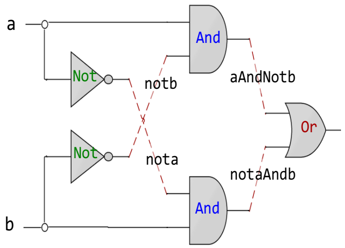
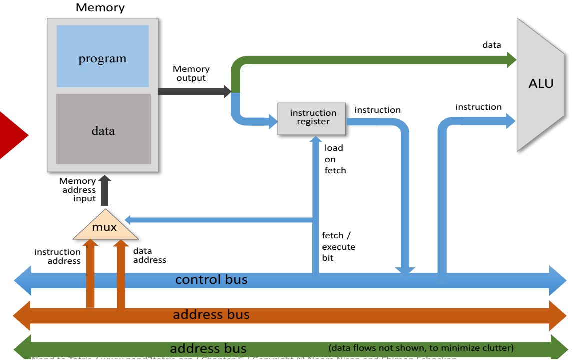
Computer Organization & Assembly Language Programming




```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not(in=a, out=nota);
  Not(in=b, out=notb);
  And(a=nota, b=b, out=w1);
  And(a=a, b=notb, out=w2);
  Or(a=w1, b=w2, out=out);
}
```




```
@R1
D=M
@temp
M=D
```

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
```

Lecture # 08

Design of ALU- I

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  printf("Learning is fun with Arif\n");
  exit(0);
}
```

```
global main
SECTION .data
   msg: db "Learning is fun with Arif", 0Ah, 0h
   len_msg: equ $ - msg
SECTION .text
   main:
      mov rax,1
      mov rdi,1
      mov rsi,msg
      mov rdx,len_msg
      syscall
      mov rax,60
      mov rdi,0
      syscall
```

```
0:  b8 01 00 00 00
5:  bf 01 00 00 00
a:  48 be 00 00 00 00 00
11: 00 00 00
14: ba 1b 00 00 00
19: 0f 05
1b: b8 3c 00 00 00
20: bf 00 00 00 00
25: 0f 05
```

Slides of first half of the course are adapted from:
https://www.nand2tetris.org
Download s/w tools required for first half of the course from the following link:
https://drive.google.com/file/d/0B9c0BdDJz6XpZUh3X2dPR1o0MUE/view

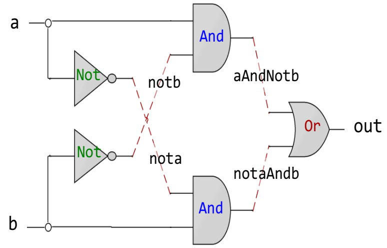
Instructor: Muhammad Arif Butt, Ph.D.

Today's Agenda

- Review of HDL for Combinational Circuits
- Designing a single bit Logic Unit
- Writing HDL for Combinational Arithmetic Circuits like
  - Half Adder
  - Full Adder
  - Full Subtractor
  - 16 bit Binary Adder (Add16 chip)
  - 16 bit Incrementer (Inc16 chip)
- Demo of above chips on H/W Simulator

Review of HDL for Combinational Circuits

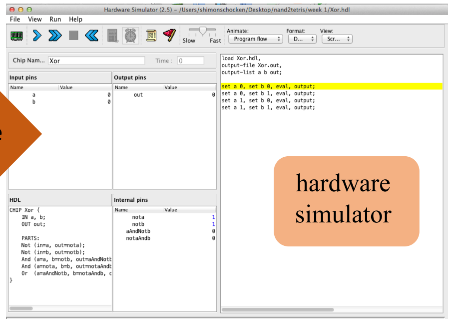
a	b	out
0	0	0
0	1	1
1	0	1
1	1	0



$$out(a,b) = a'b + ab'$$

```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not(in=a, out=nota);
  Not(in=b, out=notb);
  And(a=nota, b=b, out=w1);
  And(a=a, b=notb, out=w2);
  Or(a=w1, b=w2, out=out);
}
```

simulate



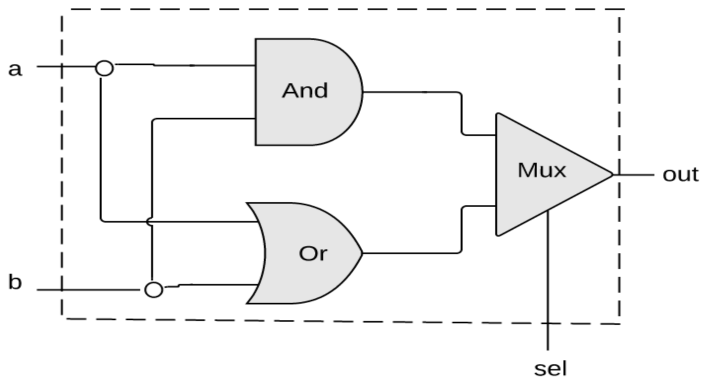
Elementary Logic Gates
- Not
- And
- Or
- Xor
- 2x4 Decoder
- 8x3 Encoder
- 2x1 Mux
- 4x1 Mux
- 1x2 Dmux
- 1x4 Dmux

