Parallel FPGA Programming over Backplane Chassis

Document Revision: 1.25

John Lockwood, Tom McLaughlin, Tom Chaney, Yuhua Chen,
Fred Rosenberger, Alex Chandra, Jon Turner

WUCS-TM-00-11

June 12, 2000

Department of Computer Science
Applied Research Lab
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130

Abstract

For systems with a large number of FPGAs, where a design is instantiated across multiple FPGAs in a chassis, an efficient mechanism of programming the FPGA devices is needed. The mechanism described herein allows multiple FPGAs to be programmed across a backplane. Only a single configuration PROM is required to store the configuration for the multiple instances of the design. When the system boots, all FPGAs are programmed in parallel. This design is applicable to any system which contains a multiple board system which has instances of identical FPGA implementations distributed across the boards. Signal integrity of signals is considered.

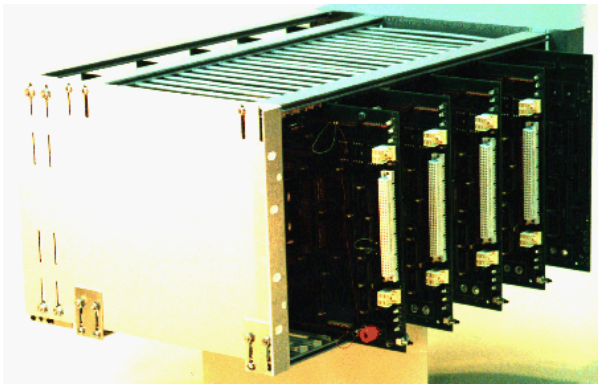

Figure 1: Backplane with multiple FPGA instances

1 Overview

Traditional mechanisms of programming multiple FPGAs are not well suited for systems that contain multiple FPGA devices distributed across a backplane. Device chains require large programming times since the concatenation of the programming bits are serially streamed through each of the devices. The use of duplicate identical PROMs is also undesirable, as it introduces the opportunity for inconsistency in the system and increase time to modify the design. Brute force Hard-wiring of the bidirectional *Init* and *Done* signals is non-scalable due to signal integrity and rise-time delays.

The mechanism described herein allows multiple FPGAs to be programmed across a backplane. Only a single PROM is required to store the configuration for the multiple instances of the design. When the system boots, all FPGAs are programmed in parallel. This design is applicable to any system which contains a multiple board system which has instances of identical FPGA implementations distributed across the boards. Signal integrity of signals is considered. A sample system is shown in Figure 1.

