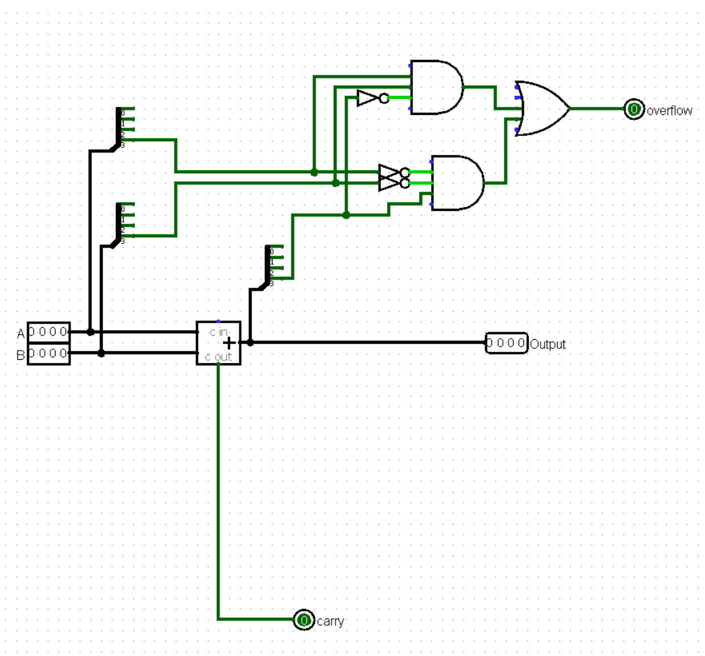
Report

1 – Arithmetic

1. **4-bit Adder:**
   We already implemented the 1 bit adder in one of the labs before, so the only thing we need to implement here is the overflow flag.
   The circuit is implemented using a full 4-bit adder with a carry flag and an overflow flag. The carry flag is a single pin which lights up whenever there is a carry by the adder.
   The circuit is designed to make the following operation:

$$A_3A_2A_1A_0 + B_3B_2B_1B_0 = F_3F_2F_1F_0$$

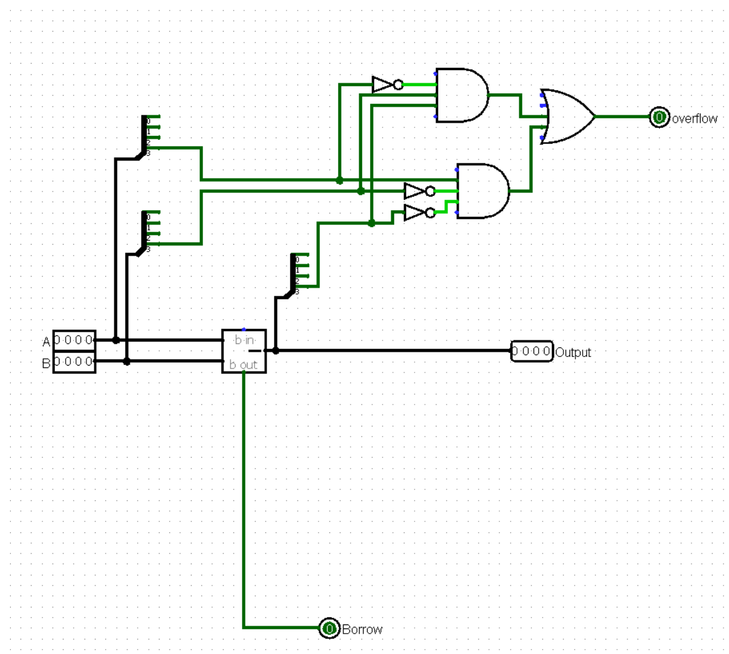
   The overflow flag $(O) = A_3B_3\overline{F_3} + \overline{A_3}\,\overline{B_3}F_3$



2. **4-bit Subtracter:**
   The circuit is implemented using a full 4-bit subtractor and a borrow flag and an overflow flag.
   The circuit is designed to make the following operation:

$$A_3A_2A_1A_0 - B_3B_2B_1B_0 = F_3F_2F_1F_0$$

   The borrow flag maps to the borrow output of the subtractor.
   The overflow bit $(O) = A_3\overline{B_3}\,\overline{F_3} + \overline{A_3}B_3F_3$



