VisiBoole

The Complete Guide

Table of Contents

[Introduction 2](#_Toc479712695)

[GUI 2](#_Toc479712696)

[General Formatting 3](#_Toc479712697)

[Setup 3](#_Toc479712698)

[Variable Declaration 3](#_Toc479712699)

[Boolean Operators 3](#_Toc479712700)

[Assignment Operator 3](#_Toc479712701)

[AND Gate 3](#_Toc479712702)

[OR Gate 3](#_Toc479712703)

[NOT Gate 3](#_Toc479712704)

[Format Specifiers 4](#_Toc479712705)

[Binary 4](#_Toc479712706)

[Hexadecimal 4](#_Toc479712707)

[Signed Integer 4](#_Toc479712708)

[Unsigned Integer 4](#_Toc479712709)

[Making a Program 4](#_Toc479712710)

[Creating our VisiBoole File 4](#_Toc479712711)

[Declaring Variables 4](#_Toc479712712)

[Basic Expressions 4](#_Toc479712713)

[Running the Program 4](#_Toc479712714)

[More Expressions 4](#_Toc479712715)

[Simple Formatting 4](#_Toc479712716)

[Run Mode 5](#_Toc479712717)

# Introduction

VisiBoole was created as a teaching aid. It was to provide a visualization of the operation of simple to fairly complex digital hardware designs. Use of a preliminary implementation of the system showed that it also provided a powerful design environment. This was due to the fact that designs can be created and tested incrementally. The interactive testing provides a very easy way to quickly test each aspect of the design as it is created. The designer may easily flip back and forth between design entry and testing. It was estimated that college seniors that used VisiBoole to create designs and then translated them to a production HDL (VHDL, Verilog, AHDL) produced a working design in one fourth the time as compared to students designing directly in the production HDL. VisiBoole consists of its own dialect of a Hardware Description Language (HDL) that lends itself to a color-coded display of the design operation. When VisiBoole is in simulation mode (simulating a design expressed in its HDL), the display and mouse serve as an interactive interface between the user and the simulation engine. It’s HDL and simulation engine provide an extremely simple and easy to learn and use digital design and verification system. Despite its simplicity it is powerful enough to create and test complex digital designs involving both combinational and sequential logic. Its features include support of hierarchical modular designs.

# GUI

In this section we will describe the different features of the GUI, and how to use it.

Within the Menu Bar you have File, Edit, and View. Within the File tab you can find New, Open, and Save. The next tab is the Edit tab, and within it you can find Undo, Redo, Cut, Copy, Paste, and Select All. The last tab is the View tab, and within this tab you can change the view of your Editor to be either the Standard view (figure 1.1), the Horizontal view (figure 1.2), or the Vertical view (figure 1.3).

On the left you have the Sub-Designs menu. This menu will hold all of your open .vbi files. From this menu you can double click on a file name and it will open the tab containing the content of the file you clicked.

Below the Editor you can find the run button, which will run the currently open file, which changes the Editor into a colorful, and clickable window which allows you to interact with your implementation.

# General Formatting

In this section we will describe the General Formatting guidelines of a .vbi file.

## Setup

All files used within VisiBoole must end with the extension .vbi.

Optionally, at the beginning of every .vbi file you can declare you variables, if you do not do this the program will automatically declare the variables for you as it comes across them.

Variables can then be used to help evaluate expressions. These expressions are composed of variables and Boolean Logic Operators.

Format Specifiers can be used anywhere within the program in order to simply see the contained variables in a different configuration.

## Variable Declaration

Variable Declaration is optional, and can be done anywhere in the file. On a new line, create a list of variables separated by spaces as shown in figure 2.1, followed by a semi-colon.

Declared variables by default are set as True. To change the default value, put an asterisk [\*] in front of the variable to change it to False.

## Boolean Operators

### Assignment Operator

The assignment operator is denoted by an equals sign, [=].

### AND Gate

The AND gate is denoted by a simple space between the variables.

### OR Gate

The OR gate is denoted by the plus sign, [+].

### NOT Gate

The NOT gate is denoted by the tilde, [~].

## Format Specifiers

Format specifiers are an easy way to show how a variable, or multiple variables separated by a space, would be represented in a different configuration. Below are the different types of specifiers currently functional within VisiBoole.

### Binary

This specifier will produce a binary number denoted by %b{ A3 A2 A1 }.

### Hexadecimal

This specifier will produce a hexadecimal number denoted by %h{ A3 A2 A1 }.

