

EE552 Final Report

“The Reading Book”

April 4, 2002

By: Jeff Bazinet, jbazinet@ualberta.ca
Reid Blumell, jblumell@ualberta.ca
Andrew Chu, akchu@ualberta.ca
Chris Ohlmann, cohlmann@ualberta.ca
Bryce Palmer, bpalmer@ualberta.ca

Declaration of Original Content

The design elements of this project and report are entirely the original work of the authors and have not been submitted for credit in any other course except as follows:

- audio amplifier drive circuitry was taken from figure 12 of reference [1]
- code for A/D converter interface was modified from a student application note submitted by Shauna Rae on October 29, 1999 [2]

Jeff Bazinet

Reid Blumell

Andrew Chu

Chris Ohlmann

Bryce Palmer

Abstract

This document outlines the technical design details of the *Reading Book* project. The *Reading Book* is a book that can detect which pages are currently open and playback a reading of those pages (either custom recorded or factory recorded). The *Reading Book* was designed so that parents can record a reading of a children's book into the book and the book will automatically play that recording back as a child flips through the pages. Overall, this project met all proposed requirements within the given timeline due to good project planning, project management, and course load balancing.

IC Datasheet

Features

- A/D convertor interface for book page detection compatible w/ADC08 family of ADCs
- Generic SPI interface
- Audio driver for use with WinBond Chopcorder family of audio ICs
- Debounced input for mode select and assert buttons
- Two story modes: factory default and user recordable
- Two playback modes: auto and manual
- Three status LED outputs: auto-playback, default story and record
- Compatible with variable page books (up to 6 pages)
- Hot-swapping book control

Specifications

Parameter	Min	Typical	Max	Units
Clock frequency	-	25.175	25.175	MHz
Number of logic cells	-	674 ¹	-	-

¹ Typical value measured using Altera FLEX10K20 FPGA.

IO Pins

Pin #	Pin Name	Description	I/O
6	LED_AUTO*	Auto-playback status LED	O
7	LED_RECORD*	Record status LED	O
8	LED_STORY*	Default story status LED	O
28	PB_MODE*	Mode select	I
29	PB_MODE_ASSERT*	Mode assert	I
45	SS*	SPI slave select	O
48	MOSI	SPI master out slave in	O
50	MISO	SPI master in slave out	I
53	SCLK	SPI clock	O
55	RAC*	Chipcorder row address clock	I
65	NO_BOOK	Low – book present, High – book not present	I
67	RESET*	Asynchronous global reset	I
70	ADC_SELECT	ADC address select	O
72	ADC_CLOCK	ADC clock	O
74	ADC_START	ADC load address and start conversion	O
76	ADC_DATAREADY	ADC conversion complete	I
79	ADC_DATA7	ADC data bit 7 – MSB	I
81	ADC_DATA6	ADC data bit 6	I
83	ADC_DATA5	ADC data bit 5	I
86	ADC_DATA4	ADC data bit 4	I
88	ADC_DATA3	ADC data bit 3	I
91	CLOCK	Global clock input	I
95	ADC_DATA2	ADC data bit 2	I
98	ADC_DATA1	ADC data bit 1	I
100	ADC_DATA0	ADC data bit 0 – LSB	I

Table of Contents

Achievements	6
Description of Operation	6
Page Detection	7
Chipcorder Driver	8
User Interface	9
Main Controller	14
FPGA Requirements	15
Experiments and Characterizations	16
Page Detection Experimentation	16
Recorder Chip Experimentation	17
Physical Construction Experimentation	19
References	20
Index to Test Cases and Design Verification	21
Diagram of Design Hierarchy	34
Index To VHDL Code	34
Index To Test Benches	35

Achievements

At the time this report was written, the base functionality of “The Reading Book” as outlined in the project specification has been achieved. This includes:

- Page detection
- Factory recorded and user recorded story playback
- Auto and manual playback modes
- Recording through internal or external microphone
- Voice feedback for user interface

Description of Operation

The Reading Book will provide voice recording and playback ability using a simple user interface. The basic model requires a mode select button, an action button, status LED's, a microphone, a non-volatile speech recorder (WinBond ChipCorder ISD4004), a speaker, and page detection circuitry. All control functions will be provided by an Altera FLEX10K20 FPGA mounted on UPI board.

The user is presented with two basic functional modes: Playback and Record. In addition to these functional modes, two configuration modes are provided to allow the user to configure the playback: Toggle Auto-Playback and Toggle Prerecorded Story. By pushing the mode button, the user may rotate through the active modes. Status LED's are used to indicate the current mode and playback configuration. In addition, the ISD4004 is used to provide voice prompts and messages to indicate the current mode and the actions that can be performed in that mode. To ensure that the record mode is not entered accidentally a confirmation is required before recording may begin. The operation of the action button will depend on the current mode. In Play mode, the Reading Book determines which page is currently open and plays audio accordingly. The story to be played is determined by the user's configuration in the Toggle Prerecorded Story mode. If the pre-recorded story has been selected, the non-erasable factory recorded audio will be used, otherwise, the user-recorded story will be used. The type of playback in the Play mode is further configured in the Toggle Auto-Playback mode. If auto-playback has been selected, audio playback begins automatically when a new page is opened, otherwise the user must press the action button to start audio playback for the current page. In any playback configuration, closing the book or opening multiple pages will automatically stop any audio playback in progress.

In the Record mode, the user must simply push the action button to begin recording audio for the currently open page. Pushing the button again will end the recording.

Whenever the book is opened the magnetic page sensors are used to determine the currently opened page and will send this data to the main FPGA controller.

A functional block diagram of the entire system is shown in Figure 1.