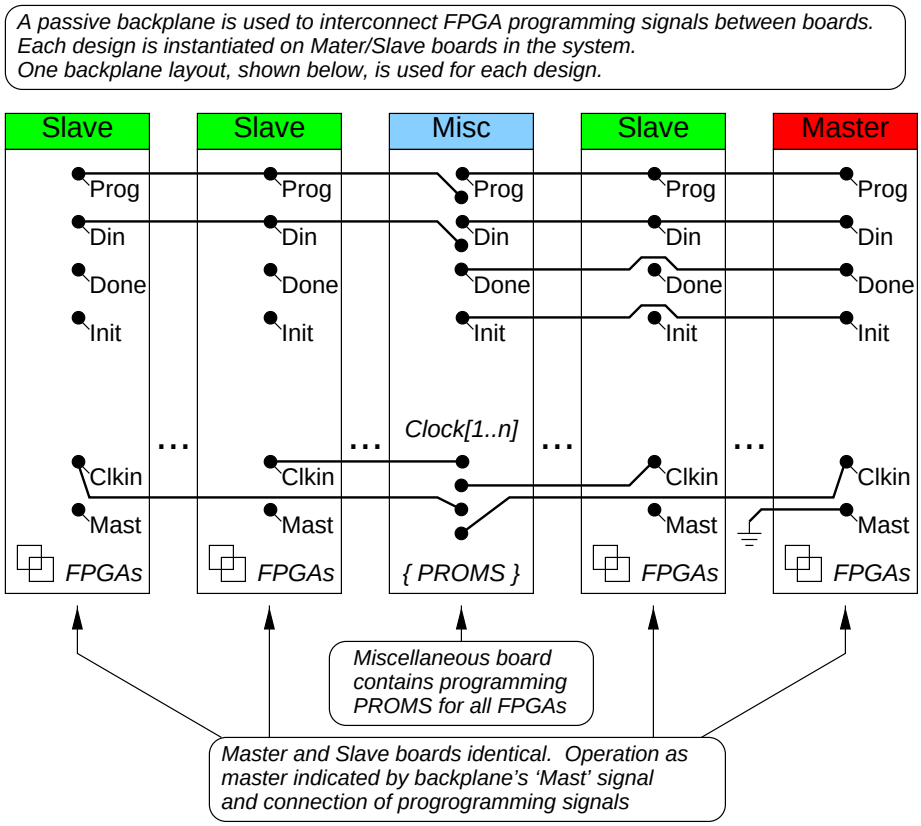

Figure 2: Backplane Configuration

2 Backplane Configuration

A diagram of the passive backplane used to interconnect the boards in the system is shown in Figure 2. The backplane has three types of slots: Miscellaneous *(Misc)*, *Master*, and *Slave*. The *(Misc)* board attaches to the slot in the center of the chassis. It contains the PROMs to program the FPGAs in the system. *Master* and *Slave* boards attach to the remaining slots in the system.

2.1 Distribution of the shared signals

The backplane distributes the program *(Prog)* and data *(Din)* signal from the *Misc* board to the *Master* and *Slave* boards using a point to multi-point line. Separately-driven signals are used to drive the signal on the left and right side of the backplane.

2.2 Distribution of signals to the Master

Each system has one *master* slot and zero or more *slave* slots. When the first board is inserted, it should be placed in the master slot. The boards that attach to the *master* and *slave* slots are identical. The slots, however, are wired slightly differently.

Firstly, the *master* slot is wired to provide a point-to-point interconnection of *Done* and *Init* signals to the *Misc* board. These signals are not connected on the *slave* boards. The *Master* slot also has the *Mast*

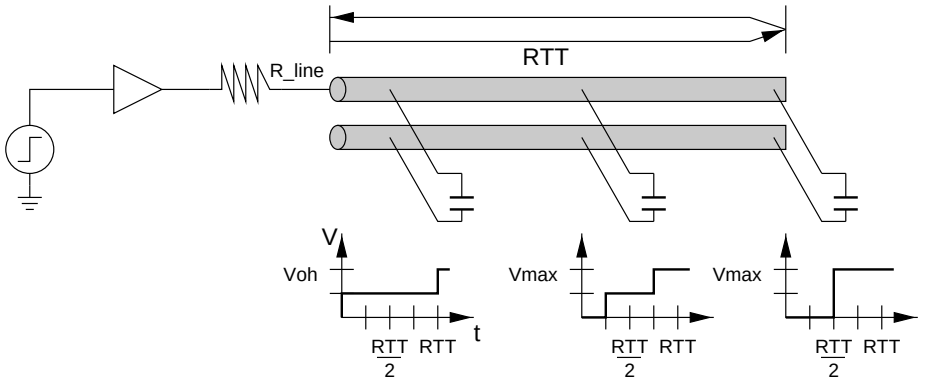

Figure 3: Signal reflections on source-terminated backplane lines

signal grounded so that logic on the board can identify that it is a master. That signal is not connected on the *slave* boards.

2.3 Distribution of clock signals

To provide glitchless operation of the programming clock, the *Misc* board has point-to-point path to each *master* or *slave* in the system. Clock-driver chips on the *Misc* board separately drive drive a signal to each clock trace of the backplane.

2.4 Backplane Signal Drivers

The I/O signals are all LVCMOS (2.5V), except where noted. All signals driving the backplane, as well as the longer traces that appear on the same board, are source-terminated.

