Mercury FPGA module Reference Manual

micronova
www.micro-nova.com

OVERVIEW

Mercury is a compact FPGA development module designed for easy project integration. Its small size and dual-inline-pin (DIP) form-factor make it easy to use in a small project on a prototyping board. *Mercury* is a complete solution, contains all the necessary support circuitry for the FPGA, along with a convenient mini-USB programmer interface, 8-channel analog capability, external SRAM, and 5-volt tolerant I/O.

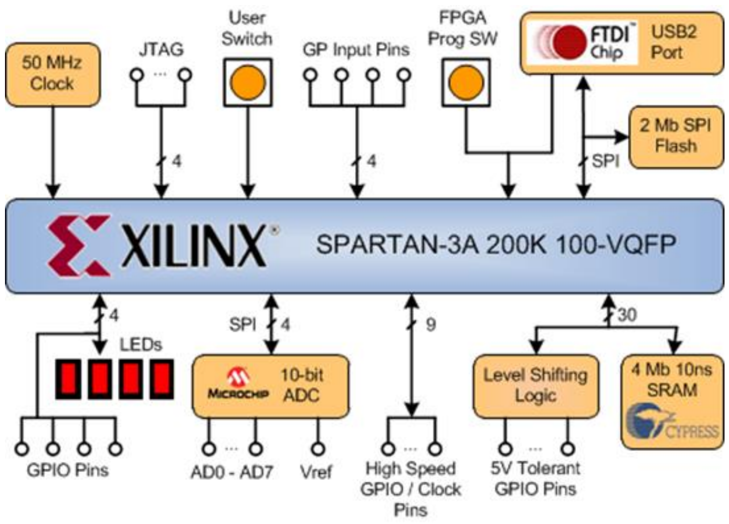
Mercury contains a powerful Xilinx Spartan 3A FPGA with 200,000 gates on 3 in x 1 in board. This, combined with the fast, simplistic interface of the 4 Mb SRAM, provides the perfect platform for rapid FPGA implementation of complex projects is super small spaces.

Mercury contains circuitry to interface with a wide range of external components. Digital I/O on *Mercury* includes level shifting logic to take away the pain of interfacing with 5V logic level devices. Eight channels of analog data can be read using the onboard ADC. The external VREF pin of the ADC removes the need for complex analog scaling circuitry.

Mercury is designed to make building with FPGAs quick, cheap, and easy. The included USB cable is used for power and programming – no other hardware is needed to start designing with *Mercury*. This manual, along with the resources at www.micro-nova.com, provide the documentation and code examples necessary to quickly understand and use the features onboard *Mercury*.

FEATURES

- Xilinx Spartan-3A FPGA (XC3S200A)
  - 200K-gate equivalent
  - 28 Kb distributed RAM
  - 16 dedicated 18x18 multipliers
  - 4 digital-clock managers (DCM)
  - 288 Kb block RAM
- Onboard USB interface for device configuration, debugging and power
- 2 Mb SPI configuration/user flash
- MCP3008 8-ch, 200 KSPS, 10 *bit ADC*
- 30 5-volt tolerant IO pins
- 9 FPGA-direct high speed IO pins
- 4 user LEDs with IO pins
- 1 user button, 1 reset button
- 4 Mb (512 K x 8 bit) SRAM



POWER SUPPLIES

Mercury requires only a single 5V power supply. Power can be supplied either though the +5V power pin (pin 64) or via USB. Acceptable external power supply range is 4.5V – 5.5V max. Voltages higher than 5.5V on the +5V power connector may damage the board. *Mercury* draws an average of 60mA, but can draw up to 275mA depending on the current FPGA configuration. When the USB cable is first connected, the lights on *Mercury* will flash on and off quickly. This is expected. Immediately after connection, the FTDI USB interface chip onboard *Mercury* will reconfigure the USB interface to provide the extra current required by *Mercury.* During this reconfiguration, the 5V board power supply is briefly disconnected and reconnected.

