University of Victoria

CENG 241

Digital Design I

Lab 4 - 4-bit Binary Adder / Subtractor

Instructor:
Dr. Amirali Baniasadi

Teaching Assistant:
Grace Hui

Yves Senechal V00213837
Tyler Stephen V00812021
A01 - B03

June 22, 2015

1 Introduction

Combinational logic circuits can be implemented to add and subtract binary numbers. In this lab, a 4-bit binary adder/subtractor was designed, implemented and tested through the Xilinx ISE and Verilog HDL. Two 4-bit binary numbers and a single-bit control, to indicate adding or subtracting, were the inputs, while a 4-bit sum and single bit carry were the outputs.

2 Discussion

2.1 Binary addition and subtraction using ones complement representation

Addition of two 1-bit binary numbers (i.e. A + B) is achieved simply by adding A + B where a sum and carry are the outputs. Subtraction of two 1-bit binary numbers (i.e. A - B) can be achieved by taking the 2's complement of B and adding it to A. The 2's complement of B is the 1's complement of B, which is simply B' with the value 1 added to it. XOR gates have characteristic functions X $\oplus$ 0 = X, and X $\oplus$ 1 = X'.

For addition, where CONTROL is set to 0, the XOR gate provides no change to B, so B is added to A. For subtraction, CONTROL is set to 1 which inverts B through the XOR gate, CONTROL is then inputed as CIN which effectively adds 1 to the inverted B to become its 2's complement. This process can be expanded from the 1-bit system explained here, to the 4-bit system explored in the lab.

2.2 Building the hierarchical adder / subtractor

A general discussion of the design and the lab.

