# Low Power Reference Platform

# Reference Guide

## Revision History

| Rev. | Date | Author | Description |
|---|---|---|---|
| 1 | 7/07 | SJK | Production Release |
| | | | |

# 1

# Introduction

It has not been previously feasible to consider FPGA's that are powered by portable battery sources. However reductions in device geometries have significantly reduced static and dynamic power requirements for these devices. The LPRP demonstrates board and chip level low power design techniques for Nios II / SOPC Builder designs that are battery powered.

Power is conserved at the chip level by

- Dynamically reducing clock frequencies during periods of inactivity
- Shutting the FPGA down (with context save) after further inactivity
- Powering the FPGA up (with context restore) once activity returns

Power is conserved at the board level by

- Lower power requirements of inactive devices
- Turning power off to unused devices
- Using efficient power conversion devices
- Using devices with very low standby power

The LPRP is powered by a number of different sources. These include

- A USB Power interface
- A Wall Power supply (not included)
- Lithium Ion rechargeable batteries

The LPRP features a Board Level ***Power Management Framework (PMF)***. This framework uses techniques to conserve power in the Cyclone III FPGA and at the board level. The PMF is a layer of software run on the Nios II processor in the Cyclone III FPGA and on an additional micro controller.

The PMF permits the highest level of power conservation while retaining the operational status of the LPRP. This is the key ingredient to obtaining extended battery life. The PMF will be discussed subsequently.

# 2

# LPRP Board Description

## LPRP Block Diagram

The LPRP block diagram is shown in Figure 1. The complementary LPRP board layout is shown in Figure 2.

*Figure 1. LPRP Block Diagram*

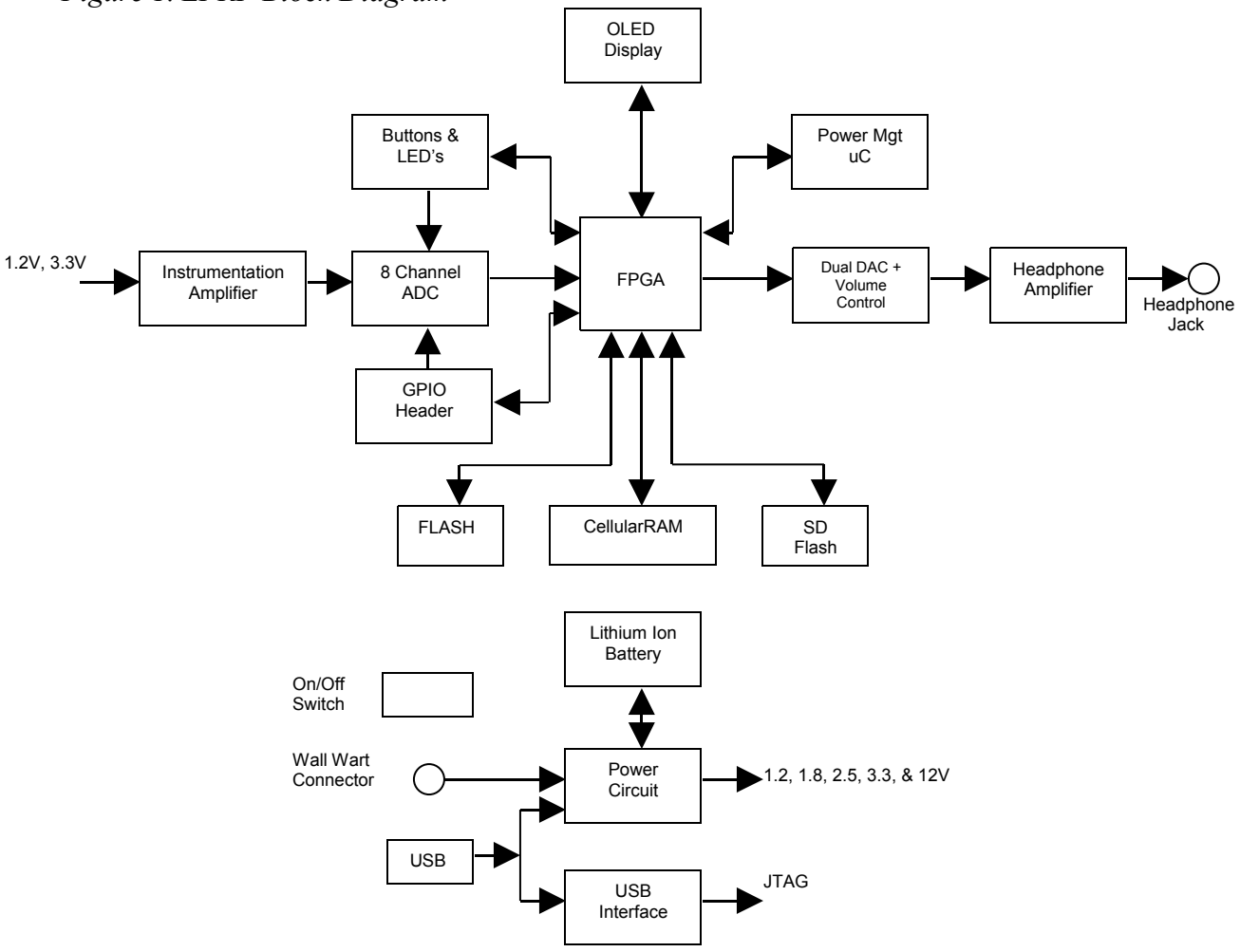
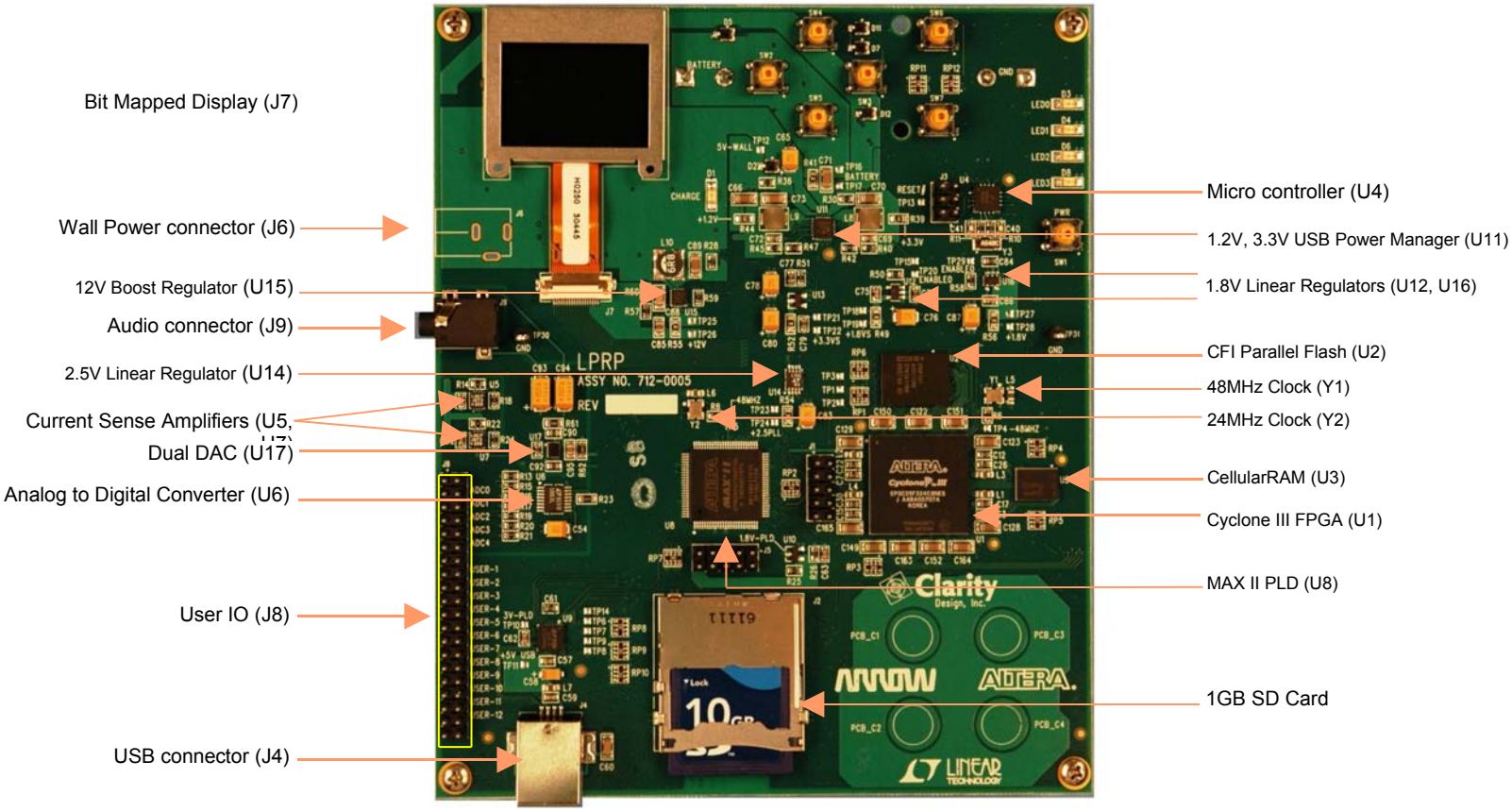