Note that the USB power rail and the external power rail are connected through J3. Power can be provided through both USB and the external power supply pin simultaneously. However, large voltage differences between the supplies could cause equipment damage.

The 5V power rail is used to power the ADC, USB chip, and bus switches, as well as the 3.3V and 1.2V voltage regulators. The 3.3V regulator powers the FPGA and remaining devices. 3.3V power is also provided for external devices on pin 63 - up to 225mA. This pin is a power output only! Do not connect a power supply to this pin. The 1.2V regulator is used for internal FPGA power only and is not available externally. Mercury's onboard regulators have been purposely oversized to support maximum current requirements across a wide temperature range.

CONFIGURATION

After power-on, Mercury's FPGA must be configured before it can perform any useful function. The FPGA is essentially a blank slate of memory cells and programmable circuit interconnects. A configuration, or *.bit, file needs to be loaded to the FPGA to map logic functions to memory cells and configure circuit interconnects.

Xilinx provides a free tool, ISE WebPACK, which can be used to create a configuration bit file from compiled VHDL or Verilog source files.

The bit file can be loaded directly to the FPGA over JTAG or it can be loaded to Mercury's onboard non-volatile FLASH chip. Any configuration loaded over JTAG is lost if board power is lost or if an FPGA configuration cycle is initiated. The FPGA will automatically load any valid bit file found in the FLASH chip during a configuration cycle. The FPGA will initiate a configuration cycle on power-up or when the PROG pin is pulled low a minimum of 0.5 µs.

Loading the bit file directly to the FPGA is a quick, easy way to test a new design. A JTAG programming cable can be connected to the JTAG interface pins on *Mercury* as shown in figure 1. Xilinx ISE WebPACK provides a software tool, iMPACT, which can be used to write a configuration bit file over JTAG to Mercury's FPGA.

Next to the JTAG pins is the PROG pin. This pin is connected to PROG_B on the FPGA. Below the JTAG interface are two buttons. The first, S01, is labeled USER. This button is connected to an FPGA input pin and serves as a general purpose user switch. The second, S02, is labeled PROG. This button is also connected to the FPGA's PROG_B pin. Refer to figure 2 for the detailed PROG schematic. Note that the FTDI USB interface chip is also connected to the PROG_B pin on the FPGA.

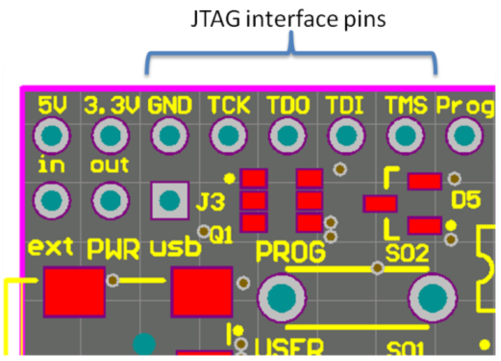


Figure 1 - *Mercury* JTAG interface and PROG pin

An FPGA configuration cycle can be initiated either by depressing S02, by grounding the PROG pin, or by driving D5 low on the FTDI USB interface chip. While PROG is low, all user-I/O pins, input-only pins and dual-purpose pins are high impedance with an internal pull-up resistor.