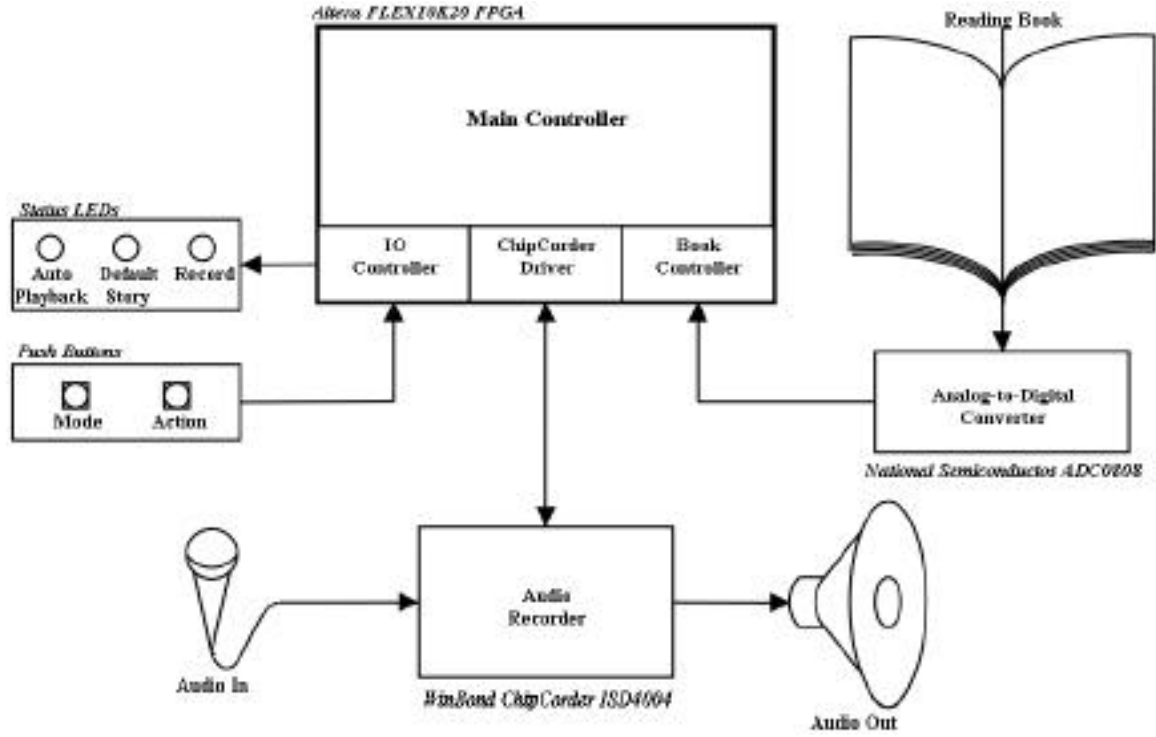


Figure 1: Functional Block Diagram of The Reading Book

Page Detection

The page detection algorithm makes use of the hardware shown in figure A in Appedix A. The two D/A converters shown are actually realized in one D/A converter with a built in mux. The output of the converter is used to detect the open page. There are 3 possible cases to detect.

Case 1 – 1 Page Open

This case is detected when:

$$N_f + N_b = N_{MAX}$$

where N_f and N_b are found using the following equation:

$$N_x = N_{MAX} \left(\frac{V_s}{V_x} - 1 \right)$$

The open page is N_f .

NOTE: The subscripts f and b refer to the front and back cover respectively.

Case 2 – More Than 1 Page Open

This case is detected when:

$$N_f + N_b < N_{MAX}$$

where N_f and N_b are found as in case 1.

Case 3 – No Pages Open

This case is detected when:

$$V_f = V_b = 0$$

Page Connections

The page connections are made with the use of magnets. The resistor will be attached to two Alnico magnets, which will provide contacts between the two pages. Once wrapped in thin copper tape, the magnets have minimal resistance and will not provide a problem with the page detection technique being implemented.

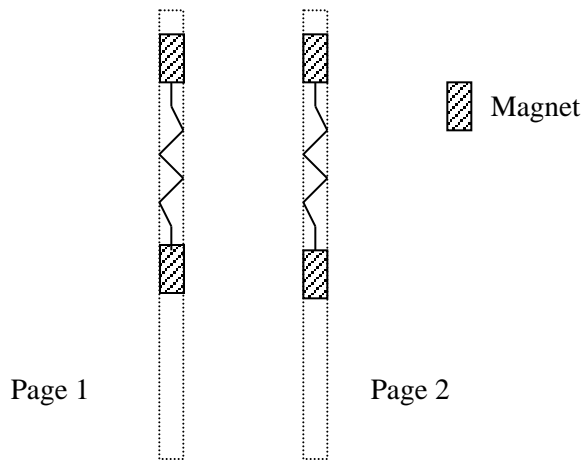


Figure 2: Page embedded resistor network

Chipcorder Driver

To communicate with the Winbond Chipcorder family of voice IC's (ISD4004 series), a ChipCorder driver has been created. In order to comply with the serial peripheral interface (SPI) standards, the driver has been layered on top of a generic SPI PHY. The SPI PHY serializes data to be sent to any SPI device, and will parallelize data from the device to be passed back up to the device driver. The ChipCorder driver receives control commands from a higher level entity. When a command is sent to the driver, it is processed and the correct sequence of ChipCorder instruction opcodes is sent out using the SPI standard protocol. The ChipCorder driver will select the ISD4004 device by asserting the Slave Select (SS) line. A LOAD signal will then be asserted to load the current instruction opcode into the SPI PHY's shift register. An SPI clock (SCLK) is

then generated. The SCLK must contain the correct number of pulses to serially send each bit of the current instruction.

User Interface

The user interface is probably the most important aspect in designs, 'The Reading Book' being no exception. User input is used to control the different functionality, and the output gives the user feedback on the current state of the system.

'The Reading Book' has both input and output for the user. There are two push buttons used as inputs and three LEDs to indicate the operating mode to the user. This module also provides voice feedback to the user for additional clarity. The main controller will keep track of a mode change and inform the Chipcoder to play the correct feedback audio.

The functionality of the I/O is as follows:

- Push Button #1: On pressing, the mode of the book is advanced through a circular queue of the three modes. (Auto/Manual Playback, Pre/User defined story, and Record On/Off)
- Push Button #2: On pressing, this button toggles the setting of the current mode. For safety, the user must press this button twice to enter Record On mode.
- LED #1: 'ON' indicates Auto Playback Mode is selected, 'OFF' indicates Manual Playback.
- LED #2: 'ON' indicates Prerecorded Story is selected, 'OFF' indicates User Recorded Story.
- LED #3: 'ON' indicates the user is in Record ON mode.

The user interface uses an FPGA for stabilization as well as mode decoding. This includes using debouncing logic to avoid errors when a push button is pressed. There is also logic to determine the current operating mode and provide that to the system's main controller. There are outputs from the mode decode block to enable LED lights which indicate the operating mode to the user.

Following is a detailed block diagram of the user I/O block. The two input signals, which are both push buttons, are fed directly into debouncing blocks. From there the mode signal is fed into the mode selector block which then provides the main controller with the current operating mode. The LEDs are also enabled appropriately.

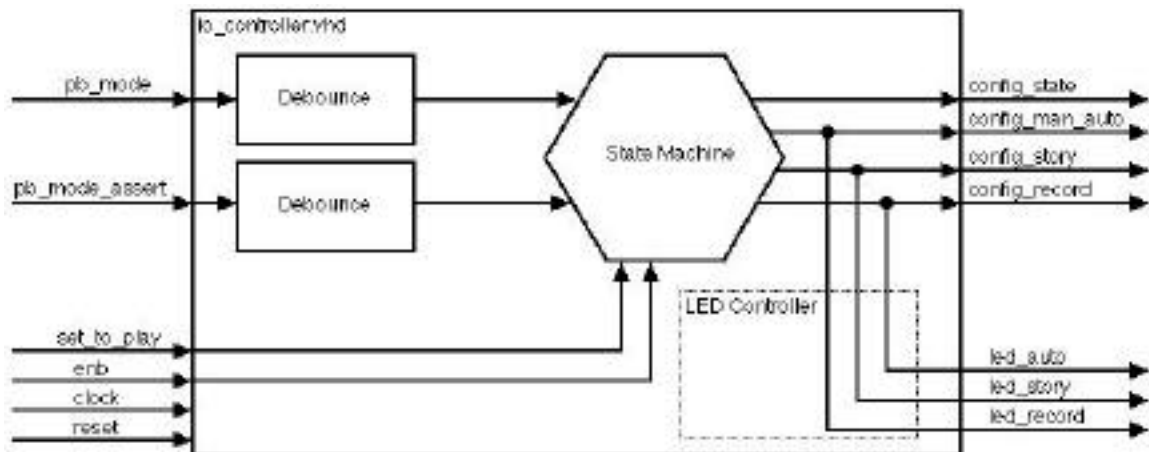


Figure 3: User I/O block diagram

The 'io_controller' has an enable signal 'enb' so that the main controller has a way to disable the block. Specifically by deasserting the enable the push-buttons will no longer have an effect. The main reason for the enable is that the main controller can prevent a user from modifying the mode or configuration while a book is not present.

