



VSI402ZX
Board Design
Considerations
Application Note



Contents

1. Introduction	4
2. References	4
3. Power Supplies	4
3.1. Power Supply Sequencing	5
4. Host Port Interface	5
4.1. General Use	5
4.2. Loading code through the HPI interface	6
5. TSerial Port	7
6. Unused Chip Select Lines as Interrupt Signals	7
7. XBUS Implementations	8
8. Board Bringup Considerations	8

Trademark Acknowledgments

© VeriSilicon Holdings Co., Ltd. All rights reserved worldwide. VeriSilicon, the VeriSilicon logo, ZSP and the ZSP logo are the trademarks of VeriSilicon Holdings Co., Ltd. in the United States and/or other jurisdictions. All other trademarks are the property of their respective holders.

Printed in P.R.China.

VeriSilicon Holdings Co., Ltd. reserves all its copy rights and other intellectual property rights, ownership, powers, benefits and rights arising or to arise from this manual. All or part of the contents of this manual may be changed by VeriSilicon Microelectronics (Shanghai) Co., Ltd. without notice at any time for any reason, including but not limited to improvement of the product relating hereto.

VeriSilicon Holdings Co., Ltd. shall not undertake or assume any obligation, responsibility or liability arising out of or in respect of the application or use of the product described herein, except for reasonable, careful and normal uses.

Nothing, whether in whole or in part, within this manual can be reproduced, duplicated, copied, changed or disposed of in any form or by any means without prior written consent by VeriSilicon Holdings Co., Ltd.

1. Introduction

Implementing the VSI402ZX on a PC board requires the system designer to consider a number of issues: system interfaces, power supplies, clock sources. This application note covers some of the basic items that should be considered when designing a VSI402ZX part into a system.

2. References

- *EB402 Evaluation Board User's Guide*
- *VSI402ZX Digital Signal Processor User's Guide*
- *Interfacing Macraigor and Corelis Emulators to the VSI402ZX/VSI403Z Application Note*

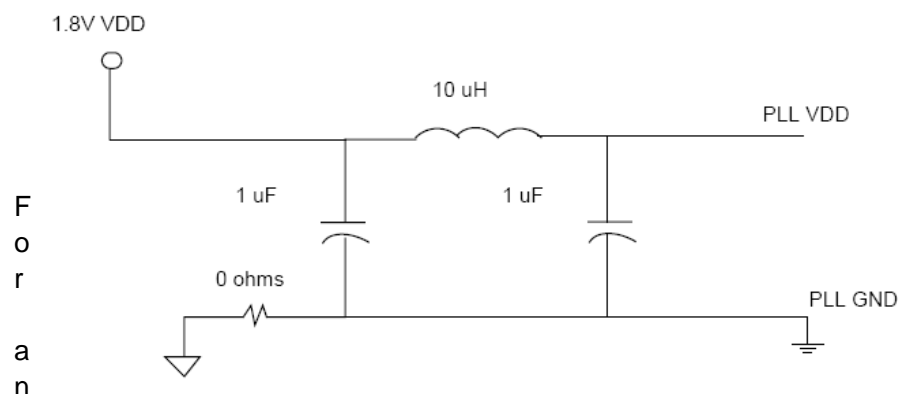
3. Power Supplies

The VSI402ZX requires three separate power supplies: 1.8V for the internal operating voltage, 3.3V for the external interface signals, and 1.8V for the internal PLL. The PLL requires an isolated 1.8V power and ground to reduce noise.

All three voltage rails should be properly decoupled. At the VSI402ZX, each rail would typically have three types of decoupling capacitors: a large (10uF) bulk cap, mid-size (0.1uF) and small (0.01uF) caps. These capacitors should be located as near the power/ground pairs as possible around the entire part.

The PLL 1.8V rail can be derived from the processor 1.8V rail through the use of a simple inductor filter. A 10uH inductor with decoupling caps will generally suffice. A separate ground signal is recommended for the PLL with a connection to the primary ground plane. [Figure 1](#) shows an illustration of a simple filter circuit.

Figure 1 PLL Filter Circuit



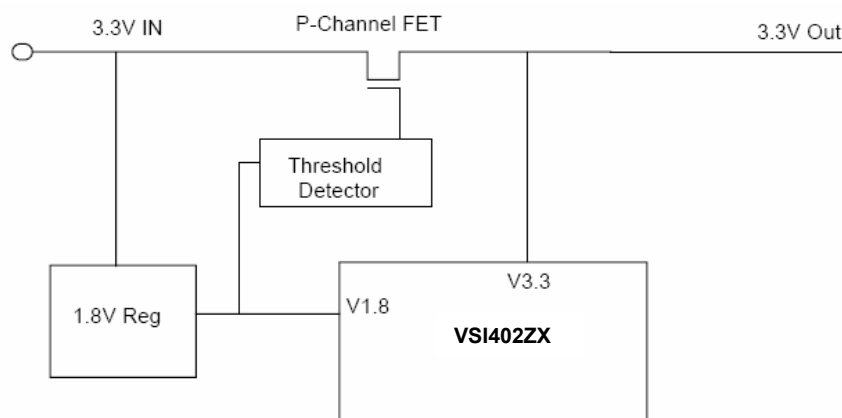
3.1. Power Supply Sequencing

The VSI402ZX is built on a technology that recommends the 1.8V internal operating voltage to be applied before the 3.3V IO voltage is applied.

