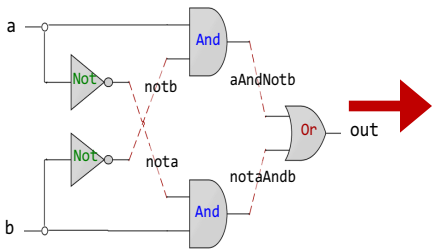
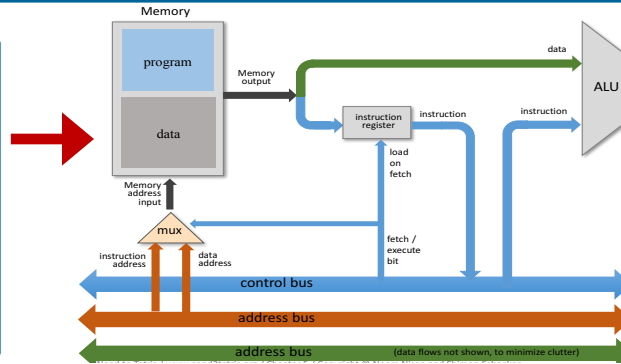




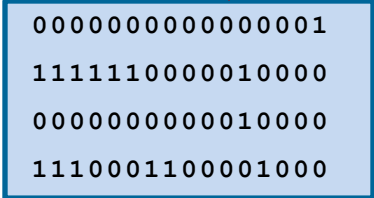
# Computer Organization & Assembly Language Programming



```
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
```

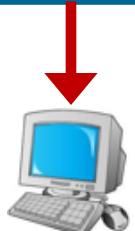
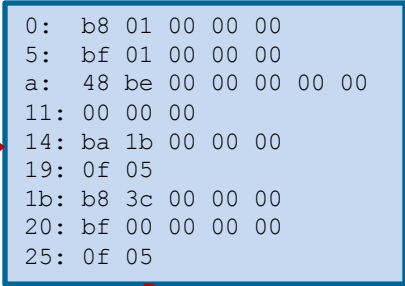


