MachXO2 Low Power Control Demo

User's Guide

April 2012
UG58_01.0

Introduction

There are many power saving features in the MachXO2™ family of devices, such as bandgap, on-chip oscillator, PLL, etc. The purpose of this demo design is to show how to save power by turning off the on-chip oscillator. The on-chip oscillator is used for configuration when the device is powered on. After configuration is complete, the on-chip oscillator is on by default but the user has the option of turning it off in order to reduce power consumption. Turning on/off this on-chip oscillator is the focus of this demo design since the resulting power savings is more significant than other power saving features. The states of ON and OFF are called wake and standby respectively and these terms will be used throughout this document.

Setup

This section describes how to set up the MachXO2 Low Power Control Demo. There are two different bitstreams: User Mode and Configure Mode. In User Mode, the power saving is controlled through user logic. In Configure Mode, the power saving is controlled through either I^{2}C or SPI hard core slave. An external I^{2}C and SPI master can be used to control the power saving through the I^{2}C or SPI bus. Another option is to set up two MachXO2 Pico Evaluation Boards, one programmed with the Configure Mode bitstream and the other used as an I^{2}C/SPI master. The I^{2}C/SPI master design for MachXO2 devices is briefly mentioned in this document but is not the focus of this demo.

Required Demo Components

- MachXO2 Pico Evaluation Board(s)
- Mini USB cable (included in MachXO2 Pico Development Kit)
- I^{2}C and SPI master
- PC
- Wires as I^{2}C or SPI bus

Bitstream Location

Unzip the demo design package. The bitstreams for User Mode and Configure Mode are located in the following folders respectively:

\power_control\user

\power_control\config

The I^{2}C/SPI master design on the MachXO2 bitstream is located in the following folder:

\Pico_I2C_SPI_master\Hardware\Implementation\Diamond_1.4\XO2_Pico

MachXO2 Pico Evaluation Board Installation

Connect the MachXO2 Pico Evaluation Board to a PC with the USB cable.

Hardware Settings

For demo operation, there is no hardware configuration necessary other than connecting the USB cable for the User Mode demo. For other modes, the board may need some pins on connector U3 to be soldered. Those pins are mentioned in the following sections.

Demo Operation

All user interaction and demo control is either through a push-button on the MachXO2 Pico Evaluation Board, standalone I^{2}C/SPI master or I^{2}C/SPI master on another MachXO2 Pico Evaluation Board. This section will discuss the expected behavior from the MachXO2 Pico Evaluation Board for the demos. Note that the value displayed on the LCD is for Icc and the units are mA. The value may vary from board to board. There is some logic clocked by the on-chip oscillator output. The following LCD display difference (2 vs. 15) is caused by the oscillator itself and the logic clocked by this oscillator.

User Mode Demo

Theory of operation for this power saving control mode: the oscillator is instantiated in the code and it has a port called STDBY. Once this port is set to ‘1’, the oscillator will be turned off to save power.

For this mode, power control is done through user logic.

Demo steps:

1. Connect the MachXO2 Pico Evaluation Board to a PC with the USB cable in the kit.
2. Locate and download the bitstream. The bitstream is located in the following folder: \power_control\user. Download the bitstream to the MachXO2 device.
3. The LCD display should be 02.0A (the oscillator is turned off).
4. Press **S1** and hold. The LCD display should be 15.0A (the oscillator is turned on).
5. Release **S1**. The LCD display should be 02.0A.

This completes the demo for User Mode.

Configure Mode Demo with a Standalone I^{2}C/SPI Master

For this mode, the power control is done through external I^{2}C or SPI bus. Table 1 lists the pins for I^{2}C and SPI on connector U3. Table 2 lists the possible ways to set the standby and wake modes. Note that the I^{2}C slave addresses in the table are the user’s choice when configuring EFB module in IPexpress™.

