XC9500 Remote Field Upgrade

XAPP 102 January 13, 1998 (Version 1.0) Application Note

Summary

This application note describes the concept and design of a remote field upgrade subsystem for an in-system programmable XC9500 CPLD. The description of the subsystem is given along with guidelines that should help with variations on it. Additional VHDL files are available for direct use of this design.

Specifically, the VHDL files include a complete IRDA receiver design fitting into an XC95108 CPLD.

Xilinx Family

XC9500

Key Terms:

Throughout this application note, frequent reference will be made to XAPP058 and VHDL code for the XC95108. These files along with others mentioned are obtainable from the CPLD Application Notes section of the Xilinx World Wide Website (http://www.xilinx.com). Also, several key terms will be used frequently as follows:

EZTag - an early Xilinx ISP download software module

JTAG - Joint Test Action Group; the developers of the IEEE 1149.1 testing standard

JEDEC - a committee that developed programming standards; a bit file in text form called a JEDEC file

XChecker - one version of a Xilinx serial download cable

SVF - Serial Vector Format, an industry standard data format

XSVF - modified SVF with some data compression to reduce code storage requirements

Introduction

In system field upgrades to an XC9500 CPLD are traditionally done by field engineers using a portable PC. The PC will have either the EZTag or JTAG Programmer Software, an XChecker or JTAG Parallel Cable with flying wires to connect to the target application, and an updated JEDEC programming file. Once all the cables, software, and files are in place, the engineer begins the field upgrade. This application note goes one step further in not requiring the field engineer to be present where the upgrade occurs. It will require advanced planning and building circuitry into the embedded system in advance, but that should be expected.

As an example of making the upgrade when the system is distant from the field engineer, the design presented combines existing embedded download technology with an infrared transceiver to introduce the remote nature of the process. Clearly, the idea can be extended to other media such as telephone/modem links, cables, laser or radio. Many off the shelf communication devices are available today. From another point of view, “remote” is relative. Several companies have expressed a desire to simply isolate their upgrade engineer from the end system, and in that sense, a simple cable to the outside world via a connector or a floppy disk interface would also qualify. In this last situation, customer service personnel have noted the benefit of making the upgrade without “opening the box.”

Design Description

This application note, coupled with the existing Xilinx Application Note XC9500 In-System Programming Using an Embedded Microcontroller, XAPP058, uses a Siemens Infrared Data Transceiver (IRM3105), to demonstrate the concept of a remote field upgrade.

The Siemens infrared transceiver receives updated XC9500 programming information in the form of an XSVF (Xilinx Serial Vector Format File) file wrapped in RS232 format (8n1). Figure 1 shows a block diagram of the XC9500 Remote Field Upgrade design. The XC95108 CPLD contains the functionality described in the VHDL code. The VHDL design contains a UART receiver (entity UART), timing interface control engine (entity TIMING), bus controller (entity BUS_CNTL), and top level connectivity file (entity IRDNLD).

At the beginning, a CPLD JEDEC design file must first be transformed into a corresponding XSVF file. Details of this are provided in XAPP058. Then, that file must be driven by software through a serial port connected to the Siemens IR data transceiver. The receiver end of the transceiver is attached to the XC95108 where the bit-stream will be intercepted by the UART receiver and loaded into the XSVF RAM. This is done with a two step protocol process. First, the UART captures each data byte, then passes it to the timing interface control engine which manages the data arrangement in the XSVF RAM memory. Once the data transfer is complete, the interface control engine flags the

embedded microcontroller, indicating that the updated XSVF data is ready in XSVF RAM. The microcontroller will drive the data down the JTAG port which is attached to the ISP parts to be upgraded. In Figure 1, the EPROM holds the 8051 code which performs the embedded download operation detailed in XAPP058.