### Signed Integer

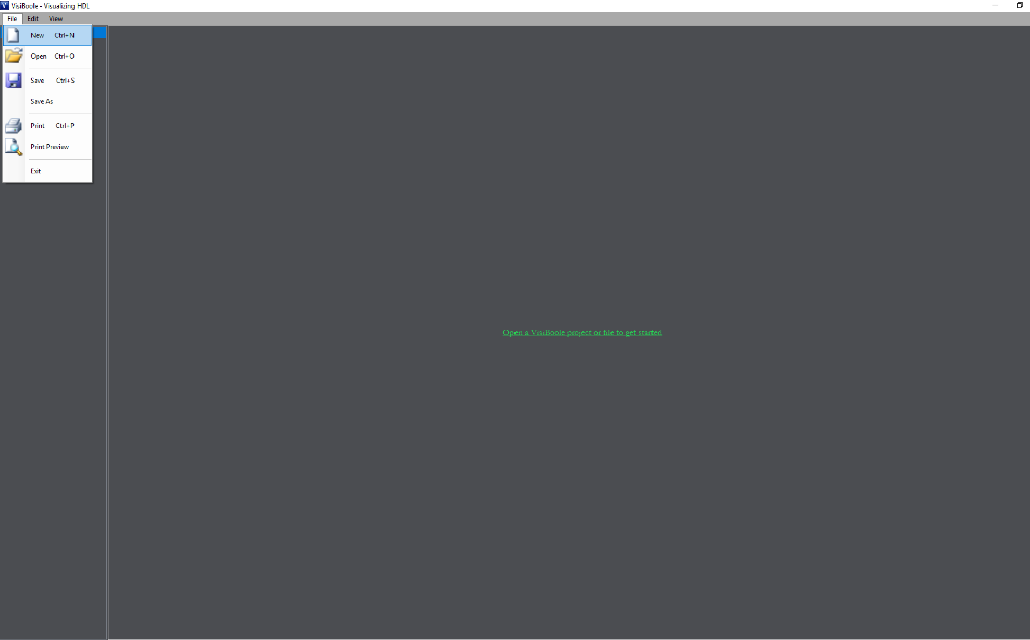
This specifier will produce a signed integer denoted by %d{ A3 A2 A1 }.

### Unsigned Integer

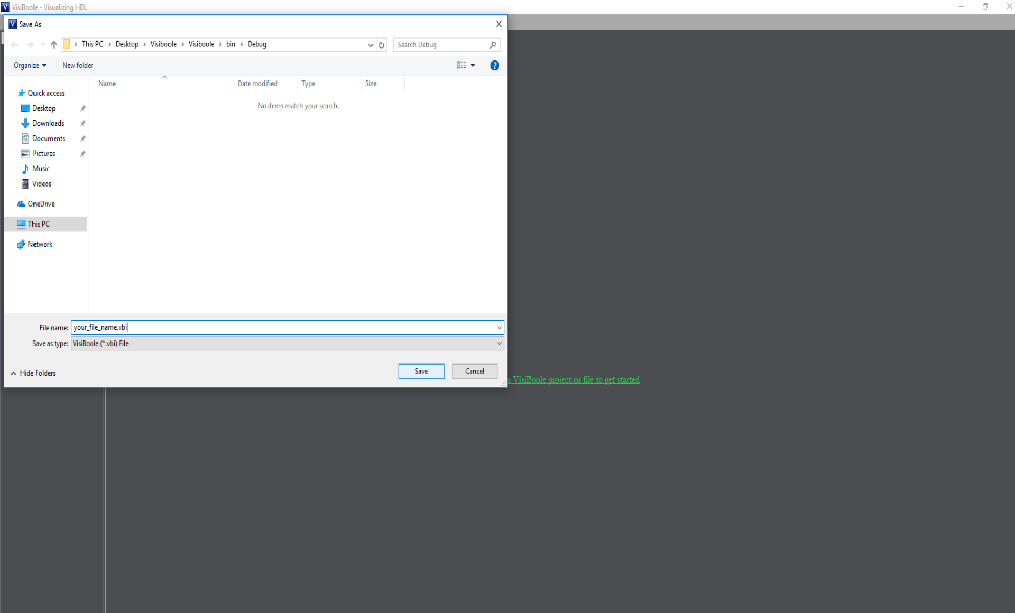
This specifier will produce an unsigned integer denoted by %u{ A3 A2 A1 }.