Table 1. I^{2}C and SPI Pins on Connector U3

Port	U3 Pin Label	Others
I^{2}C primary	SCL, SDA	Slave address: 0x08, 0x09
I^{2}C secondary	N8(SCL), M7(SDA)	Slave address: 0x0A
SPI	SO, SI, SCK	SN for chip select = ufm_sn A13 for chip select = spi_scsn

Table 2. Power Control Through External I^{2}C or SPI

Enter Mode	Action by External I^{2}C or SPI Master
Standby	Write 7D 02 00 to I^{2}C slave address of 0x08
	Write 7D 02 00 to SPI slave (chip select = ufm_sn)
Wake	Write FF to I^{2}C slave address of 0x08
	Read from I^{2}C slave address of 0x08
	Write FF to I^{2}C slave address of 0x09
	Read from I^{2}C slave address of 0x09
	Write FF to I^{2}C slave address of 0x0A
	Read from I^{2}C slave address of 0x0A
	Write FF to SPI slave (chip select = spi_scsn)
	Press and release the on-board push-button Write FF to SPI slave (chip select = ufm_sn)

Note: FF is not necessary and it is used to avoid unexpected I^{2}C or SPI response.

The theory of operation for this power saving control mode:

1. To put the device into standby mode, register bit 7D 02 00 needs to be set either through the I^{2}C or SPI port where, 0x7D is the register address to be written and bit 1 of the first data byte is the control bit for standby.
2. To put the device into wake mode, as long as one of the following occurs:
   a. One of the I^{2}C slave addresses matches (either read or write)
   b. One of the SPI chip selects becomes low

For this mode, assume the I^{2}C or SPI master is connected to the MachXO2 Pico Evaluation Board according to the pins listed in Table 1.

Demo steps:

1. Connect the MachXO2 Pico Evaluation Board to a PC with the USB cable in the kit.
2. Locate and download the bitstream. The bitstream is located in the following folder: \power_control\config. Download the bitstream to MachXO2 device.
3. Press and release **S1**.
4. The LCD display should be 15.0A (the oscillator is turned on).
5. Use one of the approaches in Table 2 to put the device into standby mode. The LCD display should be 02.0A (the oscillator is turned off).
6. Use one of the approaches in Table 2 to put the device into wake mode. The LCD display should be 15.0A.

This completes the demo for Configure Mode.

Configure Mode Demo with a Setup of Two MachXO2 Pico Evaluation Boards

Similar to the above mode, the power control is done through the external I^{2}C or SPI bus. The pinout and power control operations in Tables 1 and 2 still apply to the slave board. However, the I^{2}C or SPI master is implemented in another MachXO2 Pico Evaluation Board. Table 3 lists the pins for the I^{2}C and SPI masters.

Table 3. I^{2}C and SPI Master Pins on Connector U3

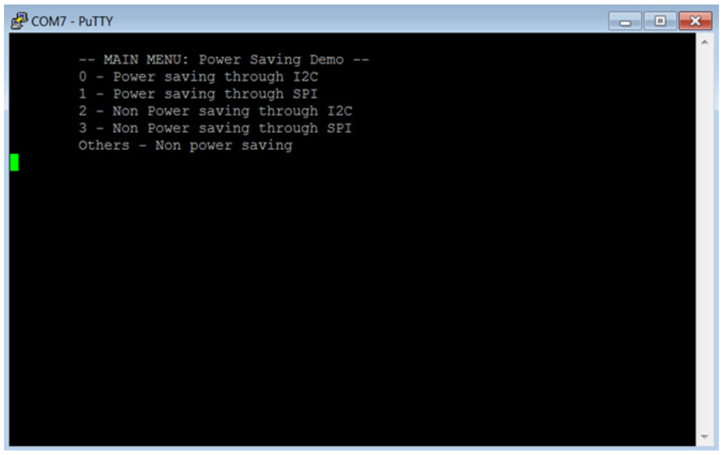
Port	U3 Pin Label
I^{2}C master	SCL, SDA
SPI master	SO, SI, SCK, CSS

For design details regarding the I^{2}C and SPI master on the board, please refer to the MachXO2 Master SPI/I^{2}C Demo. As shown in Figure 1, the UART terminal GUI on a PC has four options: 0, 1, 2 and 3. The user can strike the keys on a PC keyboard and observe the displayed values on the MachXO2 Pico Evaluation Board LCD.

