RAMPROM
A serial PROM emulator for the Xilinx Teaching Board

Quentin Stafford-Fraser

February 1992

1 Introduction

Many of the circuits used in the Fairisle project incorporate Xilinx chips. The configuration data for these chips is generally held on a small 8-pin serial PROM, which initialises the Xilinx on power-up. During testing, however, the Xilinx configuration may need to be changed several times, requiring many PROMs to be blