XILINX®

XAPP363 (v1.0) October 17, 2001

Handheld Sonic Access Module™

Summary

This document describes the implementation of the Sonic Access Module™ (SAM) design submitted to the recently publicized "Cool Module Design Contest". All development for this contest was performed using the Insight Springboard™ development platform which allows for rapid development of Handspring modules. This development platform incorporates the reprogrammable Xilinx CoolRunner™ XPLA3 CPLD and uses the Handspring Visor™ PDA expansion slot. Low power CoolRunner CPLDs are the ideal programmable logic solution for portable, handheld applications.

The SAM design is a portable audio processing module that can be used in applications ranging from real-time spectrograms to speech analysis. The CPLD and Handspring design files for the SAM are available and can be found at **Download Pack**, page 19.

A visual presentation of the SAM is provided to show operation and functionality. This on-demand video is available at: **http://www.xilinx.com/apps/video.htm**.

Introduction

In the winter of 2001, Xilinx, Handspring and Portable Design Magazine collaborated on a design contest to highlight the use of Xilinx CoolRunner XPLA3 CPLDs in quickly developing Springboard modules for the Handspring Visor PDA. About 250 contest registrations were received and nearly 100 contest design ideas were submitted. From these submissions, ten were chosen to receive a Handspring Visor and an Insight Springboard development kit. These ten finalists were given three months to complete their designs in order to compete for a "winner takes all" grand prize of $10,000. All finalists were required to included a written description of the project, all design files, and the necessary software to make their Springboard module prototypes operate. All the finalists did a great job. This application note is derived from the submission supplied by David Coode, Dustin Griesdorf, Jakob Nielsen, Todd Schneider, and Mel Witter of Dspfactory Ltd. in Waterloo, Ontario, Canada.

Appendix A outlines existing Xilinx application notes that are appropriate for understanding this application note. These are available on the Xilinx website at: **http://www.xilinx.com/**.

Design Description

The SAM Spectral Analyzer is one example use of the SAM. It is designed to display real-time data received from the Dspfactory’s SignaKlara™ chip set, perform a frequency analysis, and update the Handspring Visor PDA display. More information on Dspfactory can be found at: **http://www.dspfactory.com/**. More information on the SAM can be found at **http://www.dspfactory.com/applications/pda/sam.html**.

The Sonic Access Module (SAM) is a portable audio processing module to be used with the Handspring Visor. The module incorporates the Dspfactory's SignaKlara audio DSP chipset for all audio processing tasks. The SignaKlara chipset consists of an ultra low-power programmable DSP processor (the DELTA-2) and an ultra low-power mixed-signal device (the ALPHA). The Xilinx CoolRunner XPLA3 CPLD interfaces the SignaKlara technology, an onboard flash memory, a SmartMedia memory device, the Visor, and the Power Management Unit. The Visor microphone provides audio input and a 1/8" headphone jack provides audio output. An additional stereo audio input is also provided.

The audio input is converted to digital samples by the ALPHA. All audio processing (including Weighted Overlap-Add filtering) is done on the DELTA-2, which incorporates a dual-Harvard DSP core optimized for audio processing applications. The DSP can use the Flash memory for

© 2001 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at **http://www.xilinx.com/legal.htm**.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

storage of algorithmic libraries or audio data. The DSP produces data output that is packaged by the CoolRunner CPLD and sent to the Visor PDA to display. The DSP also produces audio output that goes through the ALPHA's D/A, where is converted to an analog signal and output through an amplifier into a headphone jack for listening via a headset.

The module is capable of analyzing audio for display of a real-time spectrogram or a spectrum analysis on the Visor screen. For this application, the DSP performs calibrated analyses on the audio data from the microphone and this data is serially sent to the CoolRunner. The CoolRunner sends the data to the Visor, which generates a real-time display of the data in the form of a spectrogram or spectral analyzer.

For real-time processing algorithms, the Visor passes any necessary control signals to the DSP, which is programmed with any appropriate algorithm. Possible applications are listed in section **End Module Applications**, page 3.

Palm Spec: A Spectrum Analyzer Demonstration Application

The Spectrum Analyzer application employs the WOLA processor filterbank co-processor on the DELTA-2 to perform frequency analysis in real-time and real-time power calculations on the WOLA analysis results are made on the Rcore DSP core, on the DELTA-2. The audio signal source is selectable between an external input and the Visor microphone. The frequency resolution is selectable between 32 and 64 uniform bands over a constant bandwidth of 8 KHz. Furthermore, a selectable range of preamplifier settings accommodates a wide range of input signal conditions. Figure 1 shows the Spectrum Analyzer operating on a Handspring Visor Prism.

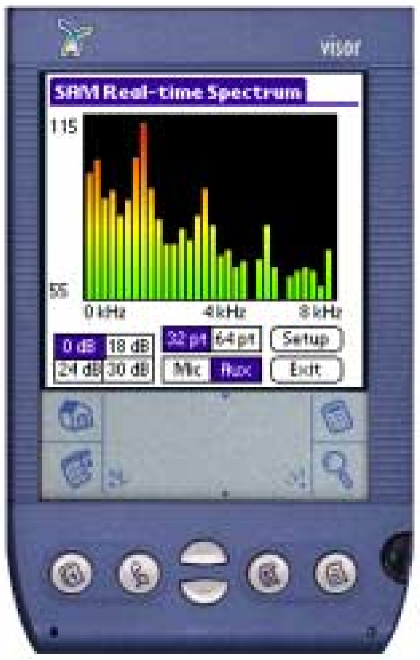

Figure 1: **Real-Time Spectrum Analyzer**

System Design Overview

The Spectrum Analyzer design employs both software and hardware components. The Visor application communicates with the DSP chipset through GPIO (General-Purpose Input/Output) lines for control and booting and through an SPI interface for real-time data transfer.

The CPLD sets the GPIO values, acts as an interface controller for the SPI port, contains an interrupt control unit, and controls system clocking. It receives real-time spectral data from the SPI port, frames that data, and generates interrupts. The Visor application reads the data from the CPLD synchronous with the received interrupts and displays the data.

A more detailed description of the CPLD and Spectrum Analyzer software is presented later in this application note (see **CPLD Design**, page 10 and **Handspring Software Design**, page 14).

End Module Applications

The complete SAM module provides the following functionality:

Audio Analysis Applications

1. Real-Time Spectrum Analyzer
2. Real-Time Color Spectrogram

Real-time Processing Algorithm Applications

3. Audio Equalizer ("EQ")
4. Sound Effects Processing
5. Noise Reduction / Speech Intelligibility Enhancement

Speech Interface Services

6. Text-to-Speech (Speech Synthesizer)
7. Voice Command Module
8. Voice-to-Text Dictation

Other Applications

9. Signal / Tone Generation
10. Audio CODEC for long storage or low-resource transmission of audio

The SAM module can be used in the following applications:

Audio Analysis Applications

1. Tuning acoustic environments (Halls, Studios, etc.)
2. Visual signal analysis
3. SPL measurements (e.g. for noise level monitoring)

Real-time Processing Algorithm Applications

4. Audio studio testing and configuration
5. Conversation in noisy environments
6. Guitar and other audio effects

Voice Services

7. Personal Voice Memo and dictation recorder
8. Voice Command Interpreter
9. E-Mail Audio Reader (for listening to emails while driving, for example)

Other Applications

10. Field audio and electrical test equipment
11. Master hearing aid

Operating Instructions

The Sonic Access Module design can be downloaded from the Xilinx website. The download pack includes the necessary files needed to run the SAM, see **Download Pack**, page 19.

