Using DESim to Simulate FPGA Projects

**Pre-Requisites:**

* Intel Quartus Prime
* Questa\*-Intel Starter Edition
  + <https://www.intel.com/content/www/us/en/software-kit/736572/intel-quartus-prime-lite-edition-design-software-version-21-1-1-for-windows.html?>
* Questa\*-Intel License
  + <https://www.intel.com/content/www/us/en/docs/programmable/683472/22-1/and-software-license.html>
* DESim
  + <https://github.com/fpgacademy/DESim>

# Preparing the Sample Project

This sample project shows a two-digit hexadecimal display driven by an 8-bit input.

A single module, named BinToHex, is created to accept a 4-bit input to display a hexadecimal seven-segment output. To complete the project, two instances of BinToHex is created to provide the necessary two hexadecimal displays with an 8-bit input. The following figure shows the Verilog code for BinToHex.v and Top.v.

NOTE: *It is important to set a common name for the top file of an FPGA project to avoid unnecessary edits to the configuration file later.*

1. Create a new FPGA project. Set devices, and simulation tools accordingly.
2. Add two Verilog files and follow the given code.

|  |  |
| --- | --- |
| **BinToHex.v**  module BinToHex(bcd, leds);  input [3:0] bcd;  output reg [7:1] leds;    always @(bcd)  case (bcd)  0: leds = 7'b1000000;  1: leds = 7'b1111001;  2: leds = 7'b0100100;  3: leds = 7'b0110000;  4: leds = 7'b0011001;  5: leds = 7'b0010010;  6: leds = 7'b0000010;  7: leds = 7'b1111000;  8: leds = 7'b0000000;  9: leds = 7'b0011000;  10: leds = 7'b0001000;  11: leds = 7'b0000011;  12: leds = 7'b1000110;  13: leds = 7'b0100001;  14: leds = 7'b0000110;  15: leds = 7'b0001110;  endcase  endmodule | **Top.v**  module Top(bcd0, bcd1, leds0, leds1);  input wire [3:0] bcd0, bcd1;  output wire [6:0] leds0, leds1;    BinToHex d1(.bcd(bcd0), .leds(leds0));  BinToHex d2(.bcd(bcd1), .leds(leds1));  endmodule |

1. Start Analysis and Synthesis. Synthesizing the project should produce the following RTL diagram.

Diagram, schematic

Description automatically generated

Figure . RTL View of the 8-Bit Binary Hexadecimal Display

After synthesis, the following figure shows the contents of the project folder.

Graphical user interface

Description automatically generated

Figure Project folder contents after synthesis.

# Preparing the Simulation to DESim

1. Download the two folders “sim” and “tb” using this link: <https://github.com/ShapePolygon/DESimConfig>
2. Copy the two folders to the project folder. The following figure shows the contents of the project folder after this operation.

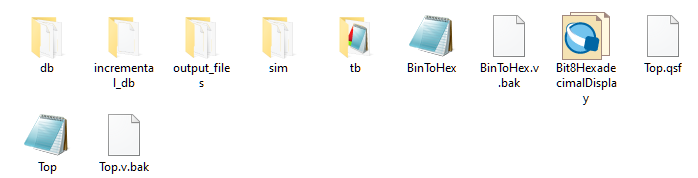
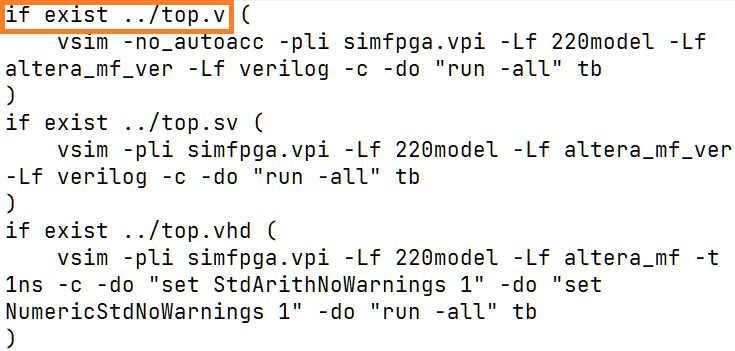


Figure Project folder contents after copying the sim and tb folders.

### Exploring the sim folder

The sim folder contains multiple files and a single work folder. Two .bat files are present containing scripts that will compile and simulate an FPGA project. It is unnecessary to modify the files in the folder unless you need to specify the file locations of the top-level file of the project.

