Introduction to Digital Design
Using Digilent FPGA Boards
─ Block Diagram / Verilog Examples

Richard E. Haskell
Darrin M. Hanna

Oakland University, Rochester, Michigan

NOTE: For the FPGA Labs you will NOT be using the Altec Active HDL software. Instead, you will be using the Xilinx tool chain as described in the Screen-shot document.

LBE Books
Rochester Hills, MI

Example 1

Switches and LEDs

In this example we will show the basic structure of a Verilog program and how to write logic equations for 2-input gates. Example 1a will show the simulation results using Aldec Active-HDL and Example 1b will show how to synthesize the program to a Xilinx FPGA on the BASYS or Nexys-2 board.

Prerequisite knowledge:
None
Learned in this Example:
Use of Aldec Active-HDL – Appendix A

1.1 Slide Switches

The slide switches on the BASYS and Nexys-2 boards are connected to pins on the FPGA through a resistor *R* as shown in Fig. 1.1. The value of *R* is 4.7 kΩ on the BASYS board and 10 kΩ on the Nexys-2 board. When the slide switch is down it is connected to ground and the input *sw*[*i*] to the FPGA is read as a logic 0. When the slide switch is up it is connected to 3.3 V and the input *sw*[*i*] to the FPGA is read as a logic 1.

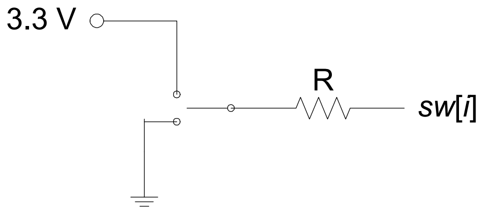


Figure 1.1 Slide switch connection

There are eight slide switches on the BASYS and Nexys-2 boards. The eight pin numbers on the FPGA corresponding to the eight slide switches are given in a *.ucf* file. The file *basys2.ucf* shown in Listing 1.1 defines the pin numbers for all I/O on the BASYS board. Note that we have named the slide switches *sw*[*i*], *i* = 0:7, which correspond to the switch labels on the board. We will always name the slide switches *sw*[*i*] in our top-level designs so that we can use the *basys2.ucf* file without change. Because the pin numbers on the Nexys-2 board are different from those on the BASYS board we will use a different file called *nexys2.ucf* to define the pin numbers on the Nexys-2 board. The names of the I/O ports, however, will be the same for both boards. Therefore, all of the examples in this book can be used with either board by simply using the proper *.ucf* file when implementing the design. Both of these *.ucf* files can be downloaded from www.lbebooks.com.

1.2 LEDs

A light emitting diode (LED) emits light when current flows through it in the positive direction as shown in Fig. 1.2. Current flows through the LED when the voltage on the *anode* side (the wide side of the black triangle) is made higher than the voltage on the *cathode* side (the straight line connected to the apex of the black triangle). When current flows through a lighted LED the forward voltage across the LED is typically between +1.5 and +2.0 volts. If voltage *V2* in Fig. 1.2 is less than or equal to voltage *V1* then no current can flow through the LED and therefore no light will be emitted. If voltage *V2* is greater than voltage *V1* then current will flow through the resistor *R* and the LED. The resistor is used to limit the amount of current that flows through the LED. Typical currents needed to light LEDs range from 2 to 15 milliamps.

