XAPP368 (v1.0) November 30, 2001

Handheld Pocket Logic Analyzer

Summary

This document describes the implementation of the Pocket Logic Analyzer design submitted to the recently publicized "Cool Module Design Contest". All development for this contest design was done 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 Pocket Logic Analyzer is an eight channel logic analyzer Springboard module that is capable of sample rates ranging from 1 millisecond to 50 nanoseconds. The CPLD and Handspring design files for the Pocket Logic Analyzer are available and can be found at **Download Pack**, page 17.

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

Introduction

In January of 2001, Xilinx, Handspring and Portable Design Magazine collaborated on a design contest to highlight use of the Xilinx CoolRunner XPLA3 CPLDs to quickly develop 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 Pocket Logic Analyzer submission supplied by NowTech, LLC. The team members are Terry Carpenter, David Hinkle, and Dan Nower from Knoxville, Tennessee.

Appendix A: Hardware Register Definitions Tables defining the hardware register definitions for the Pocket Logic Analyzer design.

Appendix B: Outlines existing Xilinx application notes helpful to understanding this application note. These are available on the Xilinx website at: **www.xilinx.com**.

Design Description

The Pocket Logic Analyzer is an eight channel Logic Analyzer implemented on the Insight Electronics Springboard Development board. Like other Logic Analyzers, this Pocket Logic Analyzer is a tool that can be used to gather operational information from a digital circuit by capturing digital waveforms and storing them in memory. The Visor handheld computer will then display this captured data in order to allow the user to evaluate operation of the device under test.

This design uses a Xilinx XPLA3 CoolRunner CPLD to control the data aquisition and the data flow to / from the Handspring host computer. The on-board Toshiba SRAM is used to store the acqusition data, and the non-volatile flash memory is used to hold the "plug and play" Logic Analyzer application. The Visor handheld computer is responsible for computing and displaying the logic waveforms.

An add-on daughter card is connected to the prototyping area of the Insight Development board. It proves a nine pin header for the eight inputs plus ground. The daughter card provides input protection and threshold support for the eight inputs. A 20MHz oscillator is also located on this board to drive the sample rate clock generation.

The Pocket Logic Analyzer supports sample rates of 50 ns, 100 ns, 200 ns, 500 ns, 1 µs, 2 µs, 5 µs, 10 µs, 20 µs, 50 µs, 100 µs, 200 µs, 500 µs, and 1 ms. Other features include user definable triggering as well as pre and post triggering.

System Design Overview

This section provides a summary of logic analyzer operation as well as a brief description of the design of the Pocket Logic Analyzer. A more detailed description of the CPLD implementation and software design is presented later in the application note.

Logic Analyzer Modes: State Analysis vs. Timing Analysis

Most logic analyzers provide two methods of collecting data: State Analysis and Timing Analysis. State Analysis Mode is mainly used for troubleshooting microprocessor-based designs. This method requires an external clock from the target system that is used to capture all the incomming data. Typically, State Analysis requires a large analyzer with a large number of input lines for the microprocessor's address, data and control lines. The Pocket Logic Analyzer does not support State Analysis mode.

Timing Analysis Mode uses the analyzer's internal clock source to capture the incomming data on all input lines. The data is collected at the sample rate, stored in memory, and displayed as time-based waveforms. Timing Analysis Mode is supported by the Pocket Logic Analyzer.

The Input Channel Circuit

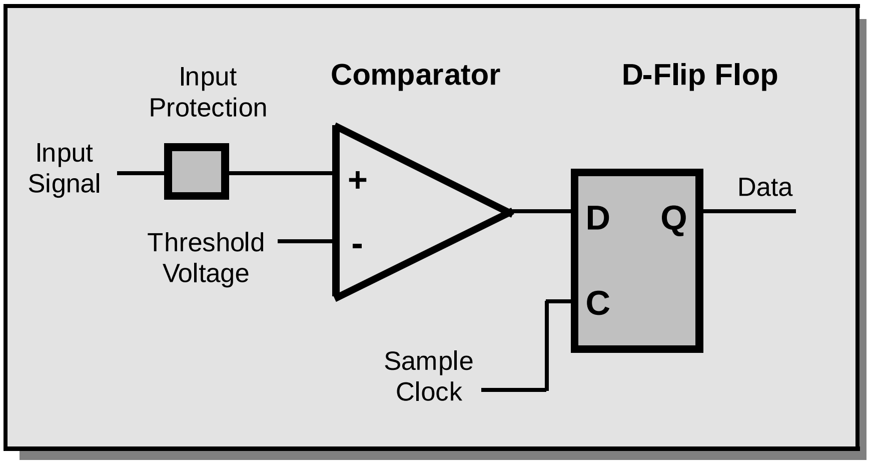


Figure 1: **Input Channel Circuit**

Shown above in Figure 1 is a simple logic analyzer input channel circuit, consisting of an input protection circuit, a comparator and a D-type register. In most logic analyzers, each of the input channels will have one of these circuits to convert the incoming signal into digital data. Each individual component of this input cicuit is described below:

Input Protection: Responsible for protecting the input comparator from excess voltage and current. This protection scheme usually consists of a current-limiting resistor and a dual mode transorb diode. This input protection circuit will have limits to its protection abilities. The user should be careful not to apply an input voltage level that exceeds these limits. Input protection was not implemented in this prototype.

Threshold Voltage: This is the voltage level used to convert the incoming signal into digital data. Any incoming signal that is lower than the threshold voltage is output from the comparator as logic level "0". Likewise, any incoming signal that is higher than the threshold voltage is output from the comparator as logic level "1". This threshold voltage can either be a fixed level or can be controlled by a digital-to-analog converter to produce a user-selectable threshold voltage level.

Comparator: As described above, a comparator will convert the incoming analog signal into digital data. The threshold voltage will determine when the output will be a "1" or a "0".

D-Flip Flop: Used to hold the captured data value. The data stored in this register will be written to SRAM.

Waveforms

Figure 2 below is a waveform representation of the input circuit described above. As can be seen, the input signal is converted into a series of ones and zeroes on the rising edge of the sample clock. This series of ones and zeroes from each channel is combined with other channels to form a data word for each sample period. This data word is then written into SRAM between each sample period. The processor (in this case, a Handspring Visor), will then read the SRAM, and convert the digital data into a waveform for the user to analyze.

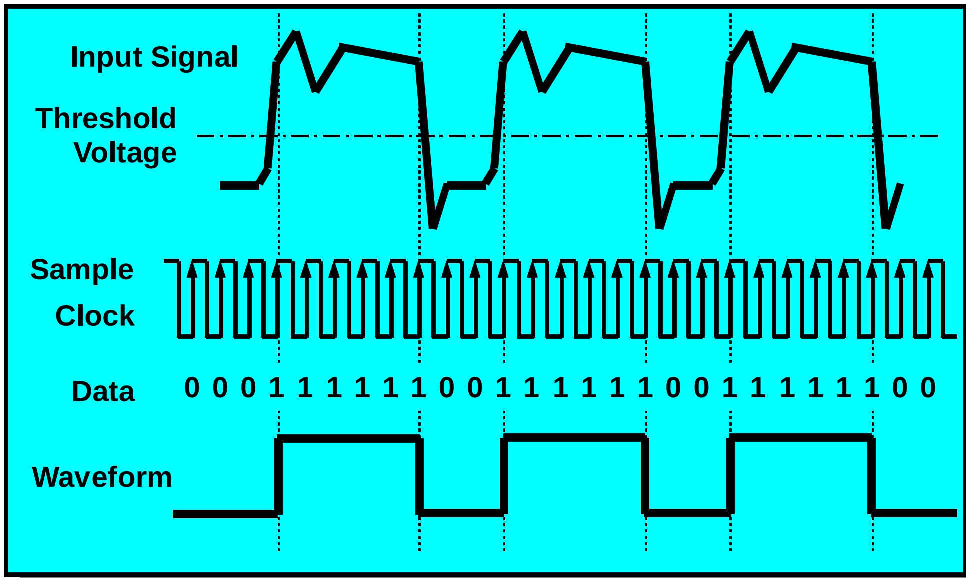
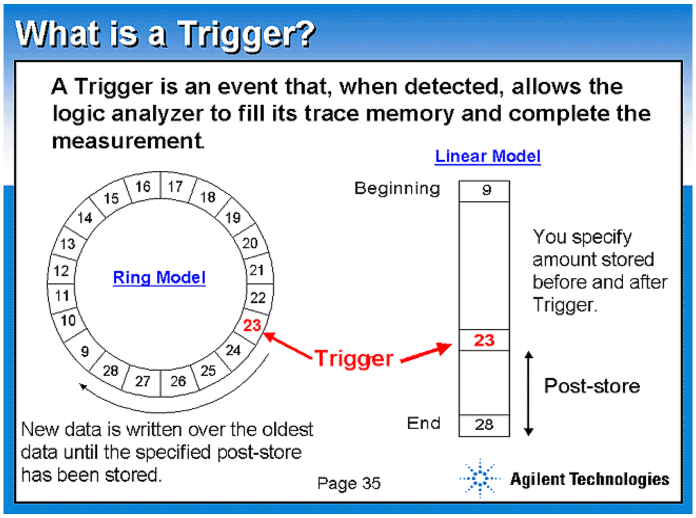


Figure 2: **Single channel Input Waveforms**

Trace Memory Buffer

The data from each channel of a logic analyzer is collected and stored into a trace buffer. This trace buffer is best described as a circular buffer as shown in Figure 3.