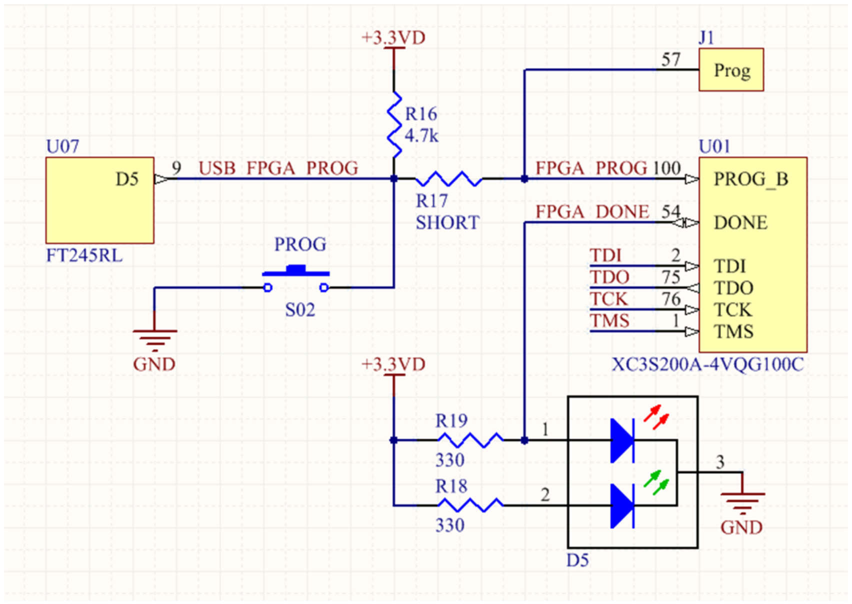
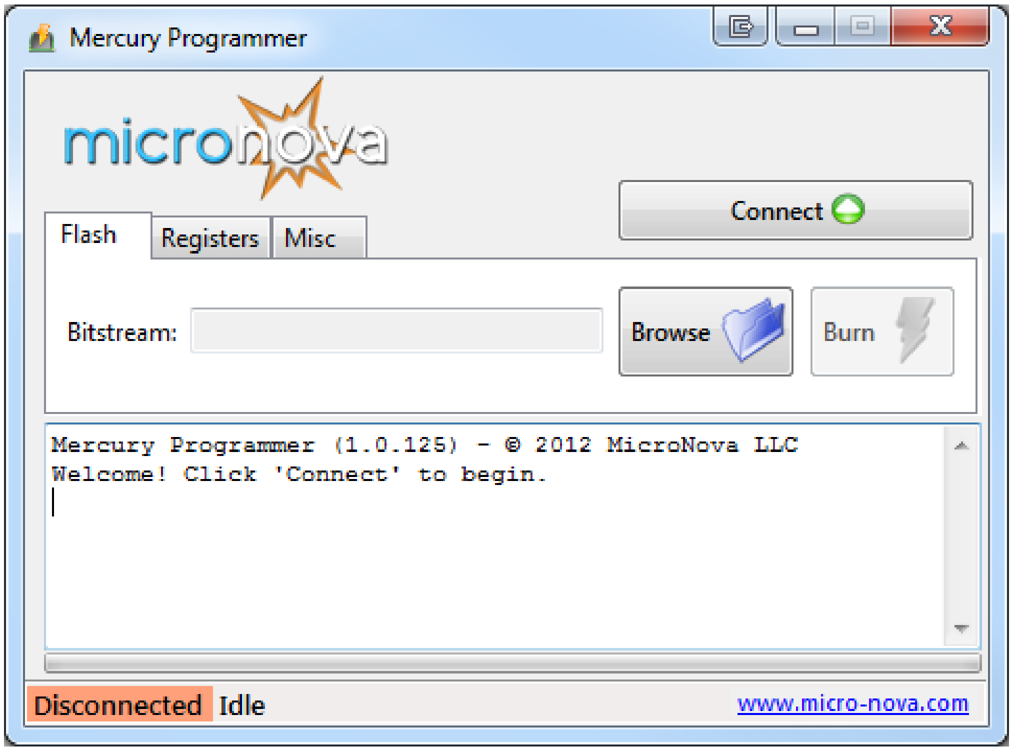


Figure 2 - *Mercury* FPGA PROG Circuit

MicroNova provides a Free, PC based program, called the Mercury Programmer, which can be used to transfer a bit file to Mercury's onboard FLASH chip over USB. The Mercury Programmer uses FTDI's D2XX drivers to communicate with Mercury's onboard FTDI USB chip - the FT245RL. While writing the configuration bit file to the FLASH chip, the Mercury Programmer will drive pin D5 low on the FT245RL, thereby holding PROG_B low. This will tristate all FPGA I/O pins, giving the FT245RL chip exclusive access to the FLASH chip over the shared SPI interface.