Multi-Way Variants
- Or4way
- And4way
- Mux4way
- DMux4way

Multi-Bit Variants
- Not16
- And16
- Or16
- Mux16
- DMux16

Mixed Variants
- Or4way16
- And4way16
- Mux4way16
- Mux8way16

Arithmetic Logic Unit



a	b	sel	out
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

When sel==0
Operation = AND

When sel==1
Operation = OR

BitLU.hdl

```
CHIP BitALU  {
     IN a, b, sel;
     OUT out;
      PARTS:
        And(a=a, b=b, out=andOut);
        Or(a=a, b=b, out=orOut);
         Mux (a=andOut, b=orOut, sel=sel, out=out);
}
```

Single Bit Logic Unit Demo

Arithmetic Logic Unit

- To design a proper Arithmetic Logic Unit, we first need to design some combinational chips that can perform some basic arithmetic operations. Later we can integrate those chips to build the complete ALU
- Let us now design and code some chips that perform some basic arithmetic operations using the already created chips so far

- HalfAdder
- FullAdder
- Add16
- Inc16
- ALU

A family of combinational chips, from simple adders to an Arithmetic Logic Unit.

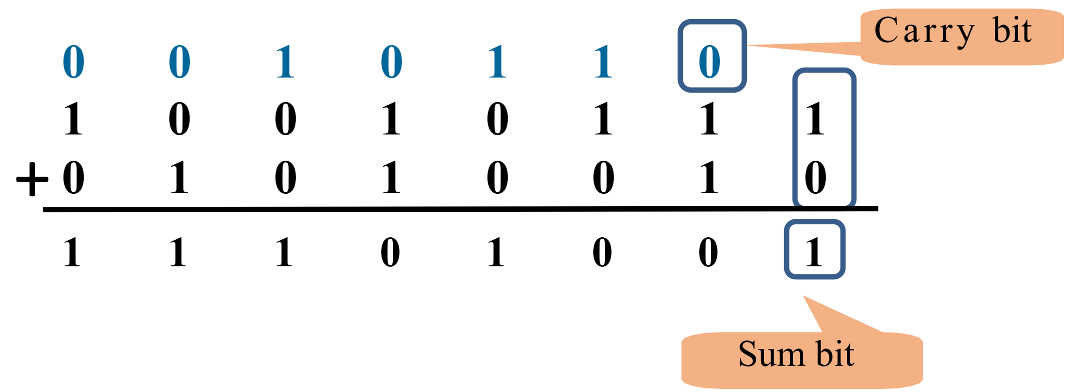
Boolean Arithmetic

- Addition — implement
- Subtraction — get for free
- Comparison (<,>,=) — get for free
- Multiplication — postpone to software
- Division — postpone to software

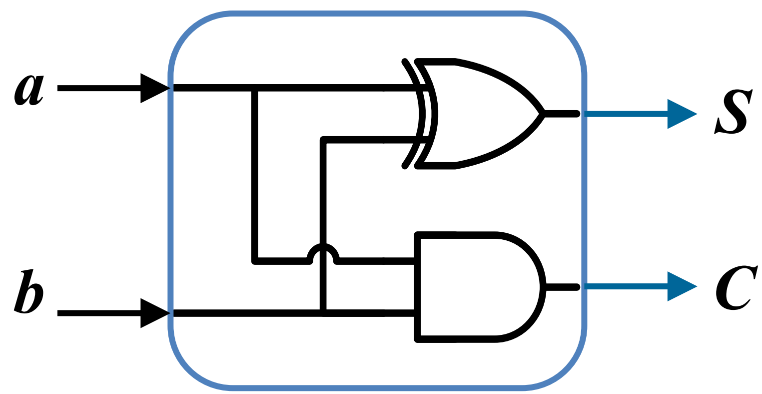
- Half adder: adds two bits
- Full adder: adds three bits
- Binary Adder: adds two integers
- Incrementer: adds one to an integer