# Making a Program

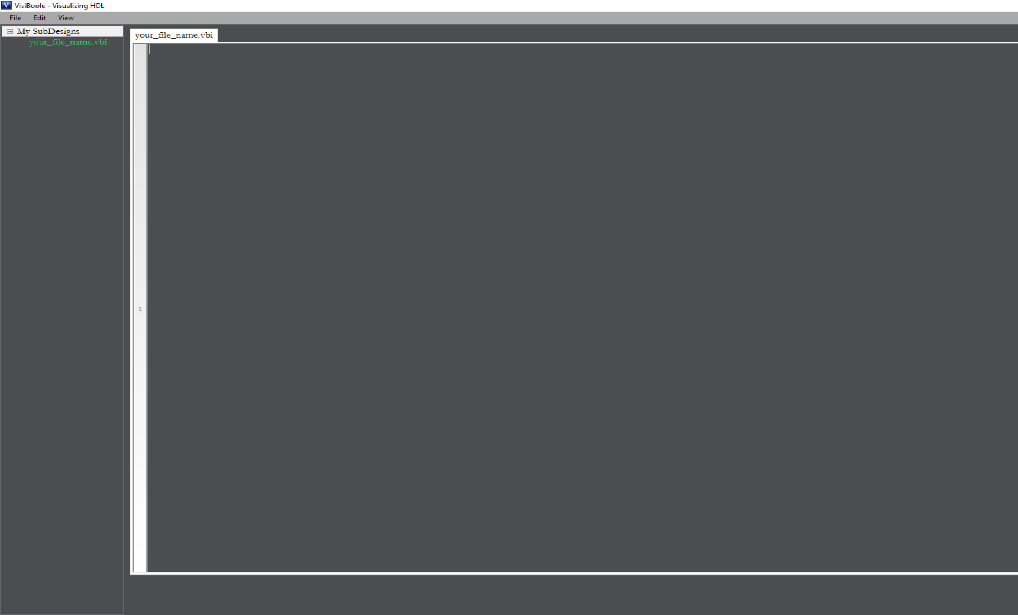
## Creating our VisiBoole File



Once you have started Visiboole, go to the top left corner of the screen with your mouse and click on the File tab and click New.



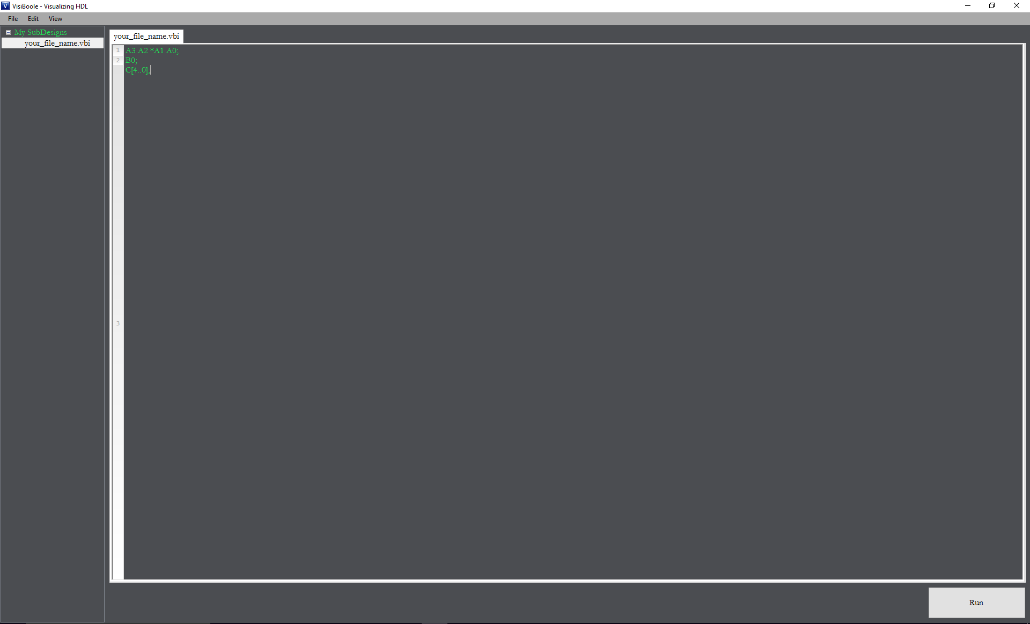
Now that you should have this dialog box appear. In the ‘file name’ section type in what you would like to name your file and then hit save.