As shown in Figure 3, this configuration allows multipoint data signals to be valid within one RTT and point-to-point clock signals to make a single transition to full voltage at RTT/2, so long as the source termination impedance matches the distributed circuit load.

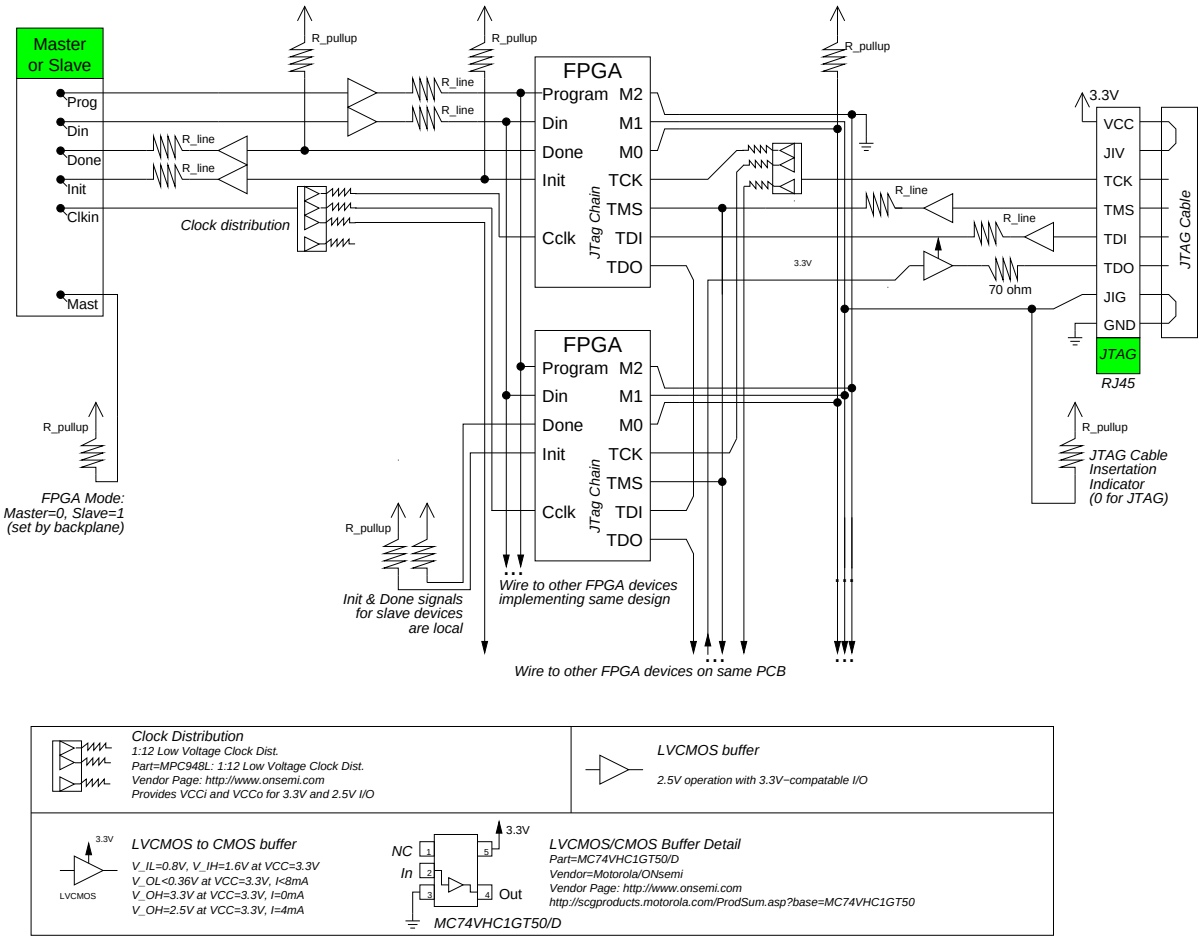

Figure 4: Master/slave Board Configuration

3 Master/Slave Board

The detail of the *master/slave*board is shown in Figure 4. This board contains the the FPGA(s) for one or more instances of a design. The slot connector on the lefthand side of the diagram matches the connector on the backplane. The JTAG connector on the right enable board-level testing.

