CSE260: Digital Logic Design

Project Title: Arithmetic Calculator

Group Number: 2

Members:

Jonathon Lenny Baroi - 20201013
Nafiun Al Amin - 20201069
Ajmain Mahtab - 20201034
Mashnoon Mayad - 20201033

Theory Section: 9

Introduction and Objective

No matter what degree one may be studying, one must always remember that Mathematics is a key component of everyday life. As such, an efficient way to calculate simple arithmetic operations is always of importance. A calculator allows us to do this, and using digital logic, we can create such a device. Our objective was to create a basic arithmetic calculator, with basic mathematical operations on two numbers.

Proposed Model:

The calculator should be capable of three operations: addition, subtraction and multiplication.

The model for the calculator was split into a few parts. Firstly, an input system so that the user can decide if the user would like to add, subtract or multiply. The user can also input the numbers that he would like to operate on and the circuit would perform the selected operation.

Coming to the operations, for addition and subtraction, we created an 8 bit parallel adder cum subtractor for those two operations, which outputs an 8 bit number at maximum.

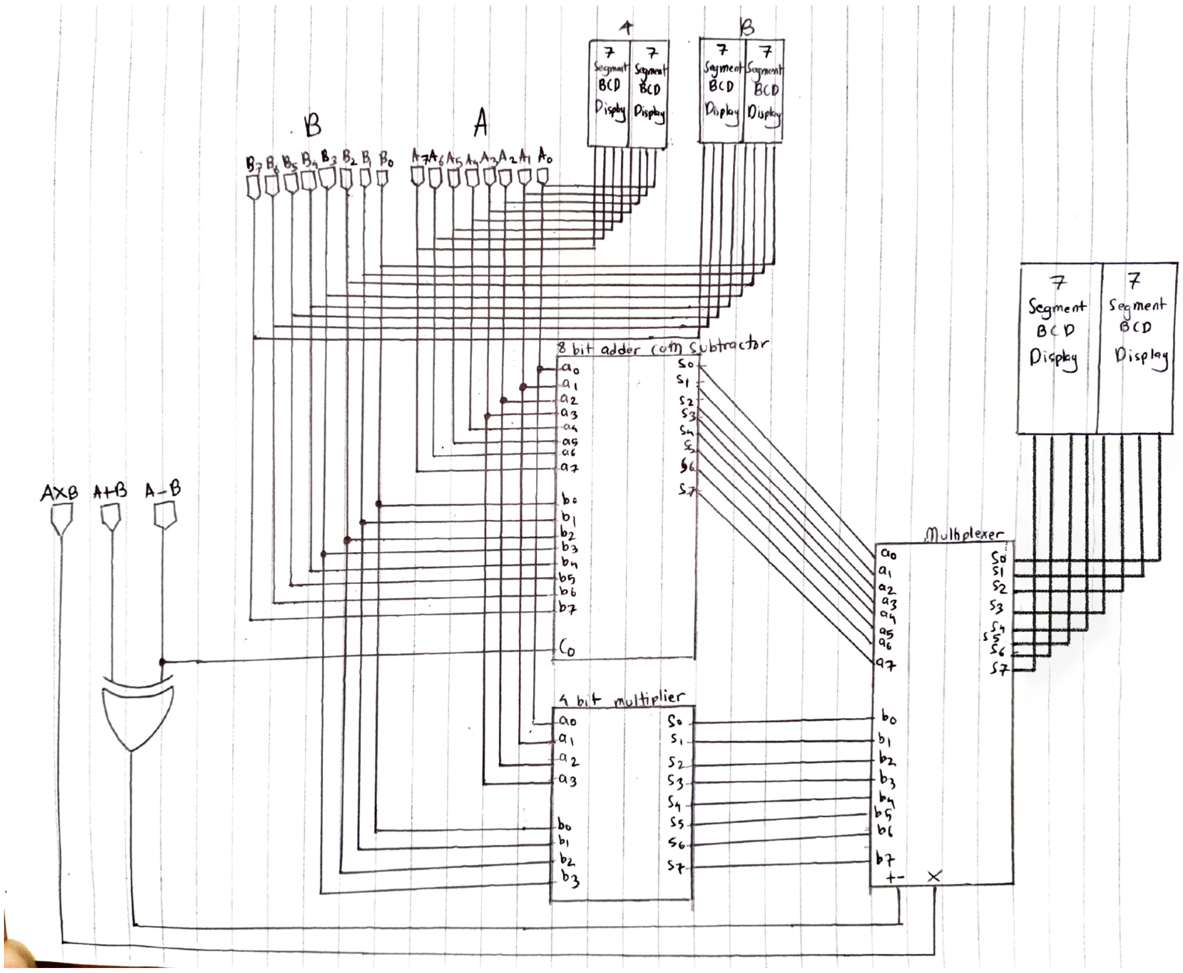
For multiplication, we created a 4 bit multiplier to multiply two 4 bit numbers and obtain a resultant 8 bit number.
There are additional displays to show the operands and the result in hexadecimal.