In instances where the 1.8V is generated from a linear regulator off of the 3.3V, a simple switch can be used to turn on the 3.3V supply to the remainder of the board after the 1.8V has reached a stable voltage. [Figure 2](#) shows an illustration of a circuit which switches on the 3.3V rail after the 1.8V rail has reached it's output value.

The RSTN signal should be held low during powerup until the power supplies are stable and the system clock is running. A minimum of 5 processor clocks should occur before reset is released. When using the internal PLL, consider the PLL lock time when determining the time required to achieve 5 system clocks. After the power supplies are stable and the core has had at least 5 clocks, RSTN can be released and the processor will start executing instructions.

Figure 2 FET Switch for 3.3V Rail



4. Host Port Interface

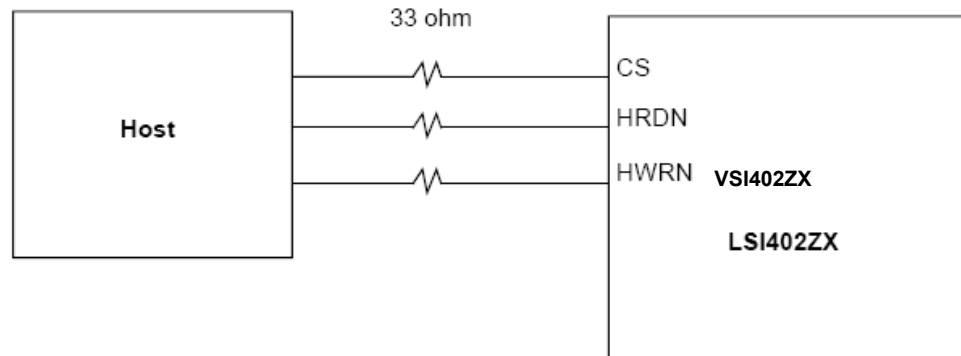
The VSI402ZX has a host port interface which enables a host processor to read and write via a parallel interface to the VSI402ZX. To the host, the interface looks like a simple, asynchronous interface, similar to a Flash or SRAM. The VSI402ZX has handshake flags (Input Buffer Full and Output Buffer Empty) which can be used in a similar manner to Ready flags on a memory device.

4.1. General Use

Since the HPI interface is fully asynchronous to the VSI402ZX, care must be taken in the design to ensure the control lines to the VSI402ZX are clean and do not allow any spurious transitions due to overshoot or undershoot. Transitions above or below the threshold can cause an inadvertent extra read or write since the HPI interface is triggered by a transition above the threshold voltage. Series termination is recommended near the source signal for all three

control signals: CS, HRDN, HWRN. Figure 3 shows an illustration of the recommended series termination.

Figure 3 HPI Series Termination



Some programmable logic parts such as FPGAs, allow adjustable slew rates on their output signals. When these parts are used to interface to the HPI port, the slew rates on the output lines should be set to a slower setting to minimize any overshoot/undershoot.

If HPI communication errors are suspected, look at the IBF and OBE signals during transactions. If either of these flags sets earlier than expected or stays on later than expected, then a communication error may have occurred due to a glitch on one of the control lines. A logic analyzer can be configured to trigger on these events.

4.2. Loading code through the HPI interface

The HPI interface can be used to load code into the VSI402ZX. The VSI402ZX user's guide describes the process. Two other details can be used to assist in this process.

When the VSI402ZX boots from ROM, it first executes a self-test sequence. After the self-test code finishes, it drives PIO0 low. The ROM code then enters an idle loop where it waits for either an HPI interrupt or a JTAG debug interrupt. After about 5 seconds at a processor clock rate of 100MHz, the PIO0 line is driven back high by the idle task.

This high to low transition on the PIO0 line can be used as a signal to the host that the VSI402ZX is ready to accept a code download. Once the code download starts, the ROM code will not automatically set PIO0 back high after a few seconds. PIO0 will remain low unless an error in the code download process is detected, at which point it will go high.

