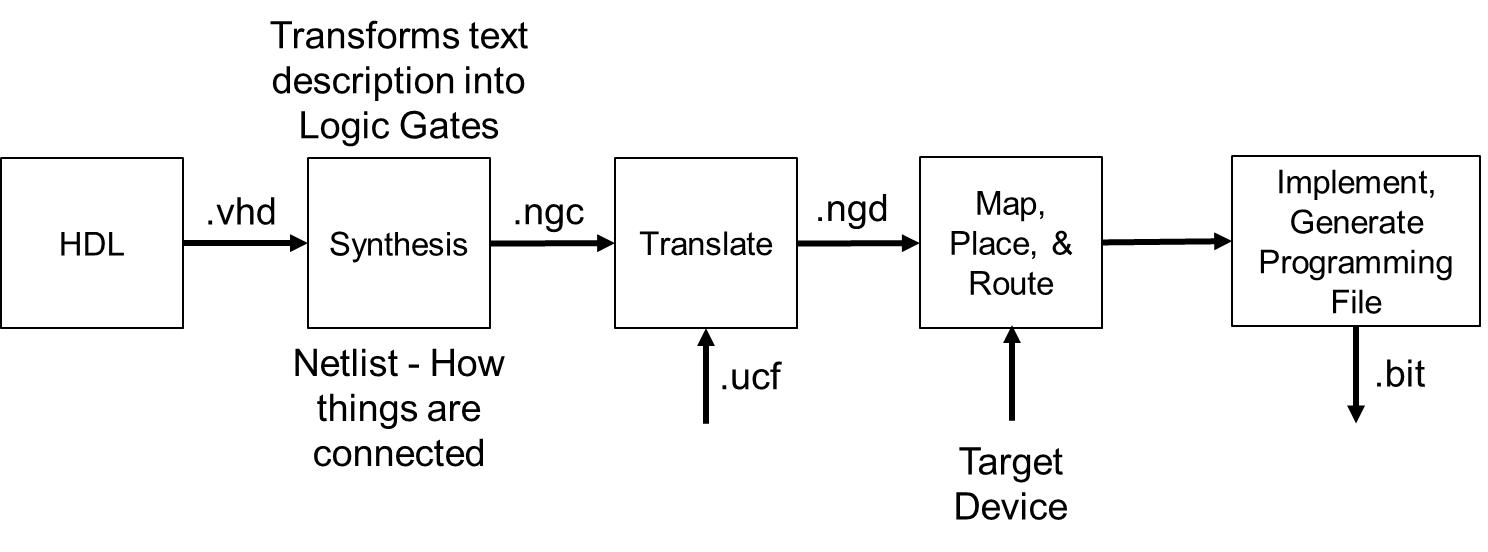
What is VHDL? A Hardware Description Language and not a programming language! **ery High Speed Integrated Circuits (VHSIC) Hardware Description Language**

Field Programmable Gate Array (FPGA)

* Look-up table
* Programmable interconnections

std\_logic:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| U | X | 0 | 1 | Z | W | L | H | - |
| uninitialized | Forcing Unknown | Forcing 0 | Forcing 1 | High Impedance | Weak Unkown | Weak 0 | Weak 1 | Don’t Care |

std\_logic\_vector – collection of std\_logic signals in a bus.

Entity – Defines inputs and outputs of box

Architecture – Describes what is inside the box

Behavioral – Describes what module does in terms of the relationships between inputs and outputs

Structural – Describes what a module does in terms of how it is composed of simpler modules

library IEEE; -- These lines are similar to a #include in C

use IEEE.std\_logic\_1164.all;

**ISE VHDL Color Code Legend:**

|  |  |
| --- | --- |
| Color | VHDL |
| Green | Comment |
| Blue | Reserved Word |
| Pink | STD LOGIC 1164 Reserved Word |
| Gray | String |
| Black | Default text  Identifier  Number  Operator |

entity **ent\_name** is

Port ( **port\_name** : *mode* *signal\_type*;

… : … …);

end **ent\_name**;

architecture **arch\_name** of **ent\_name** is

-- signal declarations (behavioral, structural)

-- component declarations (structural)

begin

-- connect your stuff here

-- component implementations

end **arch\_name**;

**Declaration: In Architecture…Before Begin (**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**)**

**component Component\_Name**

**port ( -- signal\_name : mode signal\_type;**

**Input0, Input1 : in std\_logic;**

**Output :out std\_logic);**

**end component;**

**Instantiation: In Architecture…After Begin (**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**)**

**Component\_Label: Component\_Name**

**port map ( -- Port\_name => Signal name,**

**Input0 => a,**

**Input1 => b,**

**Output => Y);**

Testbenches:

* Assert (check output)
* Report (report message)
* Severity (Error, Warning, Note, Failure)

A <= '0'; B <= '0'; C <= '0'; D <= '0' ; S <= "00";

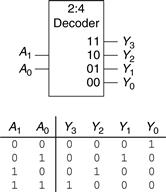
wait for 10 ns;

ASSERT Y = '1' REPORT "Test 0000 00 Failed" SEVERITY ERROR;

-- Can use Error, Warning, Note Failure

Be Careful with its use: What could be an issue?

**Homework: Exercise 4-13: Build and test a 2 to 4 Decoder Module in HDL**



----------------------------------------------------------------------------------

-- Name: Maj Jeff Falkinburg

-- Date: Spring 2017

-- Course: ECE 281

-- File: ce2\_ha\_structural.vhd

-- HW: Lecture 13

-- Purp: Half Adder - Structural Implementation.

--

-- Doc: None

-- Academic Integrity Statement: I certify that, while others may have

-- assisted me in brain storming, debugging and validating this program,

-- the program itself is my own work. I understand that submitting code

-- which is the work of other individuals is a violation of the honor

-- code. I also understand that if I knowingly give my original work to

-- another individual is also a violation of the honor code.

----------------------------------------------------------------------------------

library IEEE; -- These lines are similar to a #include in C

use IEEE.std\_logic\_1164.all;

library unisim; -- Use these libraries if you are using primitive components

use unisim.vcomponents.all;

Build Entity and Architecture of Half Adder:



entity ce2\_ha\_structural is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end ce2\_ha\_structural;

architecture Structural of ce2\_ha\_structural is

component AND2

port ( i0, i1 : in std\_logic;

o :out std\_logic);

end component;

component XOR2

port ( i0, i1 : in std\_logic;

o :out std\_logic);

end component;

All the code on the left can be summarized behaviorally like this:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ce2\_half\_adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end ce2\_half\_adder;

architecture Behavioral of ce2\_half\_adder is

begin

S <= A xor B;

Cout <= A and B;

end Behavioral;

signal s1, s2 : std\_logic; -- wires

-- which begin & end in the component

begin

unit1: AND2

port map ( -- s1 <= A AND B; (i.e. Cout)

i0 => a,

i1 => b,

o => s1);

unit2: XOR2

port map ( -- s2 <= A XOR B; (i.e. Sum)

i0 => a,

i1 => b,

o => s2);

Cout <= s1;

S <= s2;

end Structural;