The theory of operation for this power saving control mode:

1. The standby and wake modes of operation for the slave board are the same as for the Configure Mode discussed above.
2. For the master board, the communication between the PC and the master MachXO2 Pico Evaluation Board is UART through USB cable.
3. The keyboard action on the PC is sent to the master board through the UART and is translated into the I^{2}C or SPI transactions listed in Table 2.

Figure 1. UART GUI



```
-- MAIN MENU: Power Saving Demo --
0 - Power saving through I2C
1 - Power saving through SPI
2 - Non Power saving through I2C
3 - Non Power saving through SPI
Others - Non power saving
```

For this mode, assume the PC has at least two USB ports and both the master and slave boards are programmed with their respective bitstreams.

Demo steps:

1. Connect both boards to a PC with the USB cables in the kit.
2. The LCD display should be 15.0A (the oscillator is turned on).
3. Connect the boards to each other with both I^{2}C and SPI buses according to the pins listed in Tables 1 and 3. For I^{2}C, connect U3 pins SCL to SCL, and SDA to SDA. For SPI, connect master board U3 pins SI, SCK and CSS to slave board U3 pins SI, SCK and SN, respectively (SO is not used).
4. Open the UART terminal GUI according to the MachXO2 Master SPI/I^{2}C Demo instructions.
5. Strike key 0 or 1 on the PC keyboard to put the device into Standby Mode. The LCD display should be: 02.0A (the oscillator is turned off).
6. Strike key 2 or 3 on the PC keyboard to put the device into Wake Mode. The LCD display should be: 15.0A.

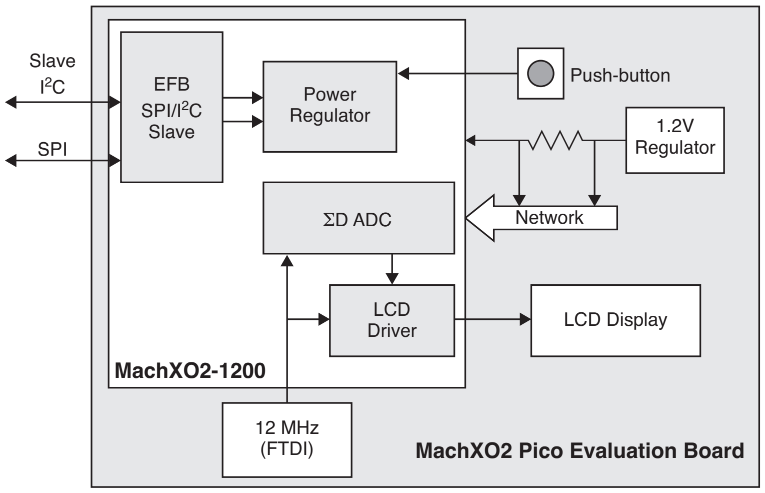
This completes the demo for this mode.

Demo Design Architecture

Overview

The demo project is based on the MachXO2 Pico Evaluation Board. As shown in Figure 2, power control can be done either through on-board push-buttons or through an external I^{2}C or SPI master. There is an on-board 1 ohm resistor and the voltage across this resistor goes to the sigma-delta ADC design in the MachXO2 device. The output data from the sigma-delta ADC module is then converted and displayed on the LCD. For the User Mode demo, the human interface for power saving control is through push-buttons. For the Configure Mode demo, the human interface for power saving control is either through a standalone I^{2}C/SPI master or through the UART terminal GUI on a PC. More design details concerning this demo are provided in the following sections.