3. **Multiplication:**

Multiplication:

A3 A2 A1 A0

× B3 B2 B1 B0

A3B0 A3B0 A3B0 A3B0 A3B0' A2B0 A1B0 A0B0

\+ 1

\+ A3B1 A3B1 A3B1 A3B1 A3B1' A2B1 A1B1 A0B1

\+ 1

\+ A3B2 A3B2 A3B2 A3B2 A3B2' A2B2 A1B2 A0B2

\+ 1

\- A3B3 A3B3 A3B3 A3B3 A3B3 A2B3 A1B3 A0B3

\+ A3B3' A3B3' A3B3' A3B3' A3B3 A2B3' A1B3' A0B3'

\+ 1

\- 0 1 1 1 1

\+ 1 0 0 0 1

\+ 1

				A3B0'	A2B0	A1B0	A0B0
			A3B1'	A2B1	A1B1	A0B1	
		A3B2'	A2B2	A1B2	A0B2		
	A3B3	A2B30'	A1B30'	A0B30'			
1	0	0	1				
P7	P6	P5	P4	P3	P2	P1	P0

- Subtract the bit of sign (B3) if it =1, and if it =0 no problem.
- Add 1 to MSB in each number to remove the extension of the sign.
- Subtract the ones I have added(in consider that each number =8bit).

Overflow flag:

- To be no overflow all bits P3-P7 must express the sign (all = 0,all=1)no one of them different.

The circuit is very big and I simply can't include it in the report.

2 – Logic

In Logisim, if you have 2 4-bit buses then by connecting them to any kind of gate would perform that gate to each bit.
For example, if you connect 2 4-bit buses to an AND gate, it would AND each bit with its corresponding bit.
This is how we did the 4 required logical operations.

3 – Bit-wise operations:

1. **Display A in 2's complement:**

   - Truth Table:

Input (A3 A2 A1 A0)	**Output (B3 B2 B1 B0)**	Zero Flag (F)
0000	0000	1
0001	1111	0
0010	1110	0
0011	1101	0
0100	1100	0
0101	1011	0
0110	1010	0
0111	1001	0
1000	1000	0
1001	0111	0
1010	0110	0
1011	0101	0
1100	0100	0
1101	0011	0
1110	0010	0
1111	0001	0

   - Minterms:
     - $B_o = \sum m(1, 3, 5, 7, 9, 11, 13, 15)$
     - $B_1 = \sum m(1, 2, 5, 6, 9, 10, 13, 14)$
     - $B_2 = \sum m(1, 2, 3, 4, 9, 10, 11, 12)$
     - $B_3 = \sum m(1, 2, 3, 4, 5, 6, 7, 8, 9)$
   - Kmaps:

− B_o :

A_3A_2 \ A_1A_0	00	01	11	10
00		1	1	
01		1	1	
11		1	1	
10		1	1	

− B_1 :

A_3A_2 \ A_1A_0	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

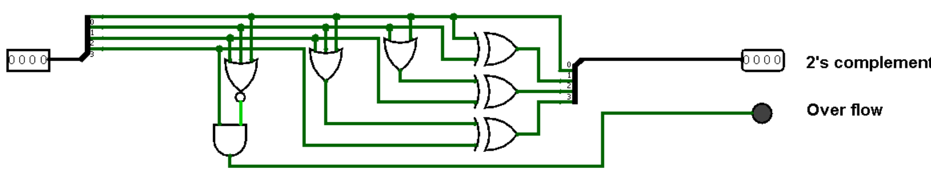
− B_2 :

