WebFPGA: Connecting the Lattice FPGA to External Devices

In any FPGA design environment, you need a way to tell the logic compiler which external pins on the FPGA should be connected to the ports (i.e., the input and output variable names) in the Verilog module. For example, suppose we want to turn on the built-in yellow LED on the WebFPGA board when the on-board user pushbutton is pressed. If you look at the schematic for the board, you can see that the white (user) pushbutton is connected to signal IOT_51a which is pin 42 on the FPGA, and the yellow LED is connected to signal IOT_42b, which is pin 31. The yellow LED is on when that FPGA pin is low.

In the WebFPGA environment, the on-board resources are assigned special port names which map the names to the correct pins. The available resource names are:

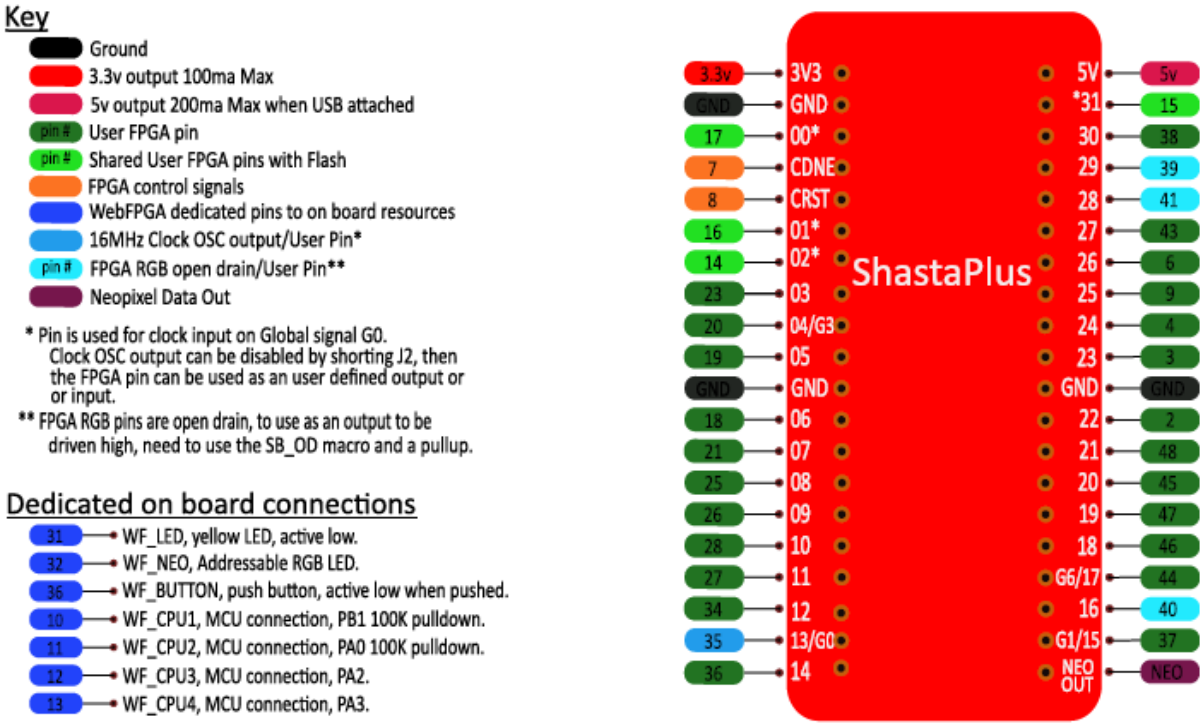
```
Name        FPGA pin
---------------------
WF_LED      31              // yellow user LED (output - active low)
WF_CLK      35              // 16Mhz clock (input)
WF_BUTTON   42              // user pushbutton (input – low when pressed)
WF_NEO      32              // NEO Pixel LED (output)

WF_CPU1     11              // These are four connections to the
WF_CPU2     12              // on-board microcontroller.  You will
WF_CPU3     13              // not need to use them in 123/223
WF_CPU4     10
```