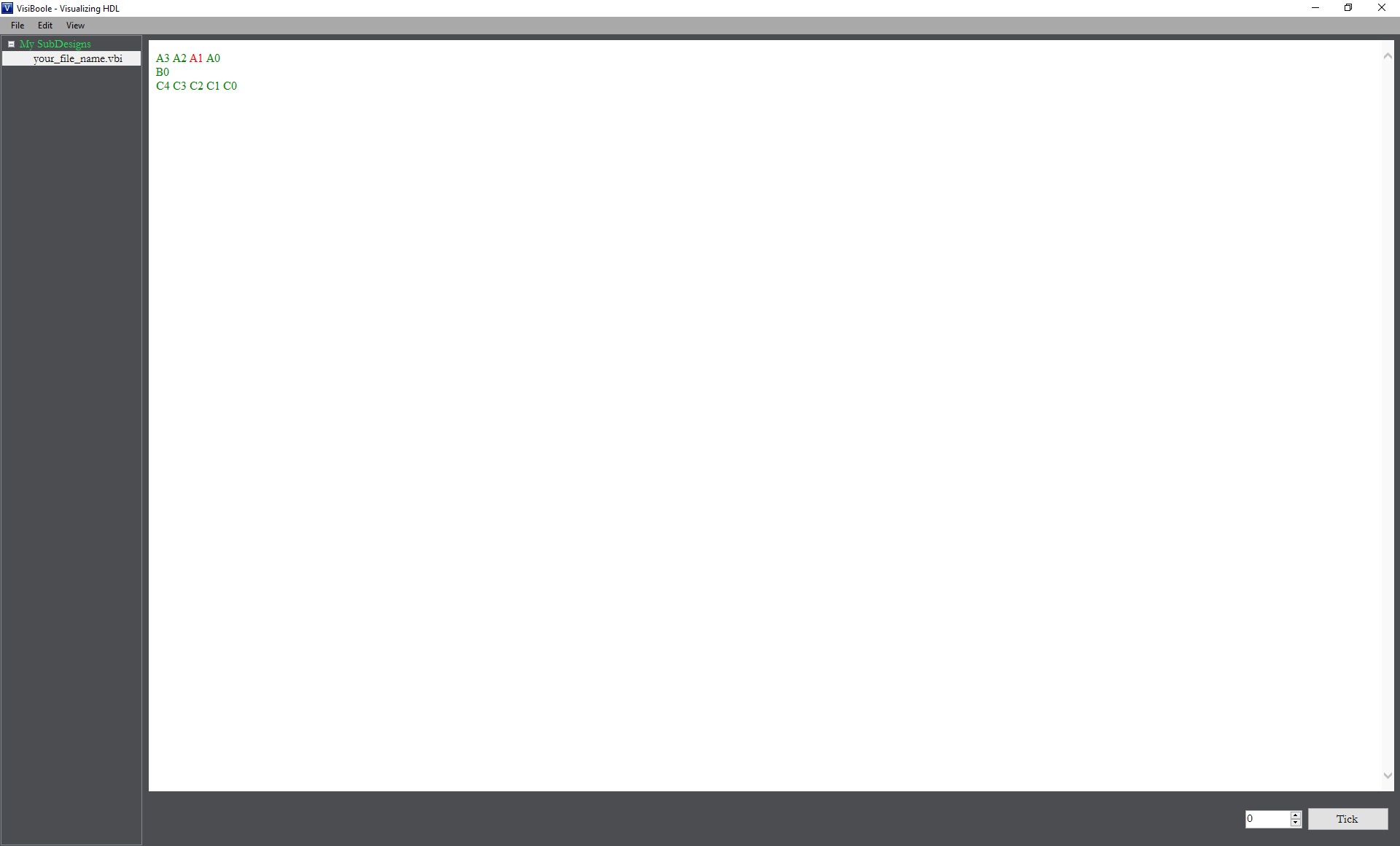
You have just created your first Visiboole file.

## Declaring Variables

In Visiboole, variables can either be declared before being used or they can be put directly into an expression. The main difference between these two is that with declaring the variable you can set its value with an ‘\*’ but putting variables directly into an expression will always initialize the variables to a value of false.