2.2.1 Half adder

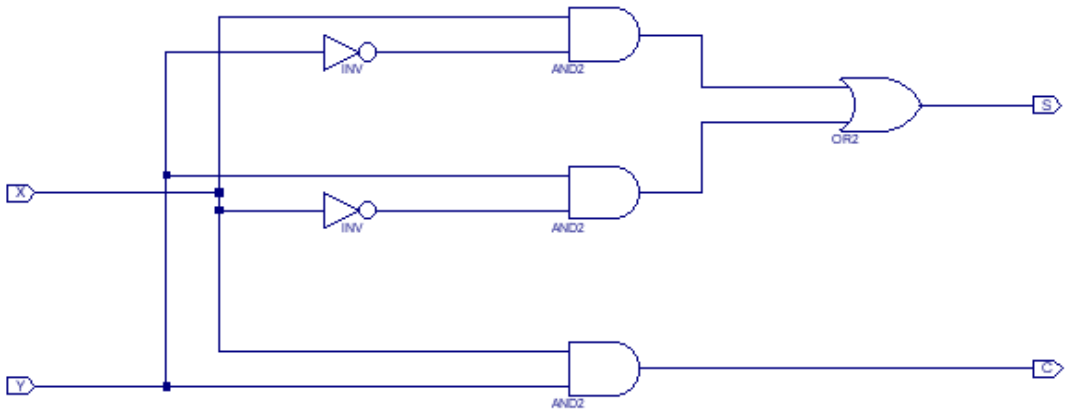


Figure 1: Schematic of the half adder

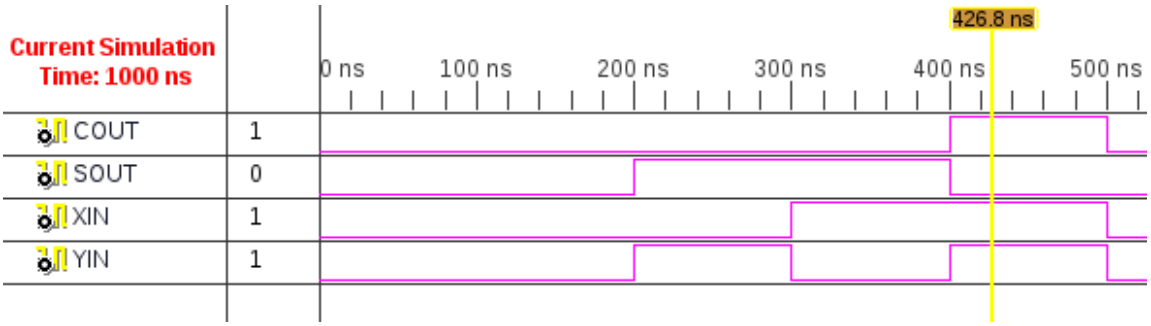


Figure 2: Test results of the half adder

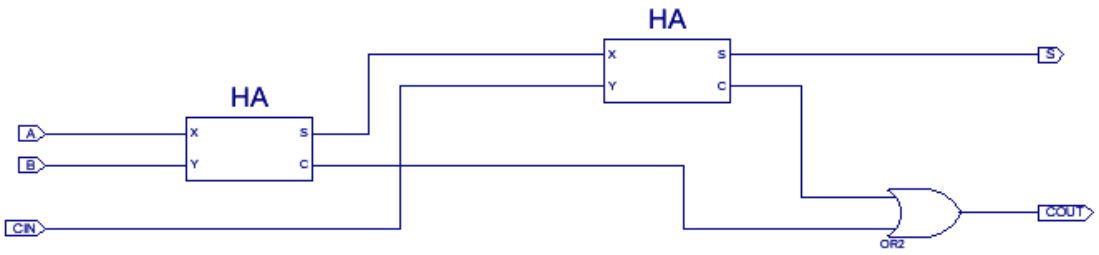


Figure 3: Schematic of the full adder

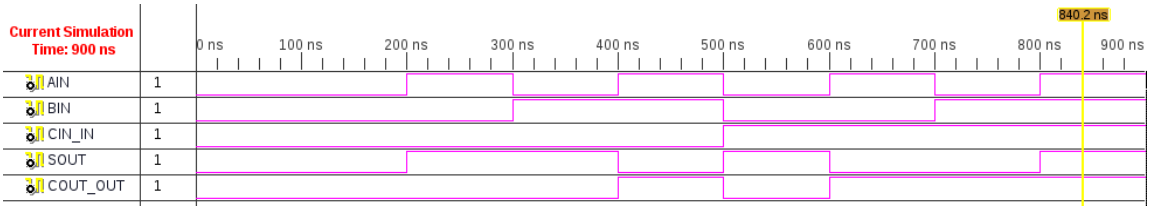


Figure 4: Test results of the full adder

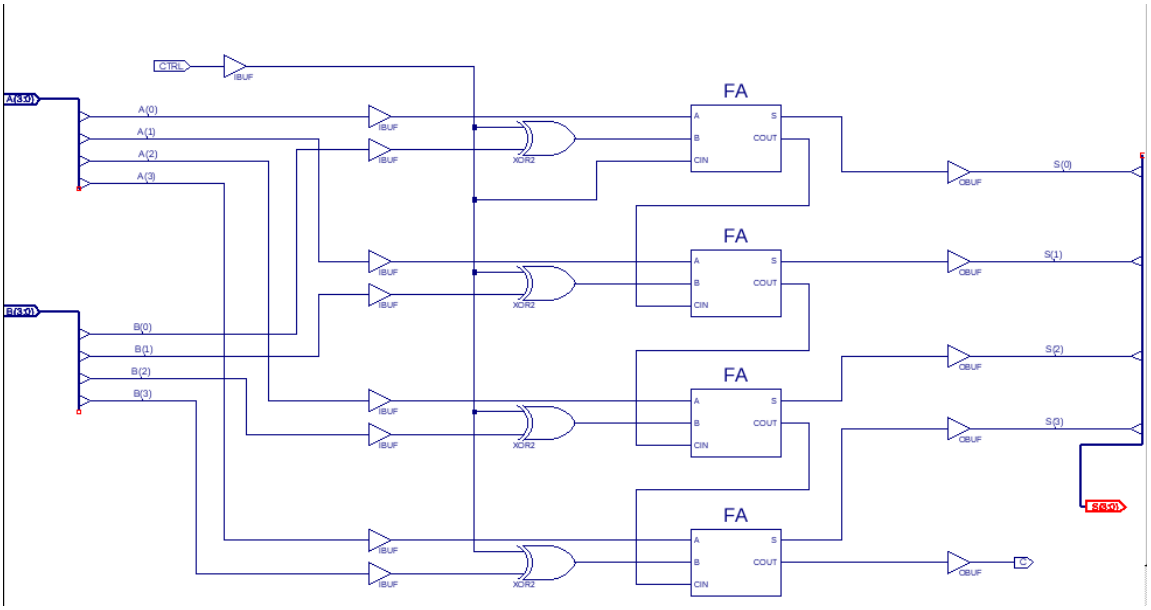
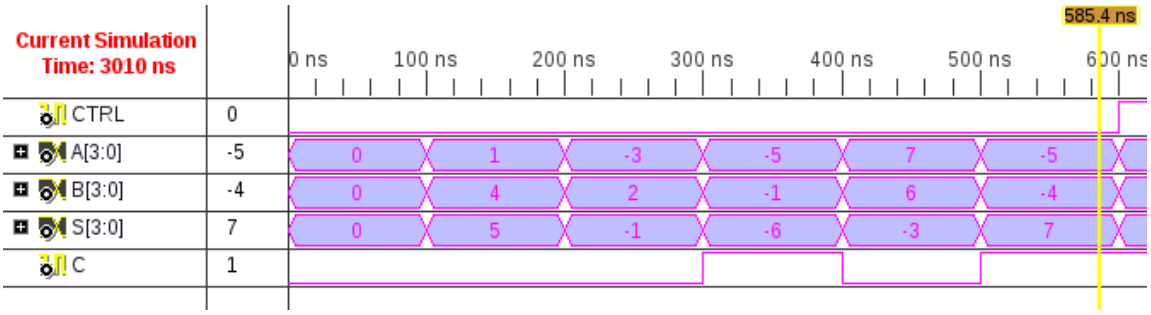
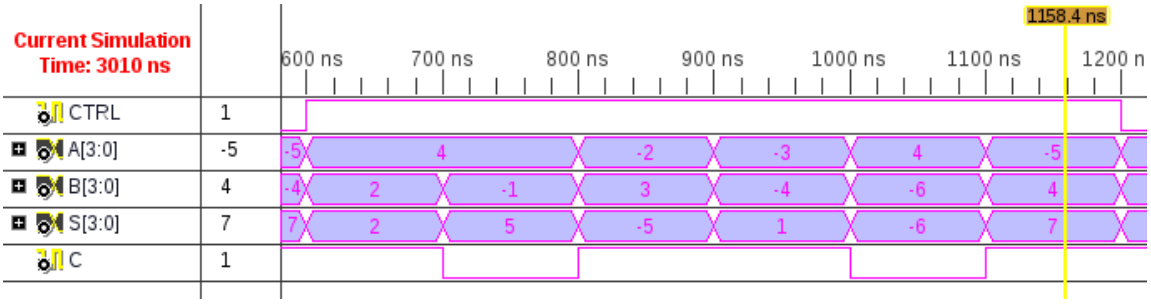


Figure 5: Schematic of the adder / subtractor



(a) Addition



(b) Subtraction

Figure 6: Test results of the adder / subtractor

CTRL	A	B	S	C
0	1	4	5	0
0	−3	2	−1	0
0	−5	−1	−6	1
0	7	6	−3	0
0	−5	−5	7	1
1	4	2	2	1
1	4	−1	5	0
1	−2	3	−5	1
1	−3	−4	1	1