3.1 Program and Data

The incoming *Prog* and *Din* signals are re-buffered on the *master/slave* board before they are fed to each FPGA device. The source-terminated trace should start at the first FPGA and be routed as shown on the diagram to preserve transmission line characteristics.

3.2 Initialize and Done

The *Done* and *Init* signals for all devices on the board have pullups, since they are driven by open-drain outputs on the FPGA. The signals from the first device are driven to the backplane, while signals from the rest of the FPGAs remain local to the device.

3.3 Component Selection

Specific parts and components that can be used to implement the circuit are shown at the bottom of Figure 4. Care must be taken for the buffering of signals between logic families. Note, for example, how the 3.3V signals on the JTAG interface drive the 2.5V signals on the FPGA. Since the FPGA I/O pads are 5.5V tolerant, it is acceptable to overdrive their input voltages.

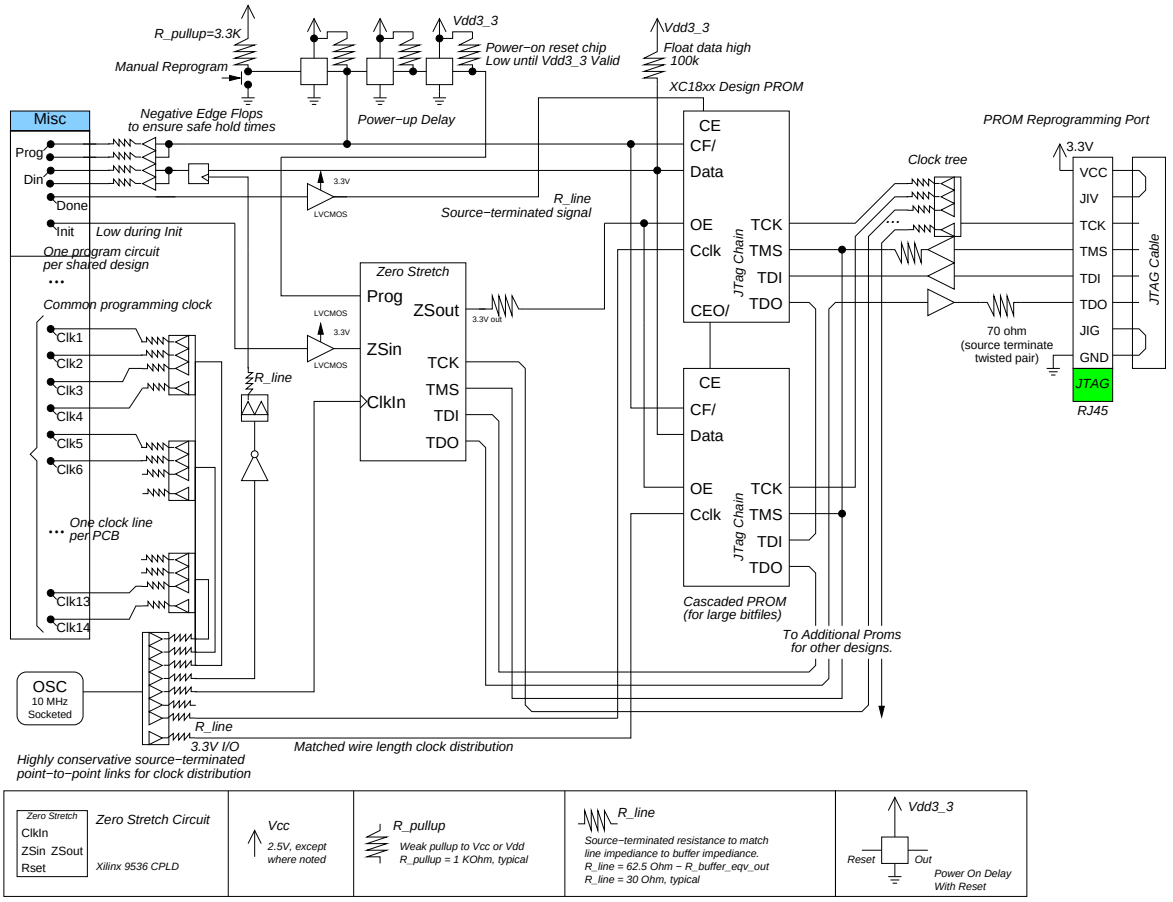

Figure 5: MISC Board Configuration

4 MISC Board Configuration

The Miscellaneous board is shown in Figure 5. The connector on the left matches the connector on the backplane. The *Misc* board contains the PROMs holding FPGA configuration data, a CPLD to program the FPGAs, and a JTAG interface. It distributes programming signals and clocks to the backplane.

4.1 Program and Data signals

Both *Prog* and *Din* are source-terminated signals distributed across the backplane. The *Prog* backplane signal is generated either by the XC18xx PROM *CF/* or as a result of manually pressing the reprogram button. The *CF/* signal is generated by the PROM when it detects the initial power-on state or when an explicit reprogramming command arrives via the JTAG interface.