Starting the Application

To start the Spectrum Analyzer application:

1. Download the PalmSpecApp.prc application (included in the download pack, see **Download Pack**, page 19).
2. Choose an input source (such as a CD player). Connect the input source to the RCA audio jack on the top of the module.
3. Choose a listening device (such as PC speakers). Connect the listening device to the 1/8" stereo jack on the top of the module.
4. Ensure that the Card Detect switch (SW2-6) is set to "On". Ensure that SW1-7 is set to "Off".
5. Carefully insert the module. The LED L3 should illuminate.
6. Run the Palm Spec application.
7. The system will boot and begin in default spectral power display mode. Audio output will be active — you should hear something on the listening device you have connected.

Display Modes

The Spectrum Analyzer display is available in either a standard spectral power plot (energy vs. frequency), as seen in Figure 1, page 2, or as a Spectrogram that plots frequency vs. time (scrolling) and energy is represented by color (red indicates high energy). Touching the central display window on the Visor screen toggles between these two visual representations.

In the standard spectral power plot, the vertical axis labels show the dB values being displayed. The horizontal axis shows the frequency range. A peak value indicator is shown atop each power bar. The peak value decays slowly, allowing the user to observe peak power in particular bands with rapid amplitude changes.

In the Spectrograph plot, the vertical axis represents frequency from 0 to 8 KHz. The frequency grows upwards. The horizontal axis shows time. The display scrolls to the left and new spectral information is drawn from the right side.

Input Modes

Two input modes are available: Mic and Aux. The desired input mode is selected by pressing one of two buttons on the PDA display. The current input selection is shown highlighted. The microphone input utilizes the built-in Visor microphone as an input source and is selected by pressing "Mic". The auxiliary input takes a standard audio input source such as a CD player or sound card. It is selected by pressing "Aux".

Audio Output

An audio output signal is provided through a 1/8" stereo jack atop the hardware implementation. This signal may be fed into standard amplified speakers, a stereo system's auxiliary input, or headphones.

To use the audio output jack, ensure that the module is not inserted in the Visor or that the Card Detect switch (SW2-6) is set to "Off".

Pre-amplification

Four pre-amplification settings are available: 0 dB, 18 dB, 24 dB, and 30 dB. The desired gain is selected by pressing one of the corresponding buttons. The current gain in the system is shown highlighted on the display.

For auxiliary inputs (through the RCA audio jack) such as a CD player or sound card, 0 dB of input gain is recommended. Quiet sources such as the Visor microphone require higher amplification. For the Visor microphone, 24 dB of gain is recommended.

Band Selection

The frequency resolution is selectable between 32 bands and 64 bands. The desired resolution is selected by pressing one of the corresponding buttons. The current band selection is shown highlighted.

Setup

Touching the "Setup" button allows the application to be configured. The minimum and maximum energy levels displayed on the Spectrum Analyzer are configurable. This is done by entering values in the Max dB and Min dB fields. Also, the peak-level display can be toggled on and off.

For the Spectrograph, an option is available to zoom the display. The zoom option is applicable to a 32-band spectrograph. The option is ignored for 64-band displays.

When the configuration options have been set, touch the "Done" button to return to the main screen.

Hardware Description

The completed hardware implementation of SAM is an extremely low-power DSP module capable of a multitude of applications. Two elements of the design are required to achieve low-power consumption during operation. The Xilinx CoolRunner CPLD acts as a system controller and communications interface. The low-power nature of the CoolRunner XPLA3 CPLD is ideal for the module. The second element is Dspfactory's ultra low-power SignaKlara DSP chipset. The complete module will contain dual DSP systems from Dspfactory, running at a clock frequency of 5.128 MHz. This will provide operation equivalent to 50-60 MIPS with both DSP systems on board. The contest submission provides one DSP device (the DELTA-2) and one mixed-signal device (the ALPHA).

In order to achieve ultra low-power operation, SignaKlara technology operates at a nominal voltage of 1.2V. Thus, in order to provide communications between the CPLD and DSPs, voltage level translation is required. In addition, non-volatile memory is included on the module. Two Flash chips, one for booting and one for data storage, are included in the design. A UART is provided for serial communications, and an output driver capable of 100 mW of power output is included. Figure 2 shows the system diagram for the complete module.

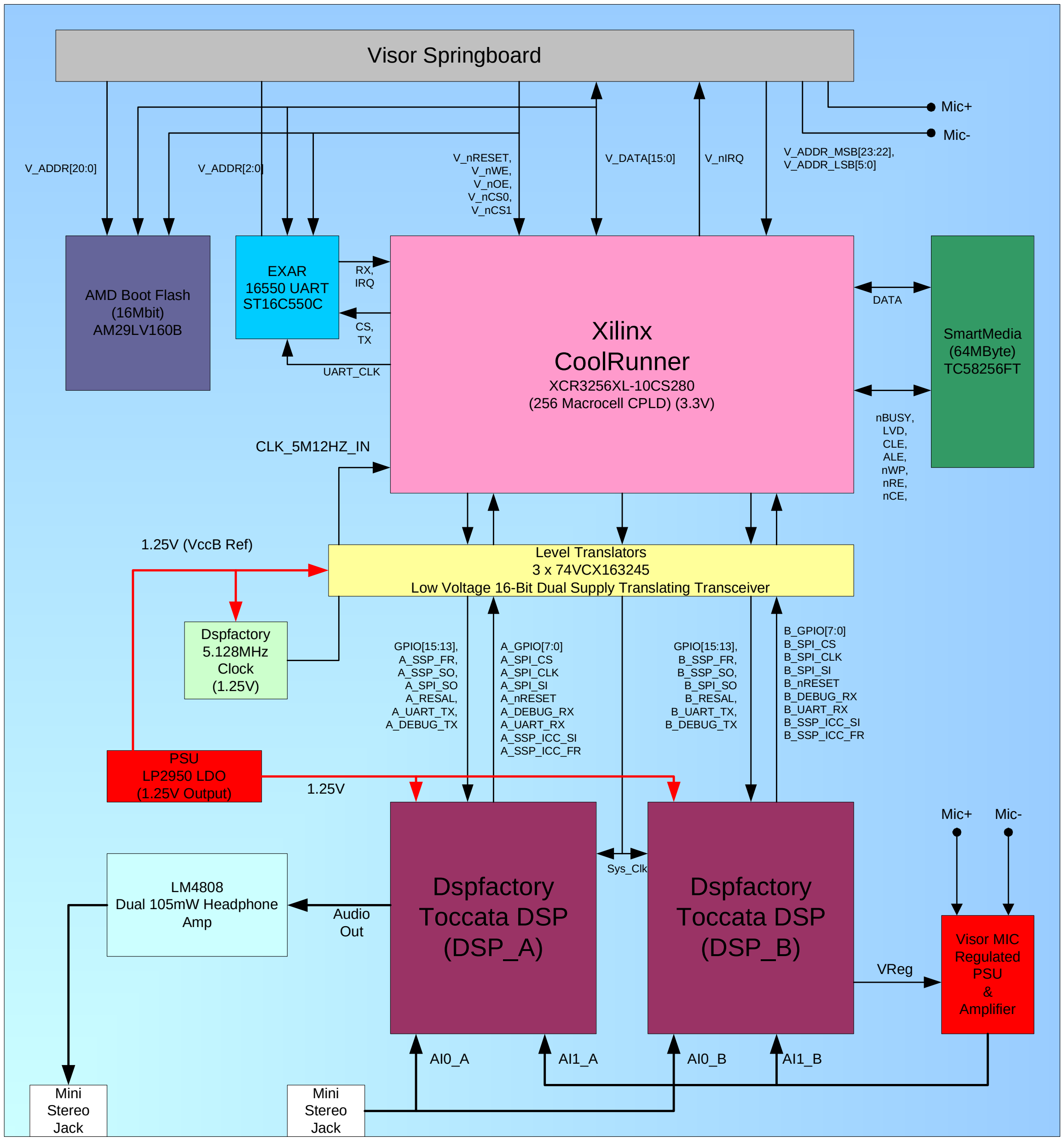
Visor Springboard
V_ADDR[20:0]
V_ADDR[2:0]
V_nRESET, V_nWE, V_nOE, V_nCS0, V_nCS1
V_DATA[15:0]
V_nIRQ
V_ADDR_MSB[23:22], V_ADDR_LSB[5:0]
Mic+
Mic-
AMD Boot Flash (16Mbit) AM29LV160B
EXAR 16550 UART ST16C550C
RX, IRQ
CS, TX
UART_CLK
Xilinx CoolRunner
XCR3256XL-10CS280 (256 Macrocell CPLD) (3.3V)
DATA
SmartMedia (64MByte) TC58256FT
nBUSY, LVD, CLE, ALE, nWP, nRE, nCE,
CLK_5M12HZ_IN
1.25V (VccB Ref)
Level Translators
3 x 74VCX163245
Low Voltage 16-Bit Dual Supply Translating Transceiver
Dspfactory 5.128MHz Clock (1.25V)
GPIO[15:13], A_SSP_FR, A_SSP_SO, A_SPI_SO, A_RESAL, A_UART_TX, A_DEBUG_TX
A_GPIO[7:0], A_SPI_CS, A_SPI_CLK, A_SPI_SI, A_nRESET, A_DEBUG_RX, A_UART_RX, A_SSP_ICC_SI, A_SSP_ICC_FR
GPIO[15:13], B_SSP_FR, B_SSP_SO, B_SPI_SO, B_RESAL, B_UART_TX, B_DEBUG_TX
B_GPIO[7:0], B_SPI_CS, B_SPI_CLK, B_SPI_SI, B_nRESET, B_DEBUG_RX, B_UART_RX, B_SSP_ICC_SI, B_SSP_ICC_FR
PSU LP2950 LDO (1.25V Output)
1.25V
Mic+
Mic-
LM4808 Dual 105mW Headphone Amp
Audio Out
Dspfactory Toccata DSP (DSP_A)
Sys_Clk
Dspfactory Toccata DSP (DSP_B)
VReg
Visor MIC Regulated PSU & Amplifier
AI0_A
AI1_A
AI0_B
AI1_B
Mini Stereo Jack
Mini Stereo Jack