Experimental Setup:

The components used in our circuit include:

- 7 segment BCD (Binary Coded Decimal) Display
- 4-bit binary Full Adders (74LS83)
- 8:4 Multiplexer (4019)
- AND gates
- LOGIC STATE
- XOR gates

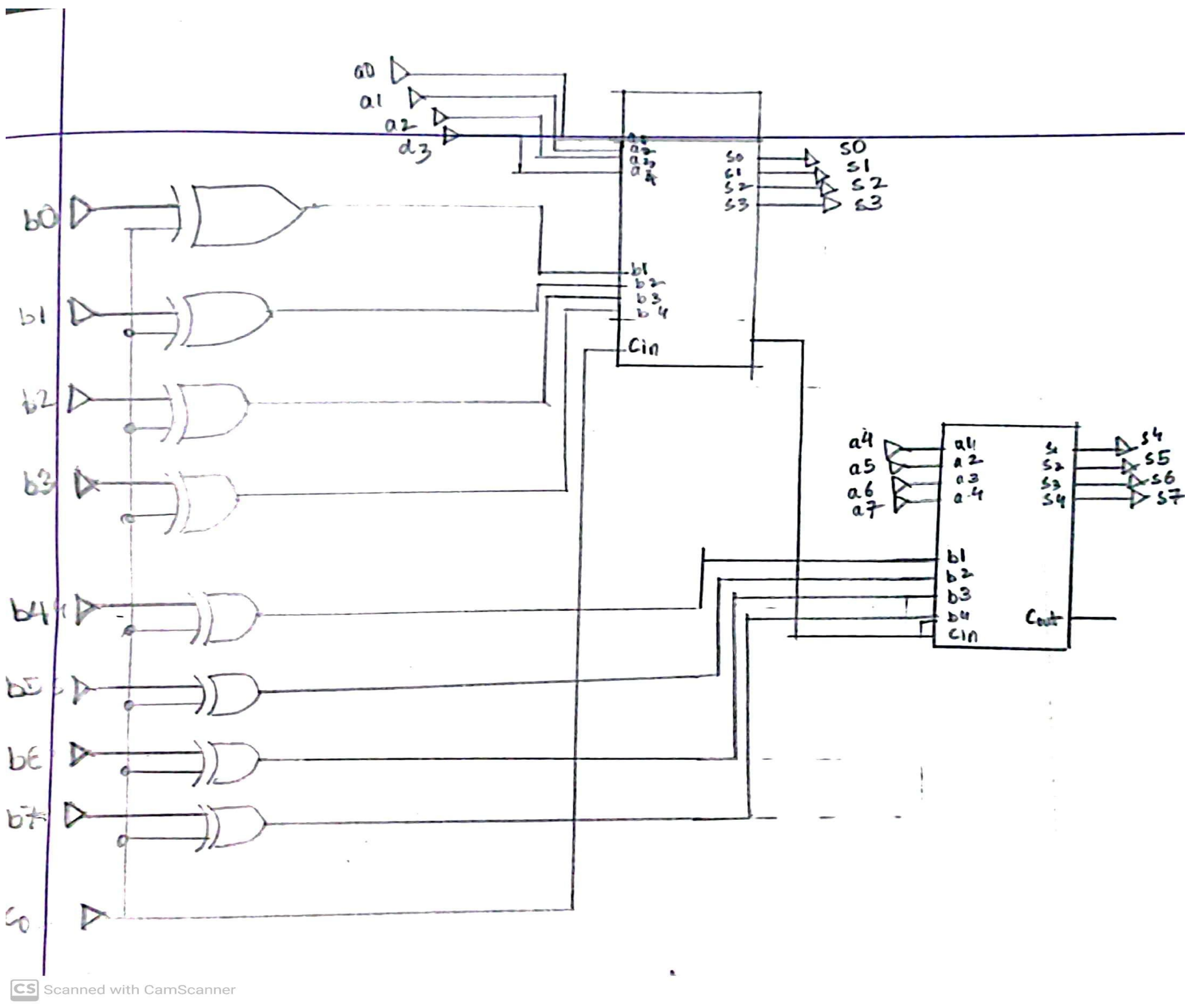
Figure 1: Main Circuit



A
B
7 Segment BCD Display
B
A
B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0
A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0
8 bit adder cum subtractor
a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7
C_0
S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
A×B
A+B
A−B
4 bit multiplier
Multiplexer
+−
×

The LOGICSTATE has mainly been used to allow the user to input their values (in binary) into the calculator. It has also been used as a switch, to choose between different operations. Each of three LOGIC STATES control the operation, and each of them should only be HIGH while the others are LOW.

