Introduction to VHDL

Announcements

- Homework #1 due today
- Homework #2 is posted
- Reading Assignment
  - Ch. 1, Ch. 2 sections 1-5
- Free homework grade for attending critical thinking lecture

Join us for the 2019 Fram Signature Lecture!

"POWERFUL STUFF: An Entrepreneurial Mindset Built Upon Critical Thinking" with Doug Melton of KEEN

Date:	Tuesday, September 17, 2019
Time:	3:30pm – 4:45pm
Place:	Ingle Auditorium *(Reception immediately following in Fireside Lounge)*

*Access Services will be providing interpreters for this event

Hardware Description Languages

- Hardware description language (HDL) is similar to a computer program except that it is used to describe hardware.
- Common HDLs
  - VHDL (VHSIC Hardware Description Language)
  - Verilog
  - *Both languages are easy to learn and hard to master. Once you have learned on of these languages, you will have no trouble transitioning to the other.*

VHDL History

- Developed in the mid-1980's – initiative from Department of Defense
    - Originally used for simulation and modeling
    - Synthesis tools were later developed to create logic circuits directly from VHDL behavioral descriptions

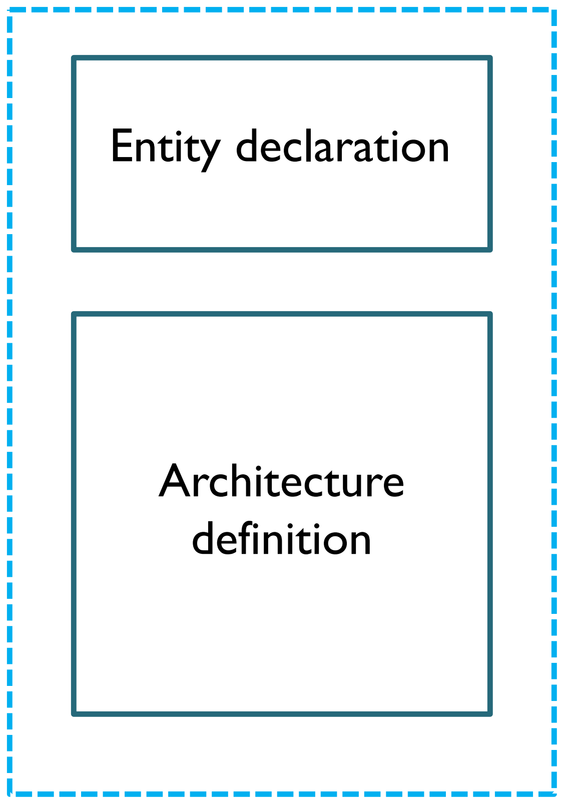
Pause for definitions

- Simulation
  - A testing procedure to ensure that the circuit you have designed functions correctly.
    - Force 1's and 0's into the circuit and observe the outputs for correctness
    - Can be done with waveforms or written in VHDL
- Synthesis
  - Translation of a source code into a hardware structure that implements the intended functionality
    - Simply put: turning a written description into underlying gates

VHDL Module Structure

- A VHDL module consists of an <u>Entity</u> and an <u>Architecture</u>
  - The entity describes the **ports** of the module
    - The direction of all I/O signals
    - The types of all I/O signals
  - The architecture describes the **functionality** of the module
    - The internal structure of the module
    - The behavior of the module

VHDL Module Structure



Text file (mydesign.vhd)

