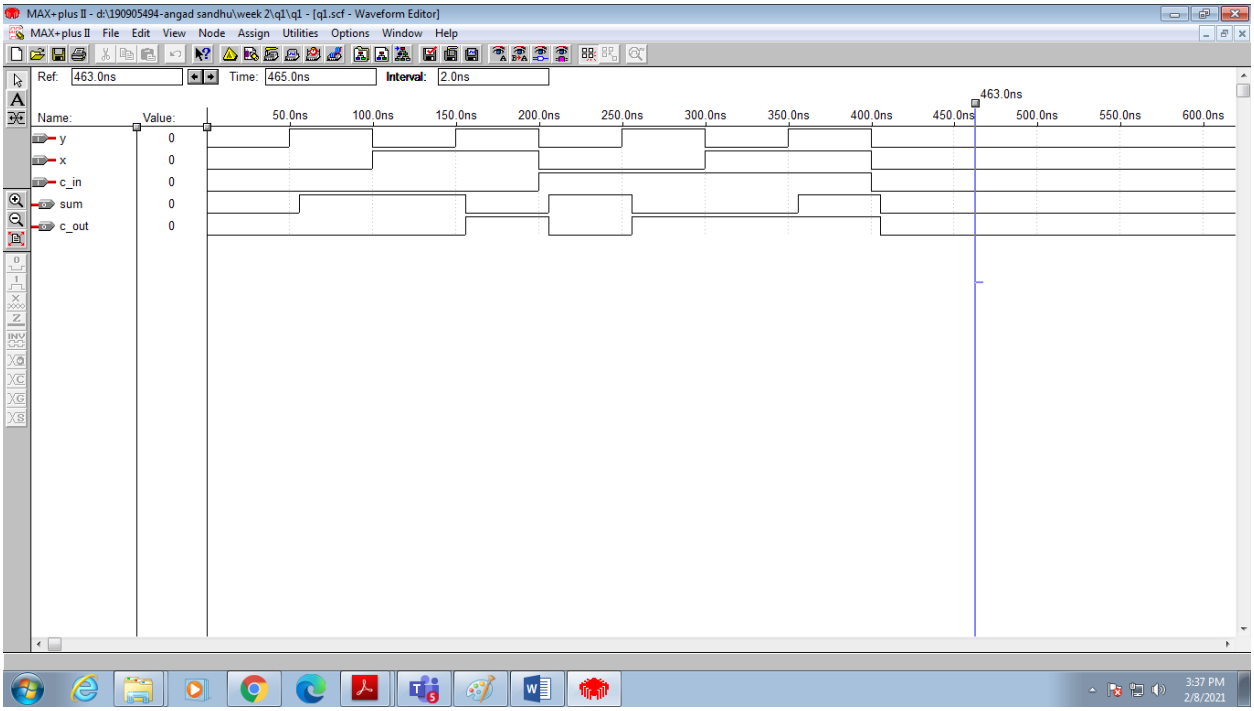
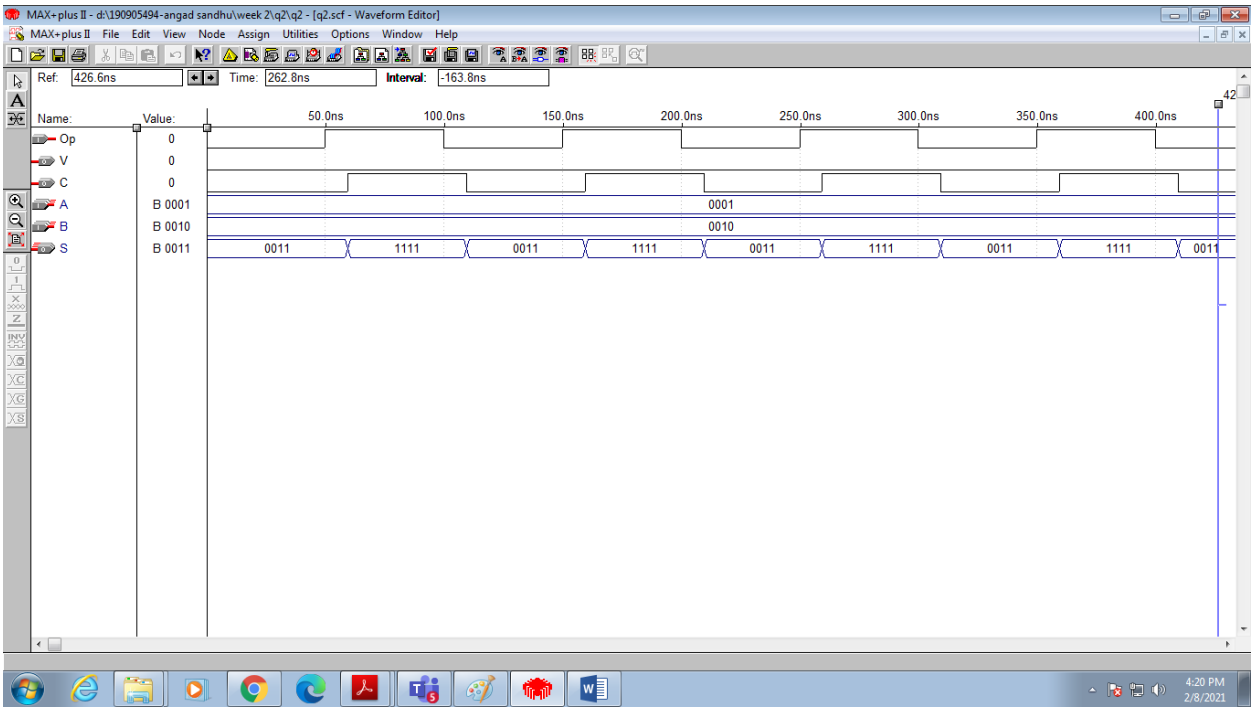
Q1

```
// code for a 1-bit full adder
module q1(c_in, x, y, sum, c_out);
   input c_in, x, y;
   output sum, c_out;

   // calculating sum by XORing
   assign sum = c_in ^ x ^ y;