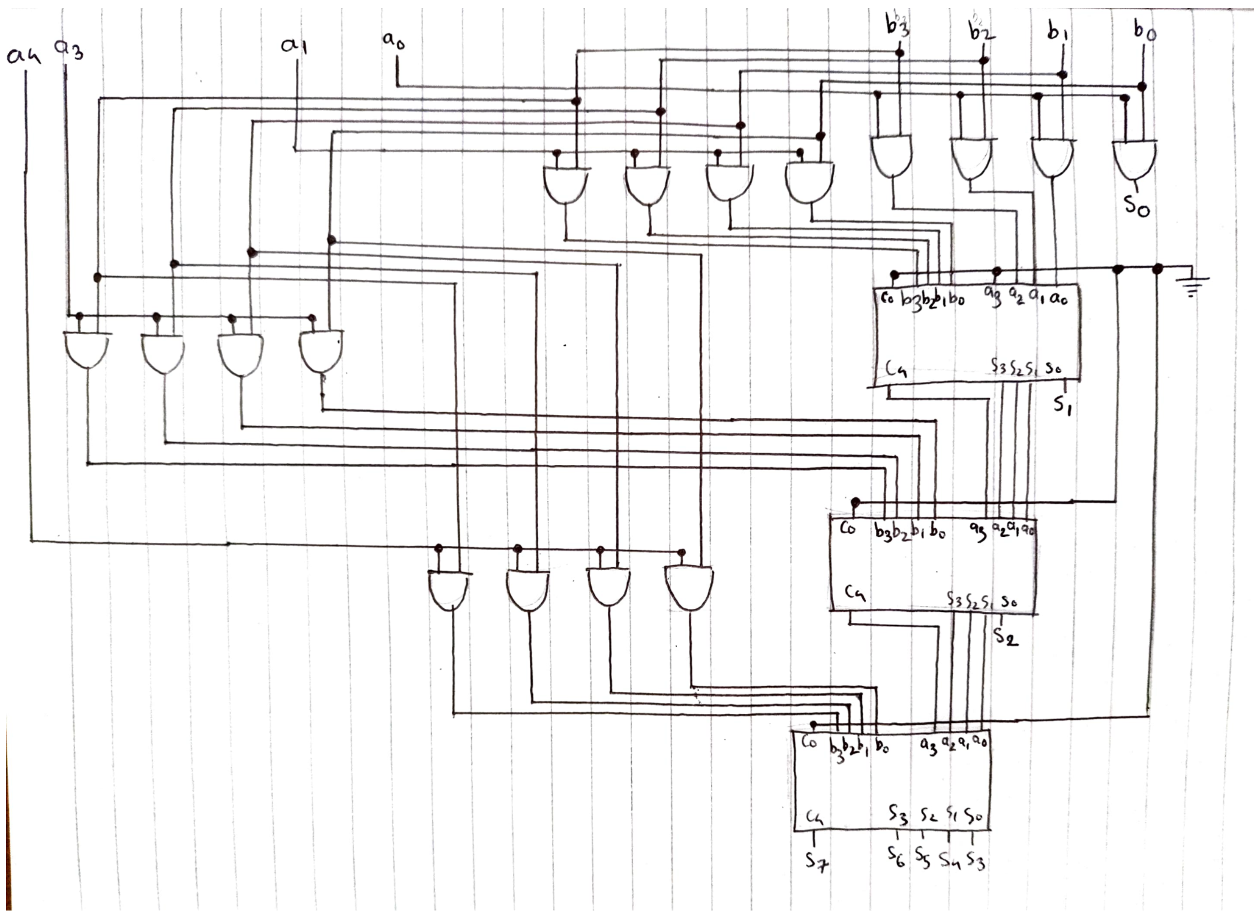
Figure 2: 8 bit Full Adder cum Subtractor



Two 4-bit full adders have been connected together to create one single 8 bit full adder cum subtractor . This is achieved by connecting the Carry-out (C_{out}) of the first full adder with the Carry-in (C_{in}) of the second full adder.