A_3A_2 \ A_1A_0	00	01	11	10
00		1	1	1
01	1			
11	1			
10		1	1	1

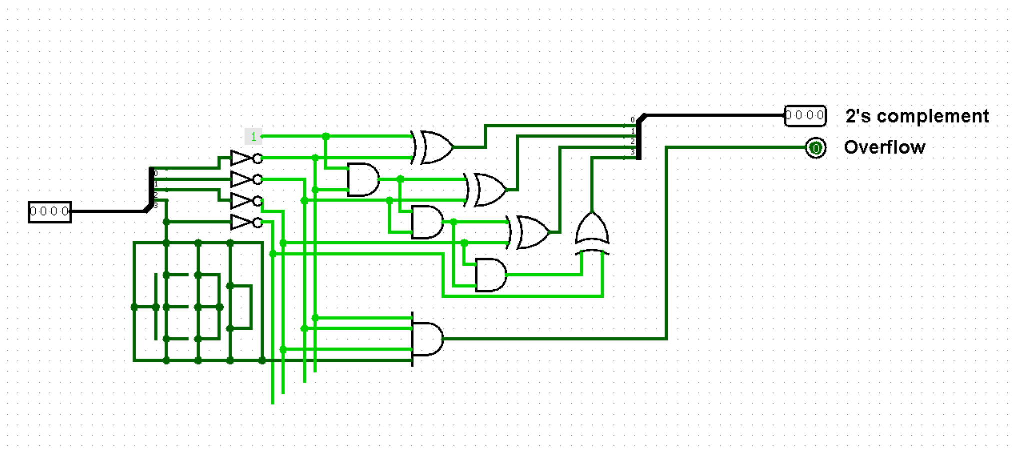
– B_3 :

$A_3A_2 \backslash A_1A_0$	00	01	11	10
00		1	1	1
01	1	1	1	1
11				
10	1	1		

- Equations:
  $B_0 = A_0$
  $B_1 = A_1 \oplus A_0$
  $B_2 = A_2 \oplus (A_1 + A_0)$
  $B_3 = A_3 \oplus (A_2 + A_1 + A_0)$
  $F = \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}$
- Final Circuit Diagram:
  The equations would give the following circuit:



But..... Another good way is to do 1's complement and add 1 to it:



2. **Display B in 1's complement:**

- Truth Table:

Input (A3 A2 A1 A0)	Output (B3 B2 B1 B0)	Zero Flag (F)
0000	1111	1
0001	1110	0
0010	1101	0
0011	1100	0
0100	1011	0
0101	1010	0
0110	1001	0
0111	1000	0
1000	0111	0
1001	0110	0
1010	0101	0
1011	0100	0
1100	0011	0
1101	0010	0
1110	0001	0
1111	0000	1

- Minterms:
  - $B_o = \sum m(0, 2, 4, 6, 8, 10, 12, 14)$
  - $B_1 = \sum m(0, 1, 4, 5, 8, 9, 12, 13)$
  - $B_2 = \sum m(0, 1, 2, 3, 8, 9, 10, 11)$
  - $B_3 = \sum m(0, 1, 2, 3, 4, 5, 6, 7)$
- Kmaps:

– B_o :

$A_3A_2 \backslash A_1A_0$	00	01	11	10
00	1			1
01	1			1
11	1			1
10	1			1

– B_1 :

$A_3A_2 \backslash A_1A_0$	00	01	11	10
00	1	1		
01	1	1		
11	1	1		
10	1	1		

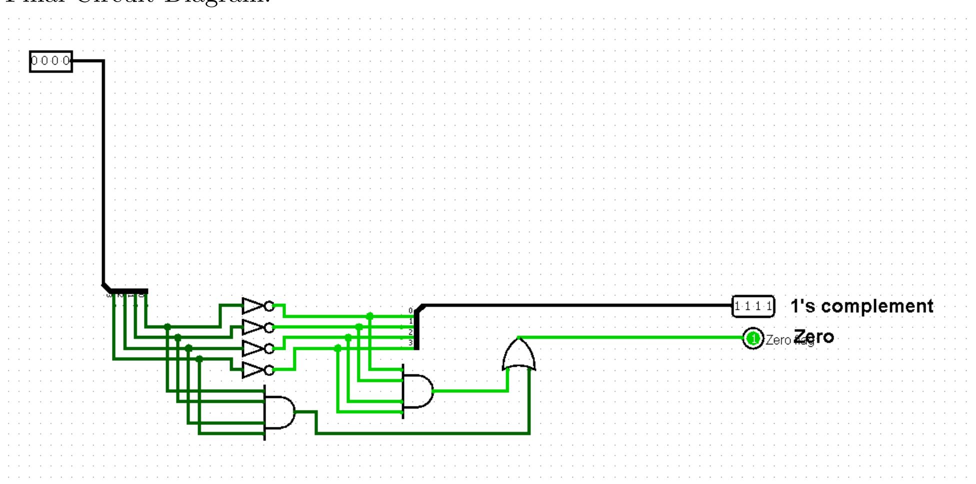
– B_2 :