*Figure 2. LPRP Component Layout*

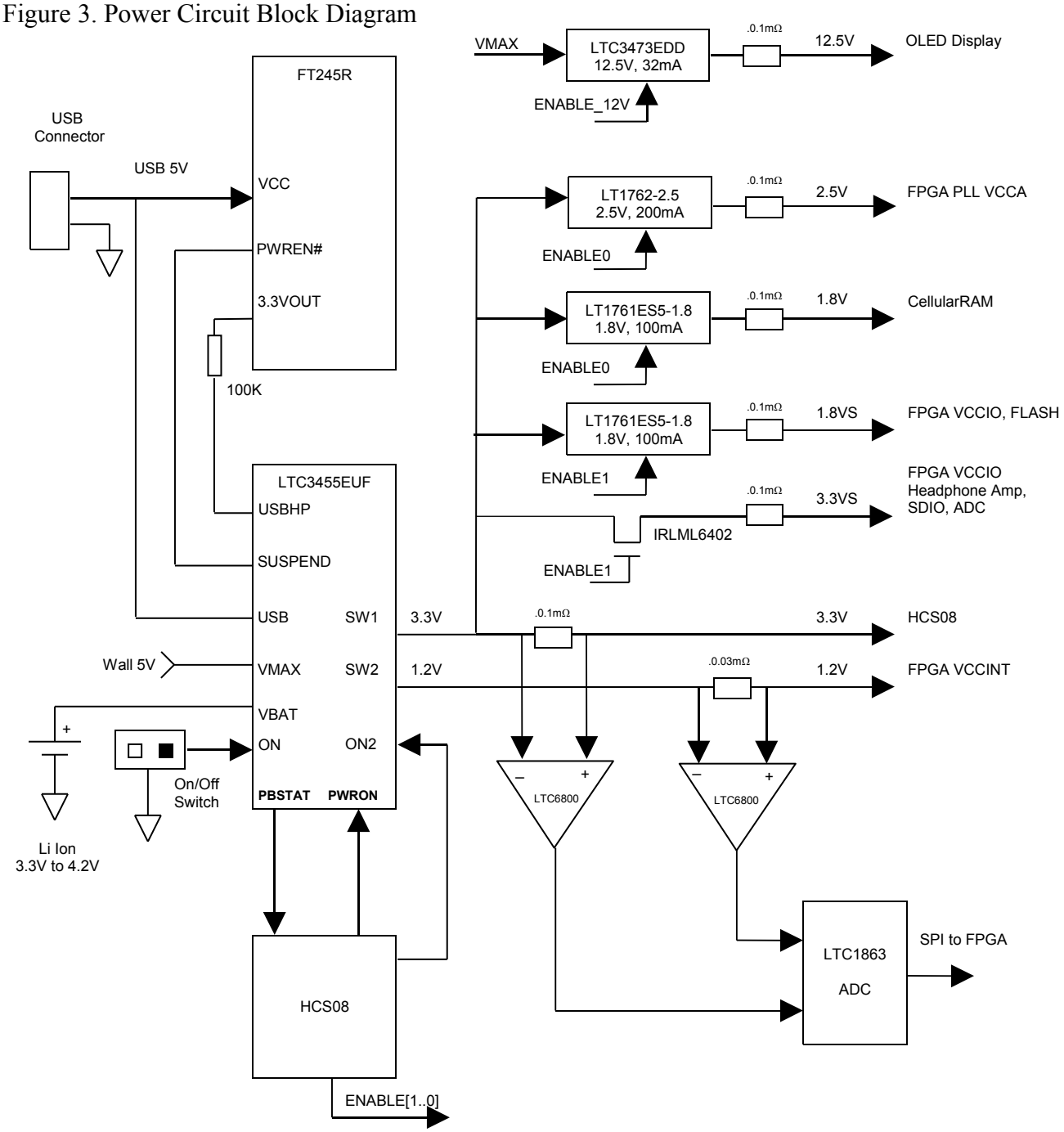



Refer to the LPRP Schematics for more board level detail.

## Power Circuit Block Diagram

The power circuit block diagram is shown in Figure 3.

Figure 3. Power Circuit Block Diagram



## Power Sources

The LPRP has three potential power sources; wall power, USB power or Li Ion power. The LTC3455 is a one chip, efficient power solution for all three sources. If all three power sources are available, the 3455 will always use wall power or USB power. If neither of these are available it will then use battery power.

Press the power button to turn the LPRP board on or off.

When the board is turned on all power circuits are activated and the applications run in full power mode. The board then transcends into other power modes based on context, time, temperature or user settings.

## LTC3455 (U11)

The LTC3455 is designed to be a complete power management solution for a wide variety of portable systems. The device incorporates two current mode step-down switching regulators, a full-featured battery charger, a USB power controller, a Hot Swap output and a low-battery comparator. Refer to Figure 4 for a functional block diagram.

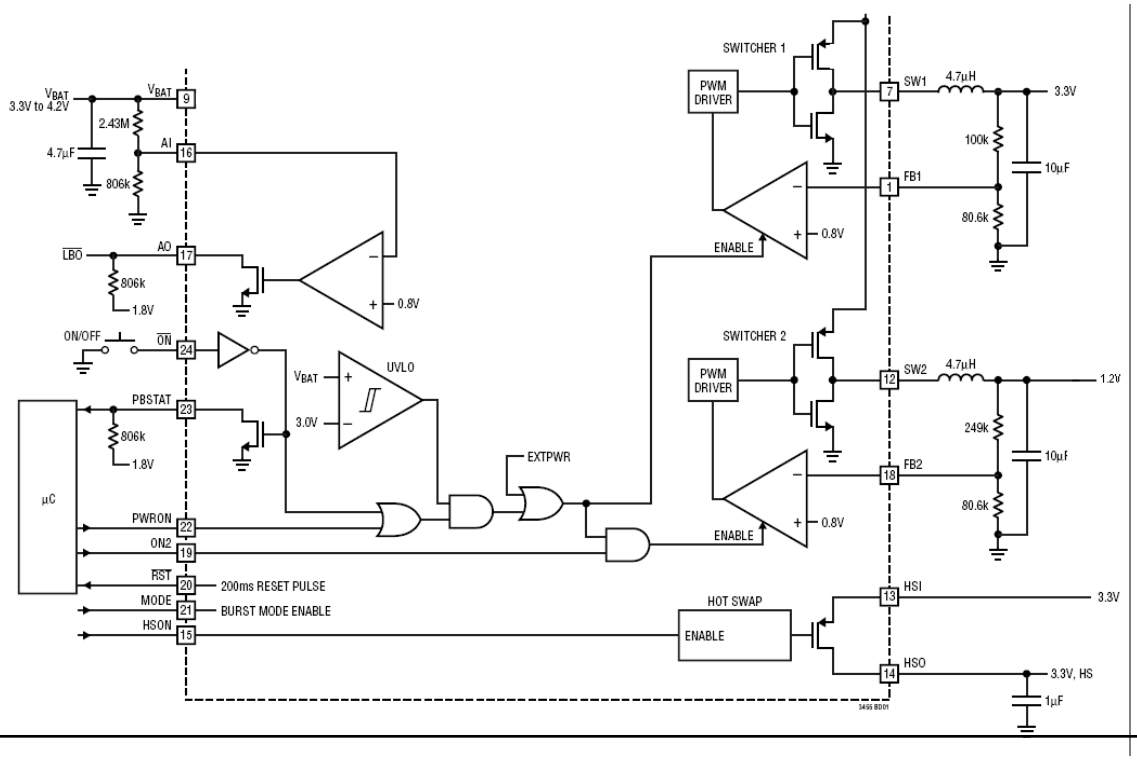
When only battery power is available, the battery PMOS switch connects the VMAX pin to the VBAT pin to provide power to both switching regulators (and any other devices powered (from VMAX). When external power is applied, the LTC3455 seamlessly transitions from battery power (a single-cell (Li-Ion) to either the USB supply or a wall adapter.

When the battery PMOS switch is turned off, the charger is activated and all internal power for the device is drawn from the appropriate external power source. Maximum charge current and charge time are programmed using an external resistor and capacitor, respectively. The USB power manager provides accurate current limiting for the USB pin under all conditions.

The USB power controller section of the LTC3345 will only be enabled when the USB interface is in USB High Power mode.