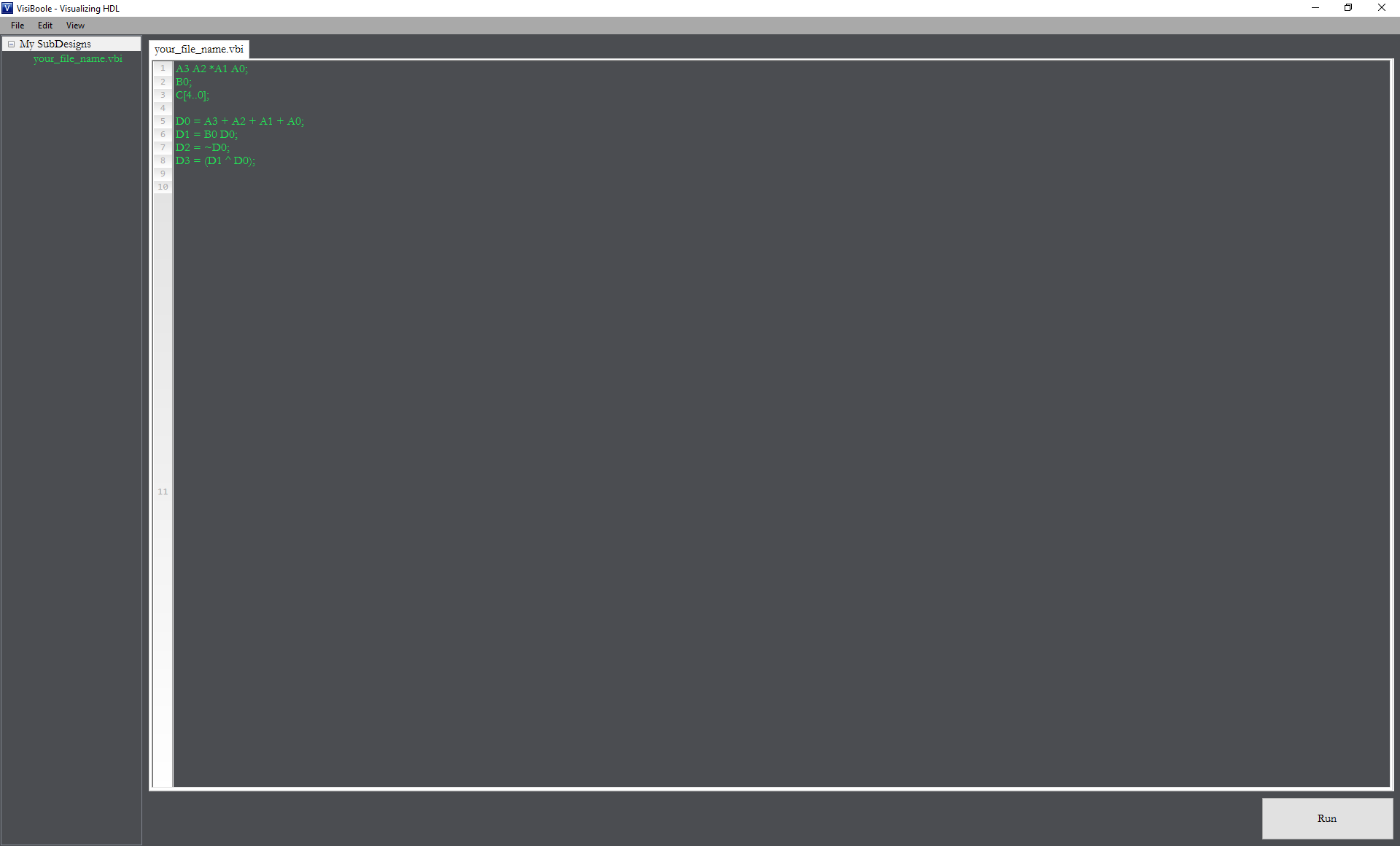


In this example code you can see there are two main ways of initializing variables. You can state them individually on the same line, or you can make a variable array like on line 3 of the example code.

