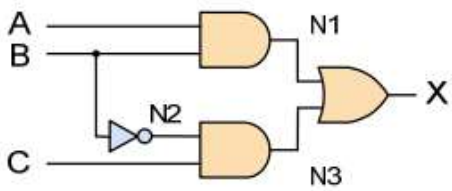
Introduction
VHDL Programming Language

This section provides a brief overview of the programming language and compilers.




```
architecture simple of example is begin
   Y <= (A and B) or (not B and C);
end simple;

architecture simple of example is begin
   Y <= (A and B) or (not B and C) after 3ns;
end simple;

architecture gates of example is
  signal N1, N2, N3 : std_logic;
begin
  N1 <= (A and B) after 2ns;
  N2 <= not B after 1ns;
  N3 <= (N2 and C) after 2ns;
  X  <= (N1 or N3) after 3ns;
end gates;
```

Introduction

Hardware Description Lenguaje

- A hardware description language is not a program, but a **code**, which describes the behavior of a digital circuit.
- **VHDL** stands for Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (HDL)
- It is used to build digital system/circuit using Programmable Logic Device like CPLD (Complex Programmable Logic Device) or FPGA (Field Programmable Gate Array)
- VHDL program (code) is used to implement digital circuit inside CPLD / FPGA, or it can be used to fabricate ASIC (Application Specific Integrated Circuit)
- In 1980s US Department of Defense (DoD) initiate the VHSIC program to standardize HDL from different companies. This standardized the HDL.
- In 1985 the first version of VHDL was created by IBM, Texas Instruments and Intermatrix under contract of DoD.
- In 1987 the IEEE 1076 standard was published. Then standard **IEEE 1164**, was added to introduce multi-valued logic system.
- Three common HDLs are Verilog (IEEE 1364), VHDL and SystemC.

Introduction

Advantages of VHDL

- VHDL is vendor independent, portable (a simple component can be exported) and reusable.
- It is designed to work modularly based on “blocks”. Thereby a complex big system can be traced down to small components.
- All statements in VHDL are executed concurrently (unless otherwise is desired by the programmer)
- It has IEEE and ANSI standard.
- Verilog and VHDL are similar. Both are useful as HDL tool. However:
  - VHDL is strongly typed. This makes it harder to make mistakes as a beginner because the compiler will not allow you to write code that is in valid. Verilog is weakly typed. It allows you to write code that is wrong, but more concise.
  - Verilog looks closer to a software language like C. This makes it easier for someone who knows C well to read and understand what Verilog is doing.
  - VHDL requires a lot of typing. Verilog generally requires less code to do the same thing.
  - VHDL is very deterministic, where as Verilog is non-deterministic under certain circumstances.

Introduction

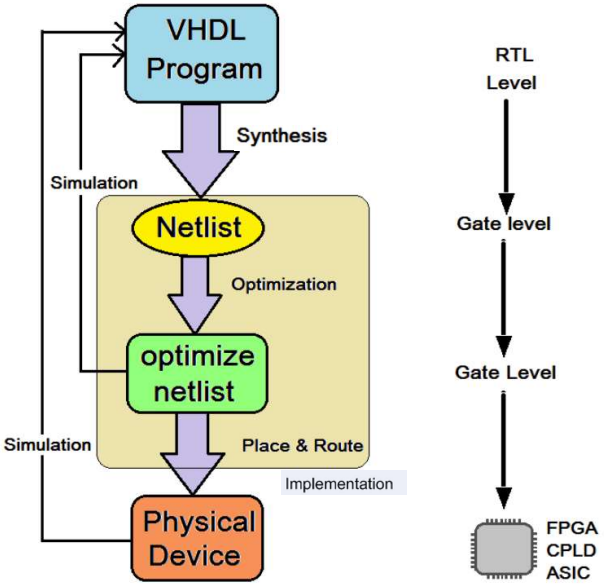
VHDL vs Verilog…There is any better?->NO

VHDL	Verilog
Strongly typed	Weakly typed
Easier to understand	Less code to write
More natural in use	More of a hardware modeling language
Wordy	Succinct
Non-C-like syntax	Similarities to the C language
Variables must be described by data type	A lower level of programming constructs
Widely used for FPGAs and military	A better grasp on hardware modeling
More difficult to learn	Simpler to learn

Finally, the decision of which is better depends on what suits better for you.

More about this:
Here, Here and Here

Introduction
Design Flow



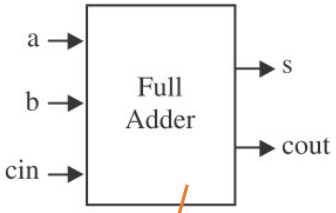
1.- First we write our VHDL code on a .vhd file. This code describe the circuit at Register Transfer Level (RTL). (abstraction level for creating circuits).
2.- Then synthesis is executed. Here VHDL code is transformed into a netlist at gate level.
3.- A netlist optimization process is executed to reduce delay signals and/or space required by the circuit. After that simulation can be done:

- Simulation can be including time delays or not (to test the logic).
- If simulation is not as expected, VHDL code must be modified.

4.- Finally the designed circuit is either: i) “printed” in a programmable device such as FPGA/CPLD or ii) transformed into a MASK to create an ASIC (Application Specific Integrated Circuit) by *placing and routing the software* (fitter). At this stage, the final device can be simulated and verified again.

Electronic Design Automation (**EDA**): Are the sofwares provided by companies to do simulation and synthesis. Like Quartus from Altera and Vitis/Vivado from Xilinx.