The LTC3345 provides a supply for hot swappable devices such as a Secure Digital (SD) Card. These devices that would appear as a short if they were hot-plugged directly to one of the supply outputs

*Figure 4 . LT3455 Functional Block Diagram*




## Explaining The LTC3455 Functional Diagram

The functionality of this power controller can be divided into two main categories. 1. External Power sources are present. 2. Only Battery power is present. The LTC3455 includes two switch mode power supplies. The LPRP uses switcher 1 to derive 3.3V and switcher 2 to derive 1.2V. Switcher 2 powers the core of the FPGA. The 3.3V from switcher 2 is used to power the HCS08 uC and is also used to derive all other board level voltages, except for 12V.

When external power is present, since power conservation is not as significant, switcher 1 (3.3V) will be turned on. This will in turn power the HCS08 uC. When only battery power is present, both switchers will be disabled until the ON# signal is asserted.

**ON# –** This signal enables either of the switch mode power supplies when asserted low. It will be asserted if the momentary POWER ON pushbutton is pressed or if USB power is present. It is over-ridden when external power is present

**PWRON** – This signal is asserted by the HCS08 uC and is used to hold switcher 1 on once the POWER ON pushbutton has been released. In this reference design the HCS08 will be used to keep real time while the rest of the board is off, so once switcher 1 is on it will not be turned off unless there is no external power and the battery has discharged. In designs where keeping real time is not an objective, this signal can be deasserted during a board level POWER OFF condition.

**ON2** – This is asserted by the HCS08 uC. It enables switcher 2 (1.2V). It will be asserted after a pushbutton POWER ON event, external power is available or an AWAKE event has occurred.

**HSON** - This signal enables power to the 3.3V HS supply. The FPGA asserts this signal when it detects that an SD card has been inserted. This allows power to ONLY be provided to the SD card once it is completely made contact with the socket VCC and GND contacts.

There are a few miscellaneous control and status signals that the FPGA interface with. They are shown in Table 1. They are connected to output and input PIO peripherals in the FPGA.

***Table 1. LTC3455 Signals***

| Signal | Direction | LTC3473 Pin | FPGA Pin |
|---|---|---|---|
| BURST_MODE | Input | 21 | U4 |
| CHRG_N | Output | 4 | U9 |
| LBO_N | Output | 17 | V9 |

## LT1761-1.8 (U12, U16)

U12 provides 1.8V for the CFI Flash device and VCCIO for the FPGA. They are enabled by the ENABLE_1 control signal from the HCS08 micro-controller .

U16 provides an isolated 1.8V supply to the CellularRAM device which remains on during Hibernation mode.

## LT1762-2.5 (U14)

U14 provides 2.5V to the FPGA that is used to power VCCA, the analog supply for its PLLs. It was chosen for this function because it provides very good Input Ripple Rejection.

## LTC3473 (U15)

U15 provides the OLED display with 12.5V. It is enabled via a signal from an SOPC Builder PIO in the FPGA (DISPLAY_POWER_ENABLE). This is shown in Table 2.

***Table 2. Enable_12V Signal***

| Signal | PIO BIT POSITION | Direction | LTC3473 Pin | FPGA Pin |
|---|---|---|---|---|
| ENABLE_12V | 0 | Input | 6 | P6 |

## Monitoring Power

Circuitry has been included to allow for dynamic monitoring of power consumption. Small resistors have been added in series with the 1.2 and 3.3V switcher outputs. The current flowing through these resistors is fed into a current sense amplifier. This in turn is fed into ADC's in the LTC1863L ADC. This information is communicated through the SPI interface to the Nios II processor. This info can be displayed on the OLED display.

### LTC6800 (U5, U7)

This is an instrumentation amplifier and is used to dynamically measure power usage on the two main voltage supplies, 1.2V and 3.3V. All other supplies are derived from the 3.3V supply. Low value (.03 Ω and 0.1 Ω) resistors are placed in series with the output of the 1.2V and 3.3V switcher supplies. The voltages measured across these resistors are then fed into the inputs of the LTC6800 and amplified by a factor of 51.

### LTC1863L Analog to Digital Converter (U6)

This is 12-bit, eight channel ADC. Two of the channels are used to sample the current drawn from the 3.3V and 1.2V supplies respectively. One of the channels is used as an analog GND reference. The remaining five channels are available for general purpose functions and are connected to the user I/O header.

The ADC communicates converted data to the FPGA via the 12 bit SPI interface. The ADC is used in a unipolar mode. Data is shifted out MSB first. It has a range of 0 to 2.5V and uses an internal reference. The power measured digitally across the current sense resistors can be expressed as

$$Power_{3.3V} = 395.0298uW * \text{converted data}$$

$$Power_{1.2V} = 478.8240uW * \text{converted data}$$

Please note that the direction is defined with respect to the discrete device and not the FPGA.

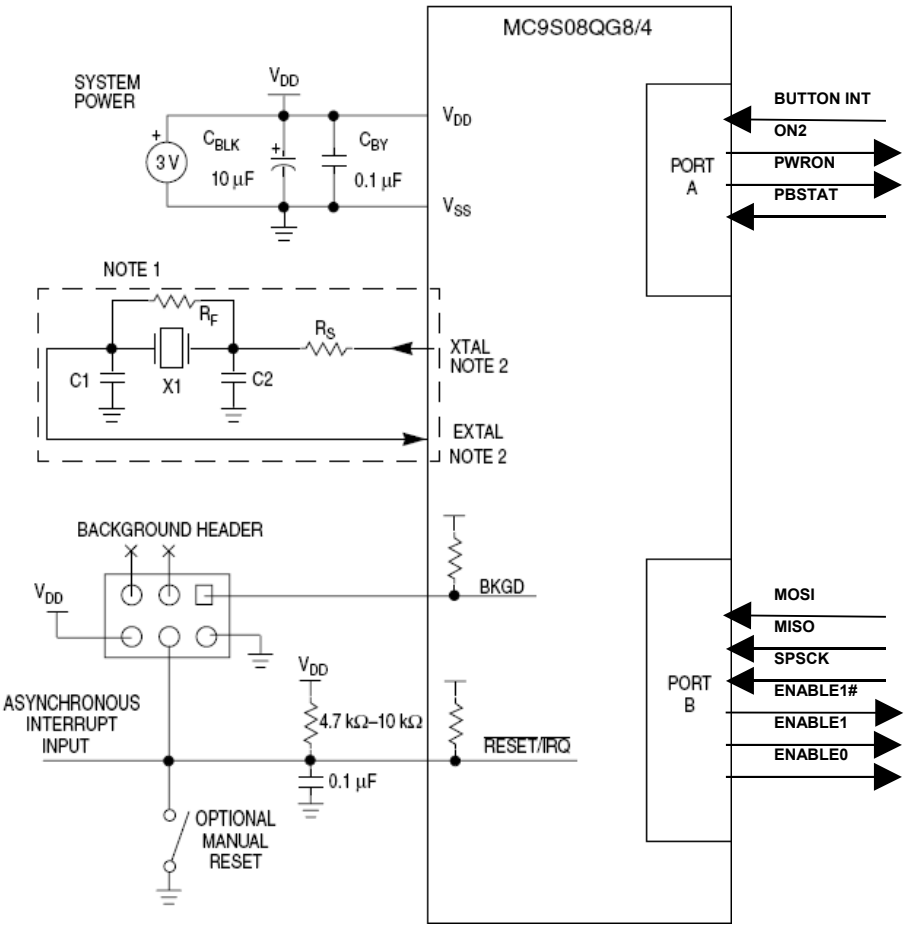
***Table 3. LTC1863L Control Signals***

| **SPI Slave interface, 12 bits, MSB first, 128 kHz** | | | |
|---|---|---|---|
| **Signal** | **Direction** | **LTC1863L Pin** | **FPGA Pin** |
| ADC_SPI_DIN | Input | 14 | U1 |
| ADC_SPI_DOUT | Output | 13 | N1 |
| ADC_SPI_CLK | Input | 12 | M6 |
| ADC_SPI_CS_N | Input | 11 | N6 |

## HCS08 Micro-controller (U4)

The HCS08 performs overall power management functions for the LPRP. The HCS08 is always on if external power is applied., running in an ultra low power mode (500nA) when the board is off. Its power is provided by SW1. SW1 will always be on if either USB or Wall power is available or if the battery voltage is above 3.0V. SW1 consumes 100uA when operational. The HCS08 determines when SW2 and all other board boost regulators and LDOs are enabled. The HCS08 block diagram is shown in Figure 5 and a pin list is shown in Table 4.