Figure 2: **Module System Diagram**

Xilinx CoolRunner CPLD

The CPLD interfaces with the Visor PDA, dual DSPs, SmartMedia, and an external UART. It is clocked from a 5.12 MHz crystal oscillator. A complete description of the CPLD design is given in the section **CPLD Design**, page 10. The CPLD operates a 3.3V, making it ideal to connect V_{CC} of the CPLD to the Springboard V_{CC} power.

Non-Volatile Memory

The module contains two Flash storage chips. An AMD Boot flash (16 Mbit) is included. Upon module insertion, the Visor will download startup code from the Boot flash. Applications and the SAM module library will be downloaded into Visor memory for execution. This flash is only used immediately following module insertion.

The module also contains a 64 Megabyte SmartMedia Flash. SmartMedia was chosen for several reasons. First, it contains only 16 data and control lines. Thus simplifying routing in the

CPLD. This is important due to the large size of the CPLD design. Further, the SmartMedia interface is an 8-bit parallel streaming protocol. This fits in well with the DSP's streaming interface (SSP_ICC). The CPLD coordinates access between the SSP_ICC interface and the SmartMedia.

Level Translator

The module contains three 16-bit dual-supply translating transceivers. Each transceiver contains two V_{CC} inputs, V_{CCA} and V_{CCB}. V_{CCcA} is the 3.3V voltage domain, connected to the Visor's power lines. V_{ccB} is the low-voltage reference. It is connected to the module's low-power line (1.25V). Two of the parts are configured for 1.25V to 3.3V voltage translation. This allows signals from the DSPs to be connected to the CPLD. One of the parts is configured for 3.3V to 1.25V voltage translation. This allows signals originating from the CPLD to be connected to the low voltage SignaKlara DSP system.

Power Supply

The module contains a micropower voltage regulator to provide the DSPs and level translators with a low-power line. The regulator is programmed to provide 1.25V. Further, to properly power the Visor MIC, Dspfactory's DSP chip provides a regulated voltage signal, V_{REG}. This power supply is further filtered and provided to the microphone. A quiet power supply is essential for low-noise microphone performance.

UART

A low-power, low-voltage UART from EXAR is included on the module to allow Visor applications to communicate with the DSP using the standard serial libraries. The UART is connected to the Visor bus. The CPLD provides the chip-select signal and a properly divided clock to the UART. Since four serial ports are available between the two DSPs, the CPLD design will mux the UART signals to their proper destinations.

Headphone Amplifier

The output stage of the module consists of a headphone amplifier connected to the SignaKlara's audio outputs. The power supply for the headphone amplifier must be well-filtered to provide low-noise performance.

Submitted Design

The prototype hardware design submitted for the Xilinx design contest is shown in Figure 3. The prototype design shown in Figure 3 is the design available for download from the Xilinx website.