Figure 2. Demo Design Block Diagram



MachXO2 Lattice Diamond® Project

There are two implementations in the project: user and config for User Mode and Configure Mode respectively. Those two implementations share the same .lpf file. The Diamond project and file structure look like the following:

User implementation:

```
power_control
    |___power_control.ldf
    |___power_control.lpf
    |___Strategy1.sty
    |___user
    |___source
            |___adc_top.v
            |___box_ave.v
            |___demo_top_user.v
            |___count16_ldrg.vhd
            |___display.v
            |___lcd_display.v
            |___lcd4digit.v
            |___LCDEncoding4to1.v
            |___lcdencoding4to1com.v
            |___pwm.v
            |___sigmadelta_adc.v
            |___ipexpress/LCDCharMap/LCDCharMap.ipx
```

Config Implementation:

```
power_control
    |___power_control.ldf
    |___power_control.lpf
    |___Strategy1.sty
    |___config
    |___source
          |___adc_top.v
          |___box_ave.v
          |___demo_top_config.v
          |___count16_ldrg.vhd
          |___display.v
          |___lcd_display.v
          |___lcd4digit.v
          |___LCDEncoding4to1.v
          |___lcdencoding4to1com.v
          |___pwm.v
          |___sigmadelta_adc.v
          |___wb_master.v
          |___ipexpress/efb_spi_i2c/efb_spi_i2c.ipx
          |___ipexpress/pwr_ctrl/pwr_ctrl.ipx
          |___ipexpress/LCDCharMap/LCDCharMap.ipx
```

Oscillator Instantiation
The following are necessary to enable the power control for the on-chip oscillator:

1. An OSCH module needs to be instantiated in the code.

2. The STDBY port of OSCH needs to be set/reset to put the oscillator into Standby or Wake mode.

The Verilog code for OSCH instantiation looks like the following:

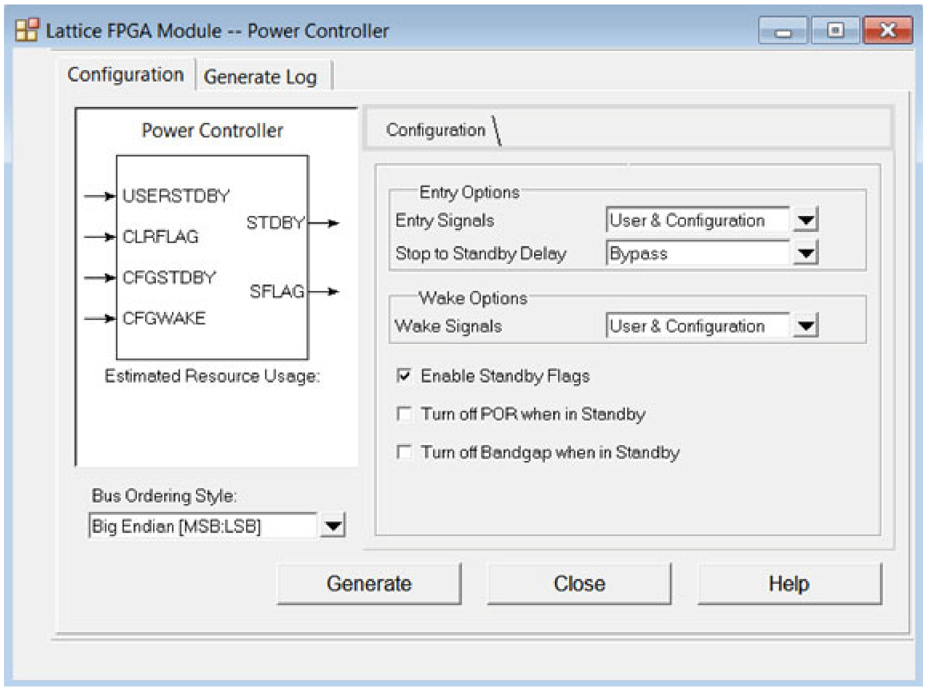
```
OSCH OSCH_inst (
    .STDBY        ( STDBY    ), // STDBY=1, stop oscillator; STDBY=0, start oscillator.
    .OSC          ( clk_osc    ),
    .SEDSTDBY     ( )
    /* synthesis syn_noprune = 1 */ ; // to prevent the oscillator from being optimized away
```