What is a Trigger?

A Trigger is an event that, when detected, allows the logic analyzer to fill its trace memory and complete the measurement.

You specify amount stored before and after Trigger.

New data is written over the oldest data until the specified post-store has been stored.

Figure 3: **Trace Memory Buffer**

Logic Analyzer Applications

Logic Analyzers comprise a special category of data acquisition tools. Like oscilloscopes, they present data in the same general manner — the horizontal axis represents time, while the vertical axis represents logic High or logic Low (not voltage). Logic Analyzers are unique in that they can trigger on a predetermined pattern of highs and lows on different signals and can also reformat the captured data in a more meaningful manner. For example, a logic analyzer is particularly useful when examining a microprocessor address, data, and control bus. It can be used to decode the address signals and present the data bus in a more meaningful hexadecimal format. The Visor handheld computer is an ideal platform for a logic analyzer because it provides convenience and portability.

Operating Instructions

The Pocket Logic Analyzer design can be downloaded from the Xilinx website. The download pack includes the necessary files needed to run the Pocket Logic Analyzer, see **Download Pack**, page 17.

Starting the Application

1. Ensure that the card detect switch located on the Insight Electronics Springboard Development card (SW2-6) is OFF (down) and that the Visor is OFF.

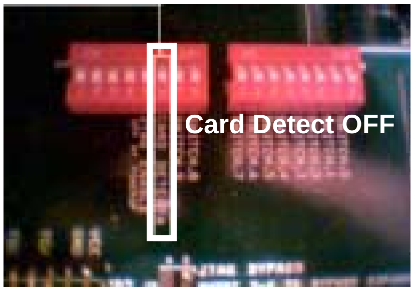


Figure 4: **Card Detect OFF**

2. Plug in the NowTech Expansion Daughter Card into the Springboard slot (See Figure 5).

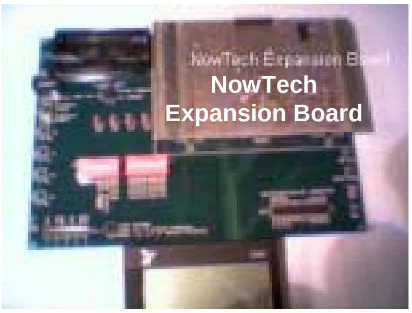


Figure 5: **Development Board and Visor**

3. Connect the NowTech Expansion Board to a signal source. As a simple test case, NowTech created a Signal Source Board consisting of a separate CoolRunner XPLA3 CPLD programmed with a counter pattern. Figure 6 below shows the NowTech Signal Source Board connected to each of the eight input channels as well as ground.

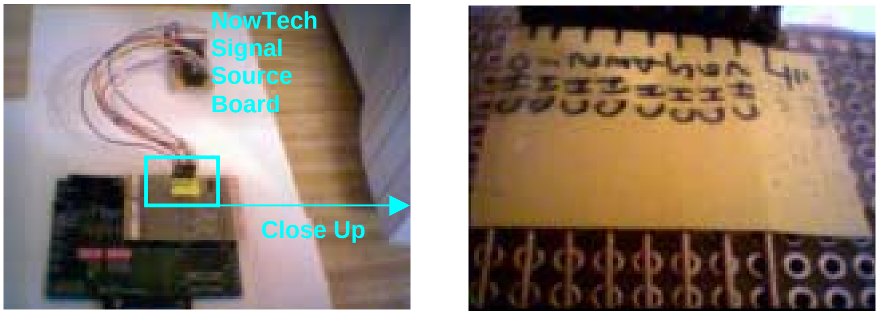


Figure 6: **Signal Source Board Setup**

4. Turn on the Visor by switching the card detect dip switch (SW2-6) to ON. The Visor will power up and attempt to read the application code stored in the Insight Development Board's flash memory.

5. Launch the Pocket Analyzer application by tapping the PAnalyzer icon in the Application Launcher.

6. Upon launch, all, none, or any of the following parameters can be modified:

a. Sample Rate

b. Percent pre-trigger

c. Trigger Word

d. Single or continuous data acquisition mode

7. Turn on the Nowtech Signal Source Test Board by moving the jumper from the OFF potion to the ON position (Figure 7). Once the signal source board is ON, the status LED will flash.

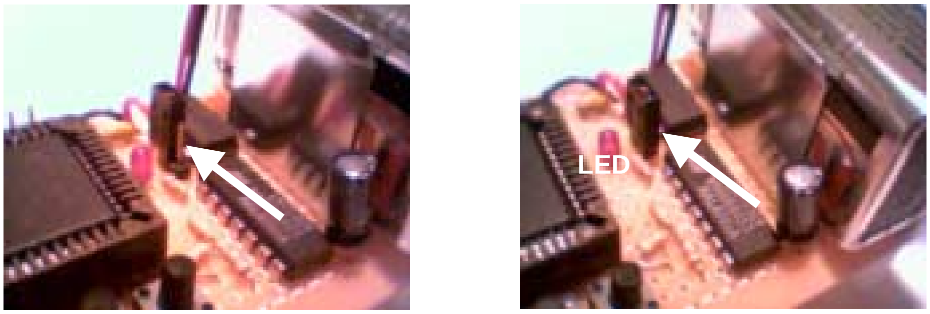


Figure 7: **Turning ON the NowTech Signal Source Test Board**

8. Tap the "Run" button to begin data acquisition.

Hardware Description

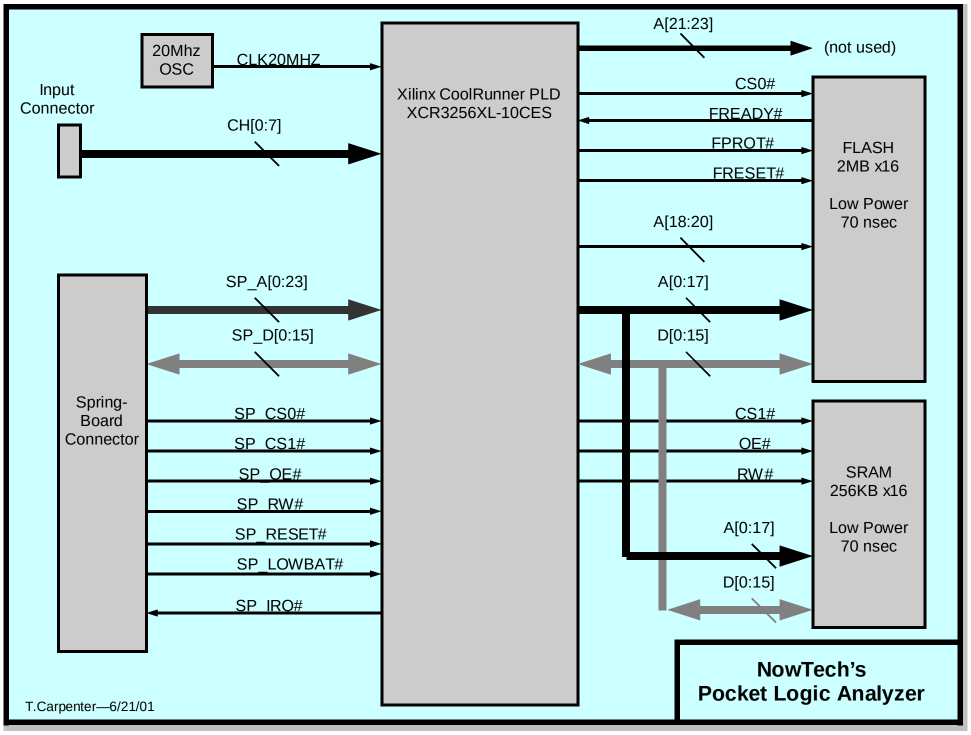


Figure 8: **System Diagram**

Figure 8 describes the hardware used for the Pocket Logic Analyzer. As shown, the CoolRunner CPLD serves as the central interface between the Handspring Visor, Flash memory, SRAM, and the input connectors. Note that in this design, the CoolRunner device is responsible for the data acquisition logic, while the Visor is used to retrieve and display voltage waveforms. The Visor also delivers trigger values and initializes the XPLA3 circuitry. The high speed, yet ultra low power nature of the CoolRunner CPLD allows for time critical tasks such as this to be completed in hardware.