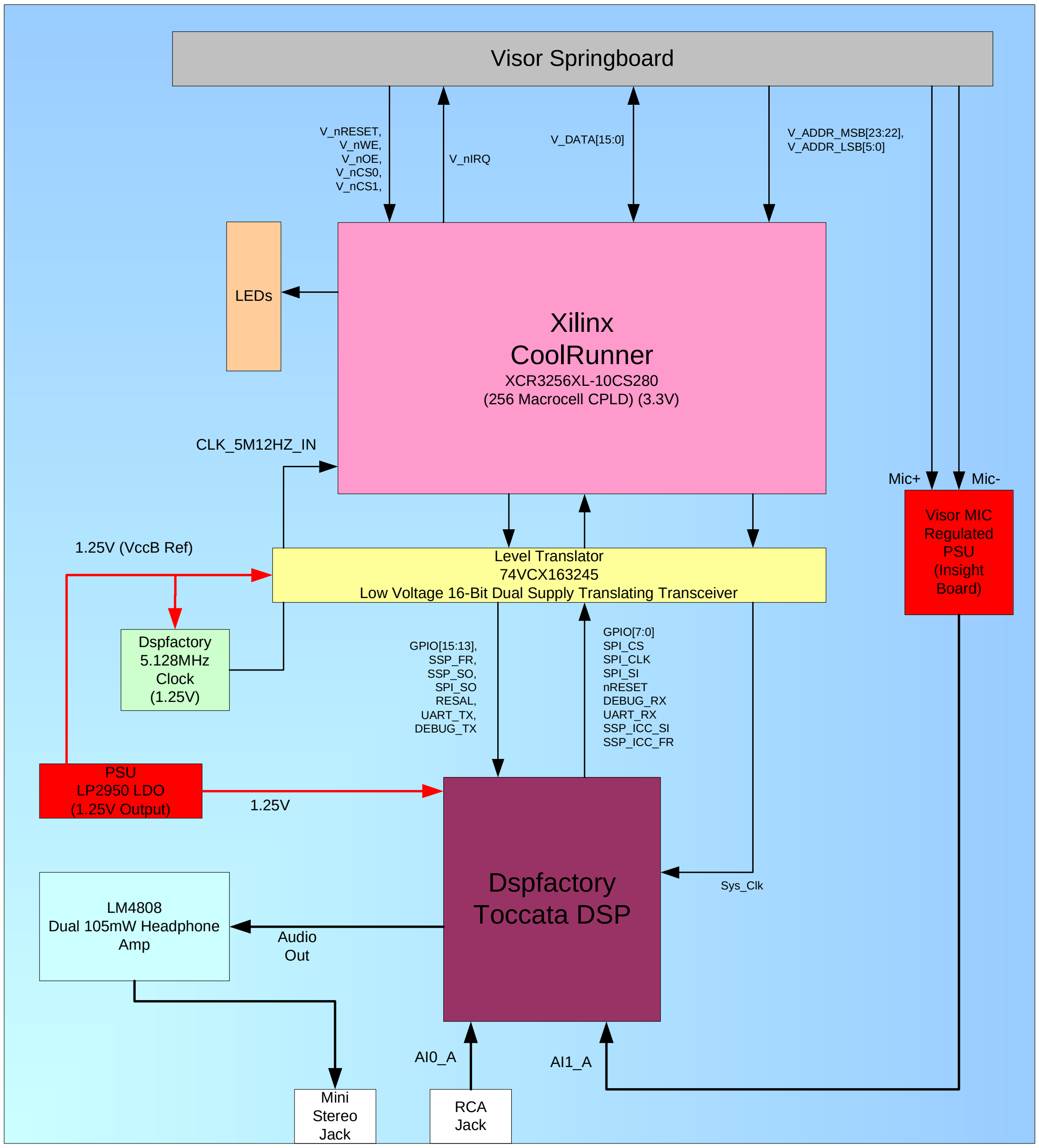
Visor Springboard
V_nRESET, V_nWE, V_nOE, V_nCS0, V_nCS1,
V_nIRQ
V_DATA[15:0]
V_ADDR_MSB[23:22], V_ADDR_LSB[5:0]
LEDs
Xilinx CoolRunner
XCR3256XL-10CS280
(256 Macrocell CPLD) (3.3V)
CLK_5M12HZ_IN
Mic+
Mic-
Visor MIC Regulated PSU (Insight Board)
1.25V (VccB Ref)
Level Translator
74VCX163245
Low Voltage 16-Bit Dual Supply Translating Transceiver
Dspfactory 5.128MHz Clock (1.25V)
GPIO[15:13], SSP_FR, SSP_SO, SPI_SO RESAL, UART_TX, DEBUG_TX
GPIO[7:0] SPI_CS SPI_CLK SPI_SI nRESET DEBUG_RX UART_RX SSP_ICC_SI SSP_ICC_FR
PSU LP2950 LDO (1.25V Output)
1.25V
Dspfactory Toccata DSP
Sys_Clk
LM4808 Dual 105mW Headphone Amp
Audio Out
AI0_A
AI1_A
Mini Stereo Jack
RCA Jack

Figure 3: **Implemented System Design**

The Insight Springboard development board plugs into the module slot on the Visor. A vector board interface layer mounts (reverse side) on the headers of the Insight board. Finally, the Dspfactory DSP Hardware Platform mounts on the obverse side of the vector board interface. These three layers together form the complete submission of the hardware contest design as shown in Figure 4.

Sna

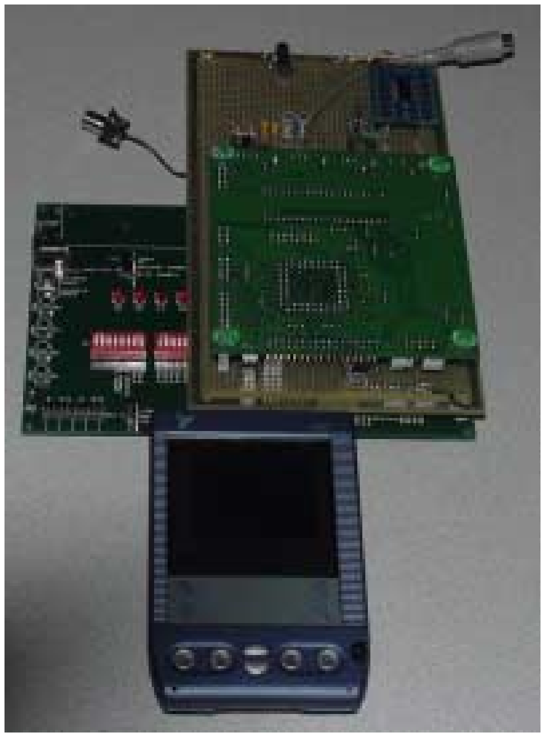

Figure 4: **System Hardware Setup**

The SAM prototype module implementation shown in Figure 4 could fit inside a standard slim Springboard Module Case, due to the small number of components and the form factor of these components.

The prototype SAM design uses the Springboard LEDs as status indicators. During normal operation, the DSP will toggle LED L3 every 0.4 seconds. This indicates that an algorithm is running. During data transfer, LED L0 will flicker indicating that the Visor is receiving interrupt requests and handling them. The system clock is brought out to LED L1. LED L2 is the active-low reset signal for the DSP. When it is off, the DSP is held in reset. The assignment of each of these LED is listed in Table 1.

Table 1: **Development Board LED Configuration**

LED	Description
L0	Module interrupt indication On: No interrupt Off: Interrupt pending
L1	System clock (5.128 MHz, 50% duty cycle)
L2	DSP Reset Signal (Active Low) On: System operating Off: Reset
L3	DSP Status Indication Blinking LED: DSP algorithm running Solid LED: DSP waiting for instruction

Two DIP switches on the Springboard are used in the hardware implementation. Card Detect (SW2-6) indicates that the module is present in the device. SW1-7 enables and disables the LED's to conserve power.

CPLD Design

The Xilinx CoolRunner XPLA3 CPLD serves as the central system controller, memory controller, and communications interface for the SAM module. The CPLD is responsible for handling the communication between:

- Two DSP subsystems
- One DSP and the Handspring Visor
- The SmartMedia memory and the DSP
- The SmartMedia memory and the Handspring Visor

The CPLD is able to communicate with these devices using three separate interfaces: SSP_ICC, SPI, and RS232.

The source code and implementation results for the CPLD design is included in the available download pack (see **Download Pack**, page 19).

The CPLD SAM design consists of various units each handling a specific function. Figure 5 shows the architecture for the CPLD. The architecture consists of an interface to the Visor, two control units, four interfaces, and three interface switches. The associated VHDL source code for each unit is shown in parenthesis.