There is a signal called 'set_to_play' which is a signal which allows the mode of the state machine to be placed into 'PLAY' mode without performing a reset, which in turn does not reset the configurations.

Signal Debouncing

At first, a push button seems like the simplest device to connect problem free with hardware. This is far from the truth. Push buttons do not have a steady transition, instead they bounce before a steady state is obtained. This bouncing can cause negative results for two separate reasons. One is that the signal may be in a transitioning state when latched and the other is that the toggling can cause a D-Flip-Flop to be enabled several times.

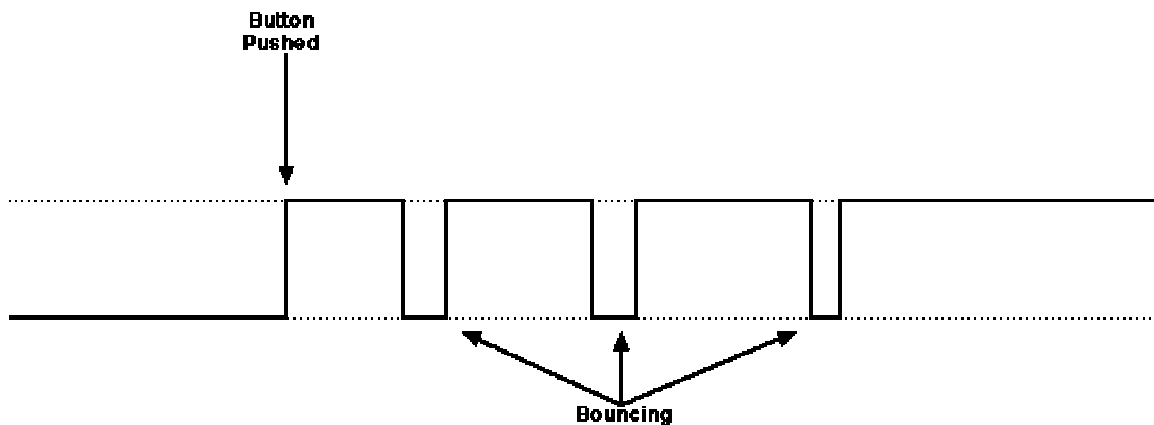


Figure 4: Example of active high push-button ringing

To solve the problem of debouncing both data buffering and less frequent sampling.

Data buffering will be the primary solution used to debounce the input signals from the push buttons. This method will simply be a history of the previous N samples of the signal line. Using this method, the determining factor that a push button was pressed and has stabilized will be to logically check for equality of all N bits in the data buffer.

The second part to this solution is to adjust the rate at which the input signal will be sampled. When reducing the clock frequency that controls the data buffer sampling rate, the data buffer will consist of values more spaced in time. Spacing out the sampling locations will reduce the probability that the debouncer will accept a large bounce as a stable state. In other words, if the sampling frequency is not reduced then the sampling time to fill the data buffer might occur completely in a single bounce cycle, obviously providing the system with an incorrect input signal.

Calculations and testing is essential to verify that the N bits of the data buffer as well as the rate of sampling will be sufficient to eliminate all errors caused by debouncing.

Mode Decoding

There are four configurations which the 'io_controller' keeps track of and reports to the main controller.

1. config_state - Whether 'The Reading Book' is in a 'play' state or whether the user is currently changing the configurations (mode_select).
2. config_man_auto - Whether the auto-page-detection is enabled (auto) or else the user must push a button in order to play a given page (manual).
3. config_story - Whether the prerecorded audio is selected for playback (default) or else the user auto will playback (user).
4. config_record - Whether the book is currently recording or not (record_on or record_off).

The state-machine which governs the 'io_controller' is shown in figure 5.

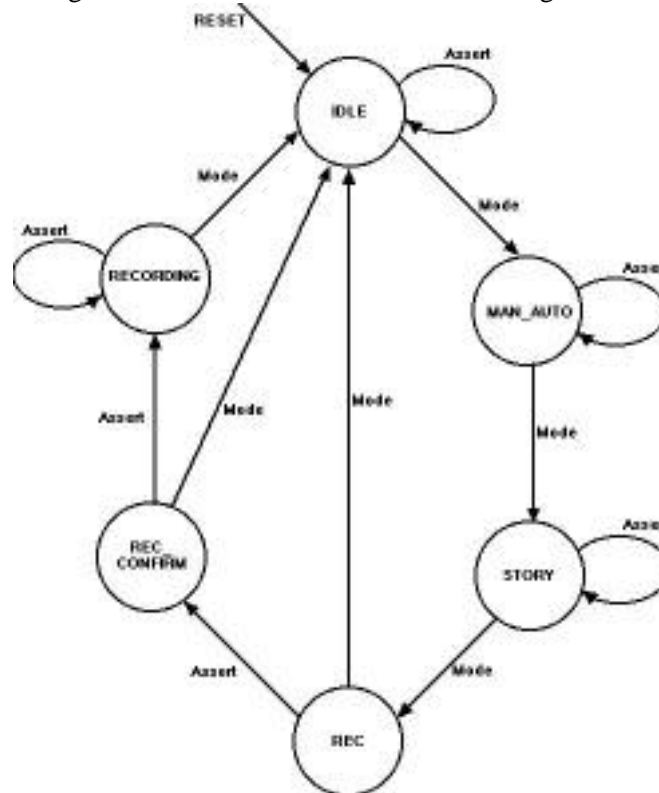


Figure 5: IO State Machine

A description of each the states are as follows:

IDLE

- Description - This is the reset state of the state machine. It is also the state for normal operation or 'non_configure' mode.
- Actions
 - Mode - This will advance the state machine to MAN_AUTO.
- set the STATE configuration to 'mode_select'.
 - Assert - This button will command the main controller to play the current page.

MAN_AUTO

- Description - This is the configuration state to adjust the current mode between Auto Play and Manual Play.
- Actions
 - Mode - This will advance the state machine to STORY.
 - Assert - This button will toggle the current configuration between 'auto' and 'manual' play.
- A LED will be set accordingly for which mode it is in.

STORY

- Description - This is the configuration state to adjust the current mode between the Default Audio and the User Audio.
- Actions
 - Mode - This will advance the state machine to REC.
 - Assert - This button will toggle the current configuration between 'default' and 'user' audio.
- A LED will be set accordingly for which mode it is in.