The on-chip oscillator can be turned on/off in two ways: through internal user logic and/or through the external I^{2}C or SPI port.

Power Controller
In RTL coding, there are two ways that users can set/reset STDBY: with and without Power Controller module instantiation, where the Power Controller is a hard-wired module in the MachXO2 device. Without the Power Controller being instantiated, user logic can set/reset STDBY directly (STDBY = 1, Standby; STDBY =0, Wake). In other words, if the MachXO2 fabric logic is chosen by the user to control the STDBY port of OSCH, there is no need to instantiate the Power Controller module in the design. There is no Power Controller module instantiation in the User Mode demo. However, if users choose to control the power saving features (oscillator, bandgap, etc.) with external I^{2}C and SPI ports, the Power Controller module needs to be instantiated in the design. To instantiate this module, the user needs to first generate it from the architecture module listed under IPexpress. With the Power Controller instantiated in the code, the STDBY signal will be set to ‘1’ either when the USERSTDBY port is set by user logic or when CFGSTDBY is set through the external I^{2}C and SPI ports. There are three options for Entry Signals and Wake signals in the GUI: User, Configuration and User & Configuration. Note that if User & Configuration is selected for options of entry signals and wake signals in the GUI, as shown in Figure 3, the I^{2}C or SPI port will

not be able to generate STDBY = 0 by setting the CFGWAKE port if USERSTDBY is set by user logic. This option is not used in the demo. Instead, the Configuration option is used for Configure Mode demo due to this limitation.

Figure 3. Power Controller GUI Options: User and Configuration



Big Endian [MSB:LSB]

