Creating and Running Circuits for the TM-3A

Dave Galloway

Department of Electrical and Computer Engineering
University of Toronto

September 2002

Compiling a Circuit for the TM-3A

Circuits for the TM-3 and TM-3A can be prepared in one of two formats: VHDL (.vhd) format, or the subset of Xilinx XNF (.xnf) format that the tmcc compiler produces.

Create one netlist file for each Virtex FPGA that you wish to use. Run either the `tm3vhd` or `tm3xnf` commands, depending on the netlist format you have produced. Both commands take the same set of arguments. The following examples use tm3vhd, but tm3xnf is used in the same way. Run your chosen command like this:

```
tm3vhd file1.vhd file2.vhd file3.vhd file4.vhd
```

This will run the Xilinx CAD software, creating a new directory called `tm3a` containing the programming files for the TM-3A. If you want to use a name other than tm3 for the directory, use the `-o dirname` option.

If you do not wish to use a particular FPGA, replace the corresponding `fileN.vhd` argument with the string `tm3_blank`.

If you are creating a design for the TM-3, use the `-arch TM-3` option.

The TM-3 and TM-3A have four independent clock generators called `tm3_clk_v0`, `tm3_clk_v1`, `tm3_clk_v2`, and `tm3_clk_v3`. Most designs should use `tm3_clk_v0` for their clock. Do not explicitly instantiate a Xilinx global clock buffer (BUFG) for your clock. The `tm3vhd` or `tm3xnf` commands will do this for you. If you are using tmcc and creating XNF files, a simple way of specifying the clock is to include a copy of `tm3clock.xnf` with your circuit. A copy of `tm3clock.xnf` can be found in the directory `~tm3/examples/counter`.

Additional options for tm3vhd and tm3xnf are documented in the "Advanced Compilation Options" section of this document.

Communicating with your Circuit

The TM-3 ports package can be used to communicate with your circuit. See *The TM-3 Ports Package* for details. If you want to use the ports package with one or more of the fpgas, create a port description file for each fpga *before running tm3xnf or tm3vhd*, giving them the same names as the corresponding netlists, but with a .ports extension.

Communicating with External Hardware

Each TM-3A system has eight 40 pin ribbon cable connectors which can be connected to external hardware. Each I/O connector has 32 signal pins, and 8 grounds. To use the signals in your circuit, you must create a file called `tm3.boilerplate` in your design directory before running tm3xnf. To do this, run the `tm3boilerplate` program:

```
tm3boilerplate > tm3.boilerplate
```

The tm3.boilerplate file describes most of the signals in the TM-3A system. An I/O connector description looks like this:

```
tm3_io_1_v0 inout 1 0
tm3_io_1_v1 inout 1 1
tm3_io_1_v2 inout 1 2
   ...
tm3_io_1_v62 inout 1 62
tm3_io_1_v63 inout 1 63
<end>
```

Use a text editor to change the names of the signals in the file to match the names used by your circuit.

The second item on each line specifies the direction of the signal on the connector. Change this to an `in` for an output from the TM-3A or an `out` for an input to the TM-3A. NOTE: THIS LOOKS BACKWARDS, but honest, that's how the software works.

The third item on the line is a number from 0 to 3 that specifies which Virtex FPGA this pin is connected to.

The fourth item on the line is a number, from 0 to 63. The mapping between this number and the physical connector pin numbers can be found on the TM-3 web page, `http://www.eecg.utoronto.ca/˜tm3/` under the I/O section.

Using the TM-3A

In order to use the TM-3 or TM-3A, the `tm3mon` daemon must be running on the workstation that is physically connected to the TM-3A hardware. This daemon is normally started when the machine is booted, and will continue running. If the daemon stops for some reason, you can re-start it by signing on to that machine and typing:

```
tm3mon &
```

If you are not using the default Transmogrifier (currently the TM-3A next to tiger.eecg), you must set the `TM3_SERVER` environment variable to point to the computer that is connected to the Transmogrifier that you want to use:

```
setenv TM3_SERVER varan.eecg
```

The `tm3` command can then be used to communicate with the tm3mon daemon. See *The tm3mon Program* document for details.

You can check the status of the TM-3A remotely by running the `tm3status` program. It will create an X-window on your screen that shows the current status of the machine.

The `tm3get` command will attempt to reserve the machine for a specified number of minutes (default 10). If it succeeds, your name will be shown on everyone's tm3status display. When you are finished, run the `tm3release` command so that others may use the machine.

To download and start a circuit in the TM-3A, run:

```
tm3run [dirname]
```

By default, tm3run will look for the programming files in the tm3a subdirectory. The optional directory name argument will cause it to look in the specified directory.

Checking Circuit Speed

The tm3xnf or tm3vhd script will run the Xilinx timing analyzer in register to register mode, leaving the output for each FPGA in the tm3a/xilinx subdirectory. The `tm3timing` script will display a portion of this output, along with some estimates of your longest chip to chip connections. The -v flag will cause tm3timing to display chip to chip timing estimates for all of the external signals in your design.

To change the clock speed of a circuit, run:

```
tm3 set_clk N MMMMMMMM
```

where N is the clock generator number from 0 to 3, and MMMMMMMM is the new clock frequency in Hertz.

Debugging your Circuit

The `tm3jtag` script will report on the values present on the pins of the Virtex chips in your design. It prints one line for each pin used. Each line contains the signal name, the name of the FPGA, the name of the pin on the FPGA, the value that the FPGA sees on the pin (in:0 for a low value, in:1 for a high), the value that the FPGA is attempting to set the pin to (only valid if the FPGA has programmed this pin as an output), and the state of the output enable for that pin (oe:0 if the pin is being driven by the FPGA, oe:1 otherwise).