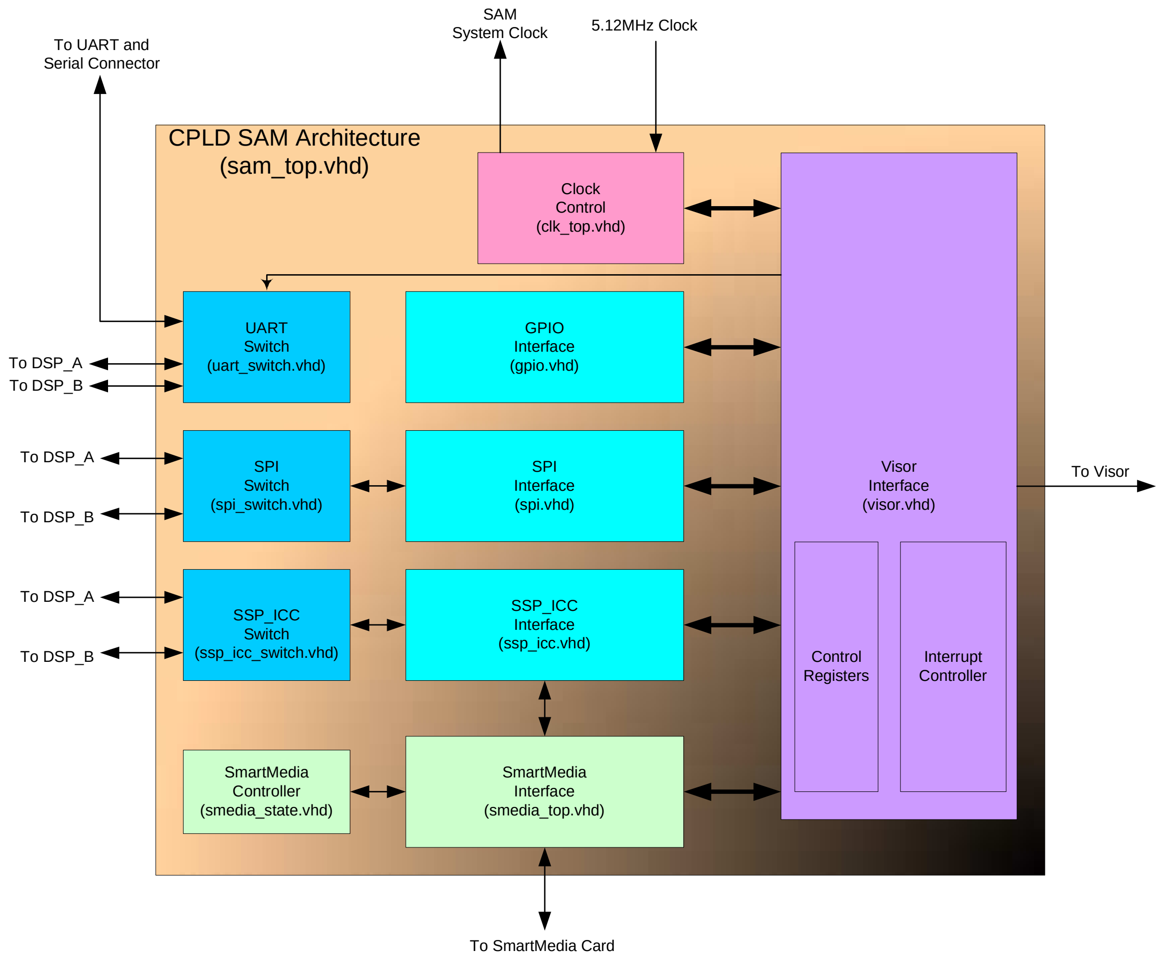
SAM System Clock
5.12MHz Clock
To UART and Serial Connector
CPLD SAM Architecture (sam_top.vhd)
Clock Control (clk_top.vhd)
UART Switch (uart_switch.vhd)
GPIO Interface (gpio.vhd)
To DSP_A
To DSP_B
SPI Switch (spi_switch.vhd)
SPI Interface (spi.vhd)
Visor Interface (visor.vhd)
To Visor
SSP_ICC Switch (ssp_icc_switch.vhd)
SSP_ICC Interface (ssp_icc.vhd)
Control Registers
Interrupt Controller
SmartMedia Controller (smedia_state.vhd)
SmartMedia Interface (smedia_top.vhd)
To SmartMedia Card

Figure 5: **CPLD Block Diagram**

A description of each CPLD design is described in the following sections.

UART Switch

The UART switch unit is responsible for controlling access to the various UARTs on the DSP. Each DSP contains two UARTs — Debug and Secondary. The SAM will contain an external low-power UART connected to the Visor bus. The CPLD will select which DSP and which serial port to connect to. Also, an external connector will be made available for debugging purposes. In debug mode, the external UART will be bypassed. Instead, the module can be connected to an external serial device so that the programmable DSP chips can be debugged.

SPI Switch and SPI Interface

The SPI interface (Serial Peripheral Interconnect) is a clocked synchronous interface. The SPI switch selects between the two DSP subsystems. The interface receives and sends data on the SPI clock edges. The interface is configured as an 8-bit or 16-bit interface. When data is received, it is latched into a storage register and an interrupt is generated.

The DSP uses the SPI interface to boot (power up). The CPLD in conjunction with the Visor software simulates booting from an EEPROM, from which the DSP was designed to boot. Due to the inherent latency in handling interrupts on the Visor, the CPLD gates the clock to the DSP subsystem through the clock control unit. Thus, the Visor is able to ensure that the SPI "send buffer" is filled and no data is missed during the boot process.

SSP_ICC Switch and SSP_ICC Interface

The SSP_ICC is a high-speed interface synchronous with the system clock. The switch can be configured to allow the DSP subsystems to communicate with each other through SSP_ICC port, or for one DSP to communicate with the Visor or SmartMedia. The SSP_ICC interface controls framing and storing data and generating an appropriate interrupt when the data has been received. The SSP_ICC interface can also be connected to the SmartMedia interface to allow the streaming of high bandwidth data.

Clock Control

The clock control unit takes care of enabling and disabling the system clock during the booting process. Several modes are available - clock enabled, clock disabled, or boot mode. Further, a clock divider may be implemented. A slower clock to the DSP decreases power consumption, and should be used with less computational intensive algorithms.

SmartMedia Control and Interface

The SmartMedia interface and associated control unit transfers streaming data to and from a SmartMedia memory. The interface is controlled by the Visor or by the SSP_ICC interface. This allows either the Visor or the DSP to read or write to the memory. The controller is implemented with a state machine and is driven by the system clock input.

Visor Interface

The Visor interface contains the appropriate logic for interfacing with the Handspring Visor bus. The Visor interface functionality includes address decoding and reading / writing to registers. Furthermore, the interface contains the module interrupt controller. The CPLD interrupt controller receives interrupts from various interface units and generates an interrupt signal to the Visor. An interrupt status register on the CPLD indicates which interrupt has been asserted. The various control registers are contained within the Visor interface unit.

The Visor controls the CPLD through various memory mapped registers. The implemented memory map is shown in Table 2.

Table 2: **SAM Memory Map**

L_ADDR_LSB[4:1]	Register	Function
0000	SAM_SYS_CTRL	Controls all module level functions, such as powering and resetting the DSPs.
0001	SAM_SWITCH_CTRL	Controls the internal switch configuration (UART, SSP_ICC, etc.).
0010	SAM_INTERRUPT	The module capable of generating multiple interrupts. The status of each is available in this register.
0100	SPI_DATA	Data for the SPI port.
0101	A_GPIO_IN	Data for the GPIOs to DSP "A".
0110	B_GPIO_IN	Data for the GPIOs to DSP "B".
0111	SSP_ICC_IN	Data for the SSP_ICC port.
1000	SMEDIA_CTRL	Control for SmartMedia.
1001	SMEDIA_DATA	Data for SmartMedia.

SAM System Control (SAM_SYS_CTRL)

The system control register controls system clocking, interface operation, and resetting the DSP subsystems. The various bit settings of the system control register are shown in Table 3.