Entry Signals: User & Configuration

Stop to Standby Delay: Bypass

Wake Signals: User & Configuration

[✓] Enable Standby Flags

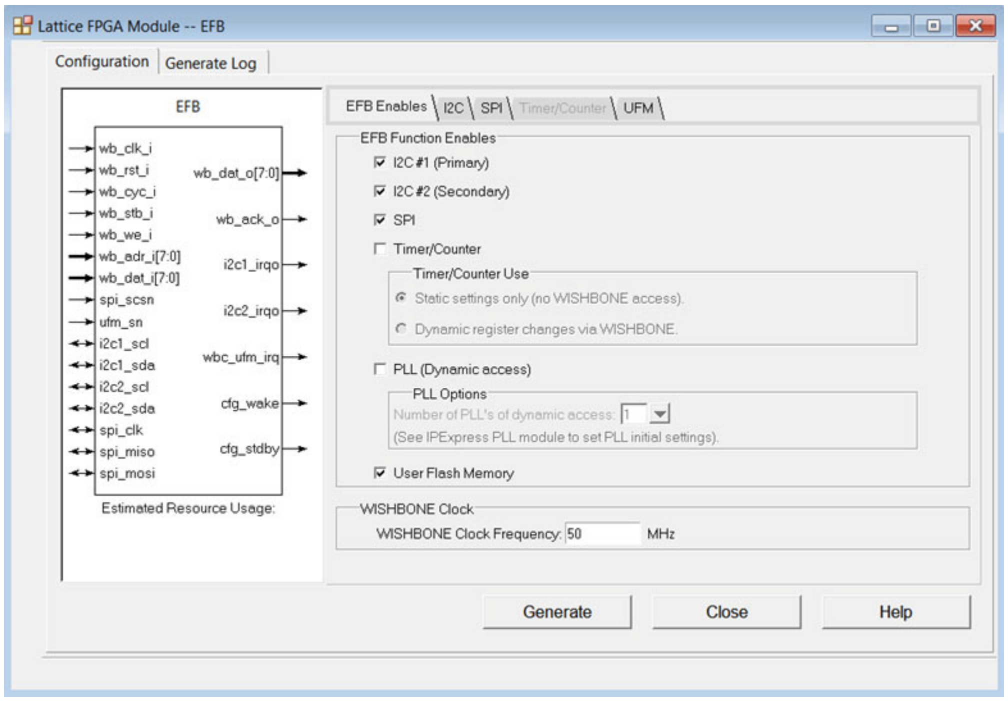
[ ] Turn off POR when in Standby

[ ] Turn off Bandgap when in Standby

EFB I²C + SPI

The MachXO2 device contains a hard-wired Embedded Functional Block (EFB). This functional block can be enabled through EFB instantiation which is an architecture module listed under IPexpress. In the Configure Mode demo, both SPI and I²C are enabled in EFB to provide the maximum flexibility of setting power saving mode. The GUI options are shown in Figures 3, 4 and 5. Note that there are two I²C slaves, primary and secondary. By default, the 7-bit slave address for primary is 000_1001 and it is 000_1010 for secondary.

Figure 4. EFB for SPI + I^2C



EFB Function Enables

- [✓] I2C #1 (Primary)
- [✓] I2C #2 (Secondary)
- [✓] SPI
- [ ] Timer/Counter

Timer/Counter Use

- ● Static settings only (no WISHBONE access).
- ○ Dynamic register changes via WISHBONE.

- [ ] PLL (Dynamic access)

PLL Options

Number of PLL's of dynamic access: 1

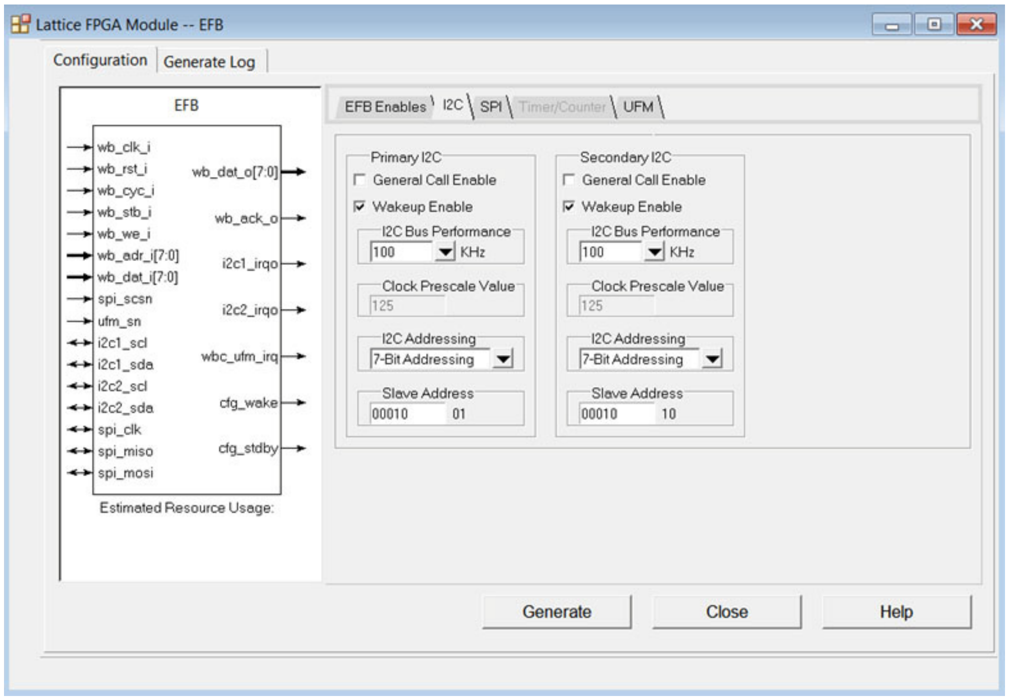
(See IPExpress PLL module to set PLL initial settings).