Listing 1.1 basys2.ucf

```
# Pin assignment for LEDs
NET "ld<7>" LOC = "p2" ;
NET "ld<6>" LOC = "p3" ;
NET "ld<5>" LOC = "p4" ;
NET "ld<4>" LOC = "p5" ;
NET "ld<3>" LOC = "p7" ;
NET "ld<2>" LOC = "p8" ;
NET "ld<1>" LOC = "p14" ;
NET "ld<0>" LOC = "p15" ;

# Pin assignment for slide switches
NET "sw<7>" LOC = "p6";
NET "sw<6>" LOC = "p10";
NET "sw<5>" LOC = "p12";
NET "sw<4>" LOC = "p18";
NET "sw<3>" LOC = "p24";
NET "sw<2>" LOC = "p29";
NET "sw<1>" LOC = "p36";
NET "sw<0>" LOC = "p38";

# Pin assignment for pushbutton switches
NET "btn<3>" LOC = "p41";
NET "btn<2>" LOC = "p47";
NET "btn<1>" LOC = "p48";
NET "btn<0>" LOC = "p69";

# Pin assignment for 7-segment displays
NET "a_to_g<6>"  LOC = "p25"  ;
NET "a_to_g<5>"  LOC = "p16"  ;
NET "a_to_g<4>"  LOC = "p23"  ;
NET "a_to_g<3>"  LOC = "P21"  ;
NET "a_to_g<2>"  LOC = "p20"  ;
NET "a_to_g<1>"  LOC = "p17"  ;
NET "a_to_g<0>"  LOC = "p83"  ;
NET "dp"  LOC = "p22"  ;

NET "an<3>" LOC = "p26";
NET "an<2>" LOC = "p32";
NET "an<1>" LOC = "p33";
NET "an<0>" LOC = "p34";

# Pin assignment for clock
NET "mclk" LOC = "p54";
```

Nexys2Simple.ucf

NOTE: When using this configuration file in a project, variable names used to define input and output wires must be exactly as defined in this file. The variable names in the Basys.ucf file are not exactly the same as those in the Nexsys.ucf file.

```
# clock pin for Nexys 2 Board
NET "clk" LOC= "B8"; # Bank = 0 , Pin name = IP_L13P_0/GCLK8 , Type = GCLK , Sch name = GCLK0
# NET "clk1" LOC= "U9"; # Bank = 2 , Pin name = IO_L13P_2/D4/GCLK14 , Type = DUAL/GCLK , Sch name = GCLK1

# Pin assignment for DispCtl
# Connected to Nexys 2 onBoard 7seg display
NET "seg<0>" LOC= "L18"; # Bank = 1 , Pin name = IO_L10P_1 , Type = I/O , Sch name = CA
NET "seg<1>" LOC= "F18"; # Bank = 1 , Pin name = IO_L19P_1 , Type = I/O , Sch name = CB
NET "seg<2>" LOC= "D17"; # Bank = 1 , Pin name = IO_L23P_1/HDC , Type = DUAL , Sch name = CC
NET "seg<3>" LOC= "D16"; # Bank = 1 , Pin name = IO_L23N_1/LDC0 , Type = DUAL , Sch name = CD
NET "seg<4>" LOC= "G14"; # Bank = 1 , Pin name = IO_L20P_1 , Type = I/O , Sch name = CE
NET "seg<5>" LOC= "J17"; # Bank = 1 , Pin name = IO_L13P_1/A6/RHCLK4/IRDY1 , Type = RHCLK/DUAL , Sch name = CF
NET "seg<6>" LOC= "H14"; # Bank = 1 , Pin name = IO_L17P_1 , Type = I/O , Sch name = CG