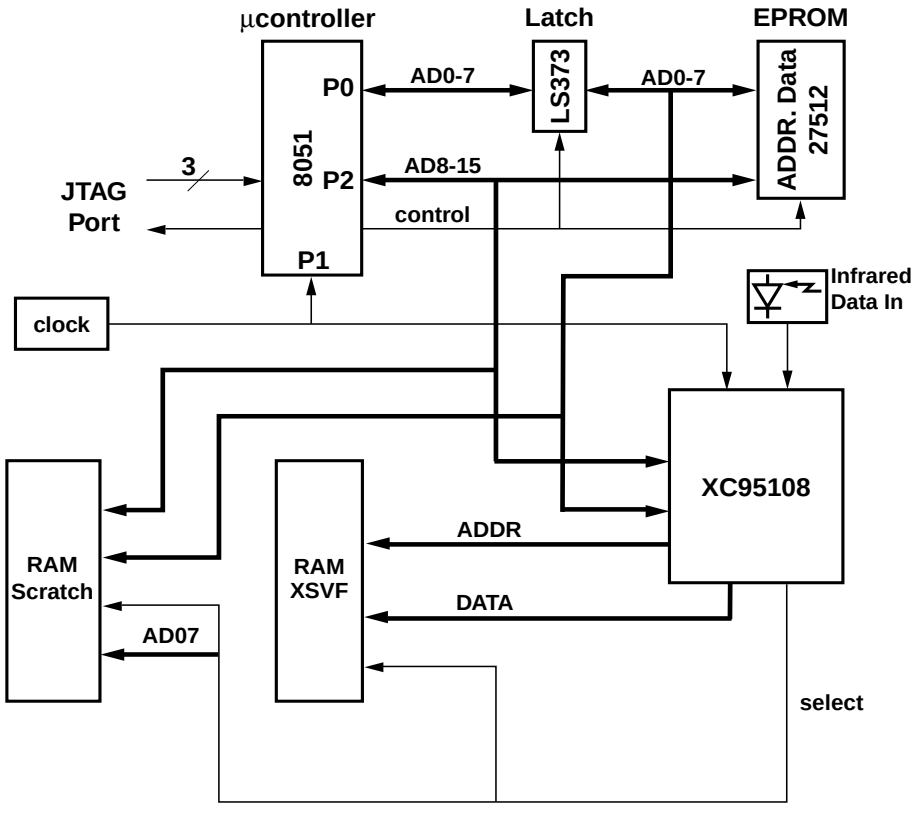

Figure 1: Block Diagram

UART Design

The UART design contains 4 states: MARK, DATA, STOP_BIT, and ERR, and is designed to parse the XSVF file, stripping out the start and stop bits, while registering the 8 data bits. The MARK state is essentially the rest state once the XSVF data transfer is complete, but also serves to verify the start bit in an RS232 data sequence. A start bit is verified by 8 counts while input IR_IN is "low". Once the start bit is verified, the state machine moves to the DATA state. A data bit is registered into an 8 bit register containing an "offset" that is incremented by one each time a bit is registered. Data bits are registered after 16 counts. Once the 8 data bits are registered, the state machine moves to the STOP_BIT state. In the STOP_BIT state, the counter counts to 16 and verifies that IR_IN is "low". If IR_IN is "low", the state machine will advance to the MARK state when IR_IN transitions to a "high". If IR_IN is "high" after 16 counts, the state machine will enter the ERR state to signal that a transmit error has occurred. While in the STOP_BIT state, the UART flags the timing interface control engine to indicate that data is ready to be written into XSVF RAM.

Timing Interface Control Engine

The timing interface control engine is the heart of the remote field upgrade design. It contains 4 states: HIGH, LOW, DOWNLOAD, and ISP. The first two bytes of the XSVF file represent the number of XSVF file bytes to be transferred. States HIGH, and LOW, register the high, and low bytes, respectively, into signal "length_count". While in the DOWNLOAD state, the "length_count" is decremented each time a byte is written into XSVF RAM. The timing interface control engine also increments the internal address counter, "add_cntr", and controls the "write" signal allowing data to be written into XSVF RAM. Once the entire XSVF file has been written into XSVF RAM, the timing interface control engine enters the ISP state. The ISP state interrupts the embedded processor, which eventually vectors off to it's interrupt service routine to reconfigure an in-