- [✓] User Flash Memory

WISHBONE Clock

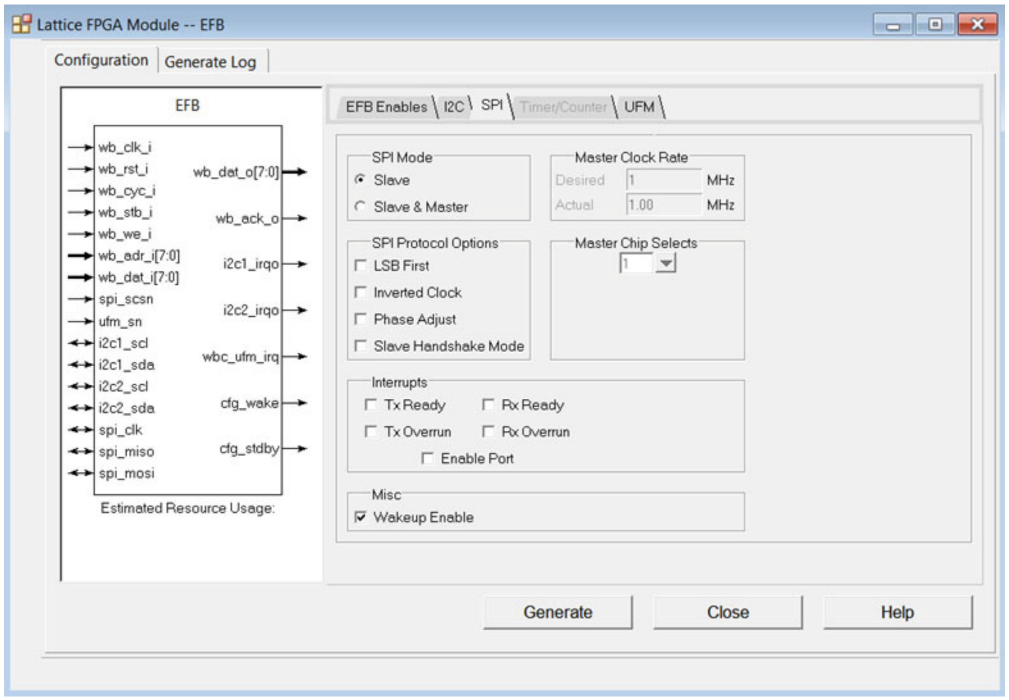
WISHBONE Clock Frequency: 50 MHz

Figure 5. I^2C Tab



	Primary I2C	Secondary I2C
General Call Enable	[ ]	[ ]
Wakeup Enable	[✓]	[✓]
I2C Bus Performance	100 KHz	100 KHz
Clock Prescale Value	125	125
I2C Addressing	7-Bit Addressing	7-Bit Addressing
Slave Address	00010 01	00010 10

Figure 6. SPI Tab



Lattice FPGA Module -- EFB

Configuration | Generate Log

EFB Enables | I2C | SPI | Timer/Counter | UFM

SPI Mode
- [✓ Slave]
- Slave & Master

Master Clock Rate
- Desired: 1 MHz
- Actual: 1.00 MHz

SPI Protocol Options
- ☐ LSB First
- ☐ Inverted Clock
- ☐ Phase Adjust
- ☐ Slave Handshake Mode

Master Chip Selects: 1

Interrupts
- ☐ Tx Ready
- ☐ Rx Ready
- ☐ Tx Overrun
- ☐ Rx Overrun
- ☐ Enable Port