## Lecture # 02

# HDL for Combinational Circuits - 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
```



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>

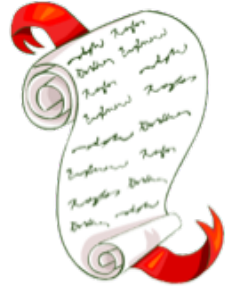
Instructor: Muhammad Arif Butt, Ph.D.



# Today's Agenda

---

- Review of Boolean Logic and Gates
- Hardware Description Language
  - SystemVerilog
  - VHDL
  - Noam / Shimon HDL
- Design and Code following gates/chips using universal NAND gate
  - AND
  - OR
  - NOT
- How to use Hardware Simulator?
- How to do Interactive Chip Testing in h/w Simulator?



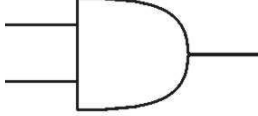
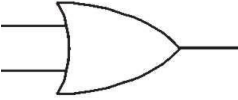
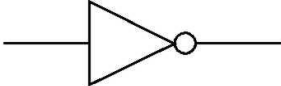
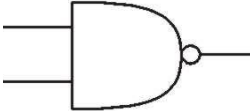
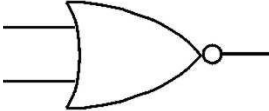



# Review

# Boolean Logic



# Elementary Boolean Operations

| Gate | Symbol   | Operator       |
|------|--|----------------|
| And  |    | $A \cdot B$    |
| Or   |    | $A + B$        |
| Not  |    | $A'$           |
| Nand |    | $(A \cdot B)'$ |
| Nor  |   | $(A + B)'$     |
| Xor  |  | $A \oplus B$   |



# Boolean Functions

---

- An expression formed by binary variables, logical operators, parenthesis and an equal to sign. The value of a Boolean function can either be zero or one

$$f(x, y) = x'y + xy'$$

$$f(x, y, z) = xy'z + xyz$$

$$f(w, x, y, z) = w'x'y'z + wxyz$$



# Boolean Identities

$$(xy) = (yx)$$

$$(x + y) = (y + x)$$

Commutative law

$$x(yz) = (xy)z$$

$$(x + (y + z)) = (x + y) + z$$

Associative law

$$x(y + z) = xy + xz$$

$$x + (yz) = (x + y) + (x + z)$$

Distributive law

$$(xy)' = (x' + y')$$

$$(x + y)' = (x' y')$$

De Morgan law



# Boolean Function



# Truth Table

$$f(x, y, z) = xy + x'z$$



| x | y | z | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



# Truth Table



# Boolean Function

| x | y | z | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f(x, y, z) = \sum(1, 3, 6, 7)$$



$$f(x, y, z) = x'y'z + x'yz + xyz' + xyz$$

$$f(x, y, z) = xy + x'z$$

You can simplify a Boolean function using Boolean Identities or Karnaugh Map methods





# Hardware Description Language



# Hardware Description Language

---

- **Hardware Description Language** is a language that describes the hardware of digital system in textual form
- There are two applications of HDL processing
  - **Hardware Simulation:** We let our HDL programs run inside a h/w simulator to simulate and debug the design. The h/w simulator interprets the HDL and produce readable o/p, that predicts how the h/w will behave before it is actually fabricated
  - **Hardware Synthesis:** The HDL programs can be compiled into h/w implementation using synthesizer and h/w compilation tools. The output of h/w synthesizer is gate level netlist, which is later used to fabricate an IC or to layout a Printed Circuit Board (PCB)
- There are a variety of HDLs available in the market. The most common are SystemVerilog (based on C) and VHDL (Very high speed integrated circuit Hardware Description Language) (based on Ada)
- In this course we will be using a simple/minimal HDL designed and developed by Noam and Shimon (Designers of the course nand2tetris)



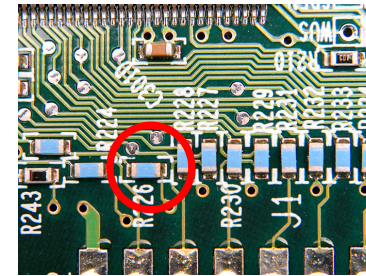
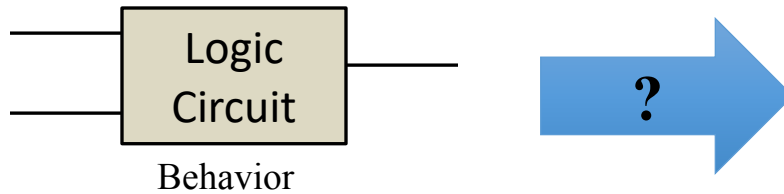
# Hardware Simulator

---

- HDL simulators are software packages that simulate expressions written in one of the hardware description languages, like VHDL, Verilog, SystemVerilog, and so on
- Hardware Simulator that we will be using is designed and developed by students of Interdisciplinary Center Herzliya Efi Arazi School of Computer Science
- It can be used to build and test many different hardware platforms. In this course, we will use it to design a complete computer, called Hack -- a 16-bit computer equipped with a screen and a keyboard
- To design and build this Hack computer we need to write hdl programs for elementary gates, combinational circuits, sequential circuits, registers, RAM, ALU, control unit and its data path. Every time, we write these hdl programs, we will test and debug them on this hardware simulator
- This is how h/w engineers build chips for real:
  - First the h/w is designed tested and optimized on a software simulator
  - Later the resulting gate logic is committed to silicon



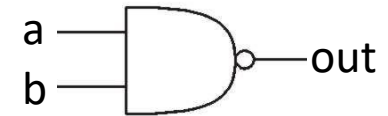
# Design Process



## Design Process:

- Use the built-in Nand gate chip having interface  $Nand(a=, b=, out=)$
- Design your logic circuit using this Nand gate only. OR. First design the And, Or and Not gates using this Nand gate and then use And, Or and Not gates to build the logic circuit as usual
- Write down the HDL program file specifying your logic circuit
- Test the chip in a hardware simulator
- Optimize the design
- Realize the optimized design in silicon

Nand.hdl



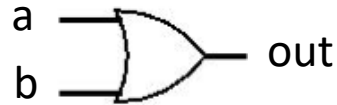
```
/* Nand gate: out = a Nand b */  
  
CHIP Nand {  
    IN a, b;  
    OUT out;  
  