NET "dp" LOC= "C17"; # Bank = 1 , Pin name = IO_L24N_1/LDC2 , Type = DUAL , Sch name = DP
NET "an<0>" LOC= "F17"; # Bank = 1 , Pin name = IO_L19N_1 , Type = I/O , Sch name = AN0
NET "an<1>" LOC= "H17"; # Bank = 1 , Pin name = IO_L16N_1/A0 , Type = DUAL , Sch name = AN1
NET "an<2>" LOC= "C18"; # Bank = 1 , Pin name = IO_L24P_1/LDC1 , Type = DUAL , Sch name = AN2
NET "an<3>" LOC= "F15"; # Bank = 1 , Pin name = IO_L21P_1 , Type = I/O , Sch name = AN3

# Pin assignment for Leds
# Connected to Nexys 2
NET "Led<0>" LOC= "J14"; # Bank = 1 , Pin name = IO_L14N_1/A3/RHCLK7 , Type = RHCLK/DUAL , Sch name = JD10/LD0
NET "Led<1>" LOC= "J15"; # Bank = 1 , Pin name = IO_L14P_1/A4/RHCLK6 , Type = RHCLK/DUAL , Sch name = JD9/LD1
NET "Led<2>" LOC= "K15"; # Bank = 1 , Pin name = IO_L12P_1/A8/RHCLK2 , Type = RHCLK/DUAL , Sch name = JD8/LD2
NET "Led<3>" LOC= "K14"; # Bank = 1 , Pin name = IO_L12N_1/A7/RHCLK3/TRDY1 , Type = RHCLK/DUAL , Sch name =
JD7/LD3
NET "Led<4>" LOC= "E17"; # Bank = 1 , Pin name = IO , Type = I/O , Sch name = LD4?
NET "Led<5>" LOC= "P15"; # Bank = 1 , Pin name = IO , Type = I/O , Sch name = LD5?
NET "Led<6>" LOC= "F4"; # Bank = 3 , Pin name = IO , Type = I/O , Sch name = LD6?
NET "Led<7>" LOC= "R4"; # Bank = 3 , Pin name = IO/VREF_3 , Type = VREF , Sch name = LD7?

# Pin assignment for Switches
# Connected to Nexys 2
NET "sw<0>" LOC= "G18"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW0
NET "sw<1>" LOC= "H18"; # Bank = 1 , Pin name = IP/VREF_1 , Type = VREF , Sch name = SW1
NET "sw<2>" LOC= "K18"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW2
NET "sw<3>" LOC= "K17"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW3
NET "sw<4>" LOC= "L14"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW4
NET "sw<5>" LOC= "L13"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW5
NET "sw<6>" LOC= "N17"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW6
NET "sw<7>" LOC= "R17"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = SW7

# Pin assignment for Buttons
# Connected to Nexys 2
NET "btn<0>" LOC= "B18"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = BTN0
NET "btn<1>" LOC= "D18"; # Bank = 1 , Pin name = IP/VREF_1 , Type = VREF , Sch name = BTN1
NET "btn<2>" LOC= "E18"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = BTN2
NET "btn<3>" LOC= "H13"; # Bank = 1 , Pin name = IP , Type = INPUT , Sch name = BTN3
```