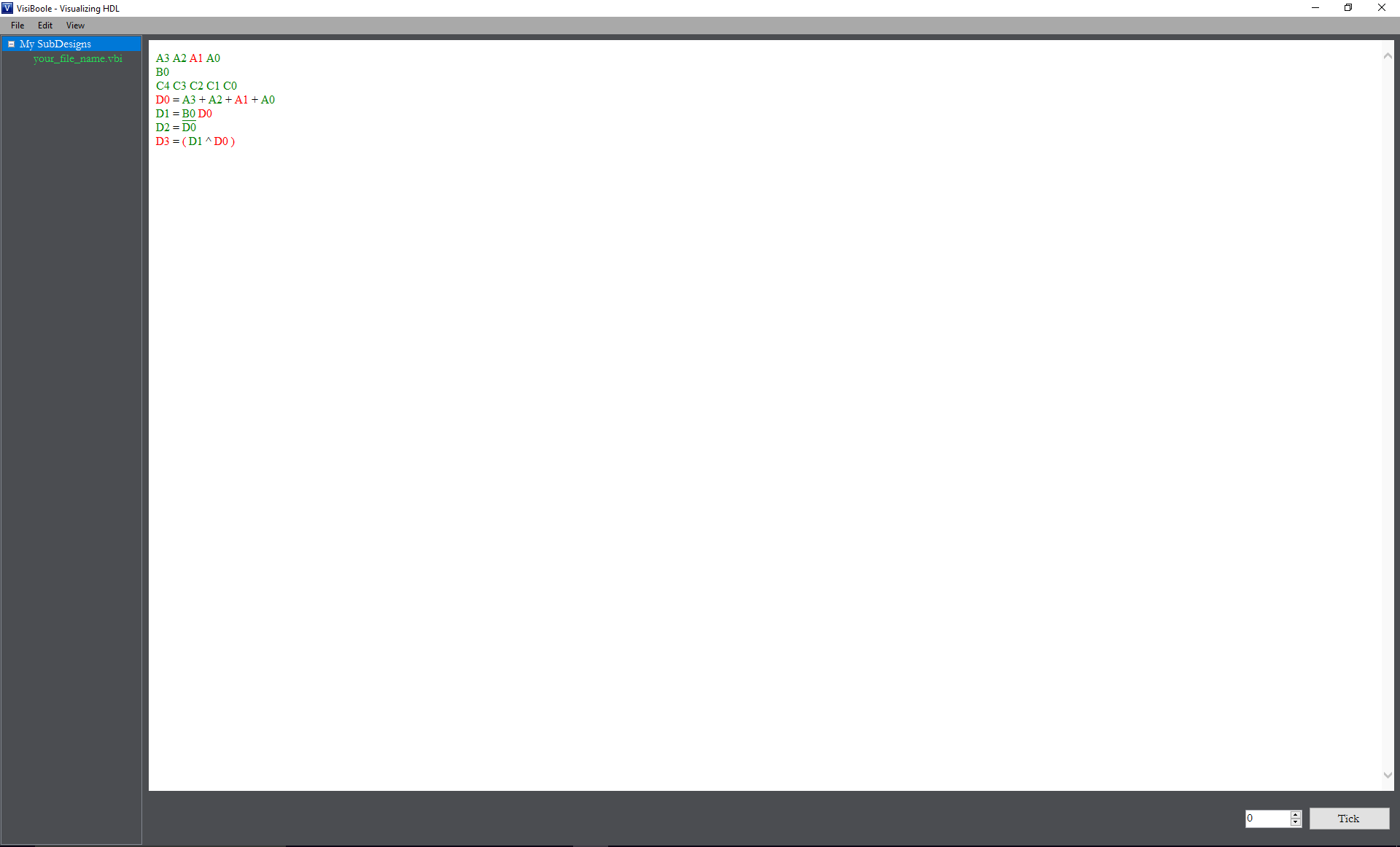


This is what the running code looks like.

## Basic Expressions



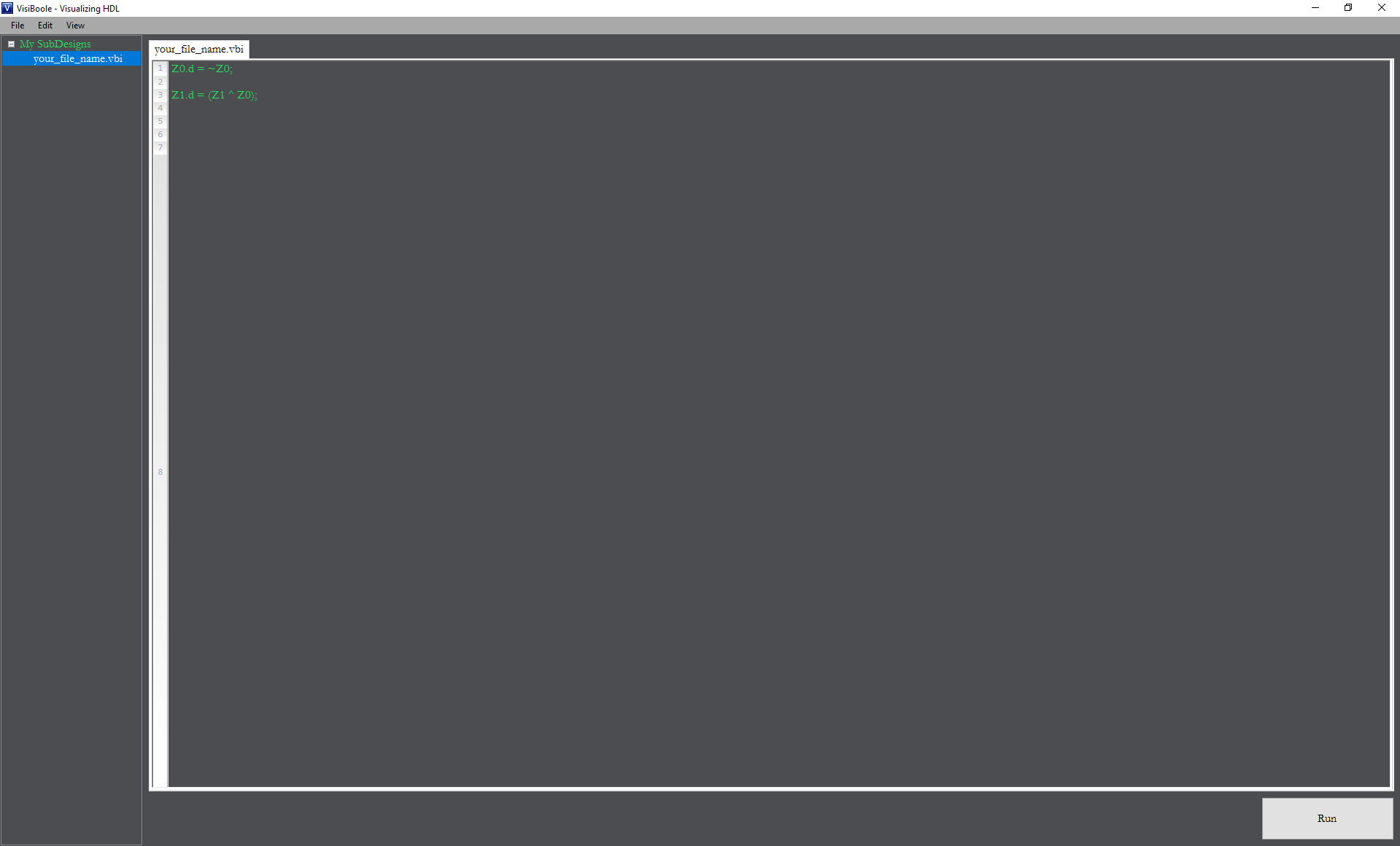
The above example demos how you would code the AND, OR, NOT, and XOR gates.