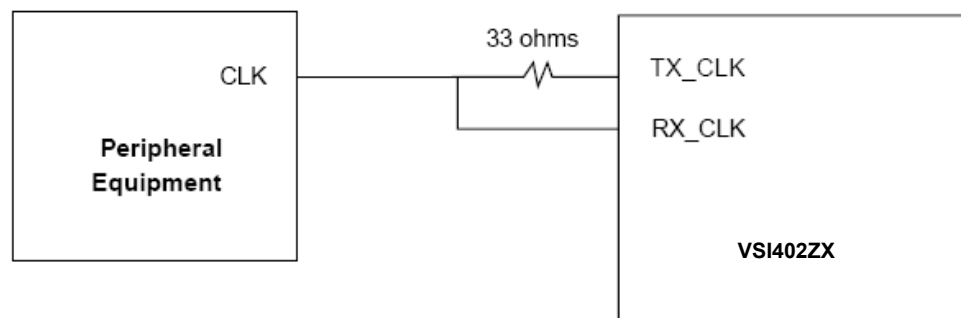
The host processor can monitor PIO0 during the code download process. If it ever goes high, then the code download process has failed. At this point, the host should reset the VSI402ZX and repeat the code download process. Monitoring PIO0 enables the host to determine if the code download has failed.

5. Serial Port

The serial ports on the VSI402ZX can be configured in different modes, with either the VSI402ZX or an external device serving as the source for the transmit clock and transmit frame sync. All input signals are asynchronous to the VSI402ZX system clock. Because of this, care must be taken to ensure the input signals do not have overshoot/undershoot/bounce attributes which could act like multiple pulses. This is particularly important for the clock signals since they cause internal state transitions on the clock edges. The data lines are sampled so they are often stable at the sampling clock edge. Because of this, they are less sensitive to noise. Frame sync signals also are sampled at clock edges, so again they tend to be less sensitive to noise.

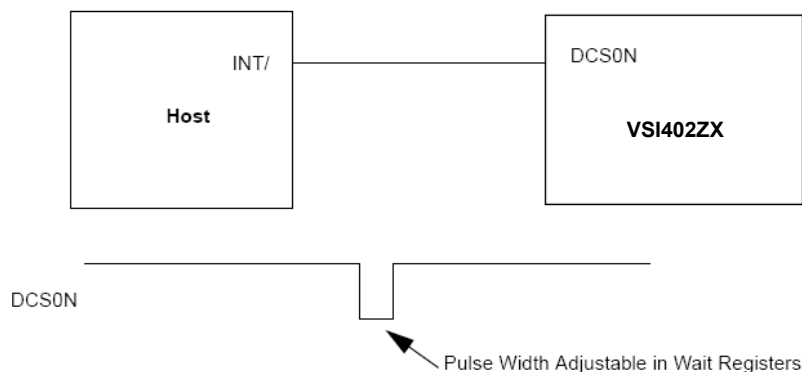
When using the VSI402ZX as the source for TX-CLK, series termination applied to the signal line near the chip can help reduce any noise due to reflections or the like. [Figure 4](#) illustrates using the VSI402ZX as the source for the TX & RX clock and a series termination.

Figure 4. Serial Port Connection



6. Unused Chip Select Lines as Interrupt Signals

Unused chip select lines (DCSxN, PCSxN) can be used as an active low strobe to an external processor. These lines can generate a variable width active low pulse which can strobe the interrupt input on a host or other DSP. By creating a memory map which has a hole allocated for this chip select line, a simple write or read from that address range can toggle the interrupt input line to another processor or chip. In the case where GPIO lines are not available, this may save the system designer from having to add additional components to drive the interrupt lines on a host. [Figure 5](#) shows an illustration of this technique. Note that the CS pulses are all active low, so the host interrupt pin must be edge triggered, not level since the CS is a pulse. The width of the CS pulse can be adjusted by writing to the appropriate WAIT register.

Figure 5 Example of DCS0N line as an Interrupt Source

7. XBUS Implementations

The XBUS (external bus interface) on the VSI402ZX differs from other ports in that it is driven by the VSI402ZX instead of acting as an input device. In this case, the VSI402ZX source signals should be terminated as appropriate. For asynchronous mode, the risk of invalid signals is moderately low due to the nature of the protocol. For synchronous mode, the MEMCLK signal should not contain any overshoot/undershoot.

8. Board Bring-up Considerations

Several additions to a PC board can simplify bring-up of the VSI402ZX. First, a connection to the JTAG interface allows a debugger to operate with the VSI402ZX. Exact pin-out depends on whether the SDK tools are used or Green Hills. The EB402 manual shows both connections. The application note: "Interfacing the VSI402ZX with the Macraigor Raven or Corelis JTAG Interface" provides details on this interface.

Second, being able to boot from either Internal ROM or External Memory can be useful. If the system normally boots from external memory, adding either a jumper or pull-up/pull-down resistor pair to the IBOOT pin allows ROM to be used instead. This can help with initial debugging.

Adding GPIO connections to test points can help when debugging. Writing to a GPIO provides a signal which can be observed with test equipment to confirm proper operation at some point. This becomes especially helpful in real-time systems when the input signals are non-periodic.