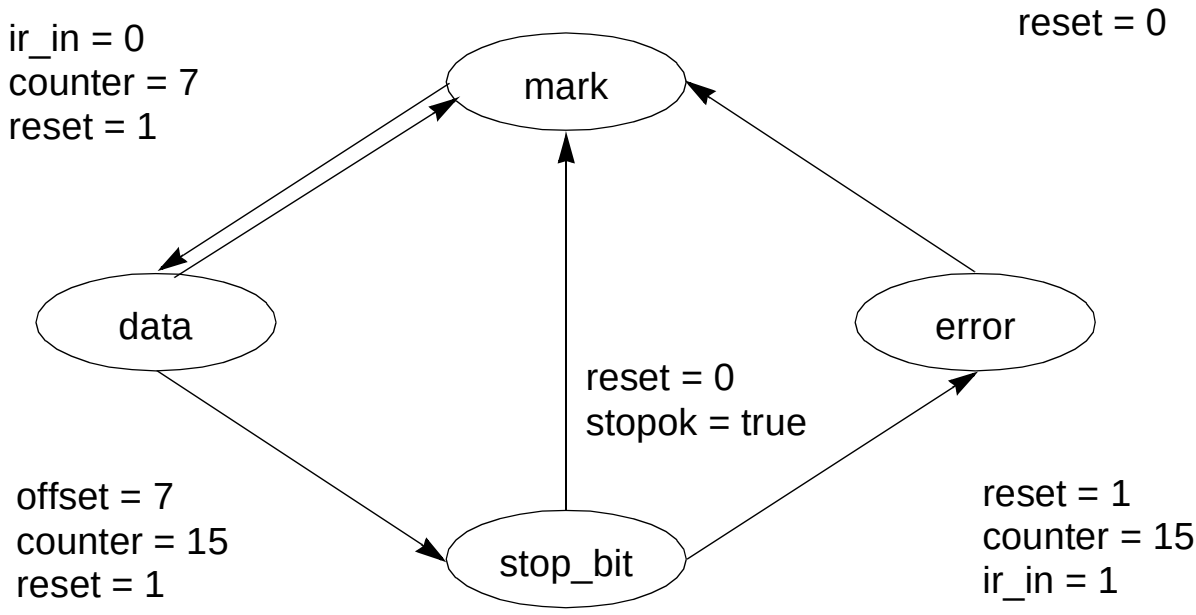

Figure 2: UART Receiver State Machine

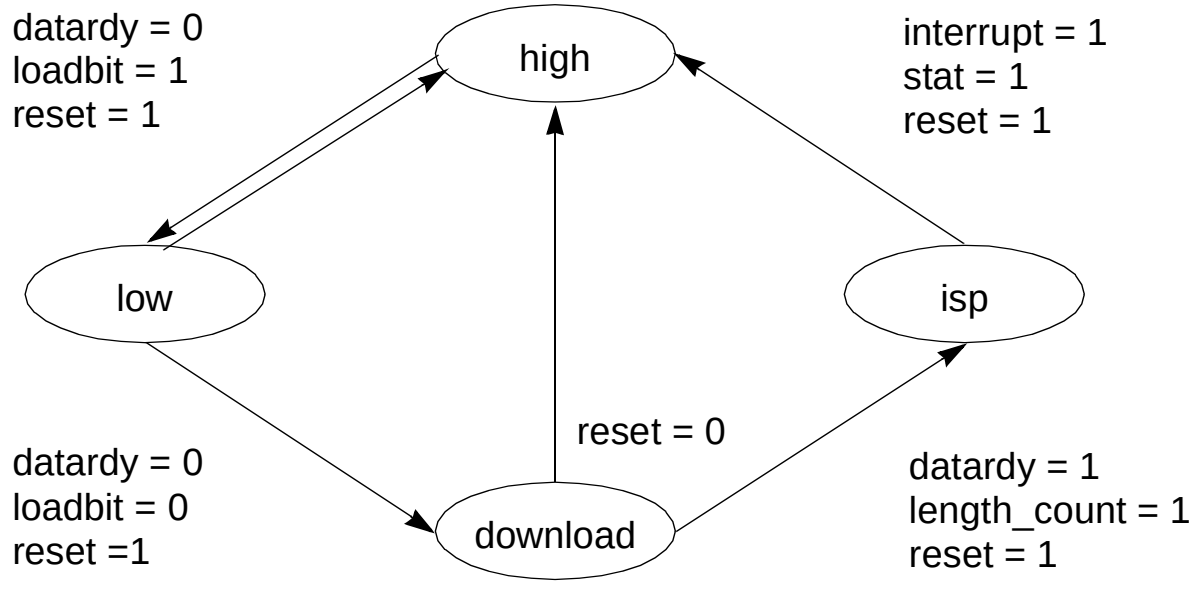

Figure 3: Timing Interface Controller State Diagram

system XC9500 CPLD with the updated information in XSVF RAM.