REC

- Description - This is the state to inform the user that they are at the point of where they can select to record.
- Actions
 - Mode - This will cancel the record mode, and take the user out of configuration mode and place the user back into regular play mode.
- set the STATE configuration to 'play'.
 - Assert - This button will allow the user to select a recording mode and advances the user to a confirmation state before they begin to record.
- This will advance the state machine to REC_CONFIRM.

REC_CONFIRM

- Description - This is the confirmation state for the user before they begin to record their audio.
- Actions:

- Mode - This will cancel the record mode, and take the user out of configuration mode and place the user back into regular play mode.
 - set the STATE configuration to 'play'.
- Assert - This will advance the state machine to RECORDING.
 - set the RECORDING configuration to 'recording_off'.
 - A LED will be asserted to inform the user that in is in recording mode.
 - set the RECORDING configuration to 'recording_on'.

RECORDING

- Description - This is a state that the user is in during the duration of their audio.
- Actions
 - Mode - This will halt the recording of the users audio and place them back in regular play mode.
 - set the STATE configuration to 'play'.
 - set the RECORDING configuration to 'recording_off'.
 - Assert - is passed through to the main controller to begin recording.

Main Controller

The main controller is the main intelligence behind "The Reading Book" which connect all the other blocks together. Specifically it connects the user interface, the Chipcorder, and the page detection logic.

A general block diagram below shows the functionality of the main controller. It can be seen that the main controller decodes the correct memory depending on the current state and configuration of the system. It then controls the Chipcorder depending directly on the inputs which supply the state of operation.

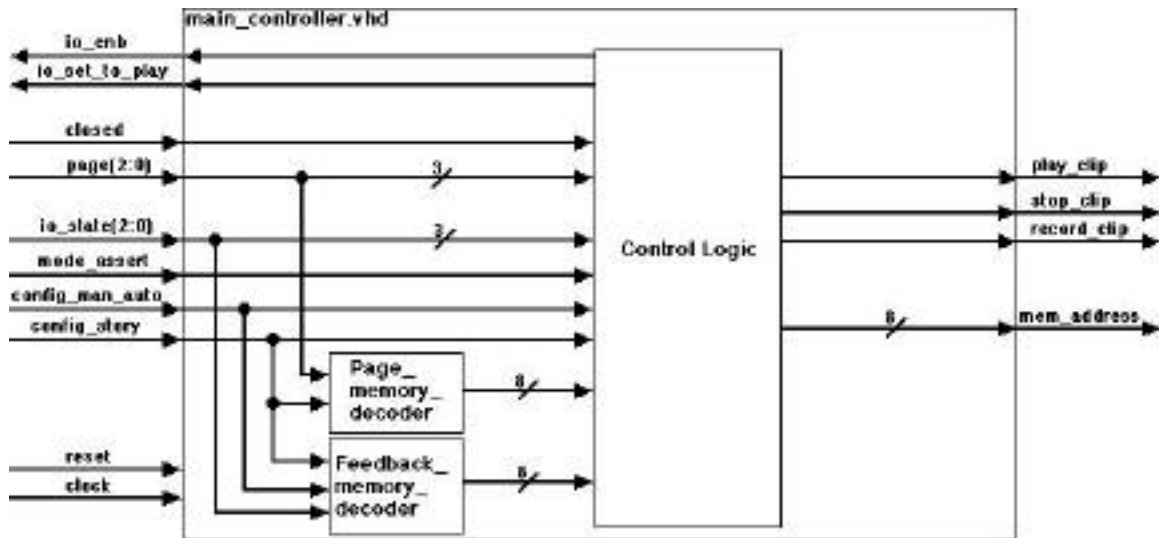


Figure 6: Main Controller Top Level Block Diagram

To enhance the performance of the main controller the design was modified to take two clock cycles to propagate data through. Although the latency increased the throughput was greatly improved. The speed factor was not an issue with this design since it is operating at a relatively slow clock frequency.

The design takes one clock cycle to calculate all the possible memory locations it may require: current page to play, current page to record, and current feedpage. During the second clock the control logic determines if a change in state has occurred and needs to inform the Chipcorder of a new event. If so, the main controller asserts the correct signal to the Chipcorder (play_clip, stop_clip, or record_clip). Also at this time if either the play or record signal is asserted the memory address bus is latched with the correct value.

The main controller only asserts the signals supplying the Chipcorder with pulses of 1 clock pulse width. The memory address bus is latched at time of change and remains constant until a new value is latched. With the interface used commands should never occur consecutive but if they do our system is capable of handling it.

Likewise, the main controller requires inputs that follow certain rules which are clearly defined by our other modules which connect to it. For example, some inputs that are provided must only be single pulses.

FPGA Requirements

The “Reading Book” will be built using the Altera Flex10K20 UP1 board. The following table summarizes the logic block requirements of each component.

Name of Component	Number of Logic Blocks (%)	Measured/Estimated
Page Detection and Book Interface	72 (6%)	Measured
Recorder Interface	208 (18%)	Measured
User I/O	219 (19%)	Measured
Main Controller	175 (15%)	Measured
Complete Project	674 (69%)	Measured

Based on our measurements and estimates we are confident that the Flex10K20 chip will meet all of our requirements.

Experiments and Characterizations

The design of the Reading Book has gone through a number of changes due to the results of experiments. This especially true in the page detection and recorder modules. The I/O section has not undergone any significant changes in design as the result of experimentation.

Page Detection Experimentation

Approach 1

The initial design of the page conversion module was based on the following equation:

$$N_x = N_{MAX} \left(\frac{V_s}{V_x} - 1 \right)$$

Where N_{max} is the number of pages in the book, V_s is the A/D value of the voltage source (255) and V_x is the A/D value of the voltage detected at either the front or back cover of the book. (Refer to figure A of the schematics section) This implementation required a divider, multiplier and adder unit. While the design produced accurate results, it required 584 logic blocks or 50% of the total available. Because of this, an alternative solution was suggested and implemented.

Approach 2

The second solution used the idea that if we could keep the number of pages in the book constant, then we could build a simple decoder to determine the page number based on precalculated expected values. The problem with this was that we wanted the capability to have several different books, with different number of pages, that could be plugged into the base unit and used. To solve this problem it was decided that no book would have more than 6 pages. Then by altering the value of the resistance in the back cover of the book, we are able to make a book with any number of pages less than or equal to 6 behave as if it were a book with 6 pages. For example, if a book has 4 pages it effectively becomes a 6 page book where the last two pages are never opened because they are embedded in the back cover. With these modifications the size of the book interface dropped to 72 logic blocks or 6% of the total available.

Conclusion of Experiment

Approach 2 is the obvious choice for our project. While it is not as flexible as approach 1, the savings in chip space makes it the better design.

Recorder Chip Experimentation

In one experiment, the SPI PHY and SPI Controller were generalized so that they could be used with various different clock configurations and so that they could be reused to interface to other peripherals, such as a digital potentiometer for volume control.