*Figure 5 . HCS08 Block Diagram*



## HCS08 Pin List

*Table 4. HCS08 Signals*

| Signal | Direction | Description | HCS08 Pin | FPGA Pin |
|---|---|---|---|---|
| **Power** | | | | |
| VCC | | | 3 | |
| GND | | | 4 | |
| **Crystal** | | | | |
| XTAL | Input | Crystal input | 6 | |
| EXTAL | Input | Crystal input | 5 | |
| **Debug** | | | | |
| BKGD | Input | Background debug | 2 | |
| **External Components** | | | | |
| PBSTAT | Input | Push Button Status | 13 | |
| BUTTON INT | Input | Button Interrupt | 16 | |
| PB/Reset | Input | 3455 Reset | 1 | |
| **LTC3455** | | | | |
| ON2 | Output | Turns Switcher #2 (1.2V) on | 15 | |
| PWRON | Output | Keeps Switcher #1 (3.3V) on | 14 | |
| **FPGA** | **SPI Master interface, 8 bits, MSB first** | | | |
| MISO | Output | SPI | 8 | U12 |
| MOSI | Output | SPI | 9 | U13 |
| SPI CLK | Output | SPI | 10 | P10 |
| **External Supplies** | | | | |
| ENABLE1# | Output | FET enable | 7 | |
| ENABLE1 | Output | LDO enable | 11 | |
| ENABLE0 | Output | LDO enables | 12 | |

## HCS08 to FPGA Communication

The HCS08 uses a SPI master peripheral to communicate with a SPI slave peripheral in the Cyclone III FPGA. It uses a simple protocol to coordinate activities with the Cyclone III. Commands included are:

- Hibernation request
- Hibernation acknowledge
- Power off request
- Power off acknowledge
- Time of Day request
- Temperature request

## LM4921 Dual Audio DAC (U17)

This device is has two interface ports. The first is a 16 bit SPI slave and is used to access the operating modes of the device as well as set its volume levels. Audio is transmitted to the LM4921 via an I2S interface. Each I2S word is 32 bits wide representing two 16 bit audio channels (left and right).

The LM4921 can be placed in low power NAP and SLEEP modes via SPI commands. The LM4921 requires an audio rate clock to drive the dual DAC's at a 44.1 or 44.8 kHz audio sampling rate. An 11.2896 Mhz clock is driven in to the AUDIO_CLK pin.

***Table 5. LM4921 Control Signals***

| **SPI Slave interface, 16 bits, LSB first, 128 kHz** | | | |
|---|---|---|---|
| **Signal** | **Direction** | **LM4921 Pin** | **FPGA Pin** |
| AUDIO_SPI_DATA | Input | E4 | M1 |
| AUDIO_SPI_EN | Input | D3 | R1 |
| AUDIO_SPI_CLK | Input | E3 | M5 |
| **I2S Slave interface, 32 bits, MSB first, 11.2896 MHz** | | | |
| **Signal** | **Direction** | **LM4921 Pin** | **FPGA Pin** |
| AUDIO_I2S_DATA | Input | C2 | L6 |
| AUDIO_I2S_WS | Input | B2 | T1 |
| AUDIO_I2S_CLK | Input | B1 | R5 |
| AUDIO_CLK | Input | E2 | M2 |
| | | | |

## Memory Devices

### CFI Parallel Flash (U2)

The LPRP uses Intel P30 256Mb StrataFash to configure the FPGA at power on and to store its embedded application. The board is layed out with one extra address line to also allow use of the P30 512Mb device. At power on the FPGA's reads configuration in 16 bit half words from the Flash device. This is performed as a burst operation and the FPGA is configured in approximately 9 mS.

After configuration the FPGA enters User Mode and the Nios II processor copies the LPRP application code from Flash to CellularRAM. At this point the Flash device can be placed in Standby mode to conserve power.

The Flash device must be connected to dedicated pins on the FPGA to allow for proper configuration. The pinout is shown in Table 6 below.

***Table 6. P30 StrataFlash Signals***

| Address | | | |
|---|---|---|---|
| **Signal** | **Direction** | **P30 Pin** | **FPGA Pin** |
| FLASH_A1 | Input | A1 | E12 |
| FLASH_A2 | Input | B1 | A16 |
| FLASH_A3 | Input | C1 | B16 |
| FLASH_A4 | Input | D1 | A15 |
| FLASH_A5 | Input | D2 | B15 |
| FLASH_A6 | Input | A2 | A14 |
| FLASH_A7 | Input | C2 | B14 |
| FLASH_A8 | Input | A3 | A13 |
| FLASH_A9 | Input | B3 | B12 |
| FLASH_A1 | Input | C3 | A12 |
| FLASH_A11 | Input | D3 | B12 |
| FLASH_A12 | Input | C4 | A11 |
| FLASH_A13 | Input | A5 | B11 |
| FLASH_A14 | Input | B5 | C10 |
| FLASH_A15 | Input | C5 | D10 |
| FLASH_A16 | Input | D7 | E10 |
| FLASH_A17 | Input | D8 | C9 |
| FLASH_A18 | Input | A7 | D9 |
| FLASH_A19 | Input | B7 | A7 |
| FLASH_A20 | Input | C7 | A6 |
| FLASH_A21 | Input | C8 | B18 |
| FLASH_A22 | Input | A8 | C17 |
| FLASH_A23 | Input | G1 | C18 |
| FLASH_A24 | Input | H8 | G14 |
| FLASH_A25 | Input | B6 | H17 |

| *Table 6. P30 StrataFlash Signals (Continued)* | | | |
|---|---|---|---|
| ***Data*** | | | |
| **Signal** | **Direction** | **P30 Pin** | **FPGA Pin** |
| FLASH_D0 | Bidir | F2 | H3 |
| FLASH_D1 | Bidir | E2 | D1 |
| FLASH_D2 | Bidir | G3 | A8 |
| FLASH_D3 | Bidir | E4 | B8 |
| FLASH_D4 | Bidir | E5 | B7 |
| FLASH_D5 | Bidir | G5 | C5 |
| FLASH_D6 | Bidir | G6 | E8 |
| FLASH_D7 | Bidir | H7 | A4 |
| FLASH_D8 | Bidir | E1 | B4 |
| FLASH_D9 | Bidir | E3 | E7 |
| FLASH_D10 | Bidir | F3 | A3 |
| FLASH_D11 | Bidir | F4 | B3 |
| FLASH_D12 | Bidir | F5 | D5 |
| FLASH_D13 | Bidir | H5 | B5 |
| FLASH_D14 | Bidir | G7 | A5 |
| FLASH_D15 | Bidir | E7 | B6 |
| ***Control*** | | | |
| FLASH_WAIT | Output | F7 | H13 |
| FLASH_ADV_N | Input | F6 | H14 |
| FLASH_CLK | Input | E6 | H4 |
| FLASH_RST_N | Input | D4 | C3 |
| FLASH_OE_N | Input | F8 | D17 |
| FLASH_CE_N | Input | B4 | E2 |
| FLASH_WP_N | Input | C6 | H1 |
| FLASH_WE_ | Input | G8 | D18 |

## CellularRAM (U3)

CellularRAM was included because of its ability to automatically transition to a very low power standby mode (65uA @ 1.8V). It is a low cost but dense memory (32Mb) device that can be run at 80 Mhz in a burst operating mode. It is based on PSRAM technology and provides a very easy to use interface.

An SOPC Builder CellularRAM controller was specially developed for the LPRP. It allows unlimited burst length operation from an Avalon master. This allows for very efficient instruction and data cache line fill and DMA operations.

When the board is placed in Hibernation mode, the CellularRAM remains powered and retains the context of the LPRP application and the Nios II processor.

## Secure Digital Card (J2)

Secure Digital defines two optional interface formats, the first a serial SPI format and the other a four bit wide parallel format (SD mode). The LPRP uses the SPI format, but the signals for the alternate format are connected from the FPGA to the SD card connector.

The SPI interface is driven by a dedicated SOPC Builder peripheral (sd_controller_0). The rest of the SD signals are connected to a standard PIO port. The SD Controller signals are defined in Table 7 and the PIO signals are defined in Table 8.

***Table 7. SD_CONTROLLER_0 Signals***

| SPI Mode Signal | SD Mode Signal | Direction | SD Pin | FPGA Pin |
|---|---|---|---|---|
| SD_CARD_SPI_MISO | SD_D[0] | Output | 7 | V16 |
| SD_CARD_SPI_MOSI | SD_CMD | Input | 2 | U14 |
| ADC_SPI_CLK | SD_CLK | Input | 5 | T6 |
| SD_CARD_SPI_CS_N | SD_D[3] | Input | 1 | V14 |

***Table 8. SD_CARD_IO Signals***

| SPI Mode Signal | PIO BIT POSITION | Direction | SD Pin | FPGA Pin |
|---|---|---|---|---|
| SD_PROTECT_N | 0 | Output | 11 | V10 |
| SD_DETECT_N | 1 | Output | 10 | U10 |
| SD_D[2] | 2 | Output | 9 | V15 |
| SD_D[1] | 3 | Output | 8 | R11 |

The SD card can be inserted or removed while LPRP power is on. Power is not applied to the SD connector until the SD_DETECT signal is asserted. A switch in the LTC3455 device is then turned on and power to the DS is applied through the dedicated VCC3HS power signal. The switch is controlled by an SOPC Builder PIO peripheral. A zero disables SD Power and a one enables SD Power. This signal is shown in Table 9.