The *Data* signal from the PROM feeds the *Din* signal on the backplane. For large FPGA devices (like the XCV1000e) that require multiple PROMs to store the configuration, the *CEO/* and *CE* signals are cascaded between PROMs. For the cascaded PROM, the *Data* signal is tri-stated by the *CE* signal. The Cclk is shared among PROMs.

4.2 Clocks

The programming clock is generated by an external crystal oscillator on the MISC board. A socketed device operating at 10 MHz is planned, though operation at frequencies closer to RTT should be possible. As noted by Peter Alfke, Xilinx Applications engineer, avoiding the master mode reduces the configuration time because the inherent timing inaccuracy of the on-chip clock.

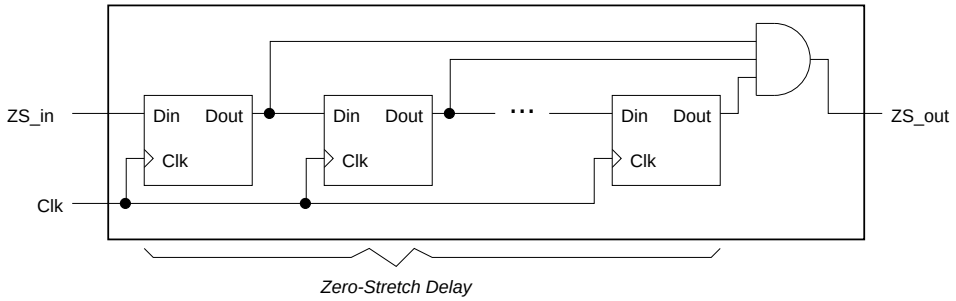

Figure 6: Zero-Stretch Circuit

4.3 Zero Stretch Circuit

The zero-stretch circuit delays a signal incident at *ZSin* to the output signal of *ZSout*. The circuit can be implemented as a simple chain of Flip/Flops followed by a logic gate to detect a sequence of zeros, as shown in Figure 6.

To reduce the number of Flip/Flops, it is preferable to implement the design of the zero-stretch circuit in a CPLD as a counter. A device with n F/Fs would allows a delay of up to $M = 2^n$.

The delay is chosen to be, at minimum, the worst-case variation between the time that the master FPGA is ready to program and the time that all other FPGAs in the system are ready to program.

This time can be estimated as the sum of the 1V rise time of the power supply plus the worst-case time for the chip to come out of the power-on reset state. The result of the circuit is used to delay programming of the parallel FPGAs until every device in the system is ready.