Directly above the PROG button (S02) is a multi-colored LED (D5). When the board has power, the LED will glow green. Once the FPGA has successfully loaded the configuration bit file, the FPGA_DONE pin will go high, which will illuminate the red section of the LED as shown in the schematic in figure 2.



```
Mercury Programmer (1.0.125) - © 2012 MicroNova LLC
Welcome! Click 'Connect' to begin.
```

Disconnected Idle www.micro-nova.com

Figure 3 - Mercury Programmer Application

After FPGA configuration, the flash SPI lines are available as user I/O. These lines can be used by the FPGA or FTDI USB chip to access the SPI flash chip to read/write user data, or to allow communication between the PC and the FPGA or FLASH chip over USB.

SPI INTERFACE - FLASH, USB, FPGA

Mercury is equipped with a 2 Mb SPI Flash chip. The four lines that make up the SPI interface for the flash chip (CS, MOSI, MISO, and SCLK) are shared between the FPGA and the FTDI USB chip. Care must be taken to be sure that only one device is driving these lines at a time. One way to accomplish this is to drive the FPGA PROG_B line low while interfacing between the FTDI and the FLASH. Pulling the FPGA PROG_B pin low will tristate the FPGA's flash interface pins.

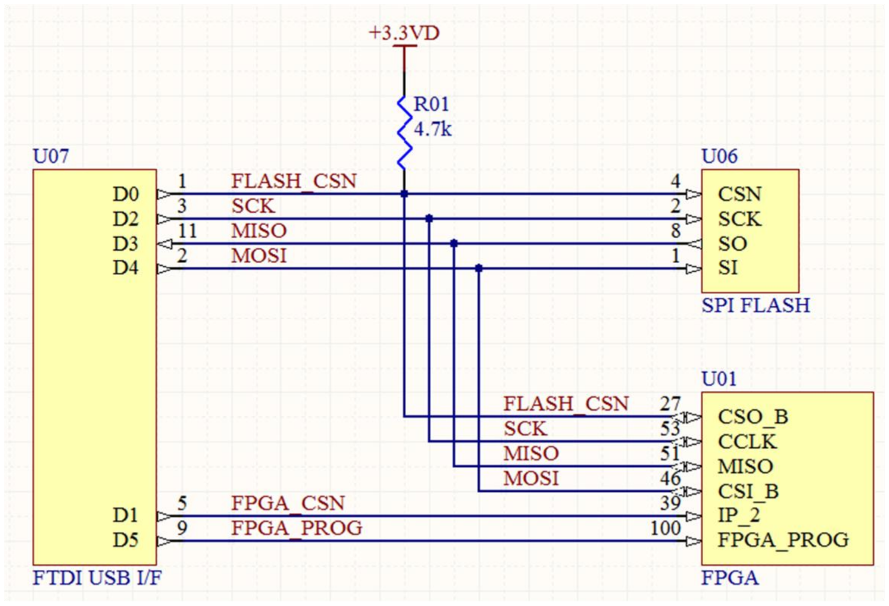


Figure 4 - FPGA, FLASH, FTDI USB SPI interface

Figure 4 shows the connections between the three chips. This connection scheme allows for three modes of SPI communication:

1. Between the FTDI USB chip and the FLASH to read/write configuration bit files and/or user data using FLASH_CSN, SCK, MISO, and MOSI. Be sure to hold FPGA_PROG low.
2. Between the FPGA and the FLASH during configuration time to load the bit file, and after configuration to allow reading/writing of user data to the FLASH using FLASH_CSN, SCK, MISO, and MOSI. Be sure to configure the FTDI data lines used for the SPI interface as inputs during this time.
3. Between the FTDI chip and the FPGA using FPGA_CSN, SCK, MISO, and MOSI. Be sure to release FLASH_CSN or drive it high on the FPGA.

The first mode is used by the Mercury Programmer to write the configuration bit file from the FTDI USB chip to the FLASH over a software driven SPI interface. Exclusive access to the FLASH is guaranteed by holding PROG_B low. The pins on the FTDI USB chip used for SPI communication (D0,D2,D3,D4) should be configured as inputs when not in use to provide the FPGA with exclusive FLASH chip access.

