

- I implemented the Full Adder by using 2 Half Adders and 1 OR gate. But before that I had to implement the half adder using a Xor and AND gate.

#### HALF ADDER

Xor (a=a, b=b, out=sum);

And (a=a, b=b, out=carry);

- Then I opened the script file for a full adder and used 2 half adders and 1 OR gate.

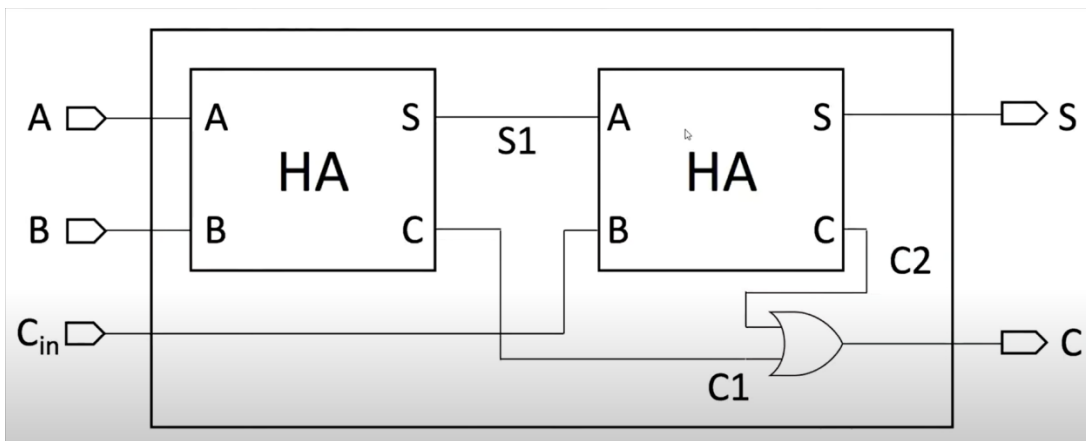
#### FULL ADDER

HalfAdder (a=a, b=b, sum=sum1, carry=carry1);

HalfAdder (a=sum1, b=c, sum=sum, carry=carry2);

Or (a=carry1, b=carry2, out=carry);

- I've provided the diagram of the full adder implemented.



- The full adder takes 3 inputs; a, b, and c. the first half adder takes a=a and b=b as inputs and outputs the sum1 and carry1. The second half adder takes inputs as a=sum1 and b=c and output the sum and carry2. Then the OR gate takes 2 inputs; a=carry1 from 1<sup>st</sup> half adder and b=carry2 from 2<sup>nd</sup> half adder. Finally outputs out=carry. So, on the whole it takes 3 inputs and gives 2 outputs as sum and carry. I've loaded and ran the file in Nand Tetris and it worked perfectly. I've provided the page below.

Hardware Simulator (2.5) - D:\Downloads\nand2tetris\nand2tetris\projects\02\FullAdder.hdl

File View Run Help

Chip Name: FullAdder Time: 0

Input pins		Output pins	
Name	Value	Name	Value
a	1	sum	1
b	1	carry	1
c	1		

HDL

```

// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems:
// by Nisan and Schocken, MIT Press"
// File name: projects/02/FullAdder.hdl
/**
 * Computes the sum of three bit
 */
CHIP FullAdder {
    IN a, b, c; // 1-bit inputs
    OUT sum, // Right bit of
    carry; // Left bit of

    PARTS:
    /**/ Replace this comment with
  
```

Internal pins

Name	Value
sum1	0
carry1	1
carry2	0

```

set b 0;
set c 0;
eval;
output;

set c 1;
eval;
output;

set b 1;
set c 0;
eval;
output;

set c 1;
eval;
output;

set a 1;
set b 0;
set c 0;
eval;
output;

set b 1;
set c 0;
eval;
output;

set c 1;
eval;
output;

set a 1;
set b 1;
set c 0;
eval;
output;

set c 1;
eval;
output;
  
```

End of script - Comparison ended successfully