***Table 9. SD HSEN Signal***

| Signal | PIO BIT POSITION | Direction | LTC3455 Pin | FPGA Pin |
|---|---|---|---|---|
| HSEN | 0 | Input | 15 | V1 |

## OLED Display (J7)

The OSRAM Full Moon Display is a 96 x 64 bit mapped display. It has a built in controller and display memory. It is connected to the FPGA via a parallel interface. The display signals are shown in Table 10.

***Table 10. OLED Signals***

| Signal | Direction | J7 Pin | FPGA Pin |
|---|---|---|---|
| DISPLAY_D7 | Bidir | 5 | L2 |
| DISPLAY_D6 | Bidir | 6 | L1 |
| DISPLAY_D5 | Bidir | 7 | K2 |
| DISPLAY_D4 | Bidir | 8 | K1 |
| DISPLAY_D3 | Bidir | 9 | K5 |
| DISPLAY_D2 | Bidir | 10 | L4 |
| DISPLAY_D1 | Bidir | 11 | L3 |
| DISPLAY_D0 | Bidir | 12 | P2 |
| DISPLAY_ERD_N | Input | 13 | P1 |
| DISPLAY_RW_N | Input | 14 | R2 |
| DISPLAY_DC_N | Input | 15 | T3 |
| DISPLAY_BSI | Input | 16 | R3 |
| DISPLAY_RES_N | Input | 17 | L5 |
| DISPLAY_CS_N | Input | 18 | R4 |

## User Pushbuttons (SW2-SW7)

These are general purpose pushbuttons that are also defined for use with the LPRP applications. They are connected to an input PIO peripheral in the FPGA and an interrupt is generated to the Nios II processor if they are pressed. They are also diode OR'ed together. This common signal, BUTTON_INT_N, is to interrupt HCS08 and remove the LPRP from Hibernation mode. The pushbutton connections are shown in Table 11.

***Table 11. User Pushbutton Signals***

| Signal | SW | FPGA Pin | LPRP Function |
|---|---|---|---|
| USER_PB0 | 2 | V6 | Left |
| USER_PB1 | 3 | U9 | Right |
| USER_PB2 | 4 | P9 | Up |
| USER_PB3 | 5 | R8 | Down |
| USER_PB4 | 6 | V5 | Esc |
| USER_PB5 | 7 | V4 | Enter |
| BUTTON_INT_N | | N2 | |

## User LEDs

These are general purpose LEDs. Relative to the rest of design they draw significant power when turned on and can shorten battery life considerably. They can be used efficiently if connected to a PWM peripheral in the FPGA where there effective brightness is reduced due to short percentages of ON time. In this design they are connected to a simple output PIO peripheral. To turn an LED ON write a one to the appropriate bit position in the PIO. THE User LED signals are shown in Table 12.

***Table 12. User LED Signals***

| User LED Signal | PIO BIT POSITION | FPGA Pin |
|---|---|---|
| USER_LED0 | 0 | T2 |
| USER_LED1 | 1 | M3 |
| USER_LED2 | 2 | U8 |
| USER_LED3 | 3 | V7 |

## User IO (J8)

The User IO header provides access to twelve FPGA IO pins and five Analog to Digital Converter inputs. The digital IO signals have 330 Ω series current limiting resistors to protect the FPGA IO pins. The User IO signals are shown in Table 13.

***Table 13. User IO Signals***

| Signal | J8 Pin | ADC Pin | FPGA Pin |
|---|---|---|---|
| USER_ADC_IN0 | 4 | 3 | |
| USER_ADC_IN0 | 6 | 4 | |
| USER_ADC_IN0 | 8 | 5 | |
| USER_ADC_IN0 | 10 | 6 | |
| USER_ADC_IN0 | 12 | 7 | |
| USER_IOT1 | 16 | | V3 |
| USER_IOT2 | 18 | | V8 |
| USER_IOT3 | 20 | | T4 |
| USER_IOT4 | 22 | | N7 |
| USER_IOT5 | 24 | | U5 |
| USER_IOT6 | 26 | | P8 |
| USER_IOT7 | 28 | | U7 |
| USER_IOT8 | 30 | | N8 |
| USER_IOT9 | 32 | | N9 |
| USER_IOT10 | 34 | | P7 |
| USER_IOT11 | 36 | | U3 |
| USER_IOT12 | 38 | | T8 |
| VCC3S | 1 | | |

*Table 14. User IO Signals (Continued)*

| Signal | J8 Pin | ADC Pin | FPGA Pin |
|---|---|---|---|
| AGND2 | 3 | | |
| AGND2 | 5 | | |
| AGND2 | 7 | | |
| AGND2 | 9 | | |
| AGND2 | 11 | | |
| GND | 13 | | |
| GND | 15 | | |
| GND | 17 | | |
| GND | 19 | | |
| GND | 21 | | |
| GND | 23 | | |
| GND | 25 | | |
| GND | 27 | | |
| GND | 29 | | |
| GND | 31 | | |
| GND | 33 | | |
| GND | 35 | | |
| GND | 37 | | |
| GND | 39 | | |
| GND | 14 | | |
| GND | 40 | | |

## Power Consumption

Power was measured on all of the supplies and tabulated in Table 15. Battery life was verified empirically and calculated based on a Li-Ion 3.7V battery @ 2300maH. The battery will charge off USB power at **395mA** when the LPRP is in the **Power Off** mode and at **259mA** when the LPRP is in **Normal** mode. There is leakage from the battery in Hibernate and Power Off modes. Efforts will be made to isolate the issue and reduce the dissipation in these two mode to less than 1mW

*Table 15. Power Consumption (in mW)*

| Mode | 1.2V | 1.8V | 1.8VS | 2.5V | 3.3VS | 3.3V | 12.5V | $V_{BAT}$ |
|---|---|---|---|---|---|---|---|---|
| Normal | 168 | 15 | 11.7 | 104 | 249.5 | 281 | 82.8 | 613 |
| Reduced | 168 | 15 | 11.7 | 104 | 237.6 | 271 | 0 | 492 |
| Standby | 35 | 3.9 | 7.2 | 104 | 206.6 | 215 | 0 | 276 |
| Hibernate | 0 | 0 | 180uW | 0 | 0 | 660uW | 0 | 7.5 |
| Power Off | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3.75 |