   // calculating carry out by ADDing and ORing
   assign c_out = (x & y) | (y & c_in) | (c_in & x);
endmodule
```



Q2

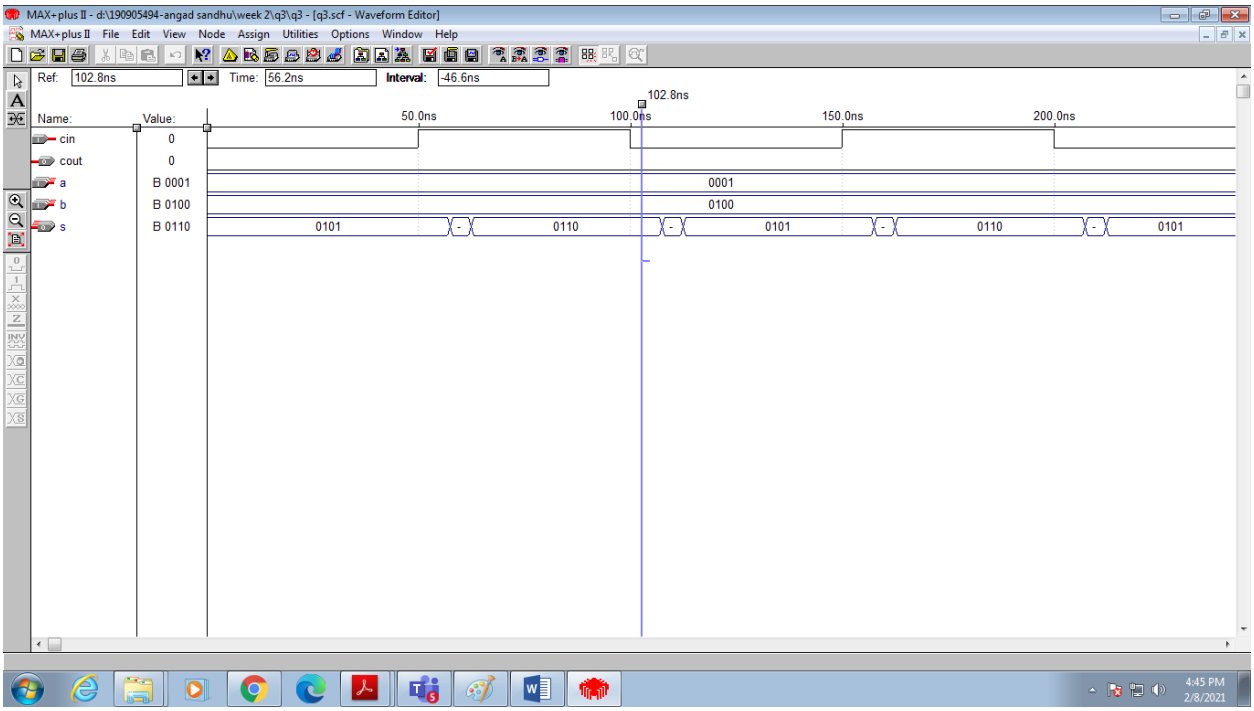
```
module q2(S, C, V, A, B, Op);
  output [3:0] S;   // The 4-bit sum/difference.