With these defined ports, we can write a Verilog module to turn the yellow LED on and off as we press the pushbutton:

```
//////////////////////////////////////////////////////////////////
// Company: Harvard University
// Engineer: David Abrams
//
// Create Date:   2020-07-27
// Module Name:   ButtonLED.v
// Project Name:  fpga_top
// Target Device:  WebFPGA
// Description: Verilog design to control on-board resources
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments: Turns yellow led on when white button pressed
//
//////////////////////////////////////////////////////////////////
```

```
module fpga_top (
  input WF_BUTTON,
  output WF_LED
);

  assign WF_LED = WF_BUTTON;

endmodule
```

This works for the on-board resources, but the WebFPGA board is not very useful if you cannot connect it to external devices. The WebFPGA board has 32 pins labeled "00" through "31," most of which can be used for general purpose external I/O (input or output). Note, however, that a few of the pins have multiple uses. Unless you are short of I/O pins, it is best to avoid using the dual-use pins (except G1 through G6).[1]

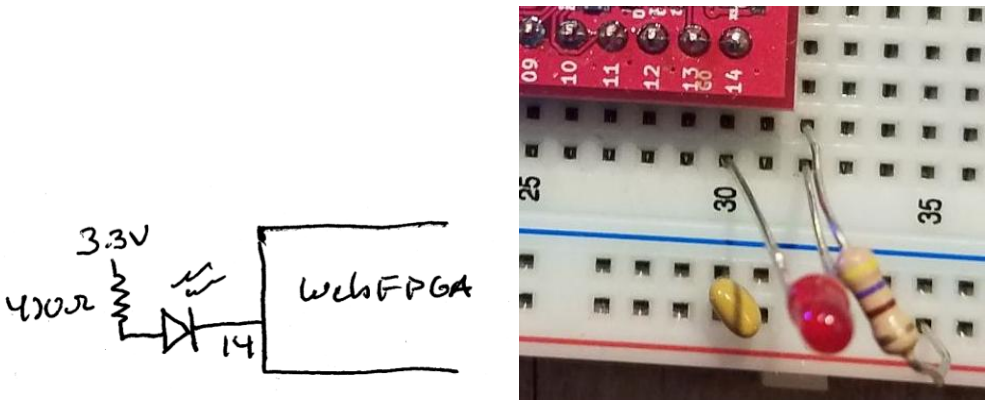


Suppose instead of the on-board LED, you wanted the white pushbutton to control an external LED connected between +3.3V and the WebFPGA board pin labeled "14" through a current limiting 470Ω resistor?[2]

---

[1] Pins 00, 01, 02 and 31 are used for SPI communications with the on-board microcontroller; pin 13 is connected to the 16Mhz clock; pins 16, 28 and 29 have open drain outputs -- see Appendix A below to use them. While the pins labeled "XX/GX" have input clock buffers connected, they can be used for normal I/O (except for pin 13/G0, which is used for the 16MHz clock input).

[2] Board pin 14 is the bottom left pin looking at the board with the USB connector up.



Pin 14 on the WebFPGA board is connected to signal IOT_48b which is pin 36 on the FPGA.

In most development environments, a separate Physical Constraints File (.pcf) is used to describe the mapping between FPGA pin numbers and ports of the Verilog module. However, since the WebFPGA environment does not support a user supplied .pcf file, it provides an alternative way to assign ports to pins. Assume we will call this port "RED_LED" in the Verilog module.[3] In a *comment* of your Verilog code, include the line:

```
// @MAP_IO RED_LED 14
```

This tells the compiler to assign the identifier "RED_LED" *to the FPGA pin connected to pin 14 on the board*. You can @MAP_IO to any pin number between 0 and 31. However, only signals that are on the top most module can become external FPGA signals.