The second mode is used by the FPGA to load the bit file during configuration. This procedure is detailed in the Xilinx user's guide UG332, starting on page 102. Once the FPGA is configured, the SPI interface can be used by the FPGA to access the FLASH. In order to communicate with the FLASH, the FPGA will need to include a SPI bus master interface as part of the FPGA configuration. The SPI bus master interface should delay a few clock cycles before driving SCK to be sure the configuration cycle has time to complete. Note that the FPGA configuration bit file will use the first 1,196,128 bits of the FLASH, any user data should come after this. If the FLASH chip is not used as part of the FPGA configuration, the FLASH_CSN pin should either be tristated or driven high.

The third SPI communication option can be used to setup a serial communication channel between the PC and the FPGA. A custom PC application can be written to communicate with the FTDI chip over USB using the DLL file provided by FTDI's website. The PC application can use the FTDI flash chip to send and receive data over a software driven SPI interface to and from the FPGA. An SPI slave interface is also needed within the FPGA configuration to send and receive data from the FTDI chip. Have a look on www.micro-nova.com for example designs highlighting this feature.

5V TOLERANT I/O

Mercury has 30 pins of 5-volt tolerant, general-purpose I/O. Series resistors of 150-ohms are present on these 30 pins for current limiting and static suppression.

The on-board clamping circuitry ensures that the FPGA I/O pins never see voltages in excess of 3.3V.

Banks of SN74CBTD buffers are used to clamp the input voltage down from 5V (or greater) to 3.3V. These buffers consist of a FET with the gate driven by 5V through a diode. This yields a gate voltage of 4.3V. The V_{GS} drop across the FET is 1V. Thus, 5V presented externally is clamped to 3.3V.

A nice side benefit of using the SN74CBTD between the FPGA and external I/O is its bus switching capabilities. FPGA pin 3 (BUSSW_OEN) has been tied to the output enable pin of Mercury's bus switches. This pin is active low. This pin has a pull-down resistor. Therefore, by default the buss switches will be closed, connecting the external I/O pins to the FPGA. Driving this pin high will open the switches, disconnecting the external I/O pins from the FPGA and putting them in a high impedance state. This allows us to share the FPGA I/O pins between external I/O and the SRAM memory. The SRAM also has an active low chip enable pin (MEM_CEN) that is connected to pin 30 on the FPGA. This pin is pulled low by an internal FPGA pull-up resistor, thereby disconnecting the SRAM from the FPGA I/O pins by default.

When interfacing to 5V logic, it is important to determine whether the device requires 5V CMOS logic levels or 5V TTL logic levels. The level shifting logic onboard *Mercury* is made to handle 5V inputs. However, Mercury's I/O pins output at 3.3V logic levels. For 5V TTL logic, this is not a problem; any voltage higher than 2V is interpreted as logic '1'. However, for 5V CMOS logic, the input voltage needs to be 3.5V or higher to be interpreted as a logic '1'. Therefore, when using *Mercury* to drive 5V CMOS inputs, use a pull-up resistor. The pull resistor should be connected between the I/O pin and +5V. Determining the pull-up resistance value is a tradeoff between current draw and I/O speed. Higher resistance values will draw less current, which could be very important when many pull-up resistors are used and/or for battery powered applications. However, higher resistance values will limit the bandwidth of the I/O pin. We recommend pull-resistances between 1K and 5K. Please refer to the following table for a listing of the general purpose I/O pins available on *Mercury.* Note that each FPGA pin has a corresponding GPIO pin and SRAM pin associated with it - use the memory chip enable (P59) and bus switch enable (P60) pins to switch FPGA pin connections between GPIO and SRAM.