Example of VHDL Module

- 3 sections
- File extension is .vhd
- Name of file is usually the entity name

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
    PORT( a          : IN STD_LOGIC;
          b          : IN STD_LOGIC;
          z          : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE model OF nand_gate IS
BEGIN
          z <= a NAND b;
END model;
```

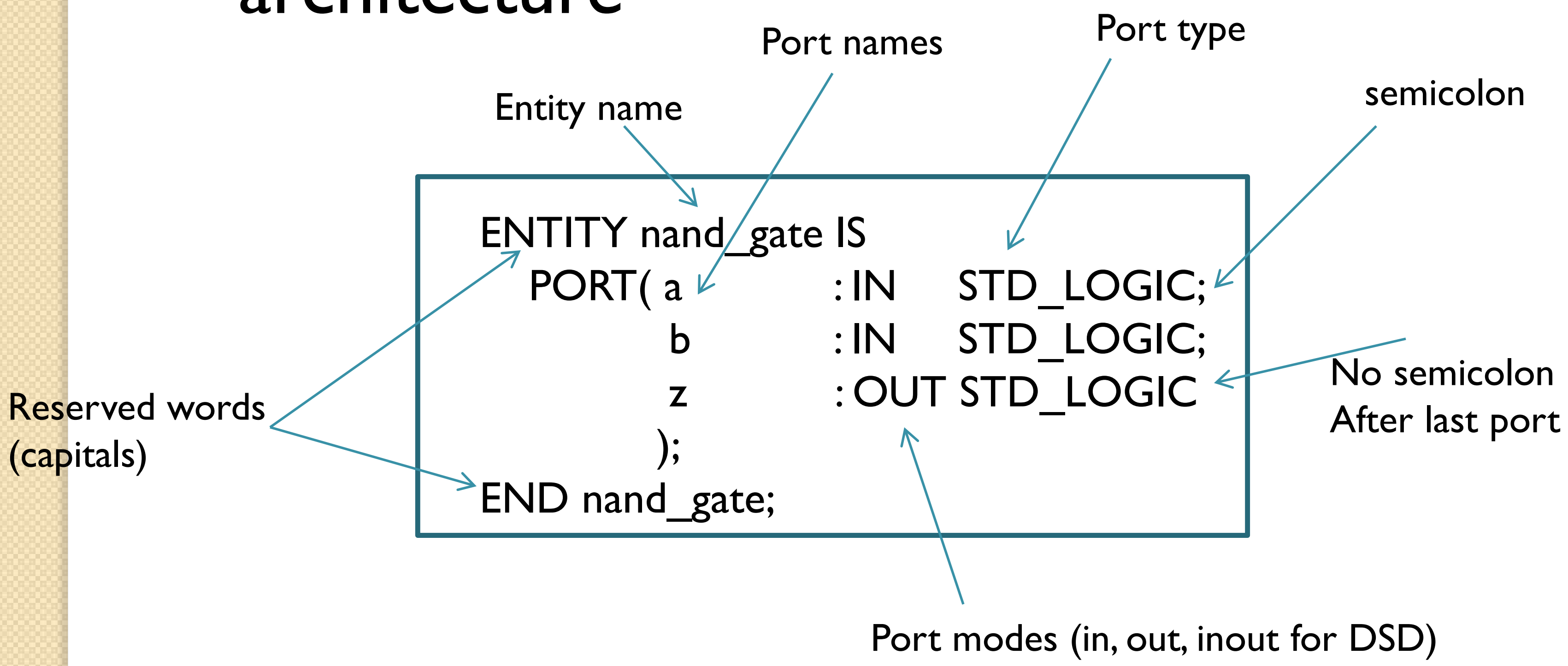
Library Declaration

Entity

Architecture

Entity Declaration

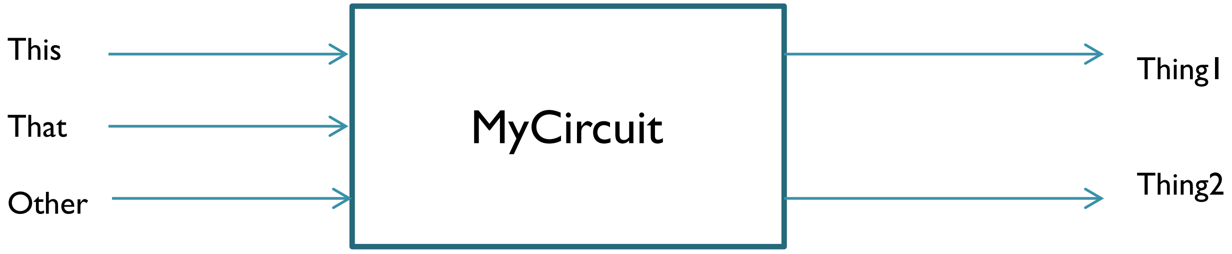
- Describes the interface of the component
- Thought of as a wrapper that exposes only external interfaces for the architecture