Figure 9 shows the hardware flow diagram for the Pocket Logic Analyzer design:

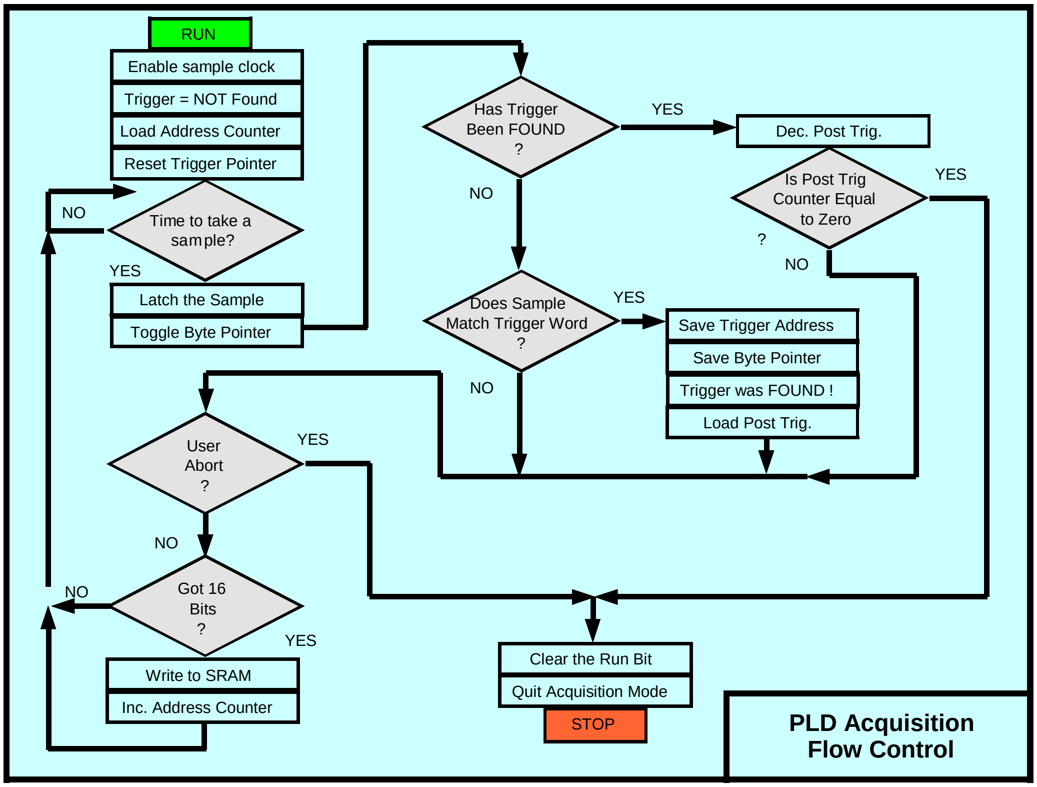


Figure 9: **Hardware Flow Diagram**

CPLD Design

The Xilinx CoolRunner XPLA3 CPLD serves as the central system controller, interfacing to the Flash memory, SRAM, Input logic, and Springboard expansion slot. The CoolRunner XPLA3 CPLD is responsible for:

- Aquisition Control Logic
- Control Register Logic
- Bi-Directional Data Multiplexer
- Address Multiplexer
- SRAM & Flash Control Signals
- Input Channels

The Top level design schematic is shown in Figure 10.

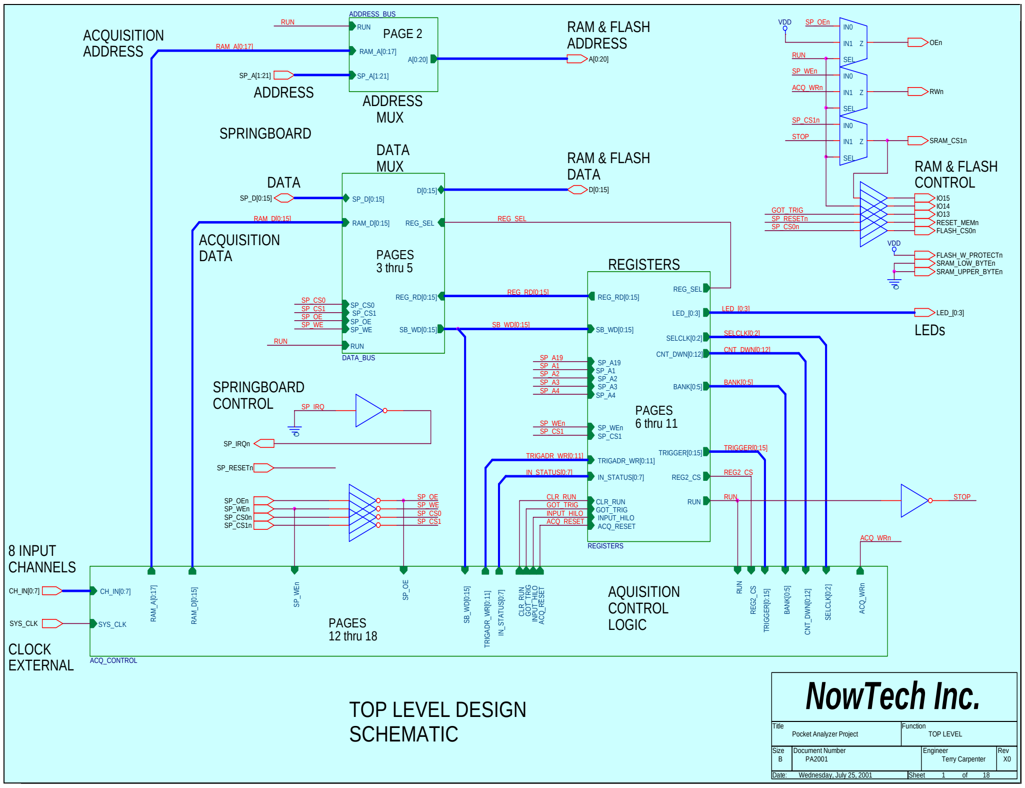


Figure 10: **Top Level Design Schematic**

Note that this design submission was completed using an eighteen-page schematic. All schematic diagrams and implementation results for the CPLD design are included in the available download pack (see **Download Pack**, page 17).

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

Address Multiplexer

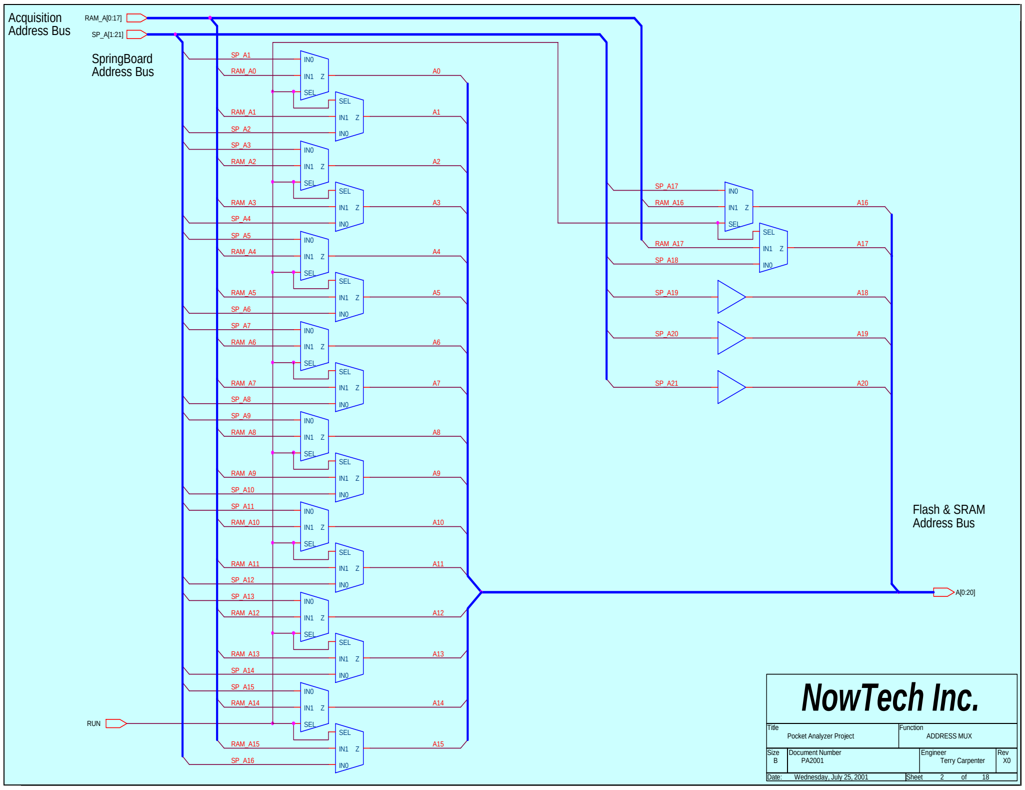


Figure 11: **Address Multiplexer Schematic**

The SRAM and Flash memory share a common external memory address bus that is connected directly to the CoolRunner CPLD. The Address Multiplexer shown in Figure 11 allows the CPLD to select the source of address data that will drive this external memory address bus. This address multiplexer will allow data to be either stored or retrieved from one of the two memory devices.