  output       C;   // The 1-bit carry/borrow status.
  output       V;   // The 1-bit overflow status.
  input [3:0]  A;   // The 4-bit augend/minuend.
  input [3:0]  B;   // The 4-bit addend/subtrahend.
  input  Op;  // The operation: 0 => Add, 1=>Subtract.

  wire   C0; // The carry out bit of fa0, the carry in bit of fa1.
  wire   C1; // The carry out bit of fa1, the carry in bit of fa2.
  wire   C2; // The carry out bit of fa2, the carry in bit of fa3.
  wire   C3; // The carry out bit of fa2, used to generate final carry/borrrow.
  wire   B0;
  wire   B1;
  wire   B2;
  wire   B3;
  xor(B0, B[0], Op);
  xor(B1, B[1], Op);
  xor(B2, B[2], Op);
  xor(B3, B[3], Op);
  xor(C, C3, Op);    // Carry = C3 for addition, Carry = not(C3) for subtraction.
  xor(V, C3, C2);    // If the two most significant carry output bits differ, then we have an overflow.

  full_adder fa0(S[0], C0, A[0], B0, Op);    // Least significant bit.
  full_adder fa1(S[1], C1, A[1], B1, C0);
  full_adder fa2(S[2], C2, A[2], B2, C1);
  full_adder fa3(S[3], C3, A[3], B3, C2);    // Most significant bit.
endmodule
```



Q3

```
module fulladd(a,b,cin,s,cout); //submodule for fulladder
input a,b;
input cin;
output s;
output cout;
wire w1,w2,w3;
halfadd g1(a,b,w1,w2);
halfadd g2(w1,cin,s,w3);
assign cout = w3 | w2;
endmodule
```

```
module q3(a,b,cin,s,cout); //Main module of one digit BCD adder
input [3:0]a;
input [3:0]b;
input cin;
output[3:0]s;
output cout;
wire [3:0]w;
wire y,c0,c1,c2,c3,c4,c5;

fulladd m1(a[0],b[0],cin,w[0],c0);
fulladd m2(a[1],b[1],c0,w[1],c1);
fulladd m3(a[2],b[2],c1,w[2],c2);
fulladd m4(a[3],b[3],c2,w[3],c3);

assign y=c3|(w[3] & (w[2] | w[1]));

halfadd m5(w[1],y,s[1],c4);
fulladd m6(w[2],y,c4,s[2],c5);
halfadd m7(w[3],c5,s[3],cout);

assign s[0]=w[0];
endmodule

module halfadd(a,b,s,cout); //submodule for halfadder
input a,b;
output s,cout;
assign s=a^b;
assign cout=a&b;
endmodule
```


MAX+plus II - d:\190905494-angad sandhu\week 2\q3\q3 - [q3.scf - Waveform Editor]
MAX+plus II File Edit View Node Assign Utilities Options Window Help
Ref: 102.8ns
Time: 56.2ns
Interval: -46.6ns
102.8ns
Name:
Value:
50.0ns
100.0ns
150.0ns
200.0ns
cin 0
cout 0
a B 0001
b B 0100
s B 0110
0001
0100
0101
0110
0101
0110
0101
4:45 PM
2/8/2021