There is one other WebFPGA compiler directive:

```
// @FPGA_TOP ButtonLED
```

Normally, the compiler expects the top level module to be named "fpga_top." This directive allows you to use more descriptive module names (a good idea). For example, here is the button controlled external LED module ready to compile for the WebFPGA board:

```
//////////////////////////////////////////////////////////////////////////////////
// Company: Harvard University
// Engineer: David Abrams
//
// Create Date:   2020-07-27
// Module Name:   ButtonLED.v
// Project Name:  ButtonLED
// Target Device:  WebFPGA
// Description: Verilog design to connect to external hardware
//
```

[3] The WebFPGA documentation recommends assigning ports connected to external pins a name in ALL_CAPS. While not required, it is a good idea to distinguish externally connected ports.

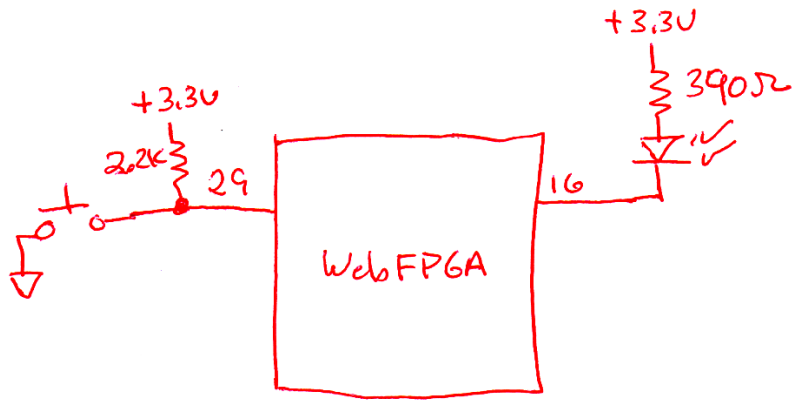
```
// Revision:
// Revision 0.01 - File Created
// Additional Comments: Uses LED connected to pin 14 of board
//////////////////////////////////////////////////////////////////////////////////

//*********************************************
// The comments in this block are directives to
// the WebFPGA environment.
//
// Set name of top level module (default is fpga_top)
// @FPGA_TOP ButtonLED
//
// Command to connect the named signal below
// to a specific WebFPGA output pin.
// @MAP_IO RED_LED 14
//*********************************************

module ButtonLED (
  input WF_BUTTON,
  output RED_LED
);

  assign RED_LED = WF_BUTTON;

endmodule
```

The compiler sees the @MAP_IO directive and assigns “RED_LED” to the FPGA pin (pin 36) connected to pin 14 on the WebFPGA board. One nice thing about placing the directives in comments is that you can use EDAPlayground to see if your code will compile, as EDAPlayground will ignore the comments. If you create a textbench, you can also see if your module works. Here is the EDAPlayground simulation of the ButtonLED design with a simple testbench that presses and releases the button. Press “Run” to see the simulation. (Since both the button and the LED are active low, you should see the output follow the input.)

Appendix A

Pins 16, 28 and 29 have open drain outputs and are designed to drive LEDs. They can be used as normal inputs and outputs (with a pullup resistor) if the SB_IO_OD module is included in your Verilog as follows:

```
// opendrain.v - turns on led when button is pressed
//
// @MAP_IO pin_out 16    // opend rain output - connected to LED and 390 ohm to +3.3V
// @MAP_IO pin_in 29     // connected to pushbutton switch

module fpga_top (
  input pin_in,
  output pin_out
);

wire mywire;             // connects pin together

SB_IO_OD #(
  .PIN_TYPE(6'b000001),
  .NEG_TRIGGER(1'b0)
) pin_in_driver (
  .PACKAGEPIN(pin_in),
  .DIN0(mywire)
);

SB_IO_OD #(
  .PIN_TYPE(6'b011001),
  .NEG_TRIGGER(1'b0)
) pin_out_driver (
  .PACKAGEPIN(pin_out),
  .DOUT0(mywire)
);

endmodule
```



mru 2020-10-29 dea