Half Adder

- A half adder is a combinational circuit that accepts two input bits and generates two outputs (a sum bit and a carry bit)



a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



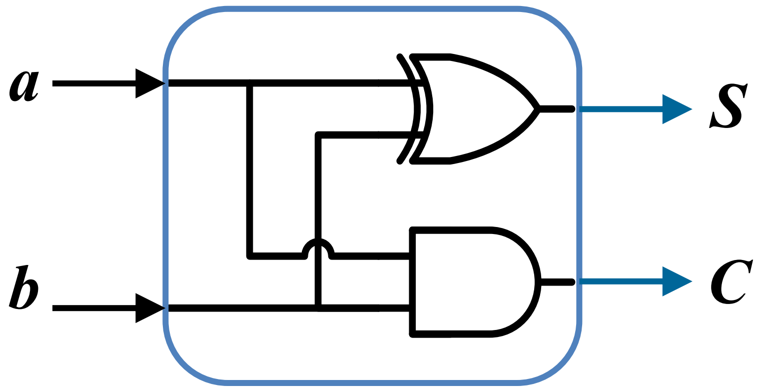
$sum(a,b) = a'b + ab' = a \oplus b$

$carry(a,b) = ab$

Half Adder Implementation

HalfAdder.hdl

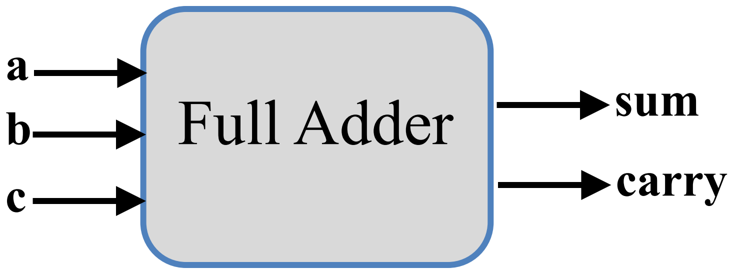
```
/**
 * Computes the sum of two bits.
 */
CHIP HalfAdder {
   IN a, b;    // 1-bit inputs
   OUT sum,    // Right bit of a + b
   carry; // Left bit of a + b
   PARTS:
     Xor(a=a, b=b, out=sum);
     And(a=a, b=b, out=carry);
}
```

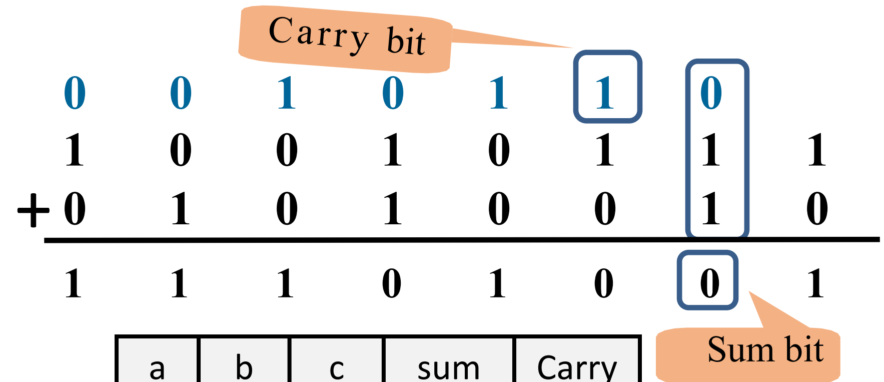


Half Adder Demo

Full Adder

- A half adder can add only two bits, it cannot accommodate the carry from the previous two bits addition. A full adder is a combinational circuit that performs the arithmetic sum of three input bits (augend, addend and carry-in) and generates two outputs a sum and a carry-out