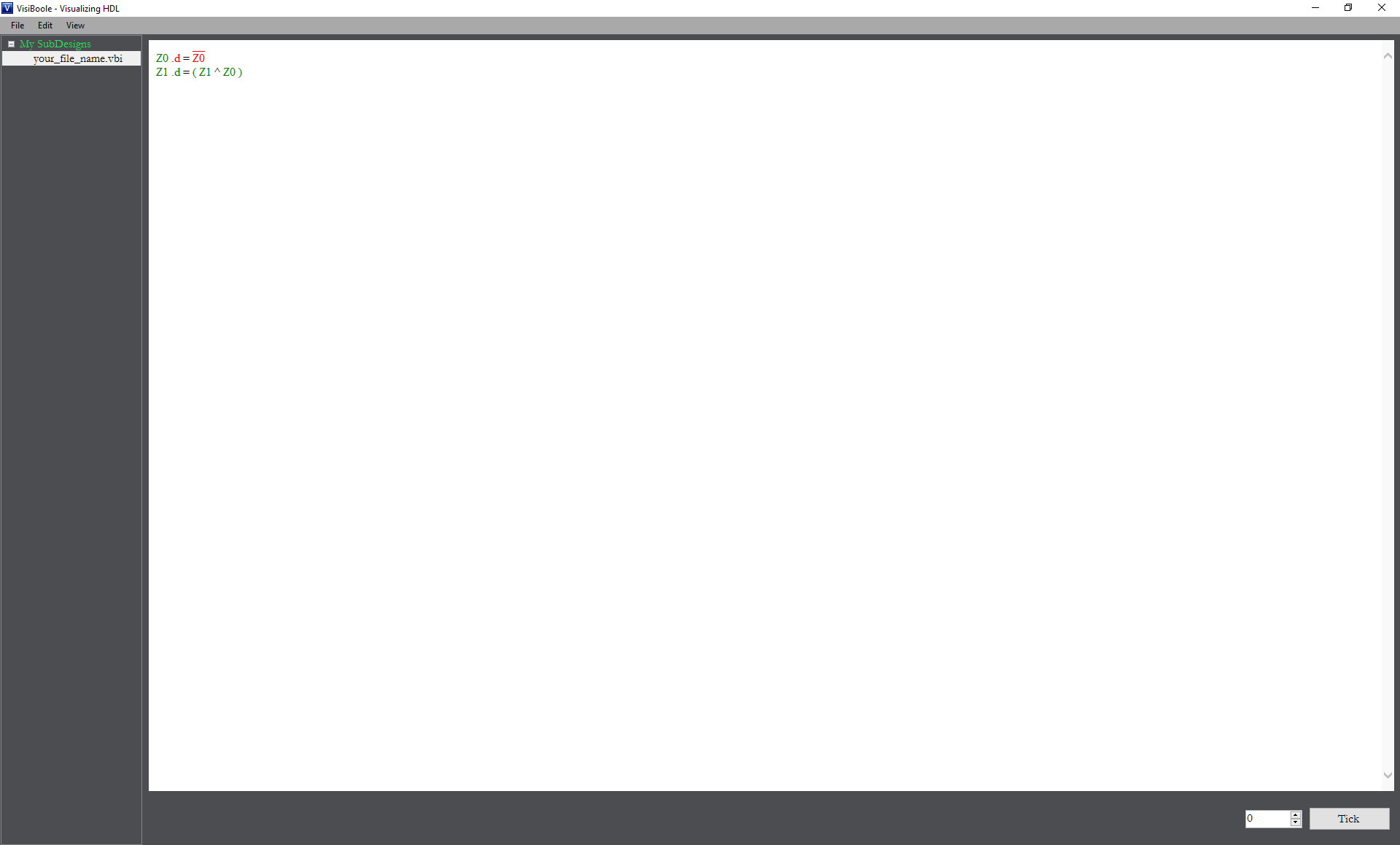
This is what the running code looks like.