There are two possible sources that can drive this external address bus: the Acquisition Control Logic (Address Counter) and the Springboard Address bus. The RUN bit is used to control which of these two sources is driving the address bus. The Handspring Application (Panalyzer.prc) sets this bit by writing to one of the internal CoolRunner CPLD registers (see **Appendix A: Hardware Register Definitions**, page 18). When this RUN bit is active High, the Acquisition Address Counter drives the external address bus. Alternately, when the RUN bit is Low, the Handspring is allowed to drive the address bus.

Bi-Directional Data Multiplexer

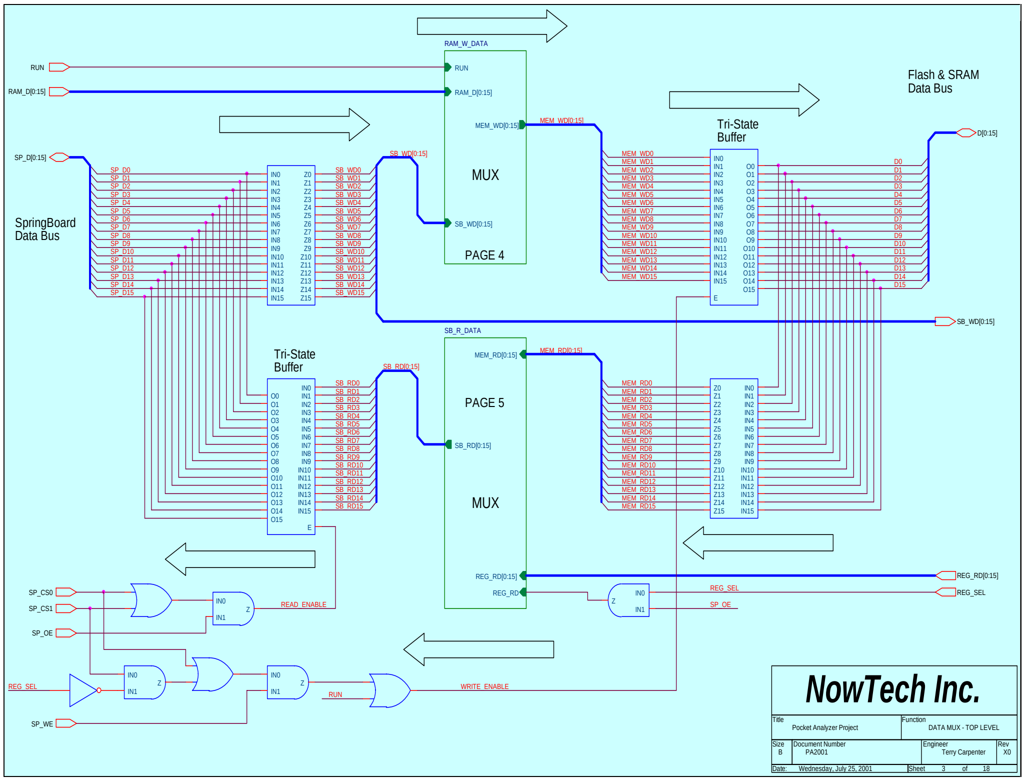


Figure 12: **Bi-Directional Data Muliplexer Schematic**

The data bus to the SRAM and flash memory need to be multiplexed for the same reason as the address multiplexers. Functionality is similar but there is one exception: bi-directionality. The data multiplexer must allow the data to both be driven and to be read to and from the external memory bus. The two sources for control of this data bus are the Data Acquisition Logic and the Handspring application. The Data Acquisition Logic is only in one direction. It is always writing to external memory, and therefore, data direction control signals are not required here.

On the other hand, the Handspring application software must be able to read and write to resources located on the Springboard module. Per the Springboard specification, the data bus direction is controlled by the read and write control lines originating from the Springboard bus. Again, the RUN bit is used to select from the two masters of this external data bus. It should also be noted that once an aquisition has begun the Handspring unit cannot access the external memory, since it will not have control over the address and data bus. However, during an acquisition, the Handspring unit does have access to the CoolRunner CPLD’s internal status registers to determine when the acquisition has completed. Figure 12 is a heirarchical view of this data bus multiplexer. This corresponds to schematic sheet number 3. Schematic sheets 4 and 5 are details of the individual multiplexers and will not be shown here. Please refer to the download pack to obtain this schematic (see **Download Pack**, page 17).

Internal CPLD Registers

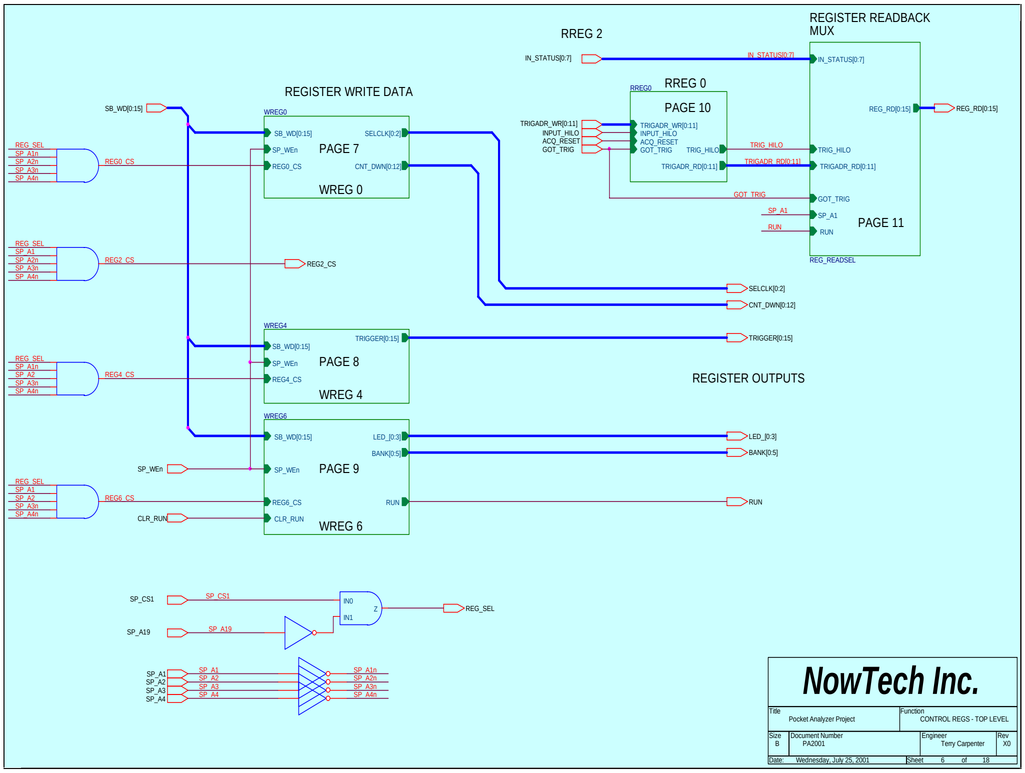


Figure 13: **Internal CoolRunner CPLD Registers**

Figure 13 is another hierarchical view of the internal registers. The internal registers in the CPLD are used to control acquisition flow and to obtain acquisition status information. A more detailed view of these internal registers can be found at the end of this document in **Appendix A: Hardware Register Definitions**, page 18.

Acquisition Control Logic

The Acquisition Control Logic is the critical aspect of the Pocket Logic Analyzer submission, as it is responsible for acquiring data and writing data to designated portions of SRAM. A hierarchical view of the Acquisition Control Logic is shown in Figure 14.

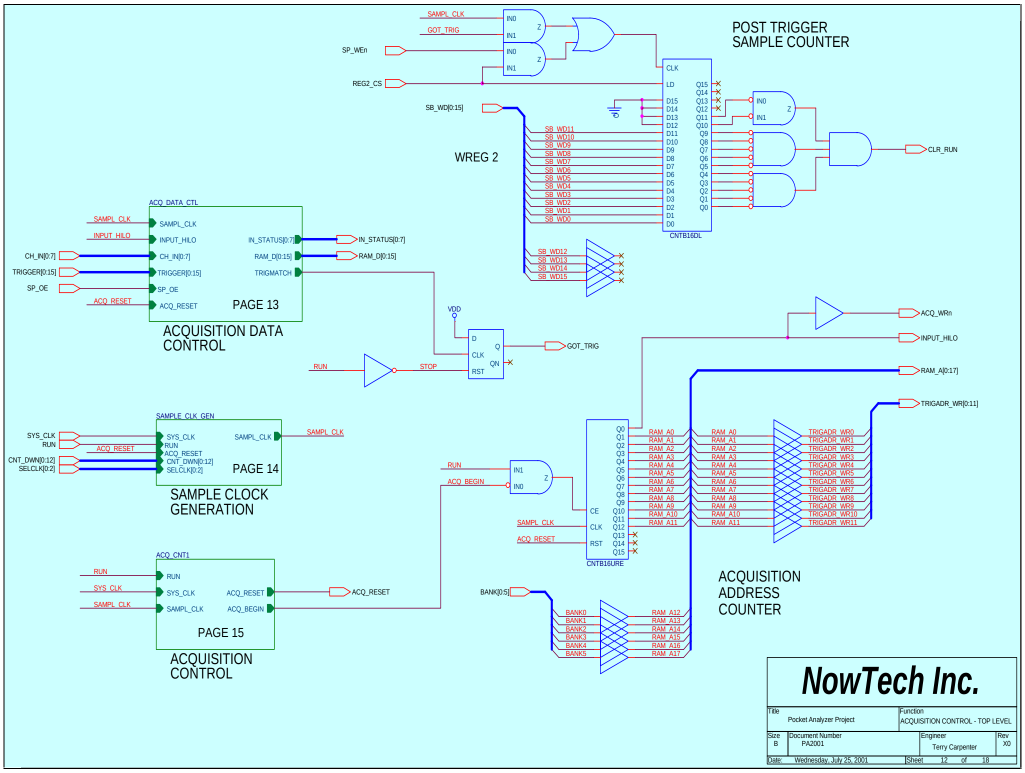


Figure 14: **Acquisition Control Logic**

The Acquisition cycle begins with the user selecting the sample rate, the amount of pre / post trigger, the trigger word, the SRAM destination location (or bank), and setting the RUN bit. Figure 15 shows the logic that was implemented in order to get the acquisition process started. Figure 15 corresponds to schematic sheet number 15.

