

Lecture Notes - Introduction to Discrete Digital Logic and Programmable Logic.

Basic Logic Gates

Similar to lectures notes from lab 1, the following is a review of basic gates truth tables. This time, you will implement a simple circuit using VHDL and Xilinx tools. The objective is for you to compare between the programmable logic implementation against the discrete logic one.

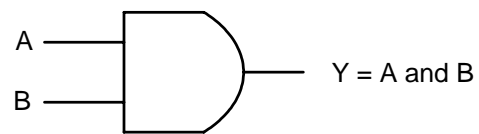
AND Gate

The notation for the AND of A and B is $A*B$ or AB .

Truth Table

A	B	Y=AB
0	0	0
0	1	0
1	0	0
1	1	1

Symbol



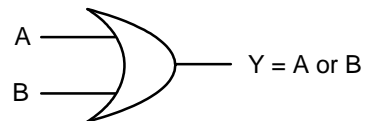
OR Gate

The notation for the OR of A and B is $A+B$.

Truth Table

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Symbol



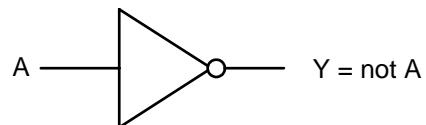
Inverter

The notation for the COMPLEMENT or NOT of A is A' .

Truth Table

A	Y=A'
0	1
1	0

Symbol



Programmable Logic Introduction

In the world of digital electronic systems, there are three basic kinds of devices: memory, microprocessors, and logic. Memory devices store random information such as the contents of a spreadsheet or database. Microprocessors execute software instructions to perform a wide variety of tasks such as running a word processing program or video game. Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform. [1]

Logic devices can be classified into two broad categories - fixed and programmable. As the name suggests, the circuits in a fixed logic device are permanent, they perform one function or set of functions - once manufactured, they cannot be changed. On the other hand, programmable logic devices (PLDs) are standard, off-the-shelf parts that offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be changed at any time to perform any number of functions. [1]

With programmable logic, designers use inexpensive software tools to quickly develop, simulate, and test their designs. Then, a design can be quickly programmed into a device, and immediately tested in a live circuit. An standard language used designers to describe the circuit or system to be implemented is VHDL.

VHDL stands for VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. This language resulted from the US Department of Defense's effort to advance and overhaul the way integrated circuits were exchanged between different companies. In 1987, the IEEE adopted as their standard IEEE Std. 1076-1987 (the language was updated in 1993).

As the name implies, VHDL is a programming language that has been designed and optimized for describing the behavior of digital systems and integrated circuits. VHDL has features for describing the behavior of electronic components ranging from simple logic gates to complete microprocessors and custom chips. One of its major advantages is that it has the ability to facilitate design description at a higher level of abstraction than the gate level. The driving benefit behind VHDL is the ability to quickly go from software to hardware. You can use the software to describe the behavior of the circuit you wish to develop and then implement the design on programmable logic devices.

Engineers that can program chips using VHDL are in extremely high demand. One of the purposes of this section of the course is to introduce you to VHDL and its use in project designs. Through the different labs we'll discuss some of the different aspects of coding in this unique language.

Today's lecture is about the pair entity/architecture present in every VHDL design file. More than one pair entity/architecture can be present in a single file, but neither entity nor architecture can be by their selves; they must have its pair.

An entity will describe the system from and outsider point of view, given information about the system's inputs and outputs; as if the system were a black box. In the VHDL for this lab session, you will have the following lines of code:

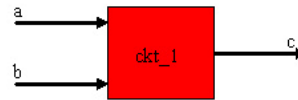
```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY ckt_1 IS PORT (
    a,b:    IN std_logic;
    f:      OUT std_logic
);
END ckt_1;

```

The above lines of code define the inputs and outputs - the ports - of this circuit. It is called a port declaration or an entity declaration. This code says that entity "ckt_1" will have two inputs. These inputs will be "a" and "b". It also says entity "ckt_1" will have one output, f. The "std_logic" type is defined in the "std_logic_1164 package", which resides in the IEEE library. In order to use it, we must first state the library in which it exists (IEEE) and then use it with the USE clause, separating the library and package names with a period. The "all" at the end of the use clause indicates that the entire contents of the std_logic_1164 package are visible, and therefore may be used.

The block diagram below is an abstract representation of the circuit to be implemented in this lab's tutorial. At this point, we do not know what is contained in the box or what it does. The above lines of code implement the box whose behavior will be described later in the code.



Architecture describes the "soul" of the system, its behavior or composition. Through the lab sessions we'll discover that there are different ways to specify it. In the VHDL for this lab session, you will have the following lines of code:

```

architecture behavioral of FEWGATES is
begin
fcn: process (a, b, c) begin
    y <= ((NOT a) AND (NOT b)) OR c ;
end process;
end behavioral;

```

VHDL is versatile language. It not only can describe a circuit but it can also be used to describe a test bench or write documentation, as we'll see in the labs through the semester.

References

[1] What is Programmable Logic? Available at: <http://www.xilinx.com/company/about/programmable.html>

Revision History

Date	Version	Comments
Spring04	1.0	Initial release
Spring05	1.1	Modification basic gates as a review from first lab.