Table 1 - GPIO Pins

GPIO	*SRAM*	*FPGA*
0	A0	**P59**
1	A1	**P60**
2	A2	**P61**
3	A3	**P62**
4	A4	**P64**
5	A5	**P57**
6	A6	**P56**
7	A7	**P52**
8	A8	**P50**
9	A9	**P49**
10	A10	**P85**
11	A11	**P84**
12	A12	**P83**
13	A13	**P78**
14	A14	**P77**
15	A15	**P65**
16	A16	**P70**
17	A17	**P71**
18	A18	**P72**
19	A19	**P73**
20	D0	**P5**
21	D1	**P4**
22	D2	**P6**
23	D3	**P98**
24	D4	**P94**
25	D5	**P93**
26	D6	**P90**
27	D7	**P89**
28	WEn	**P88**
29	(n/a)	**P86**
MEN_CEN (active low)		**P30**
BUSSW_OEN (active low)		**P60**

EXTERNAL MEMORY

Mercury is equipped with an external 4 Mb high performance CMOS Static RAM module organized as 512K words by 8-bits.

The SRAM bus is shared with the level-shifted GPIO bus. To achieve this sharing, the enable lines for the SRAM and the output buffers are independently controllable. Refer to the previous section for more information on GPIO / SRAM pin sharing. Note that address line A19 is not used internally by the SRAM chip.

Interfacing to the SRAM is done asynchronously with read and write cycle times of 10 ns. To read from the device, perform the following steps:

1. Drive BUSSW_OEN (P60) high to disconnect the external GPIO pins from the FPGA.
2. Drive MEN_CEN (P59) low to enable the SRAM while driving WEN (P88) high to disable write mode.
3. The contents of the memory location specified by A18 down to A0 will appear on D7 down to D0.

To write from the SRAM, perform the following steps:

1. Drive BUSSW_OEN (P60) high to disconnect the external GPIO pins from the FPGA.
2. Drive MEN_CEN (P59) low to enable the SRAM while driving WEN (P88) low to enable write mode.
3. Data on D7 down to D0 will be written to memory at the location specified by A18 down to A0.

For more information, please refer to the manufacture's data sheet. *Mercury* uses a Cypress SRAM, part number CY7C1049DV33.

LEDS & USER SWITCH

Four LEDs are available on-board. These pins are also available on the DIP package as general-purpose I/O. A single push button is available on-board for use in user logic.

Table 2 - LED and User Switch Pins

Device	*FPGA*
LED 0	**P13**
LED 1	**P15**
LED 2	**P16**
LED 3	**P19**
User Button	**P41**

DIRECT I/O

There are nine input/output pins that correct directly to the FPGA without any additional level-shifting circuitry or series resistors. Two of these pins are available as global clocks within the FPGA. An on-board 50MHz oscillator is also available as an input to the FPGA. Four additional pins are available as direct input-only pins. Direct input and output pins can be used for high bandwidth interfaces between external devices and the FPGA.

Table 3 - Direct I/O Pins

GPIO	*FPGA*	*Comment*
Direct 0	**P20**	I/O
Direct 1	**P32**	I/O
Direct 2	**P33**	I/O
Direct 3	**P34**	I/O
Direct 4	**P35**	I/O
Direct 5	**P36**	I/O
Direct 6	**P37**	I/O
Input 0	**P68**	Input only
Input 1	**P97**	Input only
Input 2	**P7**	Input only
Input 3	**P82**	Input only
Clock 0	**P40**	Clock I/O
Clock 1	**P44**	Clock I/O
50MHz osc	**P43**	Onboard clock

ANALOG-TO-DIGITAL CONVERTER

Mercury is equipped with an onboard Microchip MCP3008 serial ADC. It can sample 8-channels with 10-bit resolution, at a maximum sample rate of 200-ksps.

The ADC can sample voltages in the range of 0V to 5V. The dynamic range is modifiable using the external VREF pin.

Communication with the MCP3008 is achieved using SPI. A sample driver in VHDL can be found on our website.

Table 4 - ADC Interface Pins

Device	*FPGA*
ADC SCK	**P9**
ADC MOSI	**P10**
ADC MISO	**P21**
ADC CSN	**P12**