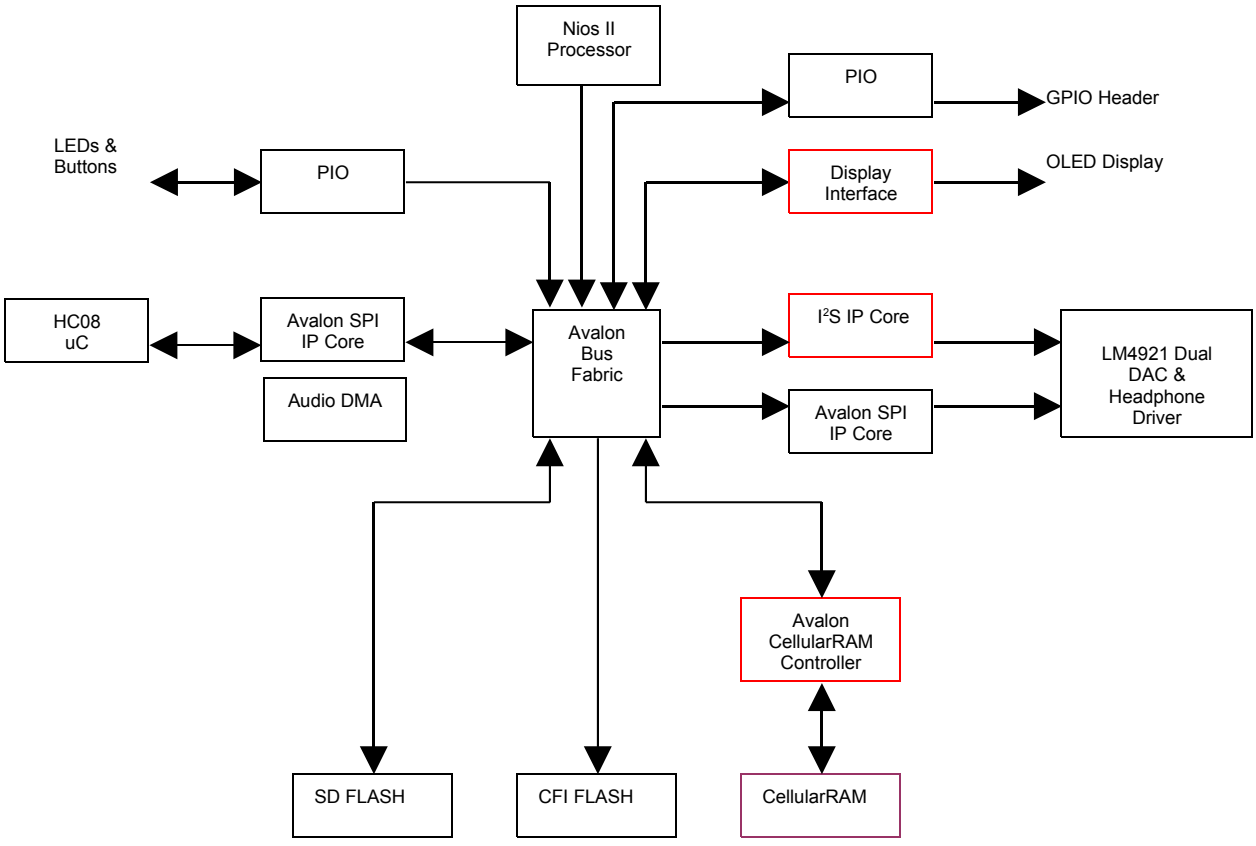
# 3

# Embedded Architecture

## Block Diagram

The embedded block diagram is shown in Figure 6. The Embedded system was constructed using SOPC Builder. It includes Nios II, a 32 bit RISC, 105 Dhrystone MIP processor and a collection of peripherals

*Figure 6. Embedded Block Diagram*



All of the blocks, except those outlined in red, are standard SOPC Builder peripherals. Documentation for them is extensive and can be found online at the Embedded Peripherals literature page. The new peripherals shown in red are detailed later in this document. A simplified SOPC Builder block diagram is shown in Figure 7.

To conserve power, peripherals were grouped into clock domains. Three main domains are defined. The fastest is the "clk" domain (80 MHz). The Nios II processor, Audio DMA and defined "clk" peripherals are grouped together. Avalon to Avalon Bridges are then used to isolate "slow clk" (20 MHz) peripherals and "ultra slow clk" (3 MHz) peripherals from the "clk" domain. Peripherals associated with the respective clock domains are shown in Table 15. These are not yet fully optimized and a few peripherals can be moved to slower domains.

*Figure 7. SOPC Builder Block Diagram*

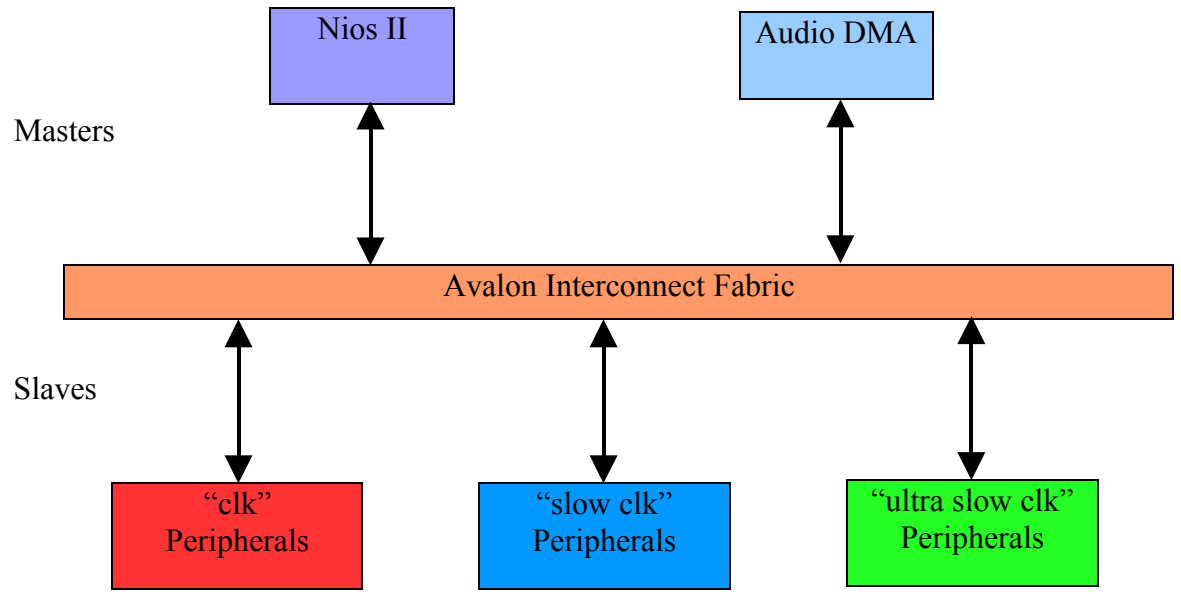


*Table 15. Clock Domains and Associated Peripherals*

| Peripheral | clk | slow clk | ultra slow clk |
|---|---|---|---|
| ext_flash | √ | | |
| ext_ram | √ | | |
| display | √ | | |
| sd_controller_0 | √ | | |
| audio_spi | √ | | |
| audio_i2s | √ | | |
| audio_dma | √ | | |
| uc_spi | √ | | |
| high_res_timer | √ | | |
| jtag_uart_0 | | √ | |
| timer_0 | | √ | |
| adc_spi | | √ | |
| display_power_enable | | | √ |
| pb_int | | | √ |
| user_pb | | | √ |
| user_led | | | √ |
| sd_card_io | | | √ |
| user_io | | | √ |
| battery_charge_indicator | | | √ |
| battery_low_indicator | | | √ |
| burst_mode_power | | | √ |
| sopc_clock_sel | | | √ |
| sopc_clock_disable | | | √ |

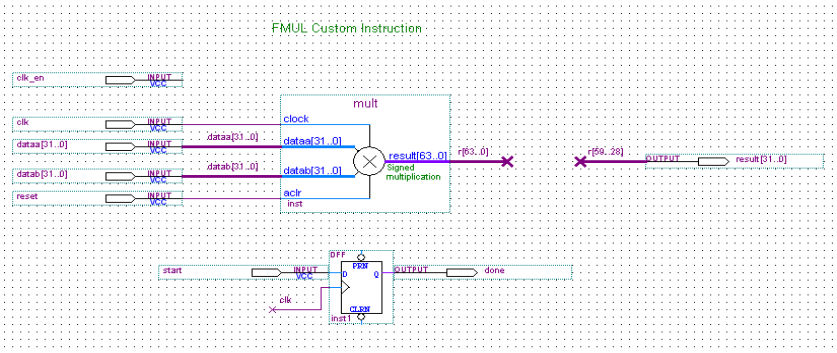
## Nios II Custom Instruction

The MP3 player application makes use of the MAD MP3 decoder to decode compressed MP3 data into hi-fidelity quality audio. The decoder makes significant use of a multiply and shift function. This function was used extensively throughout the decoder and occupied a significant portion of the processors bandwidth.

The MAD decoder is intended to run on processors at a significantly higher frequency than 80 MHz and as such they have the ability to perform the decode as a software algorithm. Nios II does not have enough processing capability to do so at 80 MHz. Nios II does however have the ability to customize its instruction set. A Custom Instruction (see Figure 8) was created and added to the LPRP SOPC Builder design. With the addition of this accelerator Nios II had the performance of a processor running at 100's of Mhz, while only using a few hundred mW.

The LPRP MP3 player application was based on work done by Nate Knight @ Altera Corporation

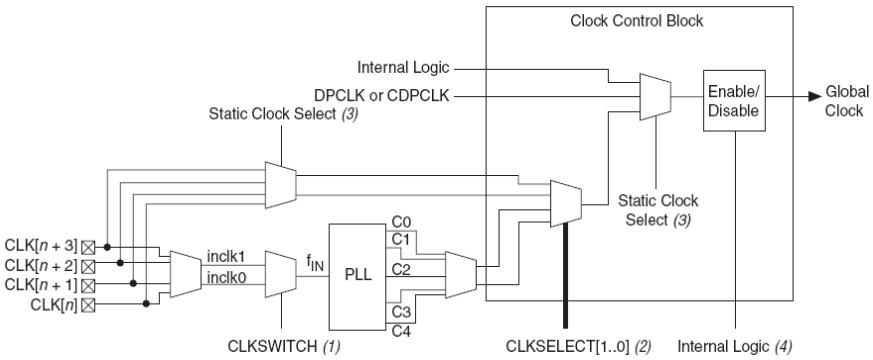
Figure 8. Fmul Custom Instruction




## Clock Control

Significant amounts of power can be saved by reducing or turning off clocks to logic that is not being utilized. The LPRP makes use of the Clock Control Blocks in the Cyclone III devices. Two PIO peripherals in the LPRP memory map are used to do this. The *sopc_clock_sel* peripheral controls the CLKSELECT inputs shown in Figure 9. The *sopc_clock_disable* peripheral controls the Enable/Disable portion of the Clock Control Block.