SPI PHY

The LPM shift register originally used proved to be insufficiently customizable for the purposes of this design. A custom shift register had to be created so that loads would occur when the load signal is asserted and shifts would occur on a selectable edge (selected at instantiation) of the shift signal. The SPI SCLK signal would be connected to the shift port of the register to shift data out of the register. In this manner, data signals in phase and out of phase with the SCLK signal can be generated by simply shifting data out onto the MOSI line on one edge or the other. Changes in polarity can be realized by simply inverting the SCLK signal and shifting data out on the desired edge. In order for a truly generic SPI PHY, ie. one that can handle any of the four combinations of SCLK phase and polarity, this design would require that two of these components be instantiated (one clocked on the rising edge and one clocked on the falling edge) and multiplexed onto the bus, or bus arbitration logic would be required to tri-state one device while enabling the other. Only the first approach was deemed to be feasible since we could not think of a way to realize the second approach on an FPGA.

SPI Controller

The SPI Controller would ideally control the edge on which to shift data out (ie. which of the two SPI PHY's described above should be in control of the bus), the loading of the selected PHY's shift register, and the generation the correct number of SCLK pulses with the correct polarity. A higher level entity would communicate to this controller by telling it what to send and when it is ready to send it. The controller would send the data and notify the higher level entity that the transmission is complete. The initial ChipCorder driver was coded according to this approach, and simulations showed that it should work. Experimentation showed otherwise: the state machines would get out of sync with each other once in a while so there are some handshaking issues to work out. When the two entities were combined, the issues could not be reproduced through experimentation and the simulated behaviour was realized. Further investigation is needed to determine the cause of the desynchronization.

Approach 1

Through the modularized approach, the following results were obtained under normal fitting with Quartus Fitter:

Component	Logic Cells Used	Minimum clock period	Maximum frequency
SPI PHY	10	8.6 ns	116.27 MHz
SPI Controller	107	58.3 ns	17.15 MHz
ChipCorder Driver	98	32.4 ns	30.86 MHz

The limiting factor is the adder and comparator used in the SPI Controller to control the counter that controls the number of SCLK pulses.

By structurally connecting up the SPI PHY, SPI Controller, and ChipCorder Driver into the Digital Recorder Module of the overall design, the following metrics were obtained for various methods of fitting.

Fitter Setting	Logic Cells Used	Minimum clock period	Maximum Frequency
Normal w/ Quartus	208	61.3 ns	16.31 MHz
Normal w/o Quartus	208	63.8 ns	15.67 MHz
Advanced w/ Quartus	208	61.3 ns	16.31 MHz
Advanced w/o Quartus	208	63.8 ns	15.67 MHz

Without custom tweaking any parameters, the module uses **208** logic cells, and runs at a Maximum of **16.31 MHz** (Minimum clock period of **61.3 ns**). The critical path is through the counter of the SPI Controller.

Approach 2

By combining the SPI Controller and ChipCorder Driver into one entity, the following results were obtained under normal fitting with Quartus Fitter:

Component	Logic Cells Used	Minimum clock period	Maximum frequency
SPI PHY	10	8.6 ns	116.27 MHz
ChipCorder Driver	304	72.0 ns	13.88 MHz

By structurally connecting up the SPI PHY and the new ChipCorder Driver into the Digital Recorder Module of the overall design, the following metrics were obtained for various methods of fitting.

Fitter Setting	Logic Cells Used	Minimum clock period	Maximum Frequency
Normal w/ Quartus	312	70.2 ns	14.24 MHz
Normal w/o Quartus	312	75.4 ns	13.26 MHz
Advanced w/ Quartus	312	70.2 ns	14.24 MHz
Advanced w/o Quartus	312	75.4 ns	13.26 MHz

Without custom tweaking any parameters, the module uses **312** logic cells, and runs at a Maximum of **14.24 MHz** (Minimum clock period of **70.2 ns**). The critical path is through the counter of the SPI Controller that is integrated into the ChipCorder Driver.

Conclusion of Experiment

If the handshaking issues can be resolved, approach 1 is the preferred approach because of the smaller area, faster speed, and reusable design.

Physical Construction Experimentation

The construction of the reading book was mainly dependent on the method of page detection that was to be implemented. The most important features of the page detection would be to be able to tell which page was open and if more than one page was open. Many different methods for page detection were discussed.

Switches on each page – This method would provide a discrete signal for each page. The difficulties of this method include design of the switches, the book would require as many wires as there are pages plus two and along with that the wiring would be difficult.

Hall effect sensors – This method would require putting hall effect sensors on both the back and front covers and inserting magnets into each of the pages of the book. The sensors would be able to detect the number of magnets present on that side of the book and therefore detect the page number and if more than one page was open. Also the number of lines to the book would be minimal as only two input lines and two output lines would be used. This method was not implemented because doubts existed on the ability of the sensor to detect all of the magnets and doubts about the resolution even if they were detected. This would still be an interesting method to research if time were to permit.

Resistor between two contacts – This is the method that was finally implemented. It consists of two contacts on each page that can both connect with the page before and the page after. Between the two contacts is a resistor. This is illustrated in Figure 2. Doing this creates parallel resistor networks that correspond to the number of pages contacting. Measuring a voltage off the front and back of the book gives the current page and if more than one page is open.

The contacts decided upon were magnets. This had the benefit of holding the pages together to ensure proper contact. This required a couple of things from the selected magnets. They must be small enough to fit in the pages of the book and they must also be conductive. Most magnets are made from ceramics or plastics so special magnets were ordered that would conduct. These magnets were not the correct size but it was assumed they could be easily machined to the right size. This ended up not being possible. The facilities to machine these magnets were not available and the magnets were harder to machine than had been anticipated. New magnets were ordered that were the appropriate size. When they arrived it was discovered that despite the fact that the size was correct they were of the wrong material. These magnets were not conductive. Due to time constraints it was decided to treat the surface so that the magnets would be conductive. Three methods were attempted.

Conductive paint – A nickel print was used to coat the magnets. This made the magnets conductive but upon soldering to the magnets the solder would pull off the print from the magnets causing it not to stay connected.

Aluminum foil – Regular household aluminum foil was glued to the magnets. This made the magnet conductive but due to the coatings and films on household aluminum foil soldering to the foil was not possible.

Copper foil – The product used here is a special adhesive backed copper foil that is used in making stained glass windows. This was very convenient, as we did not have to use

any glue. This worked very well as it made the magnets conductive and was very easy to solder to.

From this point the construction of the book went fairly well. A simple child's board book was used to construct the project. The pages were separated and grooves were cut to enable the resistors to be embedded in the pages. It was also discovered that to enable the contacts to connect the two magnetic contacts would have to be placed at on the same side of the page but at the top and bottom corners.

○ = Magnet

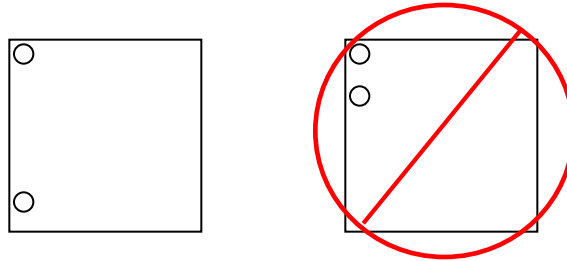


Figure 7: Magnet Positions

When the magnets were placed closer together it was difficult for both of the magnets to connect with another page.

Conclusion of Experiment

The design decisions made for the physical construction of the book proved to be effective as the final book met all of the project requirements.