There are two different ways that an I/O pin of an FPGA can be used to turn on an LED. The first is to connect the FPGA pin to *V2* in Fig. 1.2 and to connect *V1* to ground. Bringing the pin (*V2*) high will then turn on the LED. To turn off the LED the output pin would be brought low. This is the method used for the LEDs *ld*[7] – *ld*[0] on the BASYS and Nexys-2 boards.

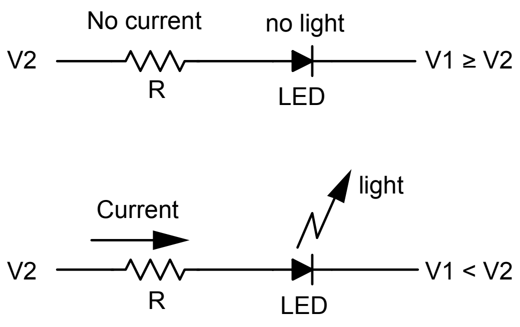
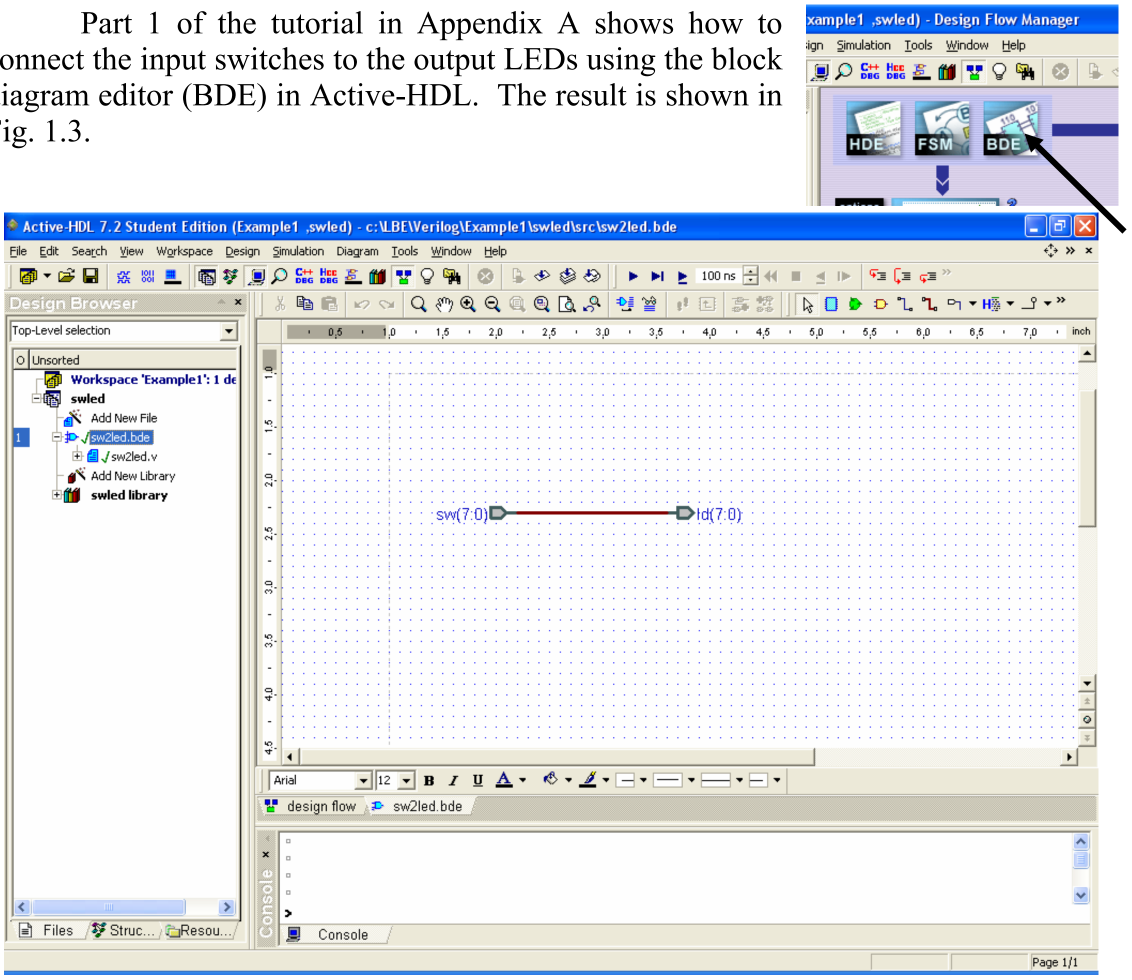


Figure 1.2 Turning on an LED

The second method is to connect the FPGA pin to *V1* in Fig. 1.2 and to connect *V2* to a constant voltage. Bringing the pin (*V1*) low will then turn on the LED. To turn off the LED the output pin would be brought high. This voltage should be equal to *V2* to make sure no current flows through the LED. This second method is the method used for the 7-segment displays on the BASYS and Nexys-2 boards. Examples 9 and 10 will show how to display hex digits on the 7-segment displays.

