Appendix: Boolean Operations

This appendix elaborates the elementary boolean operations that are used in various modules, which are the following: AND, OR, NAND, NOR, and XOR. Each operation compares two binary numbers, resulting in a new one.

Operation Basics

The first step when working with boolean operators is to convert the two provided parameters into binary form. Each 1 in a binary number, or bit, represents a certain value. The last one equals 1, the one before 2, the one before that 4. The values of each preceding number is double the value of the next. Therefore, 127 equals 1111111, 3 equals 11, and 25 equals 11001. When two values have different lengths, add as many zeros in front of that number until they do have the same size. This means that 00001010 and 1010 are the same value.

AND-Operations

An AND-operation compares each individual bit in a binary number and outputs a 1 if both bits are a 1 as well. This means that an AND-operation with the values 1010 and 1101 outputs the value 1000, or the decimal value 8.

OR-Operations

An OR-operation compares each individual bit and outputs a 1 if either of the two is or both are a 1. This means that an OR-operation with the values 1000 and 0011 outputs the value 1011, or the decimal value 11.

NAND-Operations

A NAND-operation compares each individual bit and only outputs a 1 if not both of the bits are a 1. This means that a NAND-operation with values 0101 and 0111 outputs the value 1010, or the decimal value 10.

NOR-Operations

A NOR-operation is the inverse of an OR-operation and outputs a 1 if none of the bits are a 1. This means that a NOR-operation with the values 1100 and 1001 outputs the value 0010, or the decimal value 2.

XOR-Operations

An XOR-operation is an OR-operation, however, also outputs a 0 if both values are a 1. This means that an XOR-operation with the values 0110 and 1100 outputs the value 1010, or the decimal value 10.