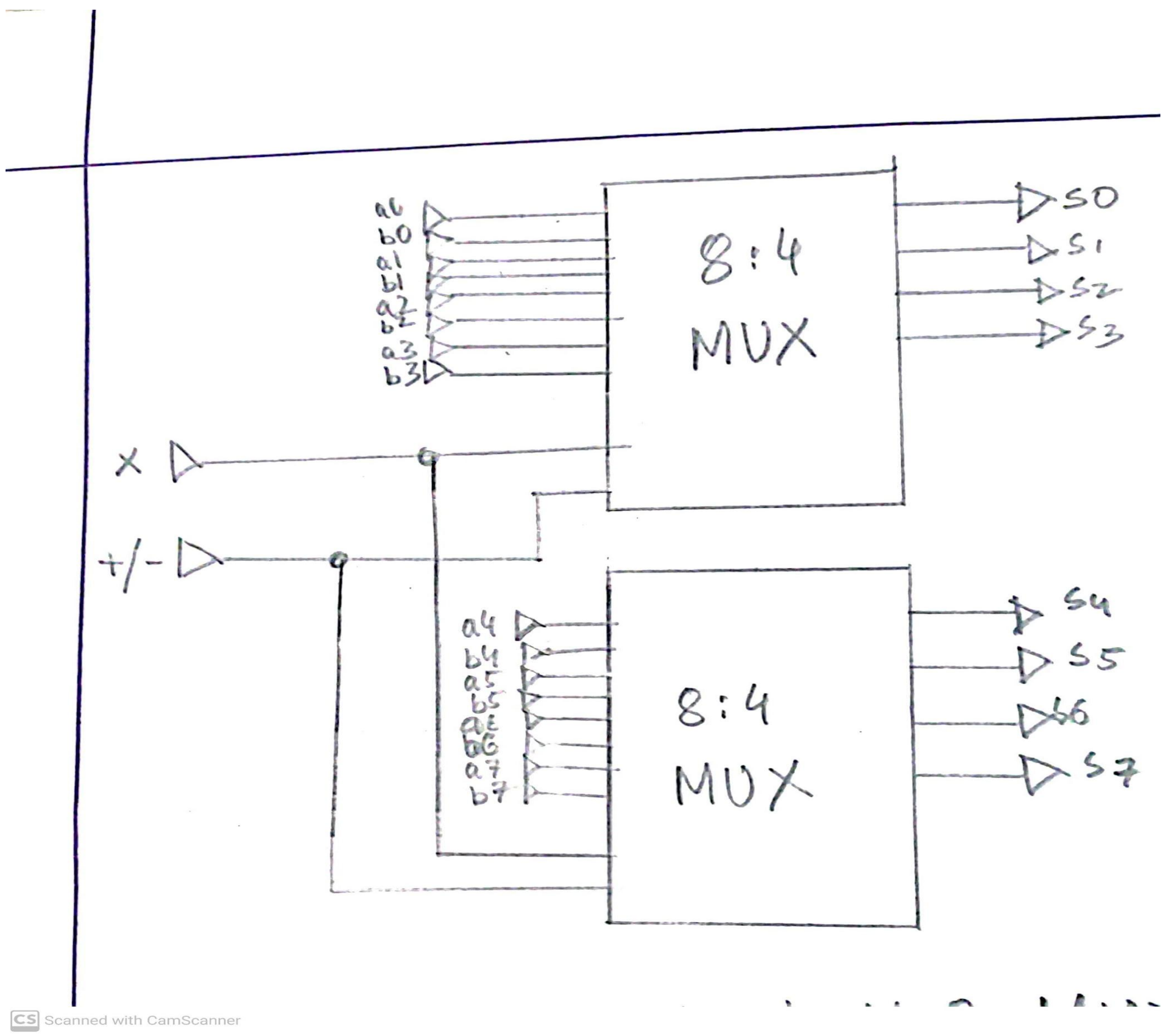
Alongside that, XOR gates have been added to the inputs to allow subtraction to take place, when needed. By default, with a C_{in} value of 0, the circuit performs 8 bit addition. However, if the user activates subtraction, the C_{in} is changed to 1 and the same circuit now performs 8 bit subtraction.

Figure 3: 4 bit Multiplier



For multiplication of 4 bit numbers, 4 numbers are created using AND gates, which are the results of multiplication of the digits of one number with another. These numbers are then shifted accordingly to the left following the rules of multiplication, and then added using 4 bit adders. The result is an 8 bit number.