References

- [1] ISD4003 Series Datasheet, Winbond Electronics
- [2] Code for A/D converter interface was modified from a student application note submitted by Shauna Rae on October 29, 1999

Index to Test Cases and Design Verification

Page_convert....	22
<ul style="list-style-type: none">- shows the operation of the page_convert.vhd module- boundary conditions for each page are tested	
Book_Controller – ADC Conversion.....	23
<ul style="list-style-type: none">- shows the operation of the book_controller.vhd module- total of four ADC conversions	
Book_Controller – Test Bench Results.....	24
<ul style="list-style-type: none">- shows the results of the book_controller test bench- a comprehensive test of the book_controller.vhd module	
SPI PHY	25
<ul style="list-style-type: none">- shows the operation of the SPI PHY- parallel data is serialized and placed on the MOSI bus line- serial data is parallelized from the MISO bus line	
SPI Controller	26
<ul style="list-style-type: none">- shows the correct generation of the SCLK SPI bus control signal	
ChipCorder Driver	27
<ul style="list-style-type: none">- shows the correct operation of the ChipCorder device driver- all possible control signals are tested verifying the correct generation of SPI bus data	
Digital Recorder Module	28
<ul style="list-style-type: none">- shows the translation of high-level control signals down to the serial bitstream of the SPI PHY	
IO_Controller	29
<ul style="list-style-type: none">- shows the operation of IO_Controller Module- all states in the I/O state machine are tested	
Main_Controller	30
<ul style="list-style-type: none">- shows the cases that the main controller may encounter	
Clock Divider	32
<ul style="list-style-type: none">- shows the generation of a clock with frequency $1/50^{\text{th}}$ of the input clock	
Book Module	33
<ul style="list-style-type: none">- shows the operation of the two components in the book module	

Page_convert

This waveform shows the operation of the page_convert.vhd module. The boundary conditions for each page are tested.

Valid input range for value is 0 to 255, this represents the ADC output for either the front or back cover voltage.

Valid output range for page is 0 to 6.

Maximum Speed: No registers to measure performance for this module specifically, but it is measured as part of the book_controller performance.

Critical Path: As above.

Improvements: N/A

Book_Controller – ADC Conversion

This waveform shows the operation of the book_controller.vhd module. Shown is a total of four ADC conversions (two for the front cover and two for the back). This shows the proper setting of the page number and the closed signal.

The sequence of events for a proper ADC conversion is as follows:

- Select line is set for either front (0) or back (1) cover.
- Start line is pulsed high to begin conversion.
- Data_Ready line goes high, indicating the conversion is complete.

This sequence is repeated for every conversion.

This test only shows, in detail, the proper execution of ADC conversion and page detection for a couple of cases. A less detailed but comprehensive test performed by a test bench is included next in this section.

Maximum Speed: 87.71 MHz

Critical Path: Loading of registers.

Improvements: N/A

Book_Controller – Test Bench Results

This waveform shows a comprehensive test of the book_controller.vhd module. This test was performed using the book_controller test bench included later in this report.

After reset, the test bench cycles through all of the possible page combinations. The page combination currently being tested is indicated by the output vector 'test_case'. Tests are performed on the rising edge of the signal 'test'. The output signal 'pass' indicates whether the module passed the test, with '1' indicating a passed test and '0' indicating a failure. The following table summarizes the test cases:

Front Page

	0	1	2	3	4	5	6
0	7	6	5	4	3	2	1
1	13	12	11	10	9	8	X
2	18	17	16	15	14	X	X
3	22	21	20	19	X	X	X
4	25	24	23	X	X	X	X
5	27	26	X	X	X	X	X
6	28	X	X	X	X	X	X

Test Cases

Notes:

- An 'X' indicates a combination that is not physically realizable.
- The closed book condition is test case 29.

Maximum Speed: N/A

Critical Path: N/A

Improvements: N/A

ack Page

SPI PHY

This simulation shows the correct operation of the SPI PHY for the Reading Book: parallel data is serialized and placed on the MOSI bus line. Serial data is parallelized from the MISO bus line. The PHY shifts and samples data on the falling edge of SCLK in this configuration.

Tests performed but not shown in waveforms (important for generalizing the PHY):

Polarity Test

Feeding in SCLK with different polarity. SCLK and MOSI signals were tested against Motorola reference timing, found a bug that does not affect our design, but affects the generic properties of the SPI PHY: First shift occurs too soon (ie. loses first bit, see waveform).

Load/Shift Priority Test

Asserting both the load and SCLK. Load takes priority.

Rising Edge Test

Clocking data out on the rising edge instead of the falling edge. Works correctly.

Maximum Speed: 125.00 MHz

Critical Path: shift register

Improvements: N/A

SPI Controller

This simulation shows the correct generation of the SCLK SPI bus control signal. The generation of the 8 SCLK pulses occurs one clock cycle after the load signal has been asserted. This allows the PHY to load the parallel data to be sent on the bus. Tests were also done to show that the ready signal must be deasserted for at least two clock cycles between transfer operations (results not shown).

Tests performed but not shown in waveforms (important for generalizing the controller):

Phase Shifted Clock Test

SCLK180 signal generated and is 180 degrees out of phase with SCLK.

Maximum Speed: 17.15 MHz

Critical Path: counter to generate SCLK pulses

Improvements: shift register instead of counter. The bits of the shift register can be ORed together to activate pulse generation. To start the sequence, a '1' is shifted into the register. The critical path in this design would be the shift operation and the chained OR operations.

ChipCorder Driver

This simulation shows the correct operation of the ChipCorder device driver. All possible control signals are tested verifying the correct generation of SPI bus data.

Tests performed but not shown in waveforms (so as not to confuse interpretation of the waveforms):

Multiple Reset Test

State machine asserts and deasserts SS many times (as it is being reset over and over) and sends partial sequences of pulses out the SCLK line, but the final reset pulse is the one that will send the correct number of SCLK pulses with the correct behaviour of SS. This behaviour is by design and is the expected behaviour.

Multiple Play, Record, or Stop Test

Pulses occurring within a command sequence are ignored. Holding one signal for longer than the duration of a command sequence results in back to back command sequences as expected.

Asserting Play, Record, and Stop at the Same Time Test

Priority is given to Record, Stop, and Play in that sequence. This will be changed in the future to: Stop, Play, Record.

Maximum Speed: 14.04 MHz

Critical Path: counter to generate SCLK pulses

Improvements: use the modular design hierarchy of ChipCorder Driver -> SPI Controller -> SPI PHY as described previously. Also, make improvements to the SPI Controller as described previously.

Digital Recorder Module

This simulation shows the translation of high-level control signals down to the serial bitstream of the SPI PHY. Operations such as reset, record, play, and stop are initiated and the SS, SCLK, and MOSI bitstreams are analyzed.

The main testing occurs in the Testbench.

Maximum Speed: 12.67 MHz

Critical Path: same bottleneck as the ChipCorder Driver.