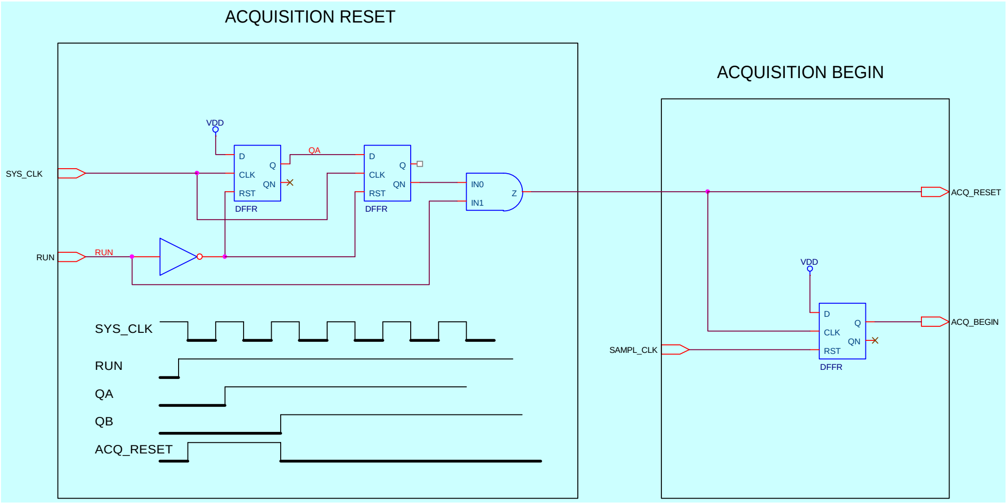


Figure 15: **Acquisition Start Logic**

Sample Clock Generation Logic

Figure 16 shows how the sample rate clock is generated. Write Register 0 (as defined in the Register Definitions Section APPENDIX) is used to determine the sample rate used during the next acquisition cycle. The three select clock bits found in register 0 are used to select either fast clocks or slow clocks. The fast clocks are SYS_CLOCK (20 MHz), SYS_CLOCK2 (10 MHz) or SYS_CLOCK4 (5 MHz). The slow clocks are generated from the 15 bit down counter and the loaded pre-count values from write register 0. The slow clock loads the pre-count value, then counts down to zero. When it hits zero, it restarts, thus producing the desired clock frequency at SAMPL_CLK.

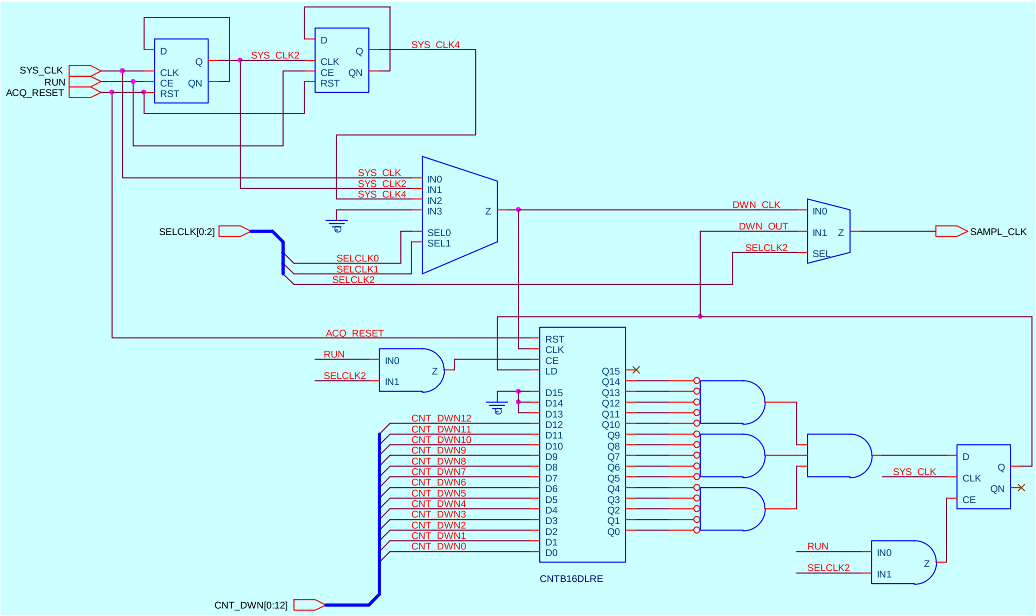


Figure 16: **Sample Clock Generation**

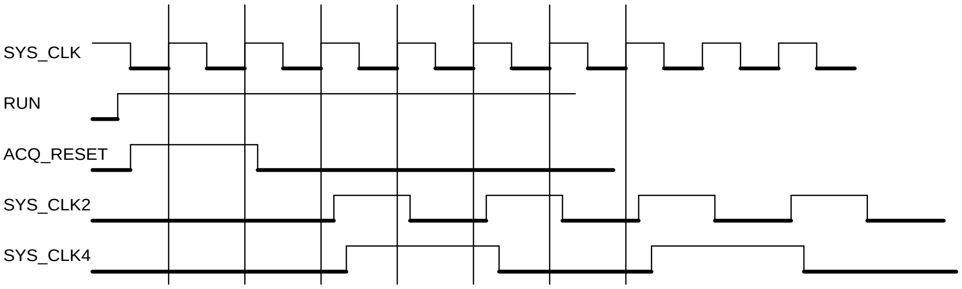


Figure 17: **Sample Clock Timing**

Acquisition Data Control

Figure 18 shows how the input data is captured and how the trigger word match is performed. In this implementation of the Pocket Logic Analyzer, the input data width is only eight bits and the SRAM memory bus16 bits wide. This is needed to achieve a 20MHz maximum sample rate.

As shown in Figure 18, the two eight-bit latches are used in a "ping-pong" arrangement to store alternating eight-bit samples. This allows two samples to be written into SRAM while collecting the next sample value. The other eight-bit latch marked "RREG2" (shown below) is used to collect input status information. This input status information is used prior to an acquisition in the form of feedback to the user of the current state of the inputs. The series of eight multiplexers shown below control the input to the trigger word comparator. The user selects the input trigger word by writing to Write Register 4 prior to setting the RUN bit (See **Appendix A: Hardware Register Definitions**, page 18).