The timing interface control engine works in concert with the bus controller. The bus controller monitors the timing interface control engine's state bits, and passes control of the XSVF data bus accordingly. The DOWNLOAD state allows the XC95108 to retain control of the XSVF data bus while keeping the processor data bus in high impedance. The ISP state passes control of the XSVF data bus to the processor when the appropriate conditions are met (i.e., PSE = 1, UC_ADD(15) = 1, and READN = 0). The remain-

ing states, HIGH and LOW, keep the XSVF data bus, and processor data bus in high impedance.

In the VHDL design, the top level connectivity file (IRDNLD) logically connects the VHDL design using named-order association. The state bits for the UART, and timing interface control engine are defined in a package called "datatype". See the comments in the VHDL code.

Programming In-System XC9500 CPLDs via a Remote Source

Serial Vector Format (SVF) is a syntax specification for describing high level IEEE 1149.1 (JTAG) bus operations. Xilinx Serial Vector Format (XSVF) is a compressed, binary version of the SVF file designed specifically for embedded applications. XC9500 CPLDs use the IEEE 1149.1 Boundary Scan Standard for in-system programming (ISP). Application note XAPP058 describes how to create both the SVF, and XSVF file. This application note extends the XSVF file with a Perl script to calculate the number of XSVF file bytes . Once the binary XSVF file is complete with number of bytes, the reader should use the binary to Intel Hex translator in Appendix D (XAPP058) to create an Intel Hex file suitable for embedded applications. The Intel Hex converted XSVF file is then transferred remotely to the embedded application described herein, and thereby upgrades an in-system XC9500 CPLD. The remote system should use a simple UART, coupled with the Siemens Infrared Data Transceiver to transfer the XSVF file.

Additional Hardware Details

As shown in Figure 1, this design requires only an 8051 microcontroller, address latch, XC9500 CPLD to decode and upgrade the XSVF RAM, and enough EPROM or non-volatile memory to contain the embedded C code detailed in application note, XAPP058.

The VHDL allows the XC95108 CPLD to operate in the background, receiving infrared data and updating the XSVF RAM, while the embedded processor works in the foreground on other bus operations. In Figure 1, the 8051 multiplexes it's port 0 for both data and address bus operations. The ALE signal causes the 74x373 to latch the lower order address, and the higher order address is output on port 2. The 8051's port 0 then floats, allowing the selected memory to drive the data inputs. The !PSEN signal goes low to activate an 8051 program read operation from the EPROM, or the SEL_RAM signal from the XC95108 CPLD goes “low” to activate a memory read from the scratch pad RAM. The EPROM contains the embedded C code detailed in XAPP058. The clock is a TTL type crystal rated at 16MHz which provides global clocking in the XC95108, and the 8051 microcontroller. The 16MHz crystal is also 16 times the infrared data transfer rate.

Modifying the Design

This application note can be used as-is to reconfigure any in-system XC9500 CPLD. If your system contains multiple CPLDs, or you are doing more then the typical ERASE/PROGRAM operations (i.e. ERASE/PROGRAM/VERIFY, or BYPASS), you may need to modify the VHDL code to include 4 bytes for the "length_count", and replace the existing XSVF RAM with a larger one. Check the size of the XSVF file generated by the M1 JTAG Programmer first. If the file is larger than 65,536 bytes, you will need to modify the VHDL code as described above. Also, you must modify the XSVF sizing utility (Perl script) with the following:

```
$ss=pack("I", $size);
```

The original code contains an "S" to signify SHORT INTEGER. Replacing the "S" with an "I" yields a 4 byte integer. Finally, add one address line to the VHDL code. In other words,

```
XSVF_ADD: standard_logic_vector(15 downto 0).
```

Planning for Field Upgrade

One additional address line may or may not cause problems. If the designer didn't leave room for growth and packed the design to 100%, the additional address line can cause problems. Some designers weigh cost and leaving room for growth as two separate entities. Printed circuit rework can be costly, and typically results from a hasty design change that expands the design.