1	4	−6	−6	0
1	−5	4	7	1

Table 1: Output of the adder / subtractor for several test cases, in decimal

2.2.2 Full adder

2.2.3 Adder / subtractor

2.3 Verilog hierarchical adder / subtractor

2.3.1 Listings

```
module FA(A, B, CIN, S, COUT)
  input A, B, CIN;
  output S, COUT;

  HA U0(A, B, S0, C0);
  HA U1(S0, CIN, S, C1);

  assign COUT = C0 | C1;
```

FA.v

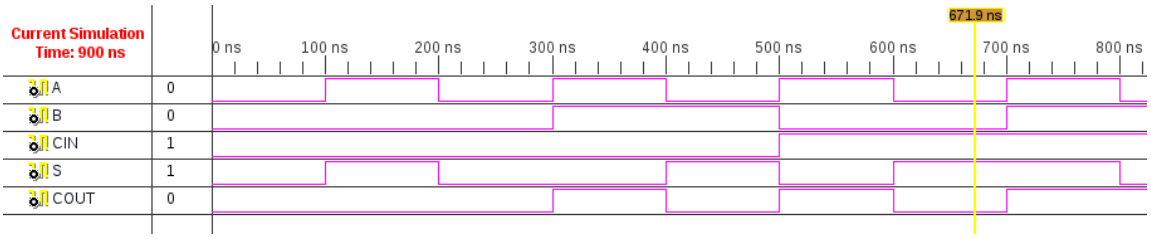
```
module AddSub(A0, A1, A2, A3, B0, B1, B2, B3, CTRL, S0, S1, S2, S3, C)
  input A0, A1, A2, A3, B0, B1, B2, B3, CTRL;
  output S0, S1, S2, S3, C;

  FA U0(A0, B0 ^ CTRL, CTRL, S0, COUT0);
  FA U1(A1, B1 ^ CTRL, COUT0, S1, COUT1);
  FA U2(A2, B2 ^ CTRL, COUT1, S2, COUT2);
  FA U3(A3, B3 ^ CTRL, COUT2, S3, C);
```