    BUILTIN Nand;  
}
```

Chip Interface

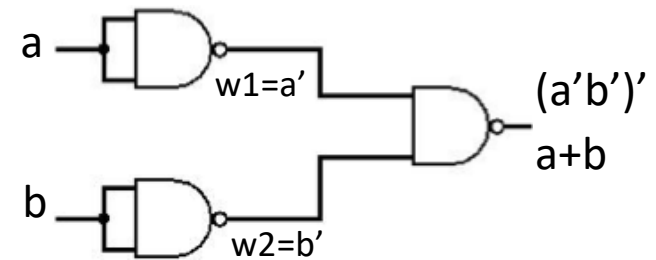
Chip Implementation



# Design of Or Gate Chip



| a | b | out |
|---|---|-----|
| 0 | 0 | 0   |
| 0 | 1 | 1   |
| 1 | 0 | 1   |
| 1 | 1 | 1   |



Or.hdl

```
/** Or gate: out = a or b */
CHIP Or {
  IN a, b;
  OUT out;

  PARTS:
    Nand(a=a, b=a, out=w1); // (a)'
    Nand(a=b, b=b, out=w2); // (b)'
    Nand(a=w1, b=w2, out=out); // (a'b')'
}
```

```
CHIP Nand {
  IN a, b;
  OUT out;

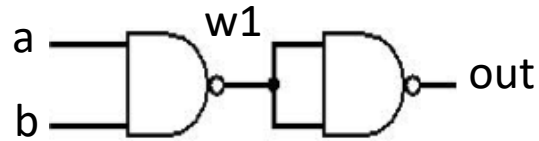
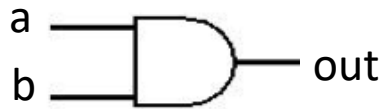
  BUILTIN Nand;
}
```

chip  
interface

chip  
implementation



# Design of And Gate Chip



| a | b | out |
|---|---|-----|
| 0 | 0 | 0   |
| 0 | 1 | 0   |
| 1 | 0 | 0   |
| 1 | 1 | 1   |

And.hdl

chip  
interface

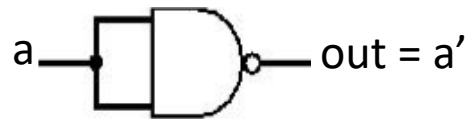
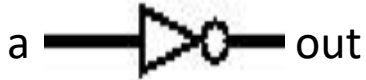
chip  
implementation

```
/** And gate: out = a And b */  
CHIP And {  
  IN a, b;  
  OUT out;  
  
  PARTS:  
    Nand(a=a, b=b, out=w1); // (ab)'  
    Nand(a=w1, b=w1, out=out); // ab  
}
```

```
CHIP Nand {  
  IN a, b;  
  OUT out;  
  
  BUILTIN Nand;  
}
```



# Design of Not Gate Chip



| a | out |
|---|-----|
| 0 | 1   |
| 1 | 0   |

## Not.hdl

chip  
interface

chip  
implementation

```
/** Not gate: out = a' */  
CHIP Not {  
  IN in;  
  OUT out;  
  
  PARTS:  
    Nand(a=in, b=in, out=out); //a'  
  
}
```

```
CHIP Nand {  
  IN a, b;  
  OUT out;  
  
  BUILTIN Nand;  
}
```



# **Interactive Chip Testing on Hardware Simulator**





# How to Download the H/W Simulator?

- Type the following URL in your browser:  
<https://bitbucket.org/arifpucit/>
- In the public repositories pain, click the **coal-repo** repository, containing all the source codes as well as the software tools used in this course
- In the left pane, click **Downloads** to download the entire repository on your system. Now on your system just check the contents of **tools** directory that you have just downloaded

```
Arif-MacBook:arifpucit-coal-repo/tools$ ls
HardwareSimulator.sh      HardwareSimulator.bat
CPUEmulator.sh           CPUEmulator.bat
Assembler.sh             Assembler.bat
VMEulator.sh             VMEulator.bat
JackCompiler.sh          JackCompiler.bat
TextComparer.sh          TextComparer.bat
builtInChips      builtInVMCode  bin      OS
```



# Starting the H/W Simulator

- Follow the following steps to start the h/w simulator on UNIX/Mac OS:
  - Open the terminal
  - Go to tools directory
  - Set execute permissions of the file `HardwareSimulator.sh`
  - Execute it