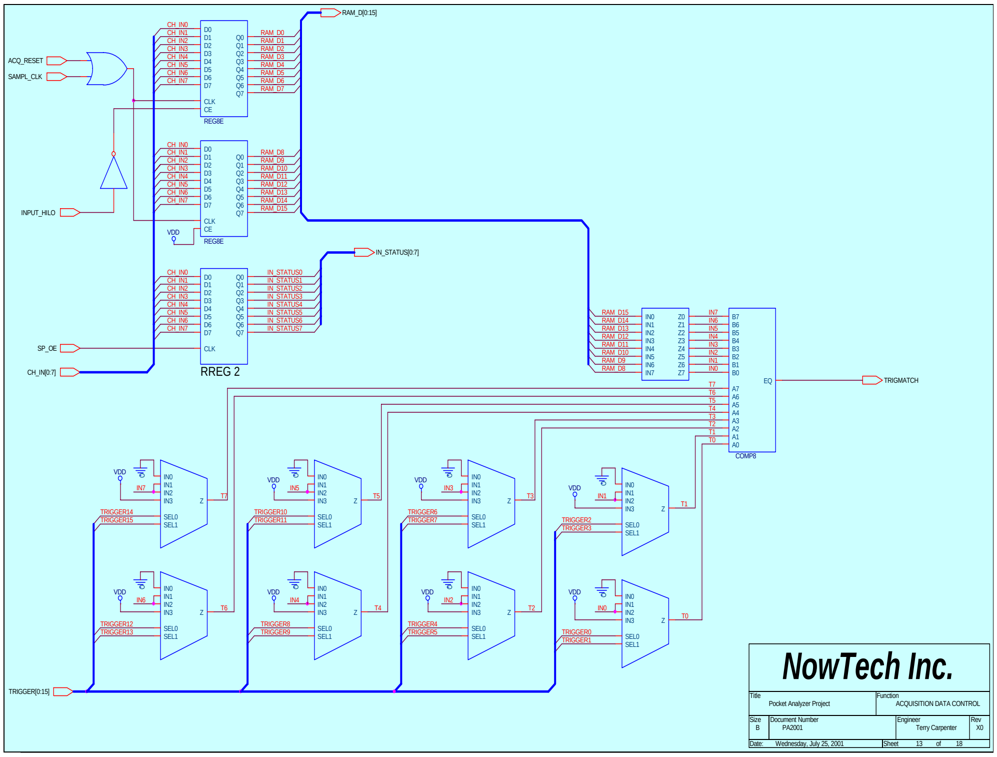


Figure 18: **Acquisition Data Control Schematic**

Acquisition Counters

Figure 19 and Figure 20 show the two main counters for controlling the acquisition process. Figure 19 shows the Acquisition Address Counter. This counter is controlled by the RUN bit and counts at the sample clock rate. As shown, "RAM_A0" uses the Q1 output of this counter to bump the RAM address at one-half the sample clock rate. This is done because the data gets collected as eight bits but gets written to memory as 16 bits. These RAM address lines are also fed to the trigger word register so that when the trigger word is detected, the address for the trigger gets stored inside this register. The bank select lines come from outputs of Write Register 6 (see **Appendix A: Hardware Register Definitions**, page 18) and are used to control where in memory the data is written. This allows for multiple banks of data for storing waveform data.

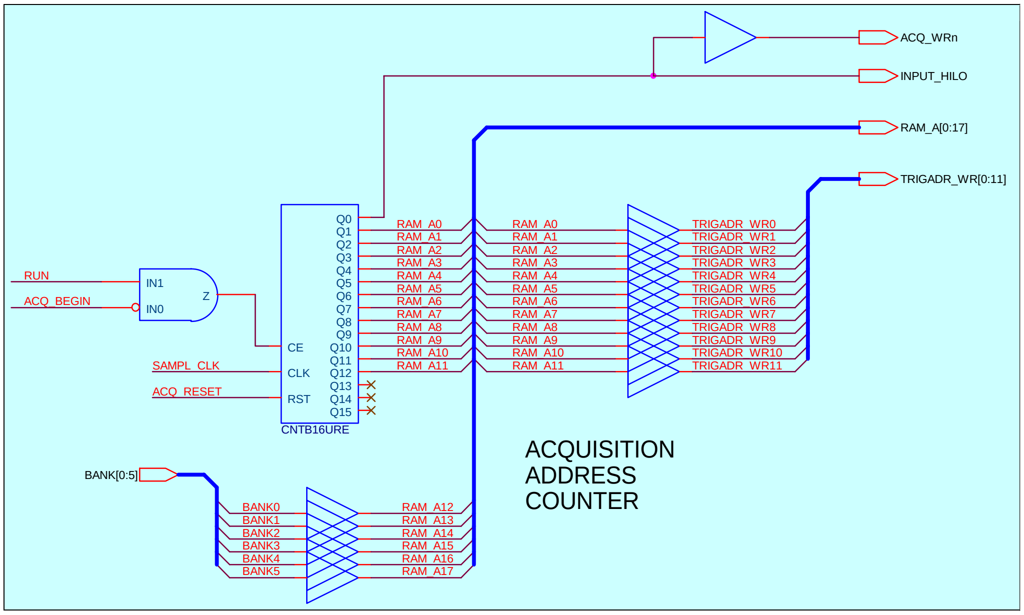


Figure 19: **Acquisition Address Counter**

Figure 20 shows the second counter, named the Post Trigger Sample Counter. This down counter is only allowed to count after the trigger is detected and counts the number of post trigger samples to collect before stopping the acquisition by clearing the RUN bit. The number of post samples to collect is written to Write Register 2 prior to starting the acquisition.

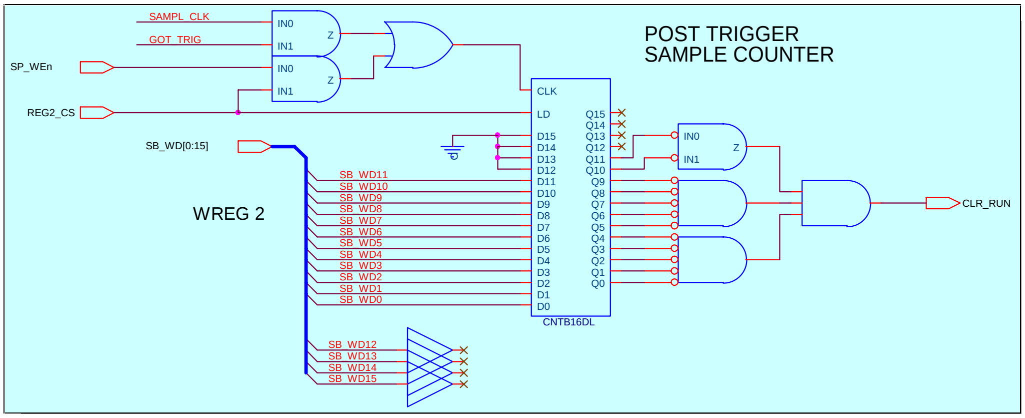


Figure 20: **Post Trigger Sample Counter**

CPLD Implementation

The CPLD implementation described above has been implemented using ORCAD Schematic Entry and the free Xilinx WebPACK software tool. Table 1 displays a summary of the fitting report. As shown, a total of 209 macrocells are used. The modularity of the design allows for particular changes and fixes without affecting the pinout. This will allow for schematic modifications, additions and fixes to be implemented without the need to re-layout the module PCB.

Table 1: **CPLD Device Resource Summary**

Resource	Available	Used	Utilization
Clock Inputs	4	1	25%
Global Control Terms	4	3	75%
Function Blocks	16	15	93.75%
I/O Pins	160	103	64.38%
Macrocells	256	209	81.65%
Registers	256	133	51.96
PLA Product Terms	768	411	53.52%
PLA Sum Terms	256	197	76.96%
Block Control Terms	128	29	22.66%
Foldback Nands	128	0	0%

Handspring Software Design

The PocketAnalyzer application is a PalmOS program compiled using the Metrowerks Codewarrior tools. It is designed to run on a Handspring Visor PDA with PalmOS® Version 3.5 or higher. The program is loaded onto the PocketAnalyzer Springboard Module and interfaces with this hardware to provide an eight channel logic analyzer capable of capturing 50 ns signals.

The software provides the user with three menus by which they can modify the operating parameters of the analyzer. The Setup menu allows the user to change the Run Mode, Trigger Definitions, Sweep Timebase (and therefore the sample rate), and select the active channels. Although not yet implemented, the Cursor menu will allow the user to turn on and/or move one of the two cursors, jump to one of the cursors or the trigger, or view the cursor measurements.

The Preferences menu will allow the user to modify their startup preferences, view help, or view information about the PocketAnalyzer application.

When the application starts it first verifies that the PDA device it is loaded on is running a compatible version of PalmOS. If this test is successful, the program reads its stored preferences from the preferences database. If a preferences record does not exist or has been corrupted, the program initializes the AppPrefs structure with factory default values. If a valid record is found, these settings are loaded into the AppPrefs structure and used by the application. The main application form is loaded and the main event loop is entered. The application remains in this event loop until the program gets a stop application event. As part of its cleanup when stopping, the application stores the latest version of the application preferences to the application database.

The main screen presents the user with the plot area with channel labels to the left, a scroll/buffer position bar at the top, cursor measurements in the lower left (not yet implemented), a plot scroll object in the bottom center and a run/stop button in the lower right. If the internal, user-selectable flag is set to indicate that the analyzer should auto-sweep, then the program initiates a data capture immediately. Otherwise, the program waits for the user to tap the run/start button. The program indicates the present run state by displaying either “RUN” on this button when the program is no longer acquiring data or “STOP” during active data acquisition. The user can pause an auto-sweep by pressing the “STOP” button. The program will then re-scale the display for different time divisions.

The program’s main form event loop changes the EvtGetEvent timeout to create a timeout to allow polling of the Springboard module for completion of data acquisition. If no acquisition is active, the timeout is set to infinity.