```
ENTITY nand_gate IS
    PORT( a          : IN    STD_LOGIC;
          b          : IN    STD_LOGIC;
          z          : OUT STD_LOGIC
          );
END nand_gate;
```

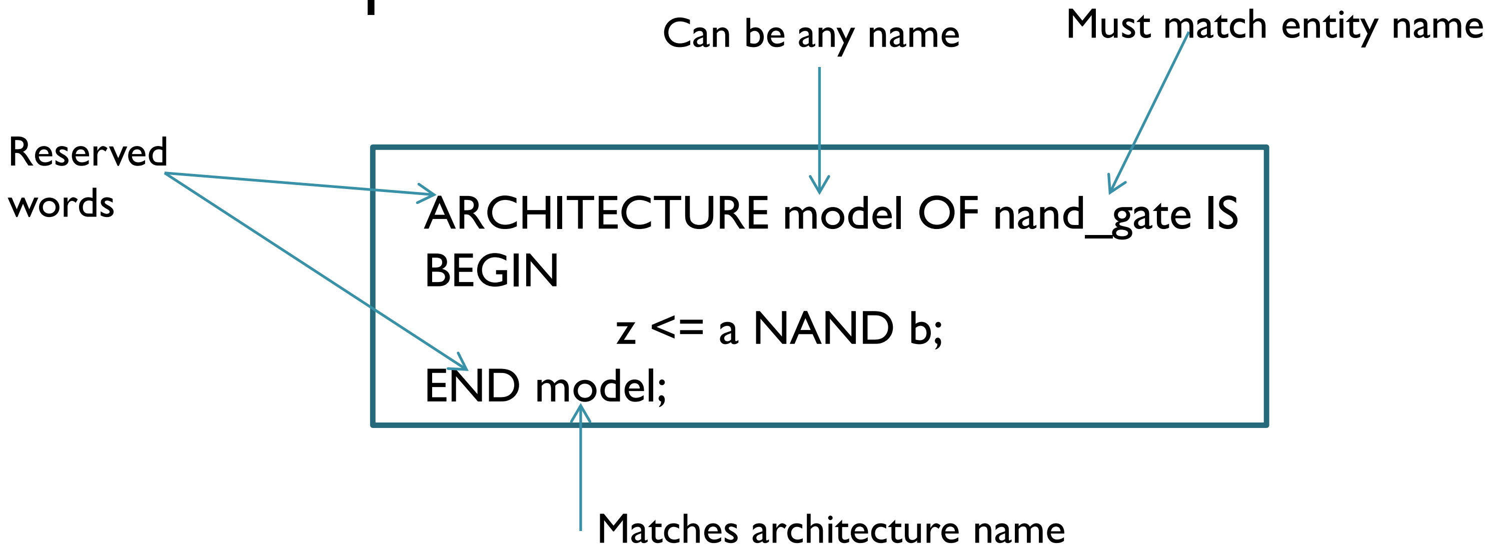
Entity Declaration

- Write the entity declaration for the following 'black box'



Architecture Definition

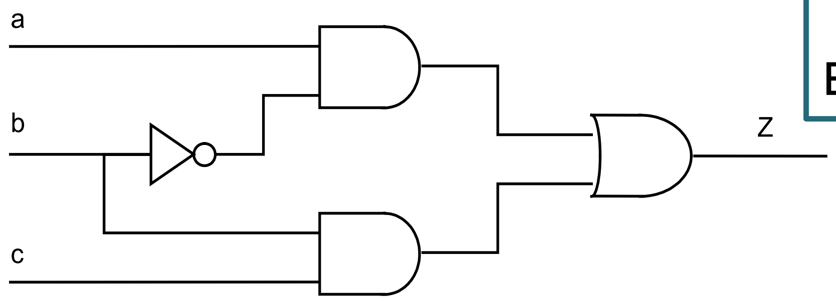
- Architecture describes functionality of the module
  - Structural – describes the system in terms of how it is built
  - Behavioral – describes the system in terms of how it operates