$A_3A_2 \backslash A_1A_0$	00	01	11	10
00	1	1	1	1
01				
11				
10	1	1	1	1

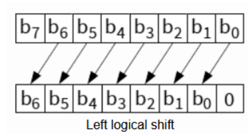
- B_3 :

$A_3A_2 \backslash A_1A_0$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11				
10				

- Equations:
  $B_o = \overline{A_0}$
  $B_1 = \overline{A_1}$
  $B_2 = \overline{A_2}$
  $B_3 = \overline{A_3}$
  $F = \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0} + A_3A_2A_1A_0$
- Final Circuit Diagram:



3. **Shift A left logical:**
   A logical shift to the left for an 8-bit number is done according to this diagram:

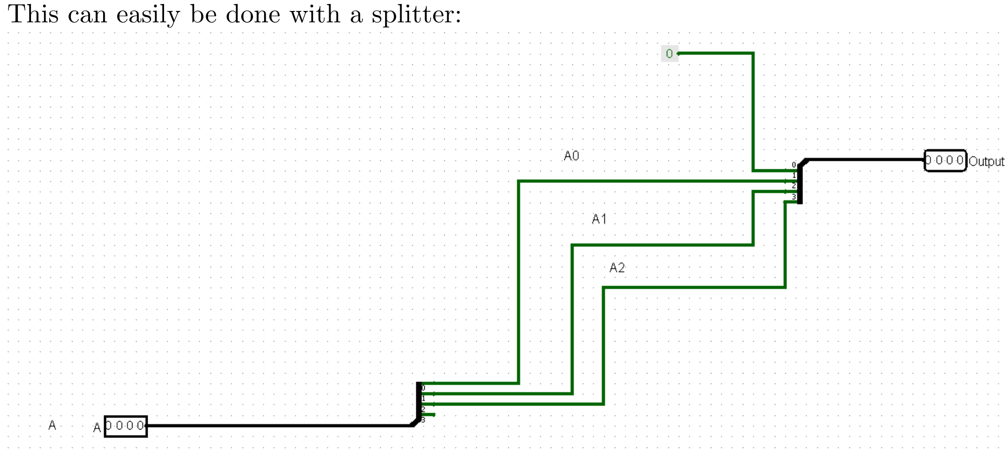


But we will only consider the 4-bit case.

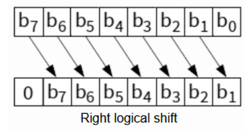
- Truth Table:

A_3	A_2	A_1	A_0	A_3'	A_2'	A_1'	A_0'
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	1	0	1	0
0	1	1	0	1	1	0	0
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	1	0
1	1	1	0	1	1	0	0
1	1	1	1	1	1	1	0

- Equations:
  $A_0' = 0$
  $A_1' = A_0$
  $A_2' = A_1$
  $A_3' = A_2$
- This can easily be done with a splitter:



4. **Shift A right logical:**
A logical shift to the right for an 8-bit number is done according to this diagram:

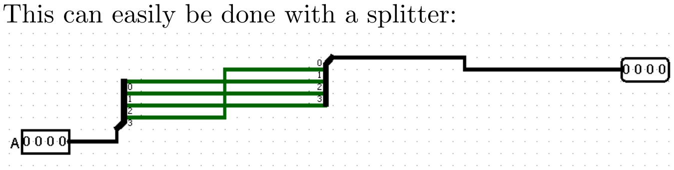


But we will only consider the 4-bit case.