```
tools — -bash — 77x18
(base) Arifs-MacBook-Pro:tools arif$ ls
Assembler.bat           JackCompiler.bat       VMEulator.sh
Assembler.sh           JackCompiler.sh       bin
CPUEmulator.bat       OS                     builtInChips
CPUEmulator.sh       TextComparer.bat      builtInVMCode
HardwareSimulator.bat  TextComparer.sh
HardwareSimulator.sh   VMEulator.bat
(base) Arifs-MacBook-Pro:tools arif$ chmod +x HardwareSimulator.sh
(base) Arifs-MacBook-Pro:tools arif$ ./HardwareSimulator.sh
```



# Interactive Chip Testing Demo

---





# Java Based H/W Simulator

Hardware Simulator (2.5)

File View Run Help

Chip Nam... Time : 0

| Input pins |       | Output pins |       |
|------------|-------|-------------|-------|
| Name       | Value | Name        | Value |
|            |       |             |       |

HDL

Animate: Program flow Format: D... View: Scr...

Slow Fast



# Loading a Chip in the H/W Simulator

Hardware Simulator (2.5)

File View Run Help

Chip Nam... Time : 0

Input pins

| Name | Value |
|------|-------|
|------|-------|

File: Or.hdl

02

| Name    | Date Modified                    |
|---------|----------------------------------|
| And.cmp | Monday, 16 October 2017, 2:28 pm |
| And.hdl | Friday, 15 May 2020, 10:09 pm    |
| And.tst | Monday, 16 October 2017, 2:28 pm |
| Not.cmp | Monday, 16 October 2017, 2:28 pm |
| Not.hdl | Monday, 16 October 2017, 2:28 pm |
| Not.tst | Monday, 16 October 2017, 2:28 pm |
| Or.cmp  | Monday, 16 October 2017, 2:28 pm |
| Or.hdl  | Monday, 16 October 2017, 2:28 pm |
| Or.tst  | Monday, 16 October 2017, 2:28 pm |

File Format: HDL Files

New Folder Cancel Load Chip

Navigate to a directory and select an .hdl file.



# Exploring the GUI of the H/W Simulator

Hardware Simulator (2.5) - /Users/arif/Documents/01 Arif-CS223-COAL/Lecture Slides (Video Sessions)/0 Lecture Codes/02/Or.hdl

File View Run Help

Animate: Program flow Format: D... View: Scr...

Chip Nam... Or Time : 0

| Input pins |       | Output pins |       |
|------------|-------|-------------|-------|
| Name       | Value | Name        | Value |
| a          | 0     | out         | 0     |
| b          | 0     |             |       |

- Names and current values of the chip's input pins;
- To change their values, enter the new values here.

| Internal pins |       |
|---------------|-------|
| Name          | Value |
| w1            | 1     |
| w2            | 1     |

- Names and current values of the chip's internal pins (used to connect the chip's parts, forming the chip's logic);
- Calculated by the simulator; read-only.

```
// File name: 02/Or.hdl
/**
 * Or gate:
 * out = 1 if (a == 1 or b == 1)
 * 0 otherwise
 */
CHIP Or {
  IN a, b;
  OUT out;
  PARTS:
    Nand(a=a, b=true, out=w1);
    Nand(a=b, b=b, out=w2);
    Nand(a=w1, b=w2, out=out);
}
```

- Read-only view of the loaded **.hdl** file;
- Defines the chip logic;
- To edit it, use an external text editor.



# Exploring The Chip Logic

Hardware Simulator (2.5) - /Users/arif/Documents/01 Arif-CS223-COAL/Lecture Slides (Video Sessions)/0 Lecture Codes/02/Or.hdl

File View Run Help

Animate: Program flow Format: D... View: Scr...

Chip Nam... Or Time : 0

| Input pins |       | Output pins |       |
|------------|-------|-------------|-------|
| Name       | Value | Name        | Value |
| a          | 0     | out         | 1     |
| b          | 1     |             |       |

1. Click any one of the chip **PARTS**