A device like the Xilinx XC9536 provides sufficient I/O and Flip/Flops to implement the zero stretch circuit.

4.3.1 Reference Design for Zero-Stretch Circuit using CPLD

A complete reference implementation of a zero-stretch circuit is illustrated in Figure 7. In this circuit, the CPLD provides a sufficient delay between the time that the one sampled FPGA indicates that it is ready to be programmed, and the time that the rest of the FPGAs should be expected to be ready.

The *Zero Stretch* CPLD generates *ZSout* such that data from the PROM is read no sooner than time it takes for the sampled FPGA to become ready for programming and the worst-case time that might be required for the any other FPGA in the of the same design to become ready for programming.

A timing diagram showing the signals for the PROMs is shown at the bottom of Figure 7. The *ZSinFF* signal is simply a flopped-version of the ZSin signal. An internal counter on the *Zero Stretch* circuit is used to count the number of pulses from the rising edge of the *ZSinFF* signal. The count is reset when *ZSinFF=0*. The count is incremented for each pulse once *ZSinFF=1*. Once the count reaches N, it holds the value and sets *ZSout=1*.

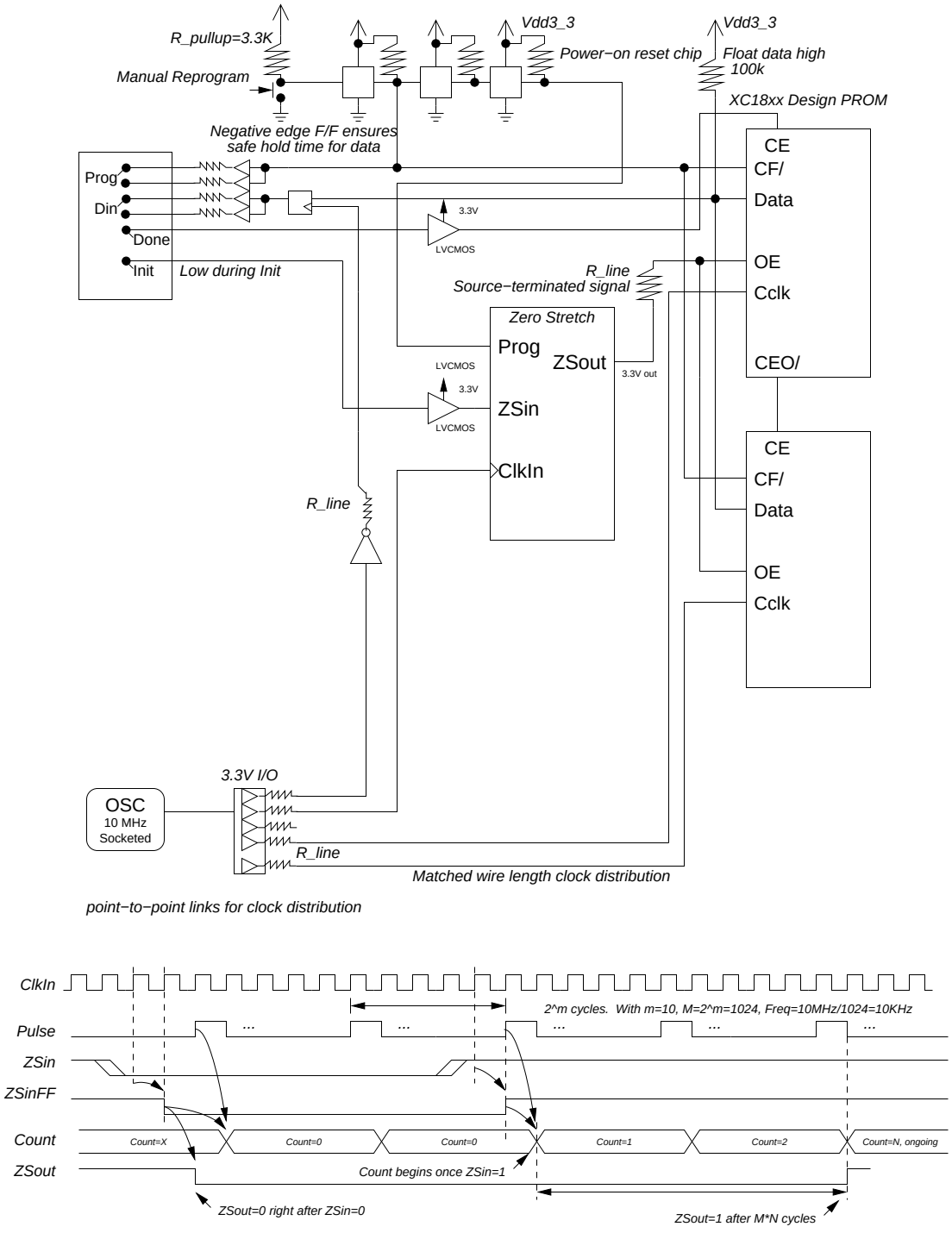

Figure 7: Zero Stretch Timing

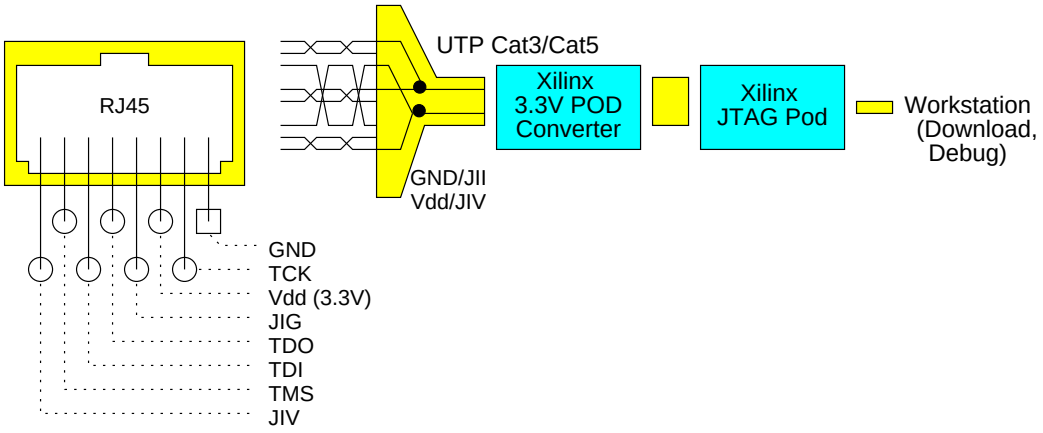