*Figure 9. Cyclone III Clock Control Block*

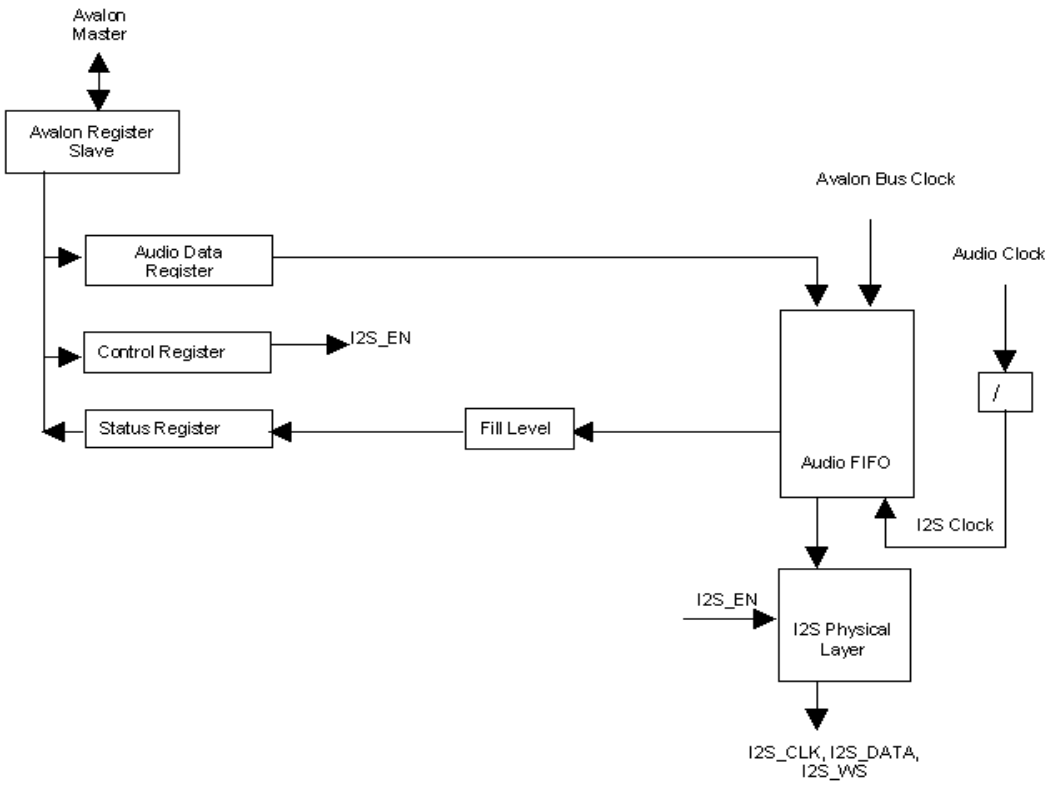


## New SOPC Builder Peripherals

### I2S Interface

The I2S Interface was created by Phillips and is used to transport stereo audio content between semiconductor devices. It is comprised of a 3 wire interface. I2S_CLK, I2S_DATA and I2S_WS.

*Figure 10: Avalon I2S Tx Functional diagram*




This IP core supports 16 bit audio resolution at a 44.1kHz or 48 kHz sample rate. The Audio clock that drives this core must be derived from the following equation

Audio clock = 16 x 2 x sample rate x 8

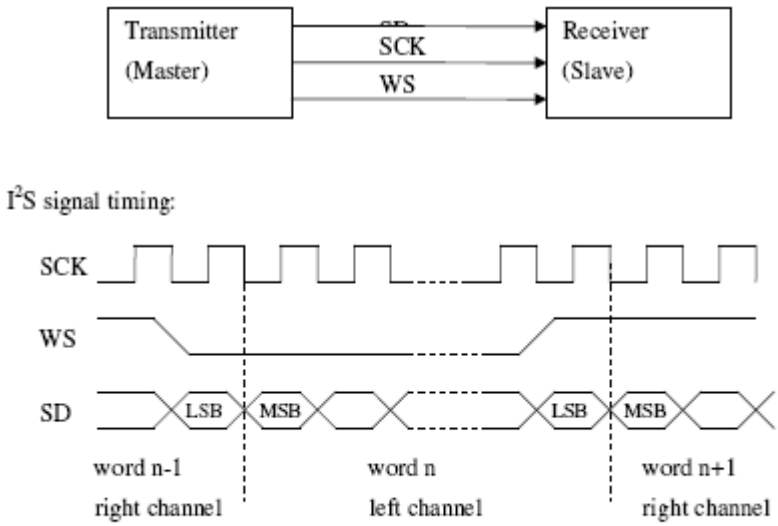
Audio clock = 11.2896MHz @ 44.1 kHz
Audio clock = 12.288MHz @ 48 kHz

The I2S_SCK is used to clock half words of data serially out of the I2S physical layer. I2S_WS identifies whether the half word is for a left or right audio channel. When I2S is low it indicates that the current I2S_DATA is for the left audio channel.

Data is written in words into the Audio FIFO via the Audio Data Register (ADR). A complete left and right audio sample is contained in one word. The data is shifted MSB first from the outbound side of the FIFO. Left channel audio samples must be written into the most significant half of the ADR and right audio samples into least significant half.

The Audio FIFO is 256 samples deep. It will set a FIFO low flag if it contains less than 128 audio samples. This flag can be polled through the status register or will generate an interrupt if interrupts are enabled. The interrupt can be cleared by reading the Interrupt Status Register (ISR).

*Figure 11: I2S Transmit Protocol*



## Register Interface

Table 16 shows the Avalon Slave Registers. All registers are 32-bit aligned to a word boundary. Unused bits should be written as zero.

*Table 16. Avalon Slave Registers*

| Address (h) | Access | Mnemonic | Name |
|---|---|---|---|
| 00 | W | ADR | Audio Data Register |
| 04 | W | CR | Control Register |
| 08 | R | SR | Status Register |

### Audio Data Register (ADR)

Table 17 shows the audio data register format.

*Table 17. Audio Data Register Format*

| Bits | Mnemonic | Description |
|---|---|---|
| 15..0 | ASR | Right channel audio sample |
| 31..16 | ASL | Left channel audio sample |

Audio samples must be written as a word to the ADR.

### Control Register (CR)

Table 18 shows the status register format.

*Table 18. Control Register Format*

| Bit | Mnemonic | Description |
|---|---|---|
| 0 | I2S_EN | Enable the I2S physical layer |

### Status Register (SR)

Table 19 shows the status register format.

*Table 19. Status Register Format*

| Bit | Mnemonic | Description |
|---|---|---|
| 0 | FL | This bit indicates if the level of the FIFO is low |

## Display Interface

The OSRAM OLED Display includes an intelligent controller. This is mapped into the SOPC system memory map and can be accessed via read or write I/O instructions from the Nios II CPU.

Table 20 shows these registers. All registers are 32-bit aligned to a word boundary. Unused bits should be written as zero. The Data Register allows access to the displays GDDRAM table. The Control Register allows access to the displays Command Table.

*Table 20. Avalon Register Slave Registers*

| Address (h) | Access | Mnemonic | Name |
|---|---|---|---|
| 00 | W | CR | Control Register |
| 00 | R | SR | Status Register |
| 04 | RW | DR | Data Register |

# 4

# Device Drivers

For further documentation on the SD, Graphics or Power Management Drivers please contact XAPLOS.

# 5

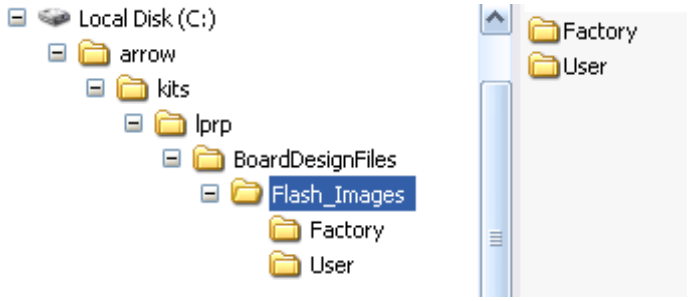
# Utilities

The LPRP is provided with a few script utilities that assist with restoring the board hardware and software images. They can also be used to flash your own .elf and .sof flash files into this design

The utilities can be found in the \arrow\kits\lprp\BoardDesignFiles\Flash_Images sub-directory. These utilities are shown in Figure 11.

Instruction on how to restore the factory image or add a new user image are shown below.