Editing the run\_sim.bat file shows the following code:



Understanding how the code works is not required but it is important to note that this script will look for a file named “top.v” in the project folder. If your top-level file has a different name or location, modify this script to the location of said file.

### Exploring the tb folder

The tb folder contains a single file named tb.v. This file provides the input and output configuration for DESim and it instantiates the top file with the specified input and output parameters.

Editing the tb.v file shows the following code:

// Copyright (c) 2020 FPGAcademy

// Please see license at https://github.com/fpgacademy/DESim

`timescale 1ns / 1ns

`default\_nettype none

// This testbench is designed to hide the details of using the VPI code

module tb();

reg **CLOCK\_50** = 0; // DE-series 50 MHz clock

reg [ 3: 0] **KEY** = 0; // DE-series pushbutton keys

reg [ 9: 0] **SW** = 0; // DE-series SW switches

wire [47: 0] **HEX**; // HEX displays (six ports)

wire [ 9: 0] **LEDR**; // DE-series LEDs

reg key\_action = 0;

reg [ 7: 0] scan\_code = 0;

wire [ 2: 0] ps2\_lock\_control;

wire ps2\_clk;

wire ps2\_dat;

wire [ 7: 0] **VGA\_X**; // "VGA" column

wire [ 6: 0] **VGA\_Y**; // "VGA" row

wire [ 2: 0] **VGA\_COLOR**; // "VGA pixel" colour (0-7)

wire plot; // "Pixel" is drawn when this is pulsed

wire [31: 0] **GPIO**; // DE-series 40-pin header

initial $sim\_fpga(CLOCK\_50, SW, KEY, LEDR, HEX, key\_action, scan\_code,

ps2\_lock\_control, VGA\_X, VGA\_Y, VGA\_COLOR, plot, GPIO);

// DE-series HEX0, HEX1, ... ports

wire [ 6: 0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;

// create the 50 MHz clock signal

always #10

CLOCK\_50 <= ~CLOCK\_50;

// connect the single HEX port on "sim\_fpga" to the six DE-series HEX ports

assign HEX[47:40] = {1'b0, **HEX0**};

assign HEX[39:32] = {1'b0, **HEX1**};

assign HEX[31:24] = {1'b0, **HEX2**};

assign HEX[23:16] = {1'b0, **HEX3**};

assign HEX[15: 8] = {1'b0, **HEX4**};

assign HEX[ 7: 0] = {1'b0, **HEX5**};

**Top DUT (KEY, SW, LEDR);**

endmodule

The code in bold (black) shows the assignable input and output ports of the simulator. These ports maybe passed as parameters to the top module of the project.

Note the sizes of the ports. Use the array notation to assign specific bits necessary for your project. Additionally, aliases are assigned to individual HEX ports (HEX0 to HEX5) based on the main HEX port. These individual HEX port may be assigned for convenience.

The most important part of the given code is the instantiation of the top module named DUT, as shown:

**Top DUT (KEY, SW, LEDR);**

This line of code represents the top-level file of the project. It is important to provide the appropriate input and output parameters to match the requirements of the project.

## Updating the tb.v to match the top-level file parameters

The top-level file of the project is given as:

module Top(bcd0, bcd1, leds0, leds1);

Therefore, two 4-bit switches, and two seven-segment displays should be provided.

1. Update the last line of tb.v to

**Top DUT (SW[3:0], SW[7:4], HEX0, HEX1);**

1. Save your changes to tb.v file.

From the given code, switches 0 to 3 controls the display of HEX0 while switches 4 to 7 controls the display of HEX1.

It is good practice to provide a Readme file to help users validate the functionality of the project. For example, in this project, the following may be saved to Readme.txt.

The circuit shows a two hexadecimal display using eight switches.

To use:

1. change the binary value of SW[3:0] to update the display   
 of the ones digit of the hexadecimal display (HEX0).

2. change the binary value of SW[7:4] to update the display   
 of the tens digit of the hexadecimal display (HEX1).

## Simulation using DESim

1. Open DESim. The figure below shows the main user interface.

Graphical user interface

Description automatically generated

1. Click “Open Project.”
2. Select the folder location of your project.
3. Click “Compile Testbench.” It should display “Compilation successful.”

Graphical user interface, text, application

Description automatically generated

1. Click “Start Simulation”

Graphical user interface, text, application, email

Description automatically generated

1. Click on the switches to change the hexadecimal displays.

Graphical user interface, text, application, email

Description automatically generated