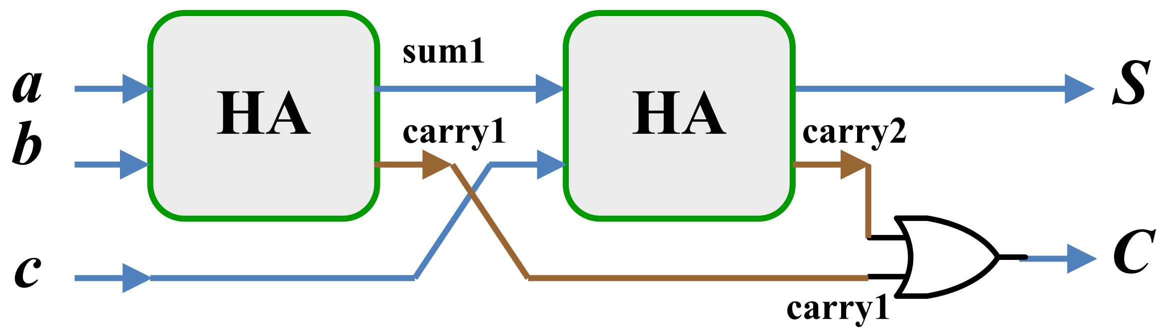


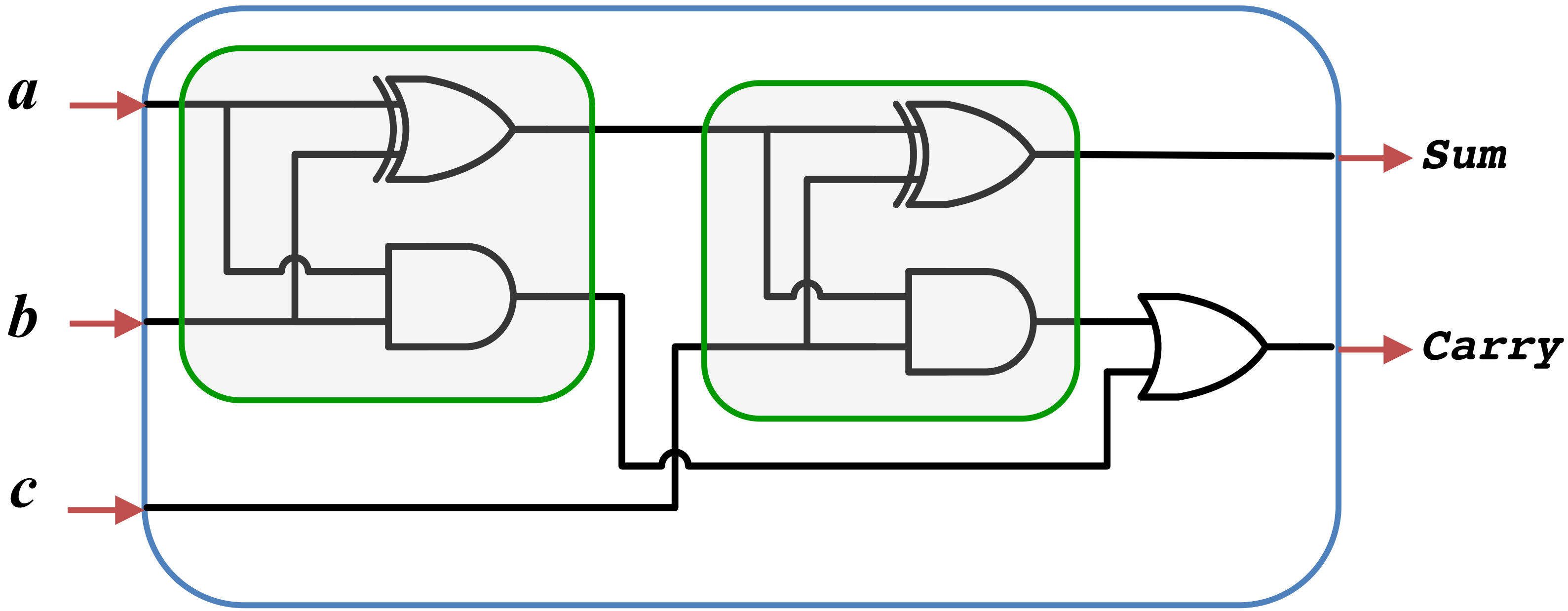
$$sum(a,b,c) = a'b'c + a'bc' + ab'c' + abc$$
$$= a \oplus b \oplus c$$

$$carry(a,b,c) = ab + bc + ac$$

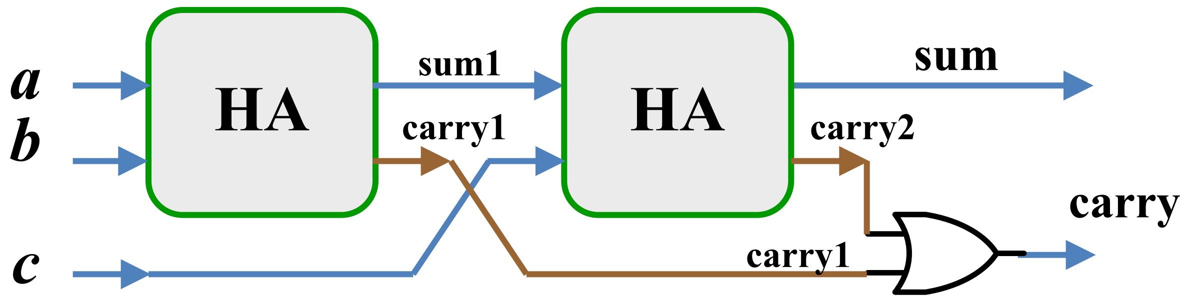
a	b	c	sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder Implementation