*Figure 11: Flash Utilities*

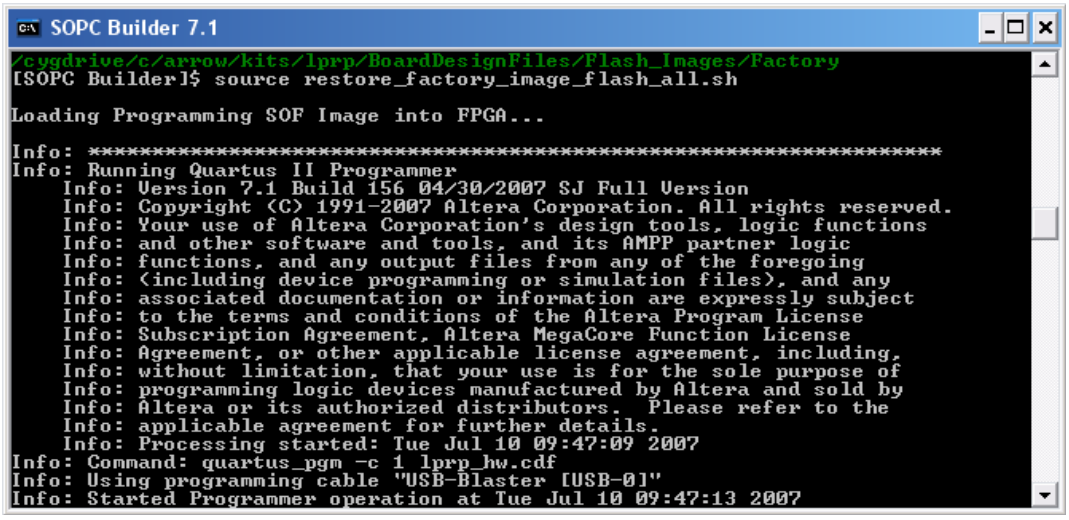


## Restoring the factory image

Follow the steps below to restore the factory LPRP image .

1.Connect a USB cable to the LPRP.
2.Turn the LPRP Power On.
3.Open a Nios II 7.1 Command Shell
4.CD to the c:\arrow\kits\lprp\BoardDesignFiles\Flash_Images\Factory directory
5.At the command prompt type the following command

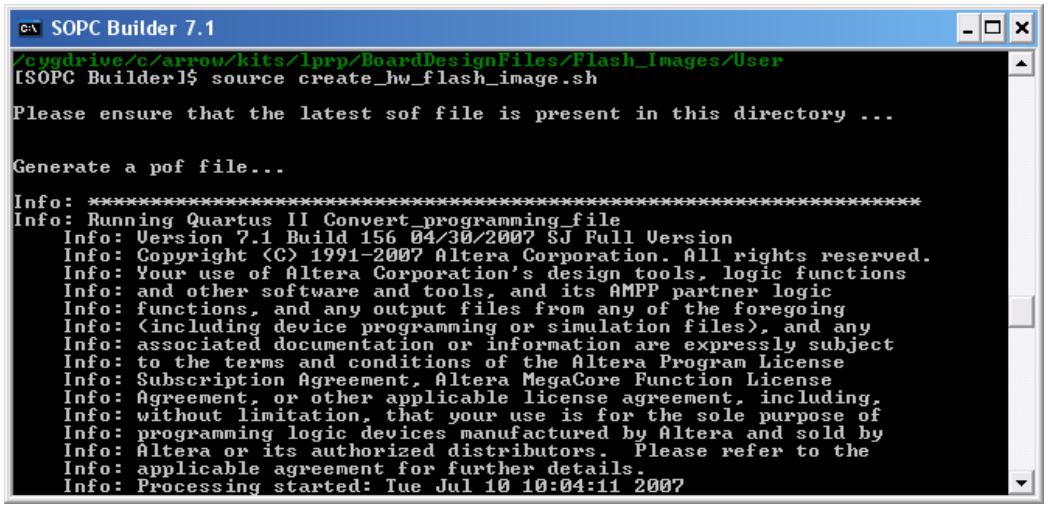
```
[SOPC_Builder]$ source restore_factory_image_flash_all.sh
```



## Preparing a new User Hardware Image

1.Copy the *.sof* file from the Quartus II project directory, rename it *low_power_demo.sof* and drop it in the User folder.
2.Open a Nios II 7.1 Command Shell
3.CD to the c:\arrow\kits\lprp\BoardDesignFiles\Flash_Images\User directory
4.At the command prompt type the following command

```
[SOPC_Builder]$ source create_hw_flash_image.sh
```

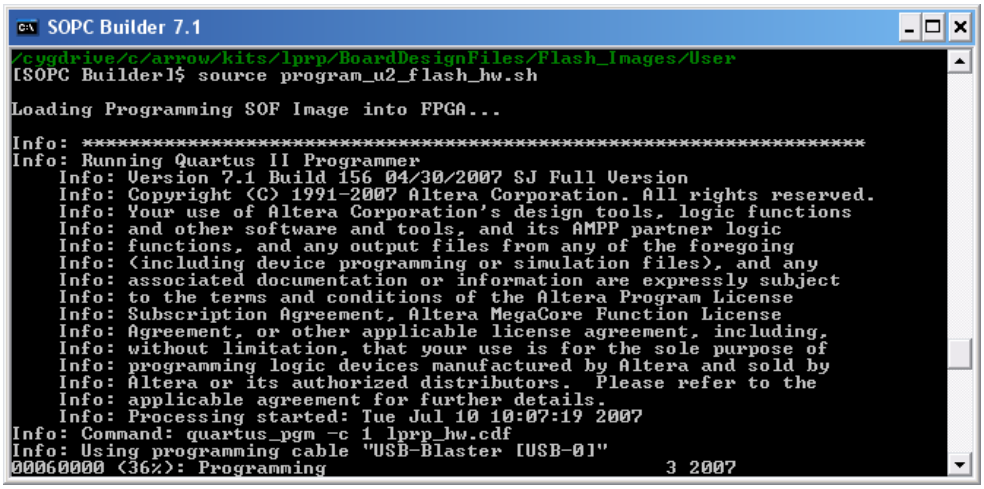


## Programming a new User HW Image

Follow the steps below to program a new user hw image into the LPRP .

1.Connect a USB cable to the LPRP.
2.Turn the LPRP Power On.
3.Open a Nios II 7.1 Command Shell
4.CD to the c:\arrow\kits\lprp\BoardDesignFiles\Flash_Images\User directory
5.At the command prompt type the following command

```
[SOPC_Builder]$ source program_u2_flash_hw.sh
```



## Preparing a new User Software Image

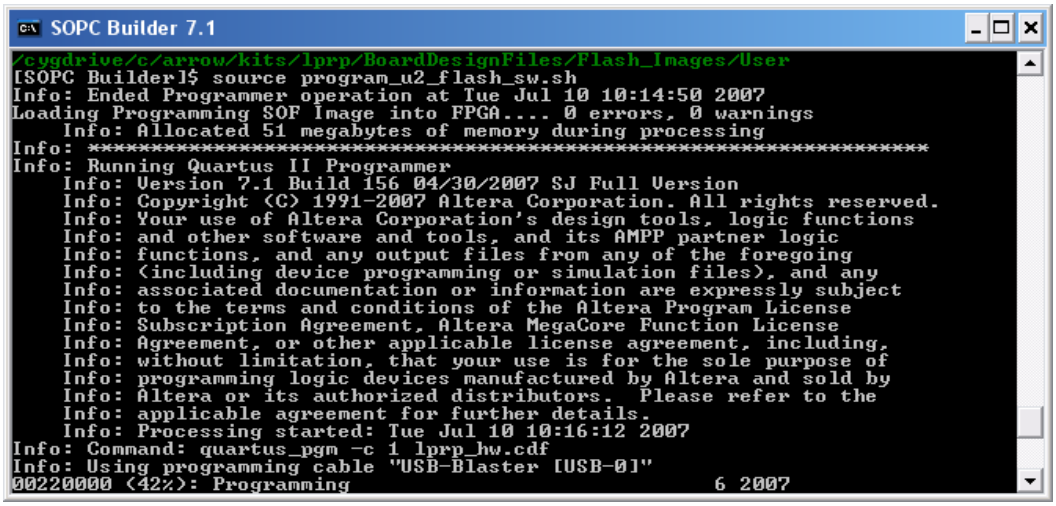
1.Copy the *ext_flash.flash* file from the <software>\Debug directory, rename it *lprp_sw.flash* and drop it in the User folder.

## Programming a new User SW Image

Follow the steps below to program a new user hw image into the LPRP .

1.Connect a USB cable to the LPRP.
2.Turn the LPRP Power On.
3.Open a Nios II 7.1 Command Shell
4.CD to the c:\arrow\kits\lprp\BoardDesignFiles\Flash_Images\User directory
5.At the command prompt type the following command

```
[SOPC_Builder]$ source program_u2_flash_sw.sh
```



# 6

# Support

For further support please contact your local Arrow FAE.

You may also post a comment or question at the Nios Message Forum

Call 800-777-2776 to connect to the nearest Arrow branch.

Click to locate a specific Arrow branch