Pin-locking refers to allowing the CPLD software to fit a design based on an algorithm that tends to spread equations throughout the CPLD. Once the design is fit, future design changes can be made without compromising the original pin-out. The XC9500 CPLD has a very robust pin-locking architecture, designed for ISP. However, it's still important to recognize that if a design is likely to grow, it's best not to force the software to override the pin-locking algorithm, and pack the design. Some techniques key to leaving room for growth are:

1. Don't pack a design more than 85%. Figure 4 below shows a partial summary of the fitting results for the VHDL code in this application note

   Notice that the Macrocells Used is 79%, or 86 out of 108. Also notice that out of 69 total user I/O, 9 are left over. This leaves a cushion for future changes.

2. Provide traces on your circuit board that physically connect I/O to future applications. If it doesn't alter the operation, tie the trace to a known logic level. Xilinx recommends not using unused I/O to board ground, and use the "Create Programmable Grounds" option in the Xilinx CPLD Software.

3. If specific I/O pins need to be reserved, provide “dummy” functions such as:

   ```
   OUT = IN
   ```

Or, if using a schematic capture package, be sure to attach the appropriate LOC constraint to the output pad and input pads, respectively. See the Xilinx on-line help system for more information on the LOC keyword.

4. Allow the CPLD software to perform an “auto device select” when initially fitting your design. This may increase run time slightly, but the software will automatically select the best device for pin-locking.

XACT: version M1.3.7 Xilinx Inc

Fitter Report

Design Name: irdnld

Fitting Status: Successful

********** **Resource Summary** **********

Design Name	Device Used	Macrocells Used	Product Terms Used	Pins Used
irdnid	XC95108-10-PC84	86/108 (79%)	362/540 (67%)	60/69 (86%)

PIN RESOURCES:

Signal Type	Required	Mapped	Pin Type	Used	Remaining
Input	:20	20	I/O	:58	5
Output	:30	30	GCK/IO	:1	2
Bidirectional	:8	8	GTS/IO	:0	2
GCK	:1	1	GSR/IO	:1	0
GTS	:0	0			
GSR	:1	1			
Total	60	60			

Figure 4: Partial Resource Summary

Summary

The XC9500 Remote Field Upgrade design was verified using the VHDL test bench provided in Appendix B, with the Synopsys VHDL Debugger Software. The design was translated using the Synopsys Design Compiler, v3.4b. It was fit to an XC95108-10PC84 CPLD with the M1.3.7 XACTcpld Software using default settings, and no time-specs. It was assembled on a generic prototyping board. Figure 5 shows a photo of the Remote Field Upgrade prototype described in this application note.

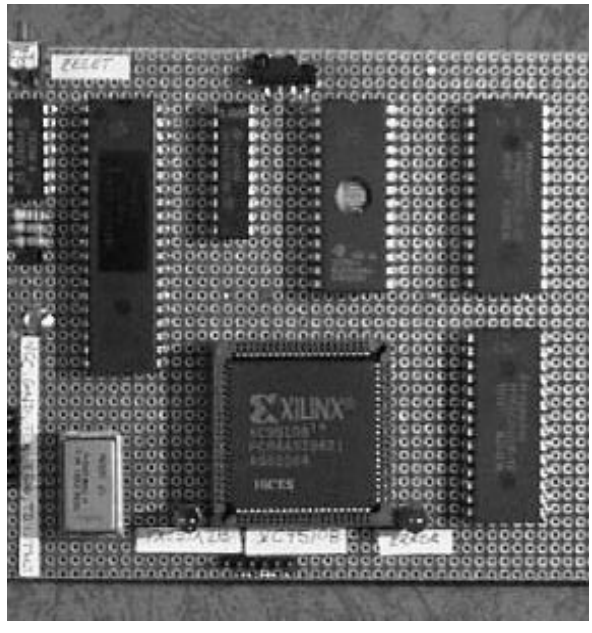

Figure 5: Field Upgrade Prototype