Full Adder Implementation (cont…)



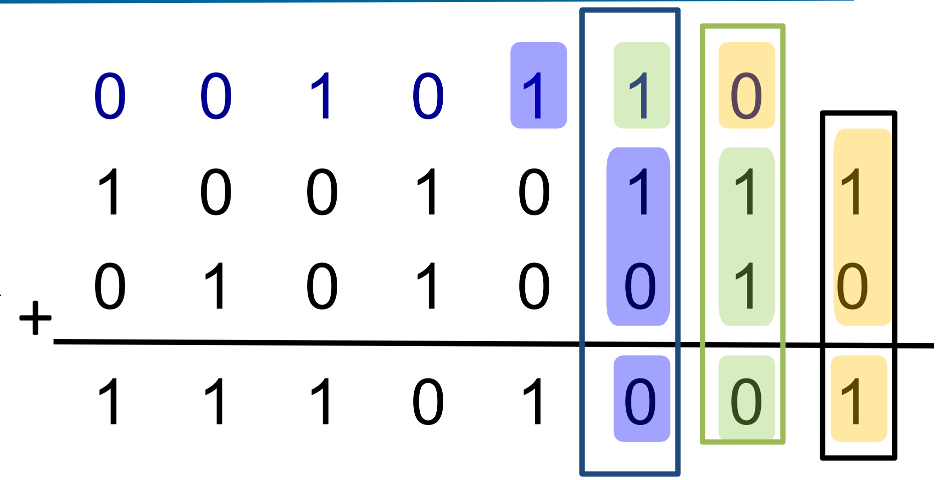
FullAdder.hdl

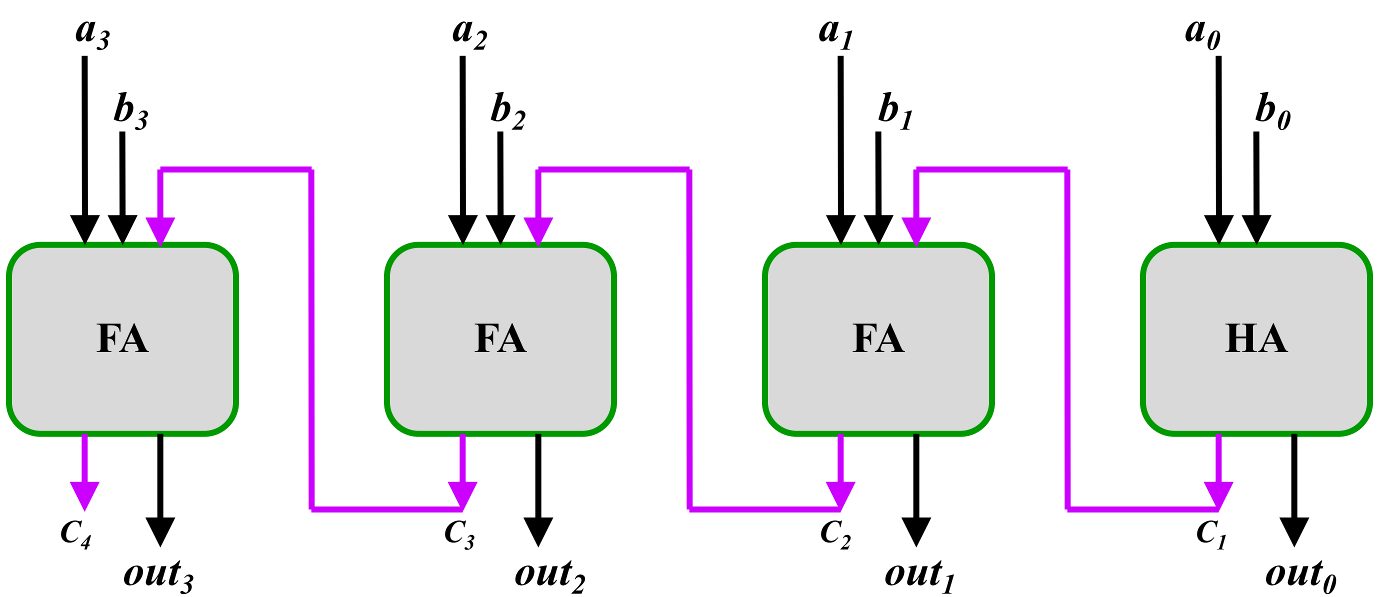
```
// Computes the sum of three bits
CHIP FullAdder {
   IN a, b, c; // 1-bit inputs
   OUT sum, carry;
   PARTS:
     HalfAdder(a=a, b=b, sum=sum1, carry=carry1);
     HalfAdder(a=sum1, b=c, sum=sum, carry=carry2);
     Or(a=carry1, b=carry2, out=carry);
}
```

