LED Manager: Additions to BSP and Toolflow

This document outlines the debugging functionality added to the current Board Support Package (BSP) for SKARAB and the CASPER Toolflow. The functionality makes use of the SKARABs front-panel LEDs, four green and four red. *Figure 1* illustrates the grouping of the LEDs (in rows) according to their BSP functionality along with their various states: 🟢 Solid Green; ✳ Flashing Green; 🔴 Solid Red. Throughout development of these additions it was decided that the various status indicators would take on one value at a time (i.e. One-hot encoding); therefore one LED can be on at any stage for each row of indicators. Lastly, the additions made to the Toolflow allows the Simulink model to drive the front-panel LEDs as well. Switching control between the BSP functionality and the DSP Design can be done via casperfpga.

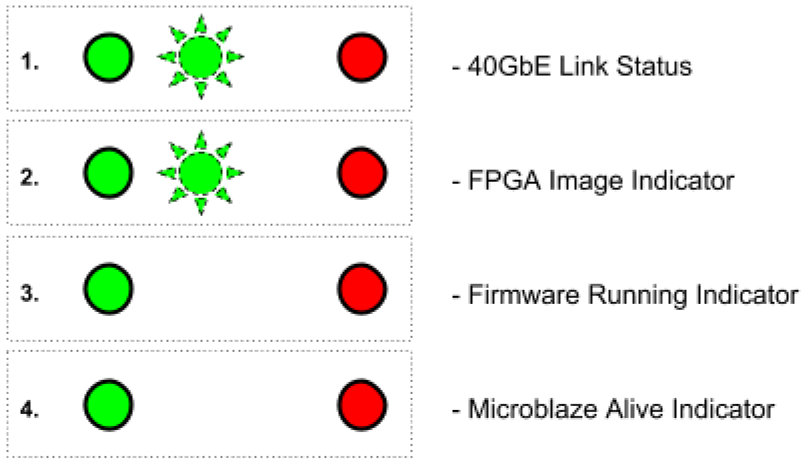

Figure 1: Front-panel LEDs and corresponding assignment

BSP Functionality

As per the grouping shown in *Figure 1*, the states of various LEDs are described as follows.

Row 1: 40GbE Link Status

The first row of LEDs are used to indicate the status of the 40GbE link(s) on its corresponding Mezzanine card. Their states are described as follows:

1. Solid RED:
    a. 40GbE Link(s) are **down**.
2. Flashing GREEN:
    a. 40GbE Link(s) are **up**, but the SKARAB has not been given an IP address/DHCP has not resolved.
3. Solid GREEN:
    a. 40GbE Link(s) are **up**, and DHCP has completed for the SKARAB.

The logic behind sensing the status of the 40GbE link(s) is done by putting each of the four PHY's link-up signals through an OR gate. It is worth noting that, in the current SKARAB BSP and CASPER Toolflow, the 40GbE interface only allows for/is configured for the first (zeroth, *Figure 2*) 40GbE link. Further additions and development will result in a change in this "link-sensing logic"; however for now the logic will stay as-is.

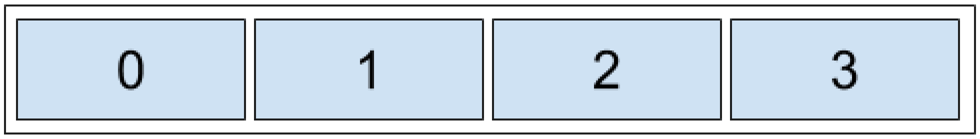

Figure 2: 40GbE Mezzanine Card QSFP Input Port Numbers

Row 2: Firmware Image Indicator

The second row of LEDs are used to indicate the status of the firmware image running on the Virtex7 FPGA. Their states are described as follows:

1. Solid RED:
    a. **Golden** Image.
        i. i.e. The image that the SKARAB falls back to if all else fails. This is the base BSP with no DSP functionality, running at 156.25 MHz clock frequency. Most notably, its boot parameters are different to that in the Multiboot image in that it boots up at a slower clock frequency to ensure the firmware comes up correctly.
2. Flashing GREEN:
    a. **Multiboot** Image.
        i. This image is similar to the Golden Image, except with a few changes in boot parameters. A SKARAB will <u>always</u> boot up into the Multiboot Image, whether it be from a reset or a power cycle.
3. Solid GREEN:
    a. **Toolflow** Image.
        i. This image is one generated by the Toolflow, and will often involve some form of DSP design. It will almost inevitably use more of the FPGAs resources, and utilise its peripherals/mezzanine cards.