Improvements: same as the ChipCorder Driver

IO_Controller

In summary the waveforms show:

- Reset
- MAN_AUTO Mode
 - manual configuration
 - auto configuration
- STORY Mode
 - user configuration
 - default configuration
- REC Mode
 - REC abort via 'mode'
- REC_CONFIRM Mode
 - REC_CONFIRM abort via 'mode'
- RECORD Mode
 - re-assert to begin recording
 - RECORD abort via 'mode'
- 'enb' functional testing
- 'set_to_play' functional testing

The values of the modes are as follows:

-- IO States		
PLAY	= b"000"	(0)
MAN_AUTO	= b"001"	(1)
STORY	= b"010"	(2)
REC	= b"011"	(3)
REC_CONFIRM	= b"100"	(4)
RECORDING	= b"101"	(5)

Performance Results:

Mim Clock Period: 21.2 ns
Max Frequency: 47.16 MHz

Critical Path is to determine which state to enter of the state machine. With out a redesign this is not easily changed. A different state machine could be designed to have less states in it, and then incorporate all the functionality into a few state-machines working together.

Main_Controller

This set of waveforms shows the cases that the main controller may encounter. The main controller must monitor the book interface and the user input to determine what to either play or record, or in general, how to control the Chipcorder.

The main controller is concerned with mode, configuration, book status (open/closed and page number). It is designed to sample each of these every clock cycle and supply the Chipcorder pulses according to what is observed. Only pulses are sent to the Chipcorder when a change in status has occurred.

In summary the waveforms show:

- Reset
- Out of reset
 - no action
- Closed book
 - disable io_controller
- Page turn (page 2)
 - plays page audio (0x10 = PAGE_2_DEFAULT)
- Inbetween pages
 - stops play
- Completed the page turn (page 3)
 - plays page audio (0x11 = PAGE_3_DEFAULT)
- Assert page play (page 3)
 - plays page audio (0x11 = PAGE_3_DEFAULT)
- Mode change to MAN_AUTO
 - plays feedback audio (0x03 = FB_AUTO)
- Change configuration to 'manual'
 - plays feedback audio (0x02 = FB_MANUAL)
- Change configuration to 'auto'
 - plays feedback audio (0x03 = FB_AUTO)
- Mode change to STORY
 - plays feedback audio (0x04 = FB_DEFAULT)
- Change configuration to 'user'
 - plays feedback audio (0x05 = FB_USER)
- Change configuration to 'default'
 - plays feedback audio (0x04 = FB_DEFAULT)
- Mode change to REC
 - plays feedback audio (0x06 = FB_REC)
- Mode change to REC_CONFIRM
 - plays feedback audio (0x07 = FB_REC_CONFIRM)
- Mode change to RECORDING (page 3)
 - begins to record (0x0A = PAGE_3_USER)
- Assert recording page (page 3)
 - begins to record (0x0A = PAGE_3_USER)
- Test REC abort (mode goes to PLAY)
 - plays feedback audio (0x01 = FB_PLAY)
- Test REC_CONFIRM abort (mode goes to PLAY)
 - plays feedback audio (0x01 = FB_PLAY)

- Test RECORDING abort (mode goes to PLAY)
 - plays feedback audio (0x01 = FB_PLAY)

For test purposes the values of the memory have been set to the following:

-- Memory locations

```
FB_PLAY      x"01";
FB_MANUAL    x"02";
FB_AUTO      x"03";
FB_DEFAULT   x"04";
FB_USER      x"05";
FB_REC       x"06";
FB_REC_CONFIRM x"07";
```

```
PAGE_1_USER  x"08";
PAGE_2_USER  x"09";
PAGE_3_USER  x"0a";
PAGE_4_USER  x"0b";
PAGE_5_USER  x"0c";
PAGE_6_USER  x"0d";
PAGE_7_USER  x"0e";
```

```
PAGE_1_DEFAULT x"0f";
PAGE_2_DEFAULT x"10";
PAGE_3_DEFAULT x"11";
PAGE_4_DEFAULT x"12";
PAGE_5_DEFAULT x"13";
PAGE_6_DEFAULT x"14";
PAGE_7_DEFAULT x"15";
```

-- IO States

```
PLAY          = b"000"    (x"0")
MAN_AUTO      = b"001"    (x"1")
STORY         = b"010"    (x"2")
REC           = b"011"    (x"3")
REC_CONFIRM   = b"100"    (x"4")
RECORDING     = b"101"    (x"5")
```

Performance Results:

```
Mim Clock Period:  30.8 ns
Max Frequency:     32.46 MHz
```

Critical Path for the main controller involved the internal variable 'page_last' which is used to inform the main controller that the page has just been turned. There are a few solutions to improve this if desired. One is to put a signal in from the book interface to inform the main controller that the page was changed. This would completely eliminate this function from the main controller. Another solution would be to divide this task among 2 clock cycles to improve the throughput. Two clock cycles are already designed into the main controller but not regarding 'page_last' which could be if speed performance was an issue.

Clock Divider

This waveform shows the operation of the clock_divider.vhd module. For this simulation the input clock had a period of 20ns (50 MHz), and it is shown that the output clock has been divided by 50 (1us, 1 MHz), which is the proper operation. This waveform also shows proper reset operation.

Maximum Speed: 92.59 MHz

Critical Path: The counter.

Improvements: N/A

Book Module

This simulation shows the correct operation of the two components in the book module.

Tests performed:

ADC Input Test

ADC stimuli were fed in to observe the correct outputs on the *page* and *closed* signals.

Digital Recorder Test

High level play, record, and stop stimuli were fed in with appropriate memory page information to observe correct output from the SPI interface.