Figure 8: JTAG Connector

5 JTAG

Every board in the system contains a JTAG connector. On the *Misc* board, this connector is used to reprogram the PROMs. On the *Master/Slave* boards, JTAG is available for board-level testing. These connectors appear on the righthand side of Figure 4 and Figure 5. The signals for *TCK*, *TMS*, *TDI*, and *TDO* signals are routed as shown. A Clock tree device is used to distribute the JTAG clock.

5.1 JTAG Connector

A RJ45 8-pin connector appears on the faceplate of the board to allow JTAG board-level reprogramming or debugging. When connected, an extra signal called the *JTAG Cable Insertion Indicator* is grounded to indicate test mode. All other signals on the connector are standard. A pinout of the connector is shown in Figure 8.

5.2 Board-Level FPGA Testing

The JTAG connector on the *master/slave* board enables programming and debugging of the FPGAs with or without connection to the backplane.

When the cable is inserted, the The mode bits for each FPGA must be set to boundary scan mode. The FPGAs operate in slave mode ($M_2, M_1, M_0 = 011$) by default, or in boundary scan (JTAG) mode ($M_2, M_1, M_0 = 001$) when the external cable is connected. Both modes utilize preconfiguration pullups to avoid floating inputs to CMOS gates during the interval the system is programming the FPGAs.