The tm4mon Program

Dave Galloway

Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto

March 2005

Introduction

The `tm4mon` program is a daemon process that acts as a front end for the TM-4 hardware. It runs on the workstation that is connected to the TM-4 (currently crunch.eecg, February 2005).

Tm4mon opens two network TCP stream sockets, and sits waiting for connections to those sockets. On one of the sockets (the housekeeping socket) tm4mon will accept requests to change or report on the state of the TM-4. On the other socket (the ports socket) tm4mon will accept data to be transferred to or from a user's design inside the TM-4. See *The TM-4 Ports Package* for a description of how the ports socket is used.

A second program called `tm4` is usually used to talk to the housekeeping socket of tm4mon. The tm4 program can be run on any host, and will transfer one or more requests to tm4mon over the network. A single request can be given as the arguments to tm4. For example,

```
tm4 getdesigndir
```

will ask tm4mon for the name of the directory containing the design most recently loaded into the TM-4.

Multiple requests can be passed to the tm4 program on the standard input by giving it a "-" argument:

```
tm4 -
getdesigndir
getdesigndate
^D
```

Starting Tm4mon

To start the tm4mon daemon, sign on to the computer that has the TM-4 board connected to it, and type:

```
cd /
tm4mon &
```

Tm4mon Housekeeping Socket Requests

This is a list of the requests that tm4mon understands on the housekeeping socket:

acquire P N	Ask for the TM-4 at priority P for N seconds. Returns 0 if successful, or the current priority and the number of seconds left in the current reservation if not.
debug N	Set the debug level to N. Debugging output will go to tm4mon's standard output.
disable_video	Disables the video output on FPGA0
disable_video_in_a	Disables video input A on FPGA0
disable_video_in_b	Disables video input B on FPGA0
EXIT	Causes tm4mon to exit.
enable_video	Enables the video output on FPGA0
enable_video_in_a	Enables video input A on FPGA0
enable_video_in_b	Enables video input B on FPGA0
getarch	return TM-4 as a string
get_clk N	return the speed of clock N
getdesigndate	return a string version of the date and time the current design was loaded into the TM-4
getdesigndir	return the name of the directory containing the current design
getportlistdir	return the name of the directory containing the list of ports used in the current design
getstatus	return a string of 1s and 0s that describe the current status of the board
getstatus1	return the arguments of the last setstatus1 command
getstatus2	return the arguments of the last setstatus2 command
read_jtag_pins	prints a human readable list of the jtag information
program_logic_chips file	program the logic FPGAs from the bits in file
read_reg N	read the value of housekeeping register N
read_temp	read the temperature of the board, along with the temperature of each FPGA

release	release the TM-4 so others can acquire it
reset	reset the board
setdesigndir dirname	set the name of the directory containing the design
setportlistdir dirname	set the name of the directory containing the port lists for the current design
set_clk N M	set the frequency of clock N to M Hz; clock 0 is tm4_glbclk0
setstatus1 args	set the string returned by getstatus1
setstatus2 args	set the string returned by getstatus2
stop	reset the FPGA chips. Also kill off any ports sockets connected to programs that are talking to the current design.

Tm4mon Ports Socket Format

The ports socket is normally used by the tm4 ports package to send data to and from a user's circuit in the TM-4. Data is sent to the ports socket as a series of packets. Each packet consists of a 32 bit packet header, followed by optional data bytes. Each header specifies a chip and port number where the data is to be transferred. The 32 bit header is sent in network byte order (most significant byte first). The most significant bit is labelled 31 in this table:

Bit Number	Meaning
31	1 for a write (data transferring to circuit); 0 for read
30	1 if the port uses handshaking; 0 if not
29-28	chip number
27-26	top two bits of packet data length
25-18	port number inside the chip
17-12	width of the port in 32-bit words
11-0	bottom 12 bits of packet data length (max length is 16383 bytes)

A header requesting a write must be followed by the number of bytes of data that were specified in the header. A read request header will cause tm4mon to block until the specified number of bytes are received from the design.

Each request (read or write) will be answered by a 32-bit word containing the number of bytes transferred to or from the circuit in the case of a successful transfer, or -1 in the case of an error. If it was a successful read request, the word will then be followed by the number of bytes requested.