Maximum Speed: 13.69 MHz

Critical Path: SCLK counter in ChipCorder Driver

Improvements: N/A

Diagram of Design Hierarchy

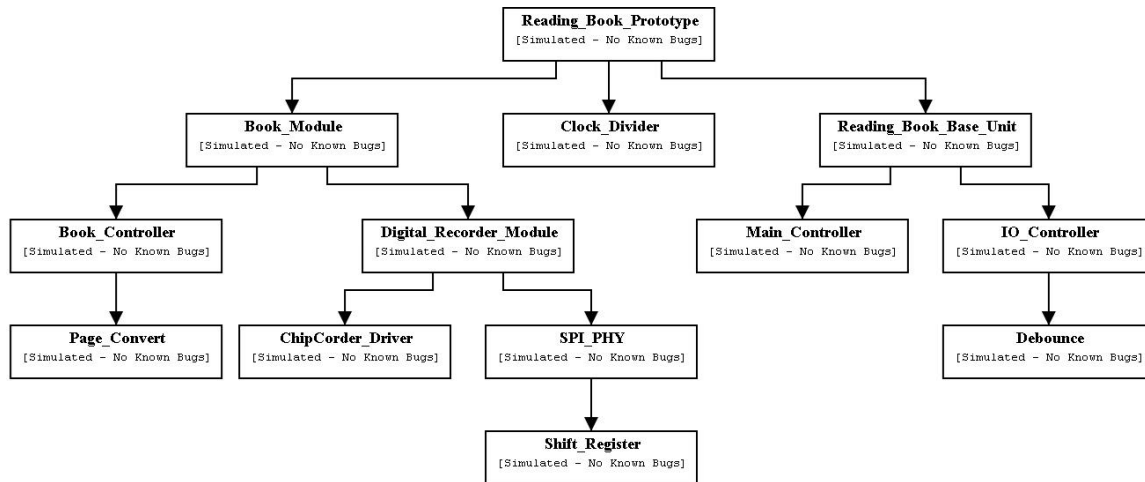


Figure 8: VHDL entity hierarchy

Index To VHDL Code

Page Detection Code	Error! Bookmark not defined.
Page_convert.vhd: Simulated – no known bugs	Error! Bookmark not defined.
book_controller.vhd : Simulated – no known bugs	Error! Bookmark not defined.
Chip Recorder Code	Error! Bookmark not defined.
shift_register.vhd: simulated - no known bugs	Error! Bookmark not defined.
register_pkg.vhd: compiled – no errors	Error! Bookmark not defined.
spi_phy.vhd: simulated - no known bugs	Error! Bookmark not defined.
spi_controller.vhd: simulated – no known bugs (not used in current design)	Error! Bookmark not defined.
spi_pkg.vhd: compiled – no errors	Error! Bookmark not defined.
chipcorder_driver.vhd: simulated – no known bugs	Error! Bookmark not defined.
chipcorder_pkg: compiled – no errors	Error! Bookmark not defined.
digital_recorder_module.vhd: simulated – no known bugs	Error! Bookmark not defined.
digital_recorder_package.vhd: compiled – no errors	Error! Bookmark not defined.
chiptest.vhd: compiled – no error (oscilloscope verifies correct signals)	Error! Bookmark not defined.
I/O Code	Error! Bookmark not defined.
debounce.vhd: simulated – no known bugs	Error! Bookmark not defined.
io_controller.vhd: simulated – no known bugs	Error! Bookmark not defined.
Main Controller Code	Error! Bookmark not defined.
main_controller.vhd – Simulated – no known bugs	Error! Bookmark not defined.
Top Level Code	Error! Bookmark not defined.
Reading Book Base Unit.vhd – Simulated – no known bugs	Error! Bookmark not defined.
Book Module.vhd – Simulated – no known bugs	Error! Bookmark not defined.
clock_divider.vhd – Simulated – no known bugs	Error! Bookmark not defined.
reading_book_prototype.vhd – Simulated – no known bugs	Error! Bookmark not defined.

Index To Test Benches

Book Controller Test Bench	36
Digital Recorder Module Testbench	Error! Bookmark not defined.
I/O Controller Test Bench	Error! Bookmark not defined.
Main Controller Test Bench	Error! Bookmark not defined.

Book Controller Test Bench

This test bench tests the book_controller.vhd module. It is not a test bench in the strictest sense because the entity contains inputs and outputs (a restriction imposed by MaxIIplus); however, it does provide an easy means for testing all possible page combinations. After reset, the test bench cycles through all of the possible page combinations. The page combination currently being tested is indicated by the output vector 'test_case'. Tests are performed on the rising edge of the signal 'test'. The output signal 'pass' indicates whether the module passed the test, with '1' indicating a passed test and '0' indicating a failure. The following table summarizes the test cases:

Front Page

	0	1	2	3	4	5	6
0	7	6	5	4	3	2	1
1	13	12	11	10	9	8	X
2	18	17	16	15	14	X	X
3	22	21	20	19	X	X	X
4	25	24	23	X	X	X	X
5	27	26	X	X	X	X	X
6	28	X	X	X	X	X	X

Test Cases

Notes:

- An 'X' indicates a combination that is not physically realizable.
- The closed book condition is test case 29.

Appendix A

Schematics

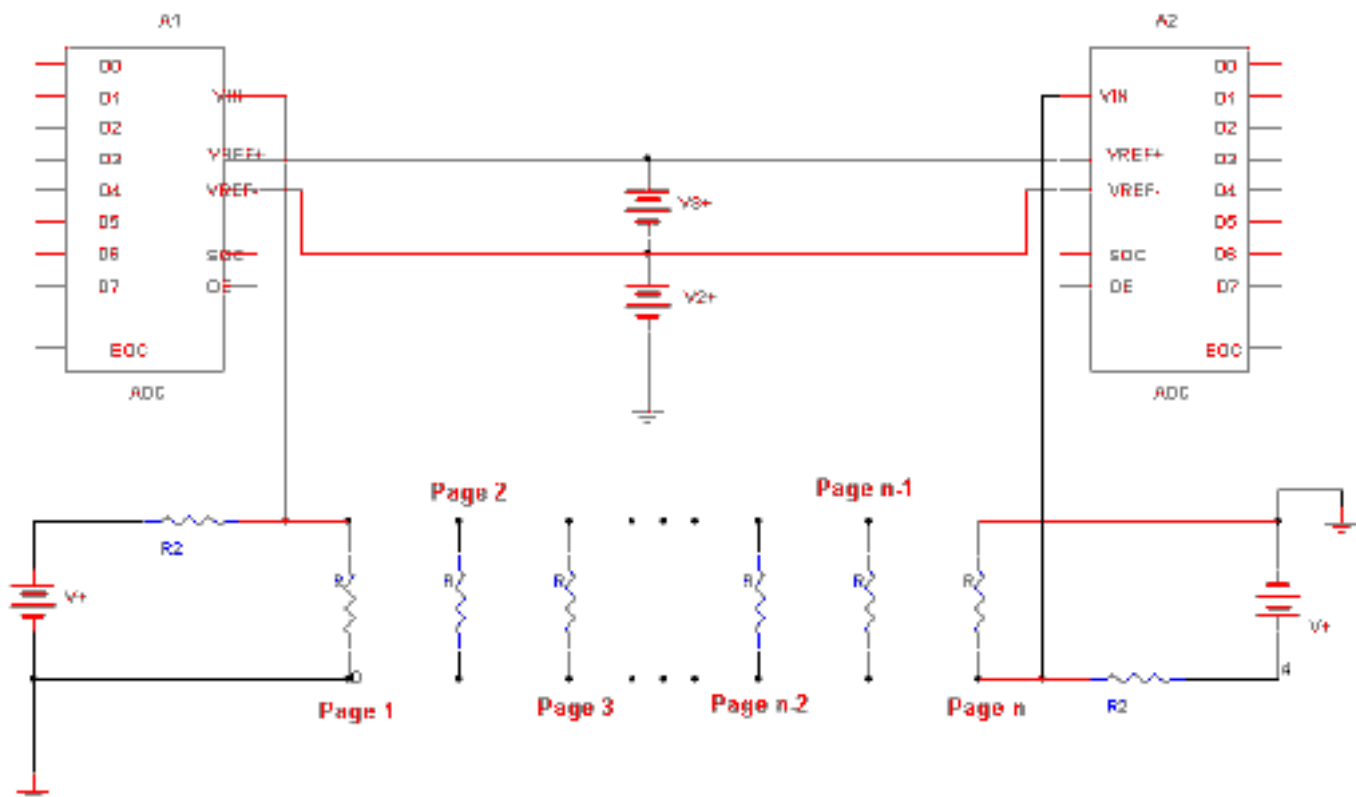


Figure A - Page Detection Circuitry

Note: The 2 ADCs shown are actually implamented using 1 ADC with an internal MUX.

Appendix B

Datasheets