Introduction
How it works



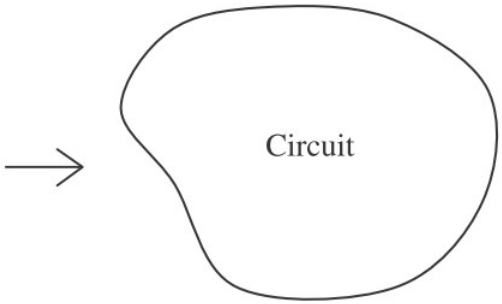
a b cin	s cout
0 0 0	0 0
0 1 0	1 0
1 0 0	1 0
1 1 0	0 1
0 0 1	1 0
0 1 1	0 1
1 0 1	0 1
1 1 1	1 1

s and count are computed as: $\text{cout} = \text{a.b} + \text{a.cin} + \text{b.cin}.$

$\text{s} = \text{a} \oplus \text{b} \oplus \text{cin}$

We generate an "**ENTITY**"(circuit structure) which has 3 input **ports** (a,b,cin) and 2 output ports (s, count). Besides that, its **architecture** (circuit functioning) describes how to assign values to s and count.

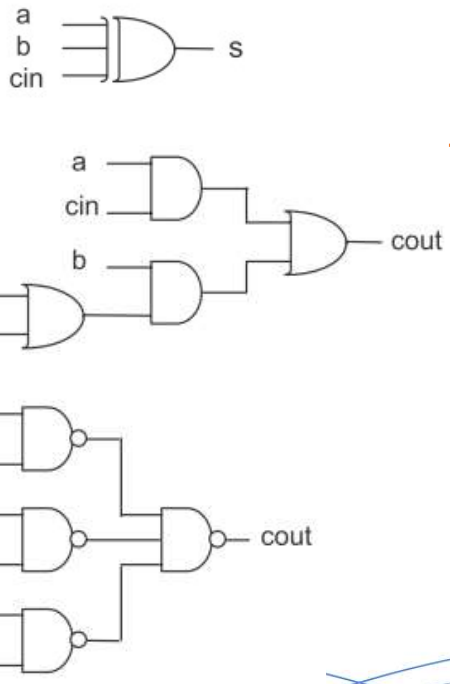
```
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
      s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
   s <= a XOR b XOR cin;
   cout <= (a AND b) OR (a AND cin) OR
           (b AND cin);
END dataflow;
```



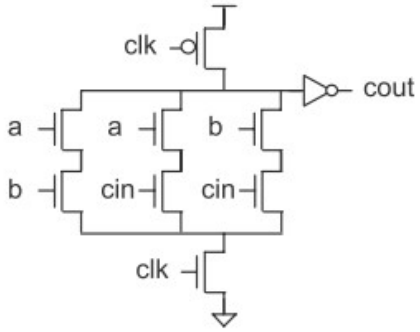
Introduction

Several possible implementations

Equtions of ARCHITECTURE described before can be implemented in several ways. The results depends on the **compiler/optimizer** and more important, the **target technology.**

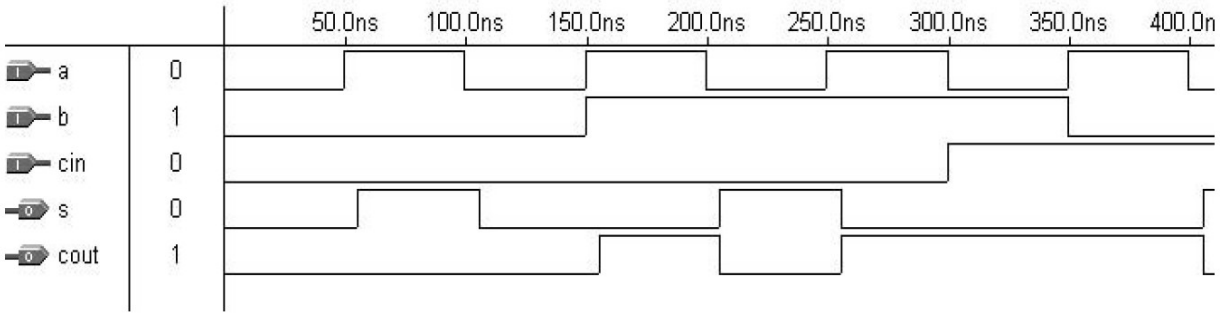


These are two possible implementations if the target technology is a FPGA or CPLD.



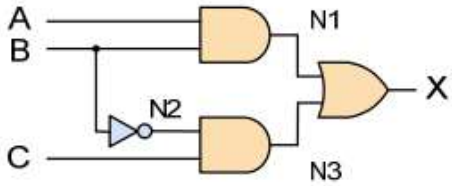
This shows a possible implementation in CMOS for an ASIC at transistor level.

Simulation results from VHDL design: Where is the error? Check it!



END-Introduction
VHDL Programming Language

This section provides a brief overview of the programming language and compilers.




```
architecture simple of example is begin
   Y <= (A and B) or (not B and C);
end simple;

architecture simple of example is begin
   Y <= (A and B) or (not B and C) after 3ns;
end simple;

architecture gates of example is
  signal N1, N2, N3 : std_logic;
begin
  N1 <= (A and B) after 2ns;
  N2 <= not B after 1ns;
  N3 <= (N2 and C) after 2ns;
  X  <= (N1 or N3) after 3ns;
end gates;
```