```
HDL
// File name: 02...
/**
 * Or gate:
 * out = 1 if (a or b == 1)
 * 0 otherw...
 */
CHIP Or {
  IN a, b;
  OUT out;
  PARTS:
  Nand(a=a, b=true, out=w1);
  Nand(a=b, b=b, out=w2);
  Nand(a=w1, b=w2, out=out);
}
```

| Part pins |          |       | Nand |
|-----------|----------|-------|------|
| Part pin  | Gate pin | Value |      |
| a         | a        | 0     |      |
| b         | true     | 1     |      |
| out       | w1       | 1     |      |

2. A table pops up, showing the input/output pins of the selected part (actually, its API), and their current values;

A convenient debugging tool.



# Interactive Chip Testing

Hardware Simulator (2.5) - /Users/arif/Documents/01 Arif-CS223-COAL/Lecture Slides (Video Sessions)/0 Lecture Codes/02/Or.hdl

File View Run Help

Calculator button circled in red

| Input pins |       | Output pins |       |
|------------|-------|-------------|-------|
| Name       | Value | Name        | Value |
| a          | 0     | out         | 0     |
| b          | 1     |             |       |

| Internal pins |       |
|---------------|-------|
| Name          | Value |
| w1            | 1     |
| w2            | 1     |

```
// File name: 02/Or.hdl
/**
 * Or gate:
 * out = 1 if (a == 1 or b == 1)
 *     0 otherwise
 */
CHIP Or {
  IN a, b;
  OUT out;
  PARTS:
    Nand(a=a, b=true, out=w1);
    Nand(a=b, b=b, out=w2);
    Nand(a=w1, b=w2, out=out);
}
```

1. User: changes the values of some input pins

2. Simulator: responds by:

- Darkening the output and internal pins, to indicate that the displayed values are no longer valid
- Enabling the *eval* (calculator-shaped) button.





# Interactive Chip Testing (cont...)

Hardware Simulator (2.5) - /Users/arif/Documents/01 Arif-CS223-COAL/Lecture Slides (Video Sessions)/0 Lecture Codes/02/Or.hdl

File View Run Help

Chip Nam... Or Time : 0

| Input pins |       | Output pins |       |
|------------|-------|-------------|-------|
| Name       | Value | Name        | Value |
| a          | 1     | out         | 1     |
| b          | 1     |             |       |

Re-calc

```
HDL
// File name: 02/Or.hdl
/**
 * Or gate:
 * out = 1 if (a == 1 or b == 1)
 *     0 otherwise
 */
CHIP Or {
  IN a, b;
  OUT out;
  PARTS:
    Nand(a=a, b=true, out=w1);
    Nand(a=b, b=b, out=w2);
    Nand(a=w1, b=w2, out=out);
}
```

| Internal pins |       |
|---------------|-------|
| Name          | Value |
| w1            | 0     |
| w2            | 0     |

1. User: changes the values of some input pins

2. Simulator: responds by:

- Dimming the output and internal pins, to indicate that the displayed values are no longer valid
- Enabling the *eval* (calculator-shaped) button.

3. User: Clicked the *eval* button

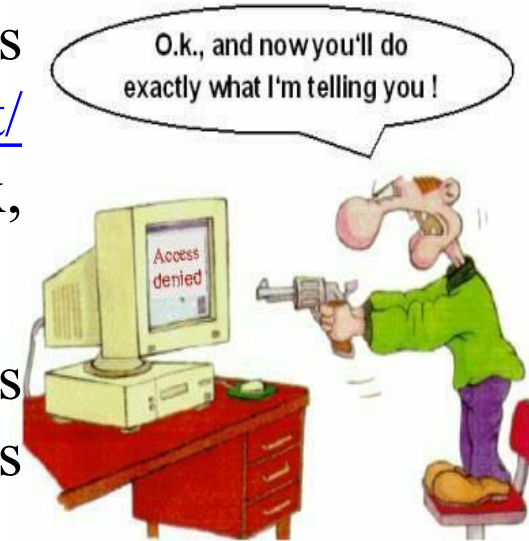
4. Simulator: re-calculates the values of the chip's internal and output pins (i.e. applies the chip logic to the new input values)

5. To continue interactive testing, enter new values into the input pins and click the *eval* button.



# Things To Do

- Download H/W simulator along with other tools and programs from <https://bitbucket.org/arifpucit/> and run it on your system (Mac, Linux, Windows)
- Perform interactive chip testing of the chips designed from built-in NAND gate in today's session
- Practice writing HDL for some basic logic circuits using the (AND, OR, NOT) chips that we have designed today, and perform interactive chip testing of these newly designed chips
- Explore the GUI of the h/w simulator



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