- Truth Table:

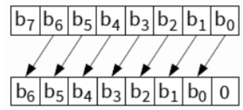
A_3	A_2	A_1	A_0	A'_3	A'_2	A'_1	A'_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	1
1	1	0	0	0	1	1	0
1	1	0	1	0	1	1	0
1	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1

- Equations:
  $A'_0 = A_1$
  $A'_1 = A_2$
  $A'_2 = A_3$
  $A'_3 = 0$
- This can easily be done with a splitter:



5. **Shift A left Arithmetic:**
An Arithmetic shift to the left for an 8-bit number is done according to

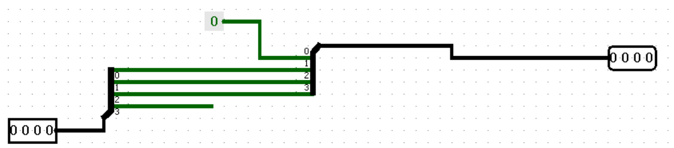
this diagram:



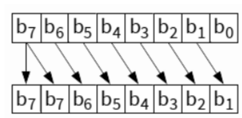
Left arithmetic shift

It can be seen that the left Arithmetic shift is identical to the left logical shift, so there is no need for a truth table.

- Equations:
  $A'_0 = 0$
  $A'_1 = A_0$
  $A'_2 = A_1$
  $A'_3 = A_2$
- This can easily be done with a splitter:



6. **Shift A right Arithmetic:**
   An Arithmetic shift to the right for an 8-bit number is done according to this diagram:



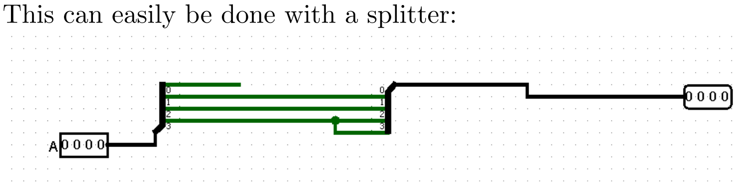
Right arithmetic shift

But we will only consider the 4-bit case.

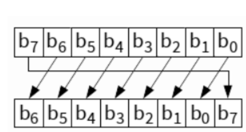
- Truth Table:

A_3	A_2	A_1	A_0	A'_3	A'_2	A'_1	A'_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	0
1	0	1	1	1	1	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1

- Equations:
  $A'_0 = A_1$
  $A'_1 = A_2$
  $A'_2 = A_3$
  $A'_3 = A_3$
- This can easily be done with a splitter:



7. **Shift A left Circular:**
   A Circular shift to the left for an 8-bit number is done according to this diagram:



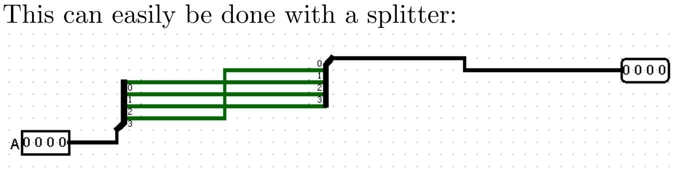
Left circular shift

But we will only consider the 4-bit case.

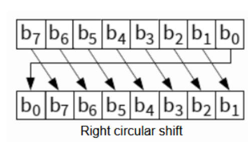
- Truth Table:

A_3	A_2	A_1	A_0	A'_3	A'_2	A'_1	A'_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	1	0	1	0
0	1	1	0	1	1	0	0
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	1	0	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	1
1	1	0	1	1	0	1	1
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1

- Equations:
  $A'_0 = A_3$
  $A'_1 = A_0$
  $A'_2 = A_1$
  $A'_3 = A_2$
- This can easily be done with a splitter:



8. **Shift A Right Circular:**
   A Circular shift to the left for an 8-bit number is done according to this diagram:

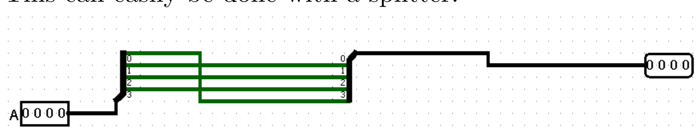


But we will only consider the 4-bit case.

- Truth Table:

A_3	A_2	A_1	A_0	A'_3	A'_2	A'_1	A'_0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	1	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	0	1	0	0	1	0	1
1	0	1	1	1	1	0	1
1	1	0	0	0	1	1	0
1	1	0	1	1	1	1	0
1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1

- Equations:
  $A'_0 = A_1$
  $A'_1 = A_2$
  $A'_2 = A_3$
  $A'_3 = A_0$
- This can easily be done with a splitter:



4 – Le finale: