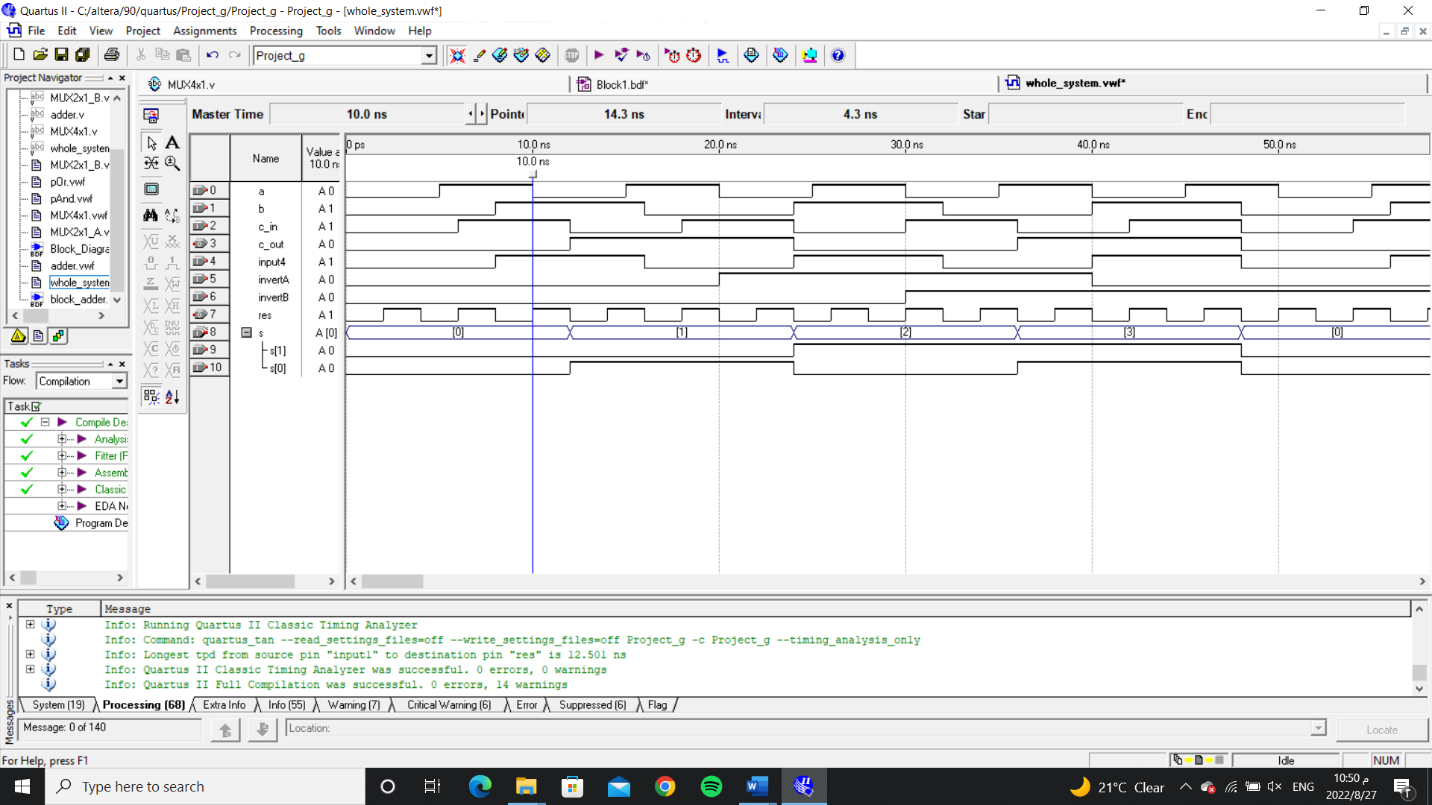
adder(w5,c\_out,w1,w2,c\_in);

MUX4x1(res,w3,w4,w5,input4,s);

endmodule

The simulated of result:

The Whole System waveform simulation-



Whole System block diagram-