## More Expressions

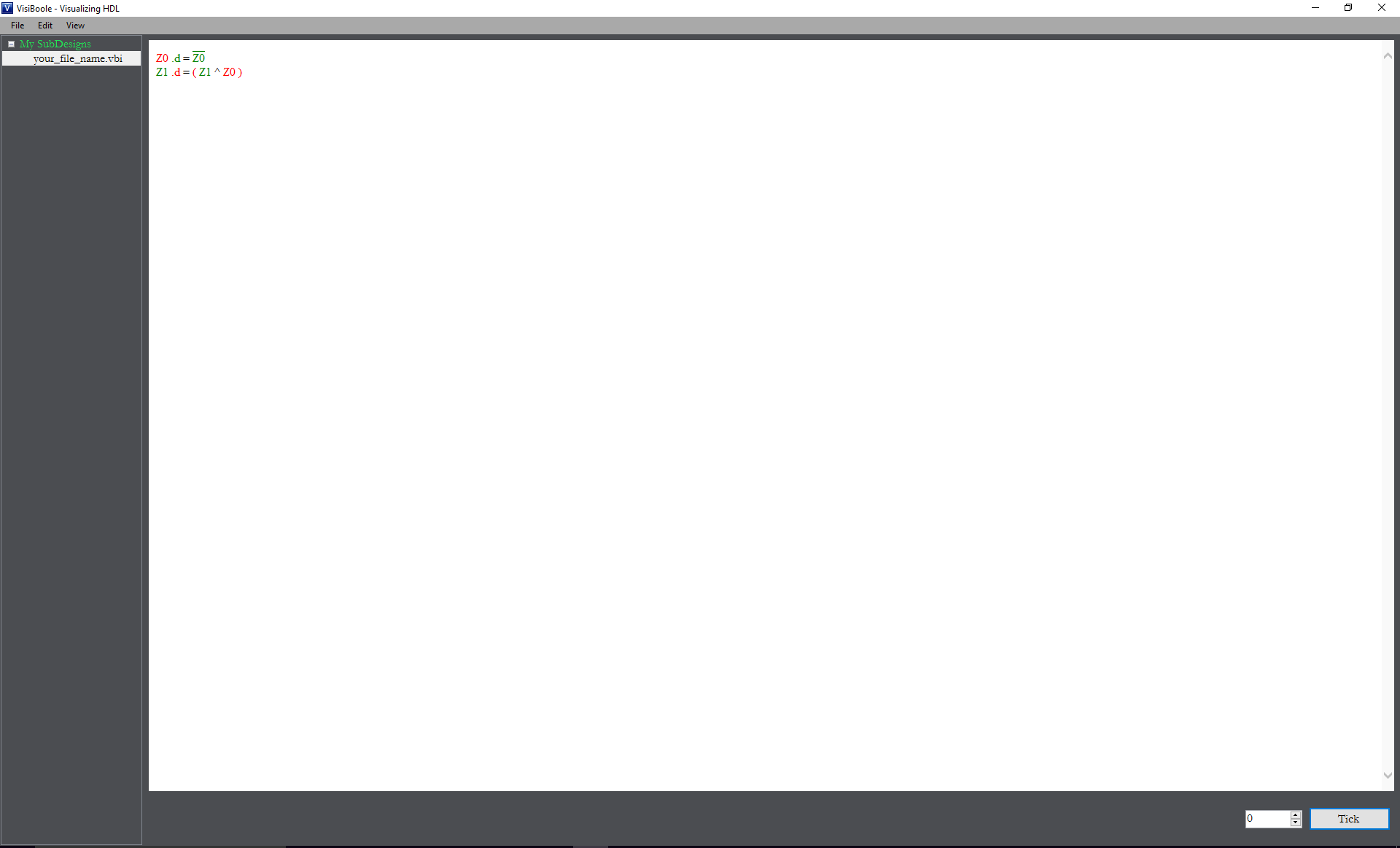
D-Flip-Flop



In order to use this feature you have to denote it in the line of code like so in the example code above.



This is what the initial run looks like.



This is what it should look like after hitting the Tick button once.

# Run Mode

Run mode is activated by clicking the ‘Run’ button shown below the Editor.

On this screen you will see the variables and expressions you have declared have now been color coded. Red represents the variable is True, green represents the variable is False, and black represents non-variables.

Within Run Mode, you can interact with the Sub-Design by clicking on the independent variables. By clicking on these variables you are switching their value, which causes the expressions to be re-evaluated in real time.