```
ARCHITECTURE model OF nand_gate IS
BEGIN
            z <= a NAND b;
END model;
```

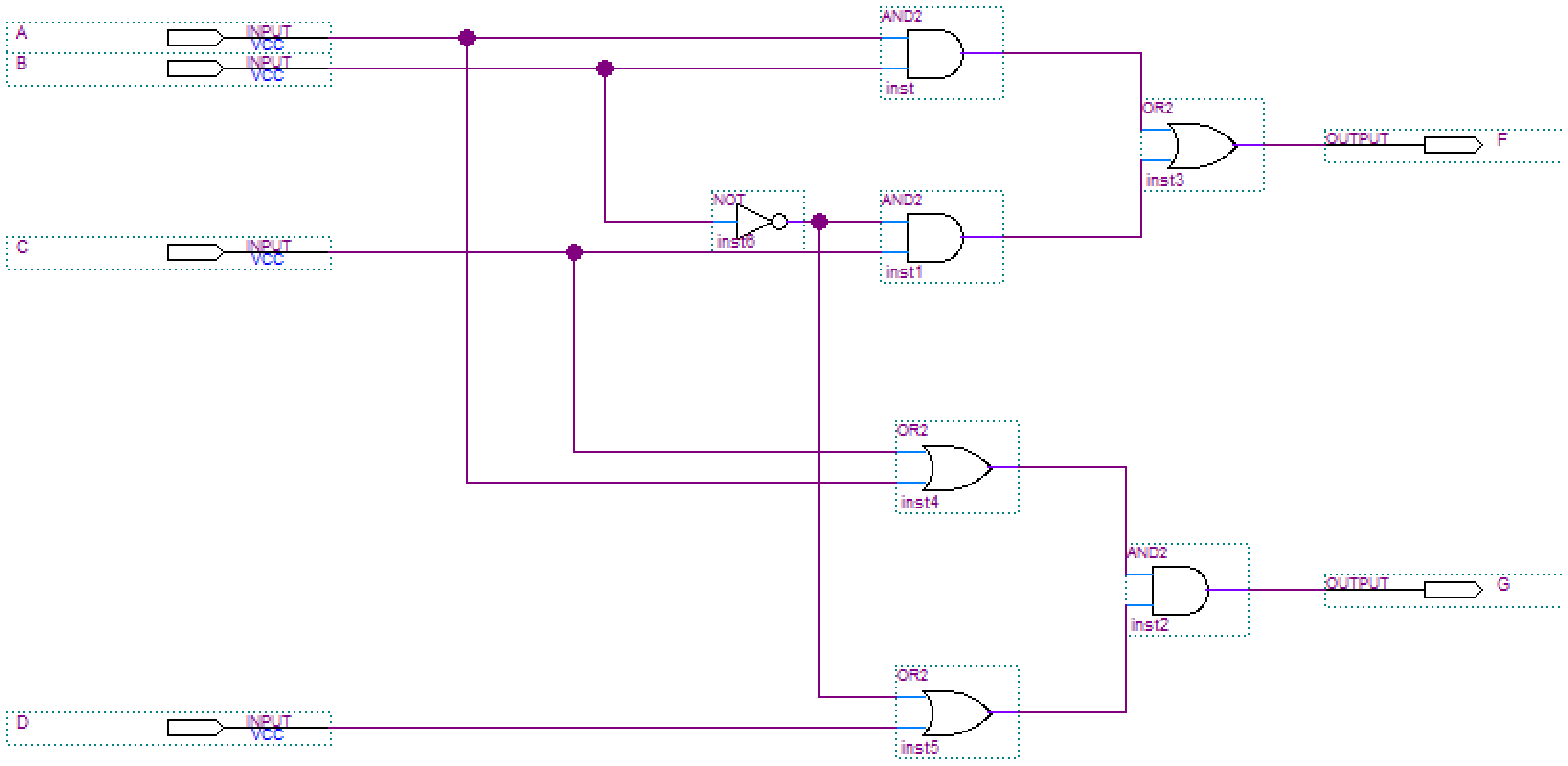
Architecture Definition

- Using AND, NOT and OR write the architecture for the circuit below

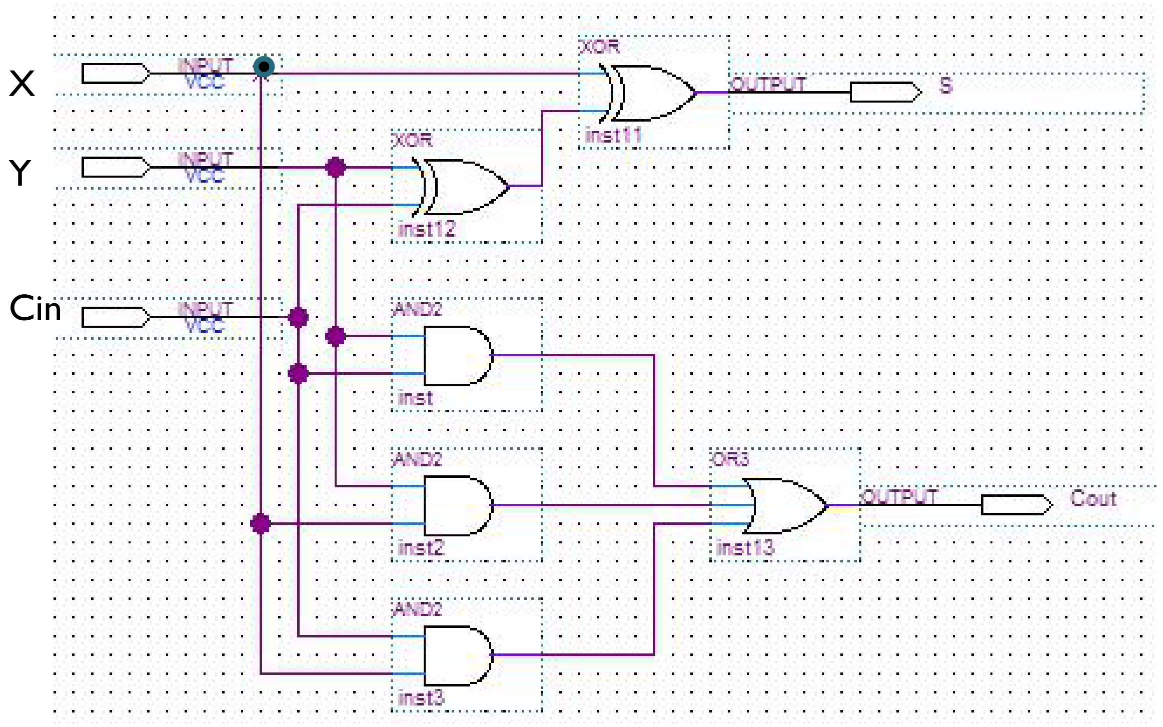