Table 3: **SAM System Control Register Bit Map**

Bit	Name	SAM_SYS_CTRL ($0000)
0-1	sys_clk_ebl	00: Disabled 01: Enabled 10: Boot-mode
2	psu_ebl	0: PSU Off 1: PSU On
3	lt_vcc_ebl	0: V_{CCA} to level translators ON 1: V_{CCA} to level translators OFF
4	spi_bit_count_rst	Write "1" to reset
5	resal_a	Reset DSP_A (via Alpha_A)
6	resal_b	Reset DSP_B (via Alpha_B)
7-8	spi_mode	00: DSP SPI-boot mode 01: SPI streaming mode 10: SPI synchronous mode
9	smedia_ctrl	0: SmartMedia controlled by Visor 1: SmartMedia controlled by SSP
10	ssp_icc_mode	0: SSP_ICC mode auto 1: SSP_ICC mode step
11	clk_div	System Clock division factors

SAM Interrupt (SAM_INTERRUPT)

The SAM interrupt register indicates which interface or other unit has generated an interrupt. Writing a '1' to the proper bit will clear the interrupt. Table 4 shows the SAM interrupt register bit map.

Table 4: **SAM Interrupt Register Bit Map**

Bit	Name	SAM_ INTERRUPT ($0010)
0	interrupt_pending	0: Interrupt pending 1: No interrupt pending
1	spi_interrupt	0: SPI interrupt pending 1: No SPI interrupt
2	ssp_icc_interrupt	0: SSP_ICC interrupt pending 1: No SSP_ICC interrupt
3	s_media_interrupt	0: s_media interrupt pending 1: No s_media interrupt
4	wdog_interrupt_a	0: wdog_interrupt_a pending 1: No wdog_interrupt_a interrupt
5	ext_uart_interrupt	0: External UART interrupt pending 1: No external UART interrupt

SAM Switch Control (SAM_SWITCH_CTRL)

The SAM switch control register controls the operation of the interface switches. Table 5 shows the various configuration settings for the switch control register.

Table 5: **SAM Switch Control Register Bit Map**

Bit	Name	SAM_SWITCH_CTRL ($0001)
0-2	uart_switch	000: External UART to DSP_A Debug 001: External UART to DSP_A Debug 010: External UART to DSP_A Secondary UART 011: External UART to DSP_A Secondary UART 100: DSP_A to DSP_B Secondary UART
3-4	spi_switch	00: Select DSP_A SPI 01: Select DSP_B SPI 10: Disconnected 11: Unused
5-7	ssp_icc_switch	000: DSP_A to DSP_B (A master) 001: DSP_B to DSP_A (B master) 010: Select DSP_A SSP (Master) 011: Select DSP_A SSP (Slave) 100: Select DSP_B SSP (Master) 101: Select DSP_B SSP (Slave) 110: DSP_A to SMEDIA (Master) 111: DSP_B to SMEDIA (Slave)

The Xilinx CoolRunner CPLD architecture allows for a multitude of signal processing applications. All of the DSP's interface peripherals are available, in addition to the GPIO (General-Purpose I/O) signals. Depending on the specific application, an appropriate interface can be used, or a combination of different interfaces. In addition, with a transparent SmartMedia controller, a SmartMedia memory card can be used in the module to store large amounts of non-volatile data.

CPLD Implementation

The CPLD design architecture just described has be implemented and tested in VHDL using the Xilinx WebPACK™ software. The design is an ambitious one, using over 80% of a XCR3256XL CoolRunner CPLD. A summary of the fitting report is shown in Table 6. The modularity of the design allows for particular changes and fixes without affecting the pinout. This will allow VHDL modifications, additions, and fixes to be implemented without needing to re-layout the module PCB.

Table 6: **CPLD Device Resource Summary**

Resource	Available	Used	Utilization
Clock Inputs	4	4	100%
Global Control Terms	4	4	100%
Function Blocks	16	13	81.25%
I/O Pins	160	97	60.63%
Macrocells	256	206	80.47%
PLA Product Terms	768	432	56.25%
PLA Sum Terms	256	203	79.30%
Block Control Terms	128	36	28.13%
Foldback Nands	128	6	4.69%

Handspring Software Design

The SAM software support layer on the Handspring Visor PDA hides many of the implementation details from the end application developer. Using library routines, each DSP subsystem can be configured, booted, and controlled with minimal effort. The implementation details of the CPLD may change without affecting the Visor applications, maximizing software reuse.

Software Implementation

Note: Since this module is reconfigurable, there are many different scenarios for software flow. The following description is for the Spectral Analysis and Spectrogram Application.

The SAM software is organized into several categories. Figure 6 shows the SAM software organization. The boot flash will contain the SAM module library and DSP algorithms. The library code will boot the module, downloading code from its algorithm database on the DSP via the CoolRunner CPLD. The Handspring Visor application loads the library and issues commands to control the CPLD and module. Thus, the end application is hidden from the details of the module and CPLD implementation.

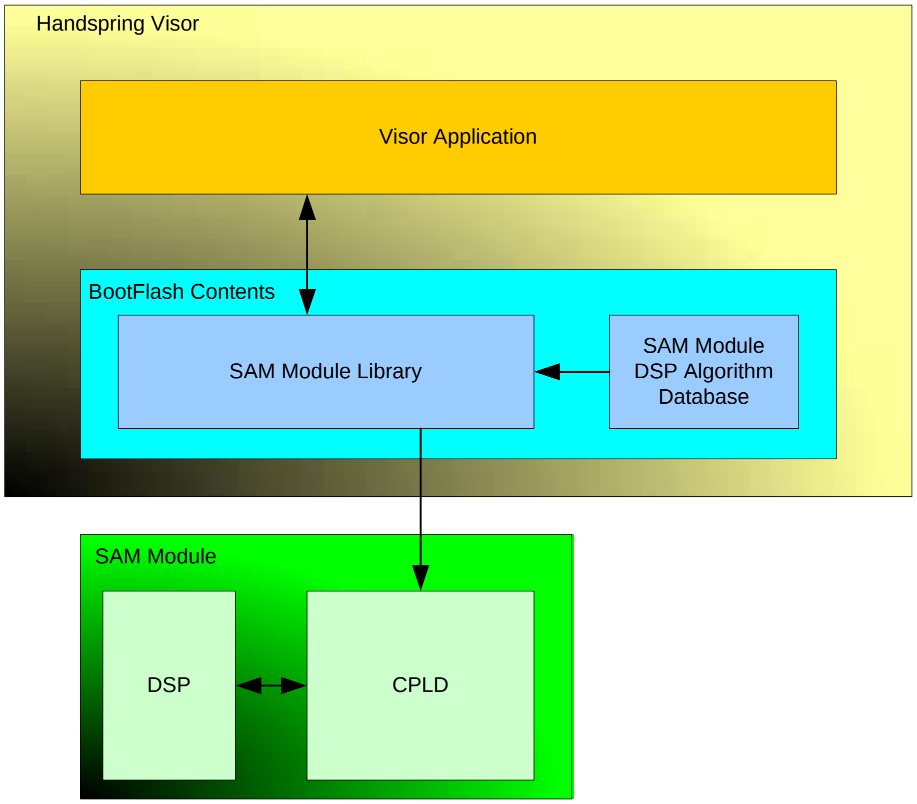

Figure 6: **SAM Software Organization**

This highly modular design incorporating multiple software levels allows for extraordinary re-programmability. The programmable nature of the DSP allows for a virtually unlimited set of audio processing algorithms to be dynamically configured and updated. The library interface allows for seamless upgrades to the hardware and CPLD design with few-to-no changes required in the Visor application.