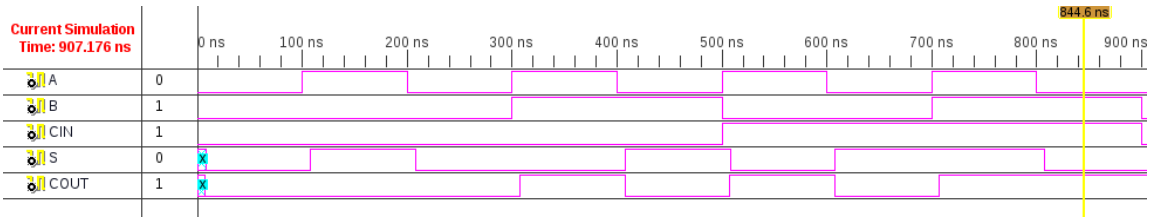
AddSub.v

2.3.2 Simulation results and delay times

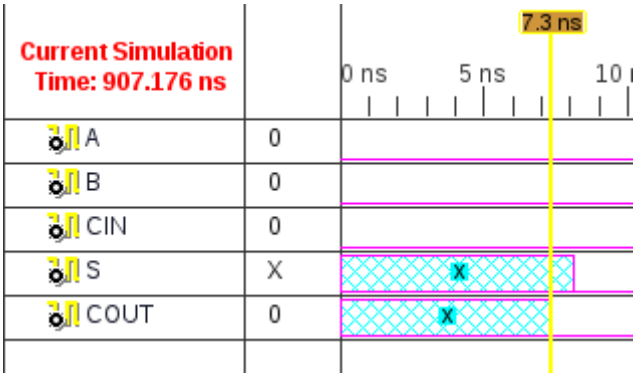
3 Conclusion



(a) Functional simulation



(b) Post place timing



(c) Delay

Figure 7: Simulation results for the full adder written in verilog

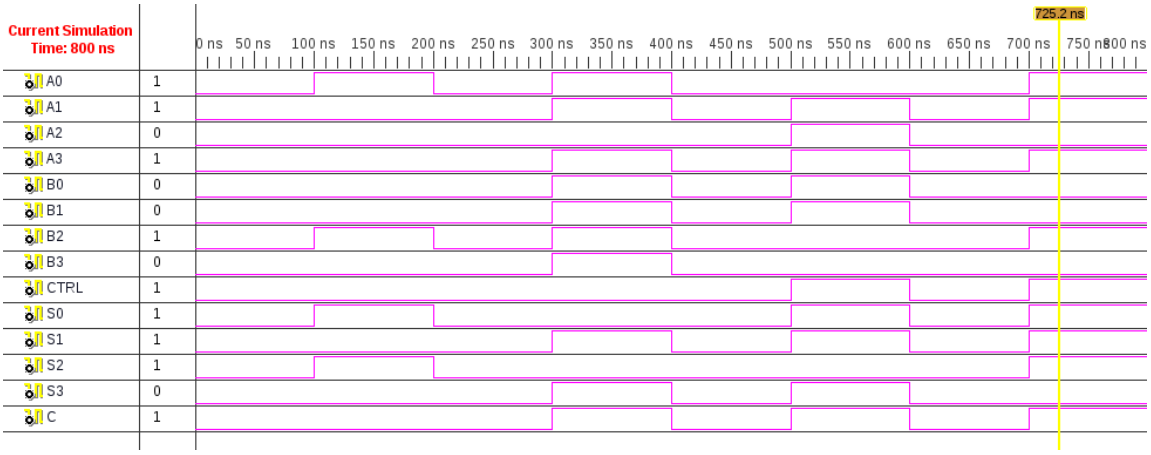


Figure 8: Functional simulation results for adder / subtractor written in verilog