Full Adder Demo

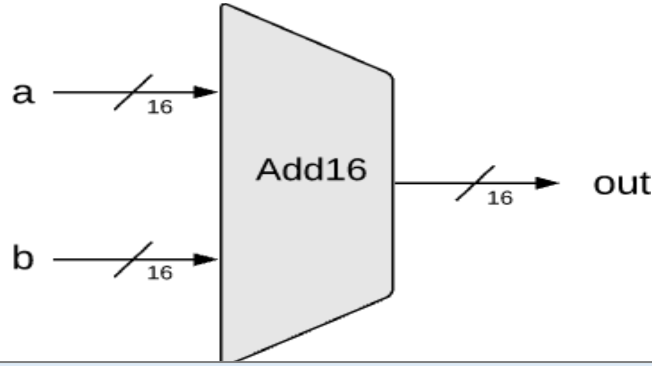
Binary Adder

- A digital circuit that produces the sum of two n bit binary numbers is called a n-bit binary adder. It can be designed with full adders connected in cascade. A 4-bit binary adder is shown below:





Binary Adder Implementation



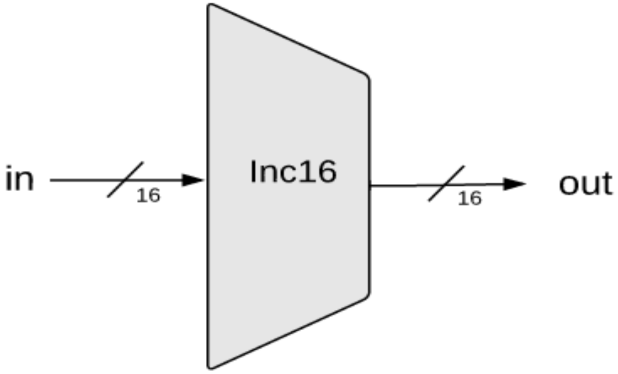
Add16.hdl

```
CHIP Add16 {
  IN a[16], b[16];
  OUT out[16];
  PARTS:
   HalfAdder(a=a[0], b=b[0], sum=out[0], carry=carry0);
   FullAdder(a=a[1], b=b[1], c=carry0, sum=out[1], carry=carry1);
   FullAdder(a=a[2], b=b[2], c=carry1, sum=out[2], carry=carry2);
   FullAdder(a=a[3], b=b[3], c=carry2, sum=out[3], carry=carry3);
   .........
   FullAdder(a=a[14], b=b[14], c=carry13, sum=out[14], carry=carry14);
   FullAdder(a=a[15], b=b[15], c=carry14, sum=out[15], carry=carry15);
}
```

Binary Adder Demo

16 Bit Incrementer Implementation

- A digital circuit that inputs a 16 bit integer and adds 1 to it, ignores the carryout from the MSb (if any)
- The single-bit 0 and 1 values are represented in HDL as `false` and `true`



Inc16.hdl

```
 /**
  * 16-bit incrementer:
  * out = in + 1 (arithmetic addition)
  */
CHIP Inc16 {
   IN in[16];
   OUT out[16];
   PARTS:
     Add16(a=in, b[0]=true, out=out);
}
```

Things To Do

- Perform interactive and script based testing of the chips designed in today's session on the h/w simulator. You can download the .hdl, .tst and .cmp files of above chips from the course bitbucket repository:

https://bitbucket.org/arifpucit/coal-repo/

- Interested students should try to design half subtractor, full subtractor and adder-subtractor chips. Also design a 16-bit binary subtractor chip that can subtract one 16 bit number from another 16 bit number

Coming to office hours does NOT mean you are academically week!