The Spectral Analysis application is a color PalmOS executable designed to display real-time spectral data received from the module. The DSP on board the SAM module analyzes the incoming signal, performs FFTs and power calculations, and transmits the resulting data stream over the SPI port. There are several elements of the software design. The Handspring Visor application code is responsible for configuring the module, receiving the data, updating the screen, and accepting user input data. Configuration of the module involves downloading an algorithm to the Dspfactory programmable DSP core. Another component is the DSP algorithm code, which is downloaded to the DSP during the boot process.

Figure 7 shows the software architecture for PalmSpec, the Spectrum Analyzer application. The software consists of several modules, shown in Figure 7. Each of the modules has an associated source code shown in parenthesis that is available for download (see **Download Pack**, page 19).

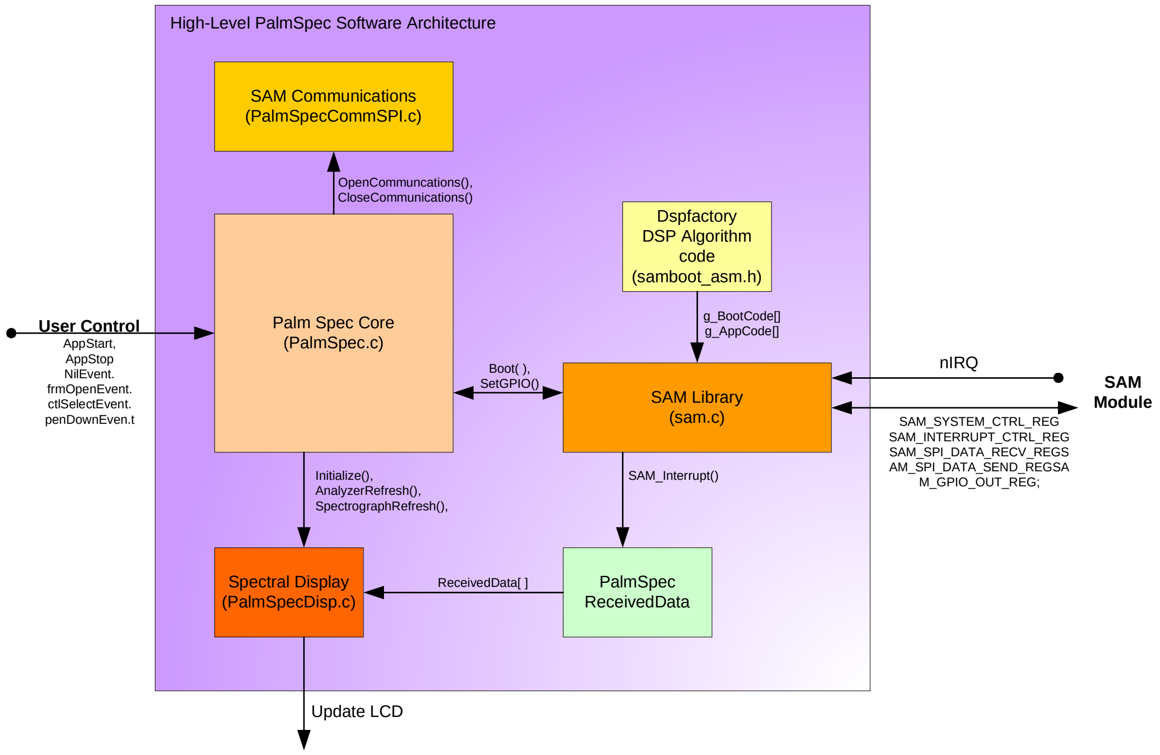
High-Level PalmSpec Software Architecture
SAM Communications (PalmSpecCommSPI.c)
OpenCommuncations(), CloseCommunications()
Dspfactory DSP Algorithm code (samboot_asm.h)
User Control
AppStart, AppStop NilEvent. frmOpenEvent. ctlSelectEvent. penDownEven.t
Palm Spec Core (PalmSpec.c)
g_BootCode[] g_AppCode[]
Boot( ), SetGPIO()
SAM Library (sam.c)
nIRQ
SAM Module
SAM_SYSTEM_CTRL_REG SAM_INTERRUPT_CTRL_REG SAM_SPI_DATA_RECV_REGS AM_SPI_DATA_SEND_REGSA M_GPIO_OUT_REG;
Initialize(), AnalyzerRefresh(), SpectrographRefresh(),
SAM_Interrupt()
Spectral Display (PalmSpecDisp.c)
ReceivedData[ ]
PalmSpec ReceivedData
Update LCD

Figure 7: **High Level PalmSpec Software Architecture**

The core of the application is contained within the PalmSpec.c file (see **Download Pack**, page 19). This module handles user input through messages received from the PalmOS. It received various events to the event dispatcher, which forwards the events to the appropriate handlers. The system is configured to generate NilEvents when no system event is pending. This allows the screen to be updated continuously. After a NilEvent is received, the application calls the appropriate refresh function depending on the display mode of the application.

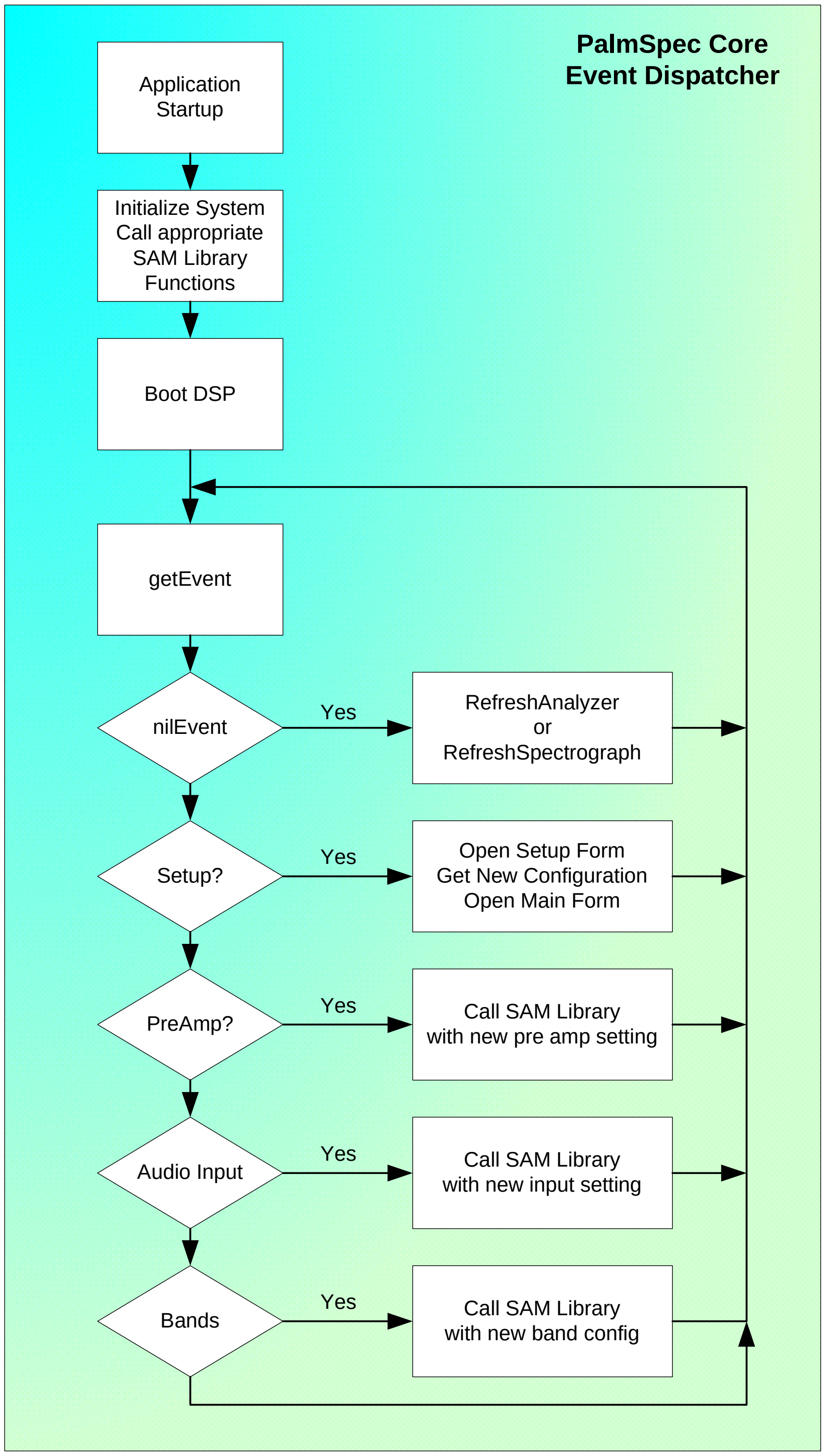
PalmSpec Core
Event Dispatcher
Application Startup
Initialize System Call appropriate SAM Library Functions
Boot DSP
getEvent
nilEvent
Yes
RefreshAnalyzer or RefreshSpectrograph
Setup?
Yes
Open Setup Form Get New Configuration Open Main Form
PreAmp?
Yes
Call SAM Library with new pre amp setting
Audio Input
Yes
Call SAM Library with new input setting
Bands
Yes
Call SAM Library with new band config