1.3 Connecting the Switches to the LEDs

Part 1 of the tutorial in Appendix A shows how to connect the input switches to the output LEDs using the block diagram editor (BDE) in Active-HDL. The result is shown in Fig. 1.3.


xample1 ,swled) - Design Flow Manager
ign Simulation Tools Window Help
HDE
FSM
BDE
Active-HDL 7.2 Student Edition (Example1 ,swled) - c:\LBE\Verilog\Example1\swled\src\sw2led.bde
File Edit Search View Workspace Design Simulation Diagram Tools Window Help
100 ns
Design Browser
Top-Level selection
Unsorted
Workspace 'Example1': 1 de
swled
Add New File
sw2led.bde
sw2led.v
Add New Library
swled library
sw(7:0)
ld(7:0)
Arial
12
design flow
sw2led.bde
Console
Files
Struc...
Resou...
Page 1/1
Figure 1.3 Connecting the eight switches to the eight LEDs

Compiling the file *sw2led.bde* generates the Verilog file *sw2led.v* shown in Listing 1.2. Alternatively, by selecting the hardware description editor (HDE) the module statement and port declarations are automatically generated but you will need to write your own *assign* statement. This can lead to the simpler Verilog program shown in Listing 1.3 where we have combined the module statement and port declarations in a single module statement that conforms to the 2001 Verilog standard. This format makes it easier to see the input and output signals. We can also write a single *assign* statement to replace the two *assign* statements in Listing 1.2. It is unnecessary to define the intermediate bus *BUS7*[7:0] and because *sw* and *ld* are the same size we don't need to include the [7:0] in the *assign* statement.

Listing 1.2 sw2led.v

```
// Title       : sw2led
module sw2led (sw,ld) ;

// ------------ Port declarations --------- //
input [7:0] sw;
wire [7:0] sw;
output [7:0] ld;
wire [7:0] ld;

// ----------- Signal declarations -------- //
wire [7:0] BUS7;

// ----------- Terminals assignment --------//
//            ---- Input terminals ---         //
assign BUS7[7:0] = sw[7:0];

//              ---- Output terminals ---        //
assign ld[7:0] = BUS7[7:0];

endmodule
```

Listing 1.3 sw2led2.v

```
// Title       : sw2led2
module sw2led2 (
input wire [7:0] sw ,
output wire [7:0] ld
) ;

assign ld = sw;

endmodule
```

In Parts 2 and 3 of the tutorial in Appendix A we show how to synthesize, implement, and download the design to the FPGA board. In summary, the steps you follow to implement a digital design on the BASYS or Nexys-2 board are the following:

1. Create a new project and design name.
2. Using the BDE create a logic diagram.
3. Save and compile the *.bde* file.
4. Optionally simulate the design (see Example 2).
5. Synthesize the design selecting the Spartan3E family and the 3s100etq144 device for the BASYS board and the 3s500efg320 device for the Nexys-2 board.
6. Implement the design using either *basys2.ucf* or *nexys2.ucf* as the custom constraint file. Check *Allow Unmatched LOC Constraints* under *Translate* and uncheck *Do Not Run Bitgen* under *BitStream*. Select *JTAG Clock* as the start-up clock under *Startup Options*.
7. Use *ExPort* to download the *.bit* file to the FPGA board.

At this point the switches are connected to the LEDs. Turning on a switch will light up the corresponding LED.

Problem

1.1 The four pushbuttons on the BASYS and Nexys-2 boards are connected to pins on the FPGA using the circuit shown in Fig. 1.4. The value of *R* is 4.7 kΩ on the BASYS board and 10 kΩ on the Nexys-2 board. When the pushbutton is up the two resistors pull the input down to ground and the input *btn*(*i*) to the FPGA is read as a logic 0. When the pushbutton is pressed the input is pulled up to 3.3 V and the input *btn*(*i*) to the FPGA is read as a logic 1. Create a *.bde* file using Active-HDL that will connect the four pushbuttons to the rightmost four LEDs. Compile and implement the program. Download the *.bit* file to the FPGA board and test it by pressing the pushbuttons.

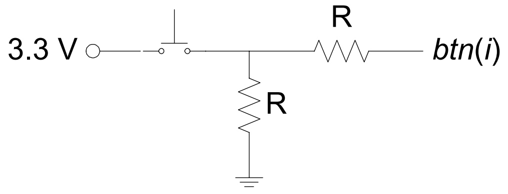


Figure 1.4 Pushbutton connection