Initialization Process

When the program initiates a data capture, either by user selection or by auto-sweep, the program first sets up the acquisition. The sample rate is downloaded to the module. The analyzer captures data at a 50 ns rate for all timebases up to 10 kHz (where it changes to 100 ns). The rate then increases with each of the additional timebases to a maximum of 1 µs at 1 kHz. The number of post-trigger samples to acquire is then downloaded to the module. The program interfaces with the user in percent pre-trigger but downloads post-trigger samples to the module. This simplifies the interface for the user and simplifies the hardware design. Next, the trigger data is sent to the module. The program allows the user to define a trigger byte that specifies the desired state of each input at the time of the trigger. The allowed states for each channel are Trigger High, Trigger Low, and Don’t Care. Finally, the command word is sent to the module that specifies the 8K data bank in which the acquired data is to be stored and the command bit to start the acquisition. If the program needs to stop the acquisition at any time, this work is sent with the Start bit cleared. The program then sets the EvtGetEvent timeout to one second. The program will use the nilEvent events generated by this timeout to control polling of the Springboard module for completion of the data acquisition.

Completion Process

Upon completion of the acquisition, the offset into the data buffer where the trigger occurred is retrieved. Then the 8K data block is transferred to an array and placed in sequential order from the circular buffer. This data is then plotted on the main application screen and the bar at the top is updated to show where in the 8K buffer the displayed data resides. At the present time scrolling of the data has not been implemented, but will be implemented using both the buffer position bar and a touch and drag method in the plot area.

If auto-sweep is active, the program initiates the next data acquisition cycle and begins the polling process again. If the user has selected to operate in single-sweep mode, the program changes the run / stop button to “RUN” and sets the EvtGetEvent timeout back to no-timeout to prevent the nilEvent events from occurring.

The program does not currently support measurement cursors. There will be two cursors that the user will be able to position on the display. The program will then calculate the time differences between these cursors and the trigger point and each other. The user will also be able to jump to either of the cursors or the trigger.

Download Pack

THIRD PARTIES MAY HAVE PATENTS ON HANDHELD POCKET LOGIC ANALYZER. 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.

XAPP368 - http://www.xilinx.com/products/xaw/coolvhdlq.htm

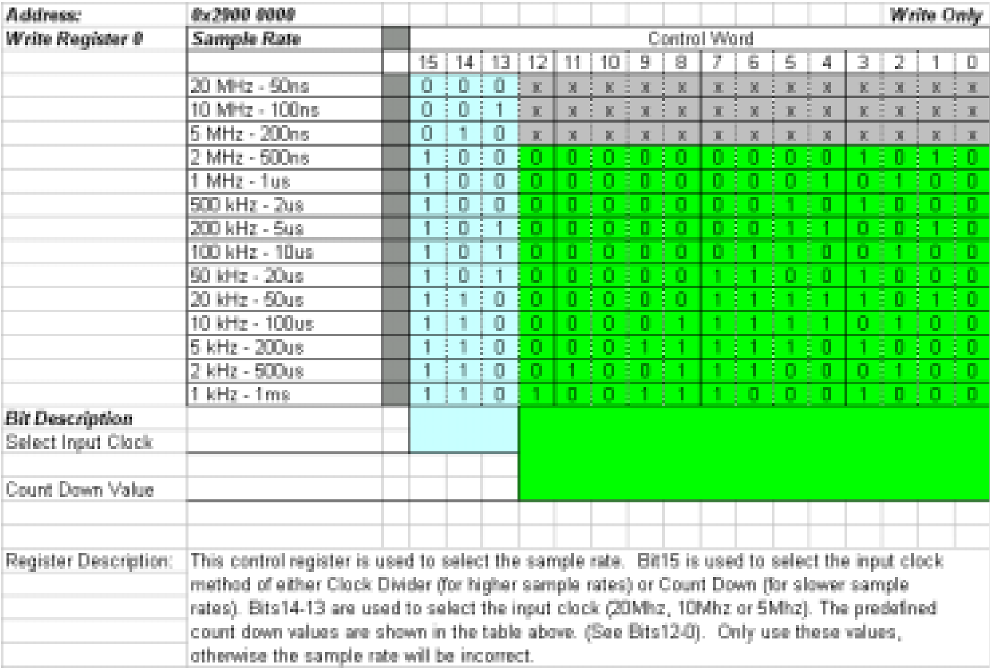
Conclusion

NowTech’s Pocket Logic Analyzer design submission is a portable, low power design that interfaces with the Handspring Springboard expansion slot. It is an ideal application for the Xilinx XPLA3 CoolRunner CPLD, as no other programmable logic device in the world is capable of delivering the Xilinx patented Fast Zero-Power (FZP) Technology, the combination of ultra-low power and uncompromised speed.

Appendix A: Hardware Register Definitions

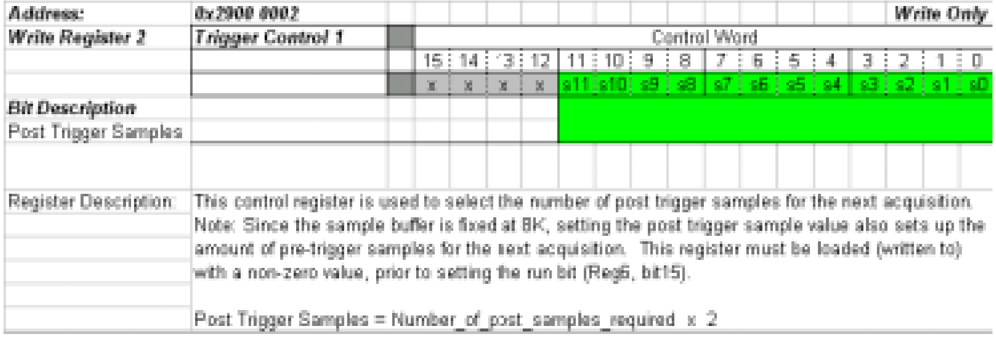
Table 2 lists the Hardware Register definitions for the internal registers in Xilinx CoolRunner CPLD application notes.

Table 2: **Sample Rate Control Register**



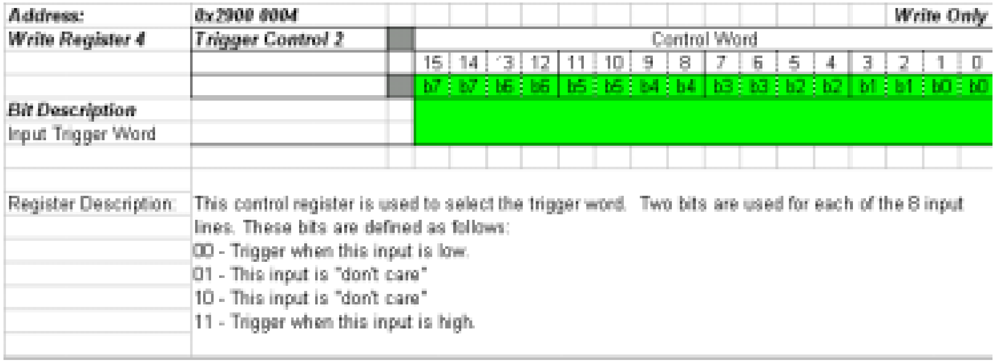
Address:	***0x2900 0000***																***Write Only***
Write Register 0	***Sample Rate***	Control Word															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	20 MHz - 50ns	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x
	10 MHz - 100ns	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x
	5 MHz - 200ns	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x
	2 MHz - 500ns	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
	1 MHz - 1us	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
	500 kHz - 2us	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
	200 kHz - 5us	1	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0
	100 kHz - 10us	1	0	1	0	0	0	0	0	0	1	1	0	0	1	0	0
	50 kHz - 20us	1	0	1	0	0	0	0	0	1	1	0	0	1	0	0	0
	20 kHz - 50us	1	1	0	0	0	0	0	0	1	1	1	1	1	0	1	0
	10 kHz - 100us	1	1	0	0	0	0	0	1	1	1	1	1	0	1	0	0
	5 kHz - 200us	1	1	0	0	0	0	1	1	1	1	1	0	1	0	0	0
	2 kHz - 500us	1	1	0	0	1	0	0	1	1	1	0	0	0	1	0	0
	1 kHz - 1ms	1	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0
Bit Description																	
Select Input Clock																	
Count Down Value																	
Register Description:	This control register is used to select the sample rate. Bit15 is used to select the input clock method of either Clock Divider (for higher sample rates) or Count Down (for slower sample rates). Bits14-13 are used to select the input clock (20Mhz, 10Mhz or 5Mhz). The predefined count down values are shown in the table above. (See Bits12-0). Only use these values, otherwise the sample rate will be incorrect.																