Figure 8: **PalmSpec Core Diagram**

The display functions for the application are contained within the PalmSpecDisp.c module (see **Download Pack**, page 19). Currently two display modes are available, Spectral Analyzer and Spectrograph. Various display functions are available for initializing, refreshing, and clearing the display. The application core coordinates the function calls based on the user configuration to create the display.

The SAM module library functions are contained within the sam.c file (see **Download Pack**, page 19). The code provided in the download pack is the prototype version of the actual library that will be embedded on the boot Flash. Functions are available to set various control parameters within the CPLD. Also, higher level functions to properly boot the DSP and download algorithm code are available. The SAM_Lib_Initialize() function initializes library variables and pointers and sets up an appropriate interrupt handler. The initialize function also starts up the system clock. The SAM_Module_Boot() function takes an application number as a parameter. Multiple algorithms may be downloaded to the DSP core by calling this function. The boot function handles configuration the CPLD clock control to boot mode and setting system variables so that the interrupt service routine knows how to handle SPI interrupts.

Software Interrupt Handler

The SAM module interrupt handler is automatically called when the module generates an interrupt. The interrupt handler is illustrated in Figure 9. The handler checks which interrupt was generated and branches to the appropriate sub-handler. Future versions will support function call-backs to allow developers to have their own interrupt handlers. For SPI interrupts, the handler checks if the system is in boot mode. If it is, it writes the appropriate boot code to the SPI_DATA register on the CPLD and since the CPLD stops the clock during the boot process, the interrupt handler re-enables it. If the system is not in boot mode, the data is stored in the received data array. This functionality is specific to PalmSpec. The actual library will contain a more configurable ISR that will handle more SPI modes and user call-backs for storing data.

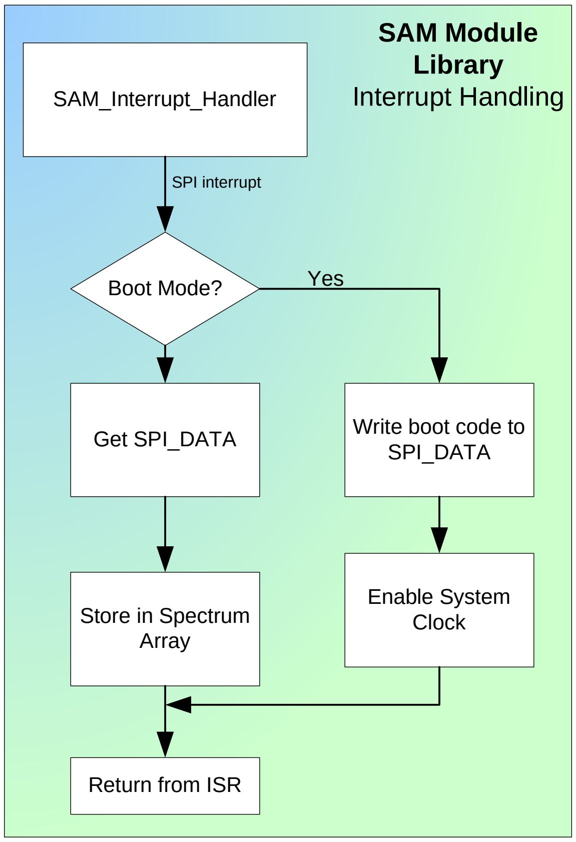

Figure 9: **SAM Module Library Interrupt Handling**

The design contains embedded code that is run on the DSP. The design of this code is beyond the scope of this document. A well-documented assembly code listing of the embedded software is included in the download pack (see **Download Pack**, page 19). The compiled code for the Spectrum Application is included in the download pack (see **Download Pack**, page 19).

Download Pack

THIRD PARTIES MAY HAVE PATENTS ON SAM. BY PROVIDING THIS HDL CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS DESIGN, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THIS DESIGN IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

XAPP363 - http://www.xilinx.com/products/xaw/coolvhdlq.htm

Conclusion

This design submission for the Cool Module Design Contest is a perfect application of a low power Sprinboard module. The SAM design using a low power Xilinx CoolRunner CPLDs in conjunction with a low power software programmable Dspfactory DSP chip set. Low power CoolRunner CPLDs are the ideal programmable logic solution for portable, handheld applications. The Sonic Access Module is a portable audio processing module utilizing the Springboard expansion port of the Handspring Visor. The SAM design submission can be used in a variety of end applications ranging from real-time spectrograms to speech analysis.

Appendix A

Appendix A lists appropriate Xilinx CoolRunner CPLD application notes. These application notes can be found by searching the Xilinx website and keying on the specific XAPP number. Many include appropriate driver software along with high level design code. All have been constructed and work.

PDA Springboard Design

XAPP147: Low Power Handspring Springboard Module Design with CoolRunner CPLDs

XAPP359: Understanding the Insight Springboard Development Kit

XAPP357: CoolRunner Visor Springboard LED Test

XAPP355: Serial ADC Interface Using a CoolRunner CPLD

XAPP146: Designing an Eight Channel Digital Volt Meter with the Insight Springboard Kit

XAPP149: Designing an Oscilloscope for the Insight Springboard Development Kit

Additional Application Notes

XAPP348: CoolRunner XPLA3 Serial Peripheral Interface Master

XAPP341: UARTs in Xilinx CPLDs

References

Dspfactory website: http://www.dspfactory.com/

Handspring website: http://www.handspring.com/

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/17/01	1.0	Initial Xilinx release.