The `tm3vars` script will read the output of tm3jtag, and print it in a more readable, condensed format. Each variable is printed on a single line, along with its value. Variables with more than one pin (ie: buses) are printed in hex.

The `tm3step` program is a circuit debugger. It will allow you to stop and start the `tm3_clk_v0` clock and run the clock for a given number of cycles. It will also display the contents of any of the flip-flops in your design.

The tm3step window has a number of buttons. The various "step NNN" buttons will stop the clock and advance it by NNN cycles. The "Capture" button will bring up a second window with two panes. The top pane is a tree of all of the flipflops in your design. If you click on the name of a variable or bus, it will be displayed along with its value in the bottom pane. The values will be refreshed each time the Capture button is pushed.

Using the SRAM

SRAM on the TM-3A

There is one 64-bit wide bank of SRAM connected to each Virtex FPGA in the Transmogrifier. Each bank is independent, with its own data, address and control busses. Each bank contains 256K 64-bit words, or 2 Mbytes.

The memory banks are synchronous, and use `tm3_clk_v0` as a clock signal. Each memory bank is disabled by default, and can be enabled with the following command:

```
tm3 enable_sram N
```

where N is the FPGA number from 0 to 3.

The ram has a 64-bit wide bi-directional data bus named `tm3_sram_data`, an 19-bit wide address named `tm3_sram_addr`, an 8-bit wide write enable control (one for each 8 bit byte) called `tm3_sram_we`, a 2-bit wide output enable control called `tm3_sram_oe` and a 1-bit address stobe control called `tm3_sram_adsp`.

A 64-bit write to the SRAM takes one cycle, and looks like this:

```
/* Write 64 bits */
tm3_sram_we = 0x00;
tm3_sram_oe = 0x3;
tm3_sram_adsp = 0;
tm3_sram_addr = address;
tm3_sram_data = data;
```

A 64-bit read from the SRAM takes 2 cycles, and looks like this:

```
/* Read 64 bits */
tm3_sram_we = 0xFF;
tm3_sram_oe = 0x1;
tm3_sram_adsp = 0;
tm3_sram_addr = address;
/* CLOCK_EDGE_GOES_HERE */
data = tm3_sram_data;
```

To disable the SRAM, set the control signals like this:

```
/* Disable the RAM */
tm3_sram_adsp = 1;
tm3_sram_we = 0xff;
tm3_sram_oe = 0x3;
```

Advanced Compilation Options

The tm3vhd and tm3xnf commands also take the following options:

`-t0 NNN` tells the Xilinx software to try to optimize the design until it will operate with a clock period of NNN ns.

`-ol N` tells the Xilinx `par` place and route software to operate at a different effort level. The `-ol 5` option will produce the best results, but will use more CPU time.

`-no_trce` prevents the Xilinx timing analyzer from running. It is usually fast, but can use a lot of CPU time on some circuits.

`-no_bufgp` prevents the automatic insertion of a global clock buffer (a BUFGP) between the tm3_clk* signals and your design.

`-no_run_in_tmp` will force the compile to be run in the file system containing the current directory. The Linux version of the Xilinx tools run faster in /tmp, so the scripts normally compile your design in /tmp and copy the results back.

`-chips "N M ..."` will recompile only the designs for the chip numbers that you specify. This option can be used to recompile one or more chips of an already compiled design. You must have compiled the whole design beforehand. You must not modify the number of pins on any of the FPGAs, and may not change the names of any of the pins. For example:

```
tm3vhd chipa.vhd chipb.vhd chipc.vhd chipd.vhd
```
< *test and modify chipa and chipc, but want to leave the other two alone* >
```
tm3vhd -chips "0 2" chipa.vhd chipc.vhd
```

Synplify design constraint files (.sdc files) can be used with your design. If a design.sdc file is present for a chip, it will be copied to the directory where synplify is run. If so, the default timing constraint will not be given to the Xilinx software. It is a "FROM FFS to PADS" constraint of 13 ns, which seems to make chip to chip communications more robust. The -t0 flag will also be ignored in this case.

Implementation Internals

CAD Tools Used

The two main commercial tools used to compile circuits for the TM-3A are the Synplify VHDL compiler from Synplicity, and the Xilinx place and route tools. They are automatically invoked by the tm3vhd and tm3xnf scripts. Look at the `tm3run_synplify` and `tm3compilexilinx` scripts in `~tm3/bin` for details.

Documentation for Synplify can be found at:

```
http://www.synplicity.com/literature/index.html
```

Documentation for the Xilinx tools can be found at:

```
http://toolbox.xilinx.com/docsan/3_1i/data/common/dev/dev.htm
```

The tm3a Directory

The tm3a directory created by tm3xnf or tm3vhd has 3 subdirectories: xilinx, download, and netlist. The download directory contains the programming files for each Virtex chip. The xilinx directory contains one subdirectory for each Virtex FPGA that holds the input and output files from the Xilinx software.

The files in each xilinx/fpgaN subdirectory will include:

File	**Purpose**
virtexN.vhd	automatically generated top level wrapper for this chip
virtexN.ucf	automatically generated pin locations and timing constraints
virtexN.srr	output from the Synplify VHDL compile of virtexN.vhd
xilinx.out	output from the Xilinx tools while compiling the chip
virtexN.twr	output from Xilinx's trce timing analysis for this chip

By default, many of the Xilinx output files are removed. The `-v` flag on tm3vhd will prevent this, and leave all of them in the xilinx/fpgaN directory.

The netlist directory contains the source for the portmux port wrapping circuit needed for the ports package (fpgaN.c), along with the port file for each Virtex FPGA (fpgaN.ports) and the modified netlist. The modified netlist connects the original user's circuit to the port wrapping circuit.