Table 3: **Trigger Control 1 Register**



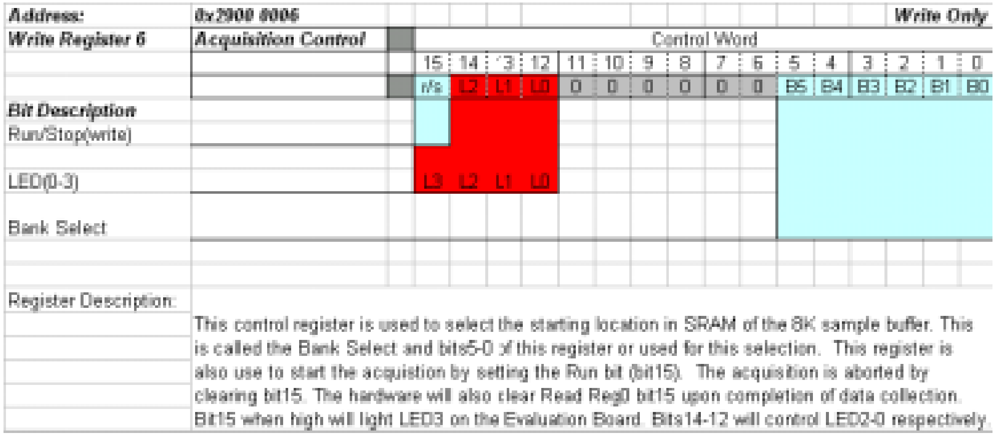
Address:	***0x2900 0002***																***Write Only***
Write Register 2	***Trigger Control 1***	Control Word															
		15	14	'3	12	11	10	9	8	7	6	5	4	3	2	1	0
		x	x	x	x	s11	s10	s9	s8	s7	s6	s5	s4	s3	s2	s1	s0
Bit Description																	
Post Trigger Samples																	
Register Description:	This control register is used to select the number of post trigger samples for the next acquisition. Note: Since the sample buffer is fixed at 8K, setting the post trigger sample value also sets up the amount of pre-trigger samples for the next acquisition. This register must be loaded (written to) with a non-zero value, prior to setting the run bit (Reg6, bit15). Post Trigger Samples = Number_of_post_samples_required x 2																

Table 4: **Trigger Control 2 Register**



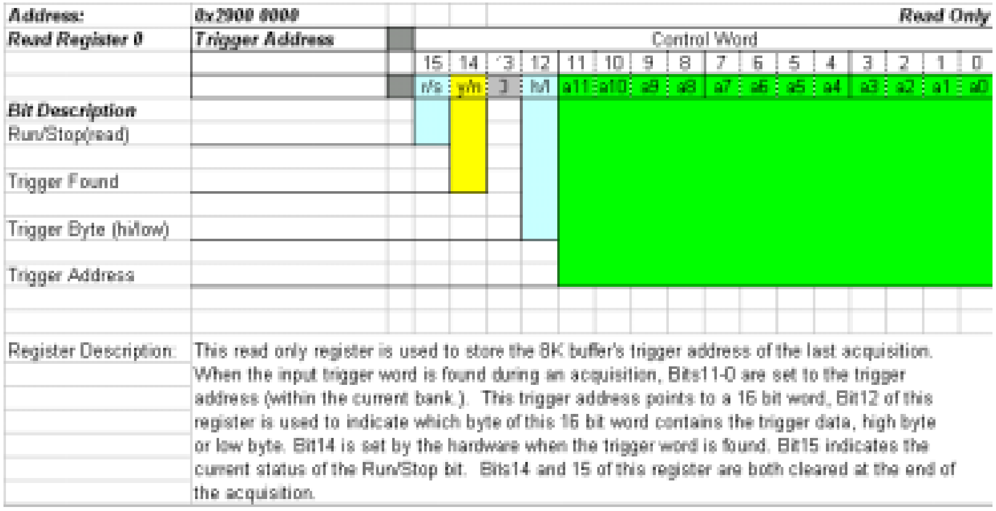
Address:	***0x2900 0004***																	***Write Only***
Write Register 4	***Trigger Control 2***		Control Word															
			15	14	´3	12	11	10	9	8	7	6	5	4	3	2	1	0
			b7	b7	b6	b6	b5	b5	b4	b4	b3	b3	b2	b2	b1	b1	b0	b0
Bit Description																		
Input Trigger Word																		
Register Description:	This control register is used to select the trigger word. Two bits are used for each of the 8 input lines. These bits are defined as follows: 00 - Trigger when this input is low. 01 - This input is "don't care" 10 - This input is "don't care" 11 - Trigger when this input is high.																	

Table 5: **Acquisition Control Register**



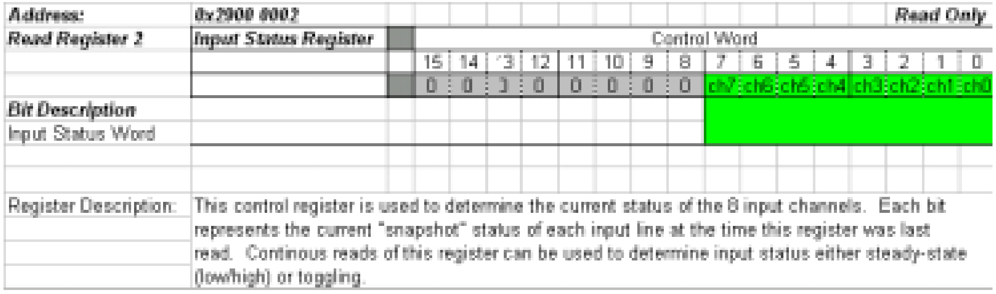
Address:	***0x2900 0006***																	***Write Only***
Write Register 6	***Acquisition Control***		Control Word															
			15	14	´3	12	11	10	9	8	7	6	5	4	3	2	1	0
			r/s	L2	L1	L0	0	0	0	0	0	0	B5	B4	B3	B2	B1	B0
Bit Description																		
Run/Stop(write)																		
LED(0-3)			L3	L2	L1	L0												
Bank Select																		
Register Description:	This control register is used to select the starting location in SRAM of the 8K sample buffer. This is called the Bank Select and bits5-0 of this register or used for this selection. This register is also use to start the acquistion by setting the Run bit (bit15). The acquisition is aborted by clearing bit15. The hardware will also clear Read Reg0 bit15 upon completion of data collection. Bit15 when high will light LED3 on the Evaluation Board. Bits14-12 will control LED2-0 respectively.																	

Table 6: **Trigger Address Register**



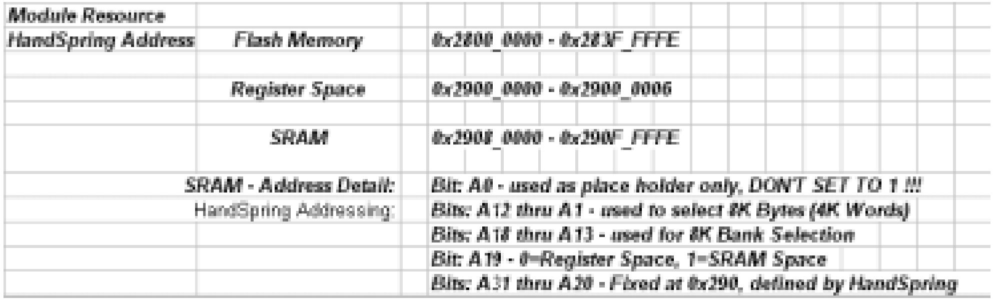
Address:	***0x2900 0000***																***Read Only***
Read Register 0	***Trigger Address***	Control Word															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		r/s	y/n	0	h/l	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
Bit Description		🟦	🟨		🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Run/Stop(read)		🟦	🟨		🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Trigger Found			🟨		🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Trigger Byte (hi/low)					🟦	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Trigger Address						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Register Description:	This read only register is used to store the 8K buffer's trigger address of the last acquisition. When the input trigger word is found during an acquisition, Bits11-0 are set to the trigger address (within the current bank.). This trigger address points to a 16 bit word, Bit12 of this register is used to indicate which byte of this 16 bit word contains the trigger data, high byte or low byte. Bit14 is set by the hardware when the trigger word is found. Bit15 indicates the current status of the Run/Stop bit. Bits14 and 15 of this register are both cleared at the end of the acquisition.																

Table 7: **Input States Register**



Address:	***0x2900 0002***																***Read Only***
Read Register 2	***Input Status Register***	Control Word															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0	0	0	0	0	0	0	0	ch7	ch6	ch5	ch4	ch3	ch2	ch1	ch0
Bit Description										🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Input Status Word										🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Register Description:	This control register is used to determine the current status of the 8 input channels. Each bit represents the current "snapshot" status of each input line at the time this register was last read. Continous reads of this register can be used to determine input status either steady-state (low/high) or toggling.																

Table 8: **Miscellaneous Address Ranges**



Module Resource		
HandSpring Address	***Flash Memory***	***0x2800_0000 - 0x283F_FFFE***
	Register Space	***0x2900_0000 - 0x2900_0006***
	SRAM	***0x2908_0000 - 0x290F_FFFE***
	SRAM - Address Detail:	***Bit: A0 - used as place holder only, DON'T SET TO 1 !!!***
	HandSpring Addressing:	***Bits: A12 thru A1 - used to select 8K Bytes (4K Words)***
		Bits: A18 thru A13 - used for 8K Bank Selection
		Bit: A19 - 0=Register Space, 1=SRAM Space
		Bits: A31 thru A20 - Fixed at 0x290, defined by HandSpring

Appendix B

Appendix B lists appropriate Xilinx CoolRunner CPLD application notes. These application notes can be found by searching the Xilinx website and keying on the specific XAPP#. 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

References

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

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/30/01	1.0	Initial Xilinx release.