Figure 4: 16 : 4 Multiplexer



After calculation, the output of the adder/subtractor and multiplier is passed through a multiplexer. Depending on which operation was picked, the multiplexer outputs the value from that specific operation. For example, if addition was performed, only the output from the adder cum subtractor is considered and the subsequent output is returned.

Finally, the output goes through a 7 segment BCD display. The results of the binary operation are displayed in hexadecimal form, and the display offers a very discernible and easy to read output.

Results and Analysis:

Op input				Input A									Input B									Output							
c0	c1	c2		A7	A6	A5	A4	A3	A2	A1	A0		B7	B6	B5	B4	B3	B2	B1	B0		S7	S6	S5	S4	S3	S2	S1	S0
0	0	1		0	0	0	0	1	1	1	1		0	0	0	0	1	1	1	1		1	1	1	0	0	0	0	1
0	0	1		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	0	1		0	0	0	0	0	1	0	0		0	0	0	0	0	0	0	1		0	0	0	0	1	0	0	0
0	0	1		0	0	0	0	0	1	0	1		0	0	0	0	1	0	1	0		0	1	1	0	0	1	0	0
0	0	1		0	0	0	0	1	0	0	1		0	0	0	0	1	0	0	1		0	1	0	1	0	0	0	1
0	0	1		0	0	0	0	1	1	0	0		0	0	0	0	1	1	0	0		1	0	0	1	0	0	0	0
0	0	1		0	0	0	0	1	0	1	1		0	0	0	0	1	0	1	1		0	1	1	1	1	0	0	1
0	0	1		0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	1
0	1	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0	1	0		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	0
0	1	0		1	0	1	0	1	0	1	0		1	0	1	0	1	0	1	0		0	1	0	1	0	1	0	0

Op input				Input A									Input B									Output							
0	1	0		1	1	1	1	0	0	0	0		1	1	1	1	0	0	0	0		1	1	1	0	0	0	0	0
0	1	0		0	0	0	0	1	1	1	1		0	0	0	0	1	1	1	1		0	0	0	1	1	1	1	0
0	1	0		0	0	1	1	0	0	1	1		0	0	1	1	0	0	1	1		0	1	1	0	0	1	1	0
0	1	0		0	0	0	1	1	1	0	0		0	0	0	1	1	1	0	0		0	0	1	1	1	0	0	0
0	1	0		1	1	1	0	0	0	1	1		1	1	1	0	0	0	1	1		1	1	0	0	0	1	1	0
1	0	0		1	1	1	0	0	0	1	1		1	1	1	0	0	0	1	1		0	0	0	0	0	0	0	0
1	0	0		1	1	1	0	0	1	1	0		1	0	1	0	1	0	1	0		0	0	1	1	1	1	0	0
1	0	0		1	1	1	1	1	1	1	1		0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1
1	0	0		0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1		0	0	0	0	0	0	0	1
1	0	0		1	1	1	0	1	1	1	1		0	0	0	1	0	0	0	0		1	1	0	1	1	1	1	1
1	0	0		1	1	1	1	0	0	0	0		0	0	0	0	1	1	1	1		1	1	1	0	0	0	0	1
1	0	0		1	1	1	1	1	1	0	0		1	1	1	1	1	0	0	0		0	0	0	0	0	1	0	0
1	0	0		1	1	1	0	0	0	0	0		0	0	0	0	0	1	1	1		1	1	0	1	1	0	0	1

The operator inputs are c0, c1 and c2. These control which operations are being done. When only c0 is 1, input B is subtracted from A. When only c1 is

1, the inputs are added together. When only c2 is 1, input A and B are multiplied.

It is to be noted that during multiplication, the number being used as input should be a 4-bit number at maximum, so as to not exceed the output range of 8 bits when operated.

Conclusion:

While the calculator can perform most arithmetic operations, **it cannot perform divisions**. It also displays the output in **hexadecimal**, and not decimal. **The multiplier also cannot take 8-bit inputs** like the adder/subtractor, and is limited to taking 4 bits, as multiplication of two 8-bit numbers would result in a 16-bit number, which is beyond the output range of the calculator.

For **subtraction**, negative results cannot be represented.

It is to be noted that the selector also requires the user to have a HIGH input on the corresponding operation, and a low on the non-corresponding ones; if not, the output does not match, but the output does not notify anything about the error. A low input on all 3 operator values also results in an error, where an output is still generated (given that a number is still being input).

Also, the input numbers have to be input in binary, **missing the user-friendliness** of a decimal **numpad** with numbers and operations.