Misc
- ☑ Wakeup Enable

Code Example

The Verilog code instantiating power control and EFB modules looks like the following:

```
// power control (configuration only)
pwr_ctrl pwr_ctrl_inst (
    .CLRFLAG     (         ),
    .CFGSTDBY    (         ),          // cfg_stdby  hardwire connected to EFB
    .CFGWAKE     (         ),         //  cfg_wake   hardwire connected to EFB
    .STDBY       ( STDBY   ),
    .SFLAG       (         )
    );

// EFB SPI + I2C
efb_spi_i2c efb_spi_i2c_inst (
    .wb_clk_i    ( clk_USB   ),
    .wb_rst_i    ( 1'b0      ),
    .wb_cyc_i    ( wb_cyc_o ),
    .wb_stb_i    ( wb_stb_o ),
    .wb_we_i     ( wb_we_o  ),
    .wb_adr_i    ( wb_adr_o ),
    .wb_dat_i    ( wb_dat_o ),
    .wb_dat_o    ( ),
    .wb_ack_o    ( wb_ack_o ),
    .i2c1_scl    ( i2c1_scl ),
    .i2c1_sda    ( i2c1_sda ),
    .i2c1_irqo   ( ),
    .i2c2_scl    ( i2c2_scl ),
    .i2c2_sda    ( i2c2_sda ),
    .i2c2_irqo   ( ),
    .spi_clk     ( spi_clk  ),
    .spi_miso    ( spi_miso ),
    .spi_mosi    ( spi_mosi ),
    .spi_scsn    ( spi_scsn ),
    .ufm_sn      ( ufm_sn   ),
    .wbc_ufm_irq ( ),
    .cfg_wake    ( cfg_wake ),
    .cfg_stdby   ( cfg_stdby)
    );
```

I^{2}C and SPI Master on the MachXO2 Pico Evaluation Board

In the demo setup of two MachXO2 Pico Evaluation Boards, one is programmed with the Configure Mode bitstream and the other is used as an I^{2}C and SPI master. The I^{2}C and SPI master design is located under Pico_I2C_SPI_master. In this design, the platform for the LatticeMico8™ is the same as the one used in the MachXO2 Master SPI/I^{2}C Demo (see references). For more technical details regarding this design, please refer to the MachXO2 Master SPI/I^{2}C Demo.

Known Limitations

This section explains some of the limitations of the MachXO2 Low Power Control Demo.

Separate User Mode and Configure Mode for Power Control

There are two separate bitstreams for the demo: one for User Mode and the other for Configure Mode. They do not coexist in the same design due to the following limitations:

1. The power controller module needs to be instantiated in the code for the above situation.
2. Once USERSTDBY is set by user logic to put the device in standby mode, CFGWAKE will not be able to reset it and wake up the device.

Limited Range for LCD Display of Icc

There are two decimal digits on the LCD display: XX.0A, however the range of those two digits is not a full range of 0-99. This is due to the fact that in the code, it is one digit of hexadecimal which has a range of 0-F and the corresponding converted decimal range is 0-15. If a display range larger than this is needed, either of the following modifications to the code can be made:

1. Use hexadecimal directly instead of decimal

2. Replace the current conversion logic with multiple-digit hexadecimal-to-decimal conversion logic

Extra wb_master Module

There is a Verilog file in the Diamond project called wb_master.v. This module is used to set a certain register value through the WISHBONE bus. This module is necessary to make this bit set such that power control through SPI is enabled.

References

- EB61, MachXO2 Pico Development Kit User's Guide
- TN1198, Power Estimation and Management for MachXO2 Devices
- UG54, MachXO2 Master SPI/I^{2}C Demo using 'C' and LatticeMico8

Technical Support Assistance

Hotline:	1-800-LATTICE (North America)
	+1-503-268-8001 (Outside North America)
e-mail:	techsupport@latticesemi.com
Internet:	www.latticesemi.com

Revision History

Date	Version	Change Summary
April 2012	01.0	Initial release.