```
ENTITY example IS
    PORT( a,b,c        : IN     STD_LOGIC;
               z              : OUT STD_LOGIC);
END example;
```

Write the VHDL for the following



Now Consider This



```
ENTITY fulladd IS
    PORT( Cin, X,Y  :  IN STD_LOGIC;
              S, Cout    :  OUT STD_LOGIC);
   END fulladd;

ARCHITECTURE structural OF fulladd IS
    BEGIN
              S <= X XOR ( Y XOR Cin);
              Cout <= (X AND Y) OR (X AND Cin) OR (Y AND Cin);
    END structural;
```

VHDL for full adder

- Problem with previous example is that equations get long
- A better way to write it is to use signals
- Signals are declared after the ARCHITECTURE line but before the BEGIN
- Signals represent <u>internal</u> wires and busses in the design

Full Adder Arch. with signals

Entity is the same

```
ENTITY fulladd IS
    PORT( Cin, X,Y  :  IN STD_LOGIC;
               S, Cout    :  OUT STD_LOGIC);
    END fulladd;

ARCHITECTURE structural OF fulladd IS
     SIGNAL xor1, and1, and2, and3 : STD_LOGIC;
     BEGIN
          xor1  <= Y XOR Cin;
          and1  <= Y AND Cin;
          and2  <= Y AND X;
          and3  <= X AND Cin;
          S <= X XOR xor1;
          Cout <= and1 OR and2 OR and3;
     END structural;
```