Row 3: Firmware Running Indicator

The third row of LEDs are used to indicate the status of the firmware image running on the Virtex7 FPGA. The process that implements this functionality continually writes a HIGH ('1') to a flag on each rising-edge of the clock. This process runs at the system clock frequency (*sys_clk)* of 156.25MHz. Therefore it can be safely assumed that if the firmware isn't running the process will not be running, and **all** the LEDs will be <u>off</u>. That being said, a case of the firmware being up but held in reset could be attributed to an issue with the clock itself.

The states of the LEDs corresponding to this functionality is as follows:

1. Solid RED (Basically, off):
    a. Firmware **not running**.
2. Solid GREEN:
    a. Firmware **running**.

Row 4: Microblaze Alive Indicator

The fourth row of LEDs are used to indicate the status of the Microblaze running on the SKARAB's Virtex7 FPGA. This process is implemented using a free-running counter running in the LED Manager entity, and its corresponding reset being triggered by the Microblaze itself. Essentially, the free-running counter relies on the Microblaze to trigger a counter-reset in order to 'prove' that it is still alive. If it doesn't, the counter will overflow which the process interprets as 'Microblaze down'. The counter itself is a 33-bit counter whose overflow corresponds to ± 55 seconds; whereas the Microblaze triggers the counter-reset as often as its main-loop executes (in the order of nanoseconds). Their states are described as follows:

1. Solid RED:
    a. Microblaze **down**.
2. Solid GREEN:
    a. Microblaze is **up** and responding.

Toolflow Additions

Changes have been made to the following files in the Toolflow to accommodate for switching between the control of LEDs via the BSP and the DSP Design:

1. Yellow Blocks
    a. gpio.py
    b. forty_gbe.py, which contains the BSP
2. HDL Sources
    a. gpio_simulink2ext.vhd
    b. forty_gbe.vhd
    c. led_manager.vhd (created)
    d. parameter.vhd

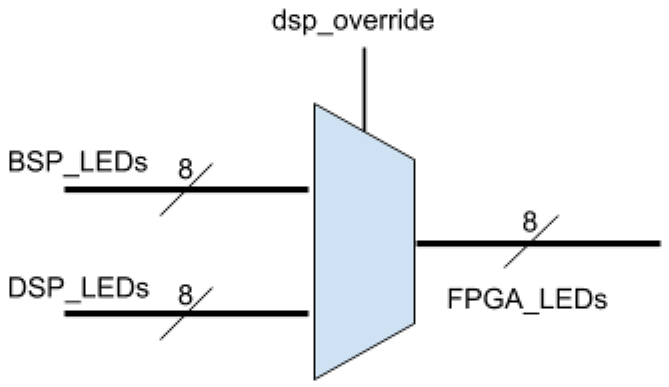

Figure 3: Multiplexing Logic for switching LED signals

The changes revolved around GPIO yellow blocks being defaulted to synthesise their output signal to pins on the FPGA. This is not desired in the case of the ‘LED Manager’, where the signals driving the LEDs can be from either the BSP or the DSP Design. *Figure 3* depicts the basic logic behind the ‘LED Manager’ entity; the signals provided by the BSP and DSP Design are multiplexed for the final output with a ‘DSP_override’ select-signal. The dsp_override signal is the zeroth-bit of a wishbone register at address 0x34, mapped in parameter.vhd for ease of reference.

It is also worth mentioning that as the DSP_LEDs are being used to drive something in the BSP’s clock domain, the signals themselves are triple-registered in order to cross clock domains. Often the clock frequency chosen for the DSP Design is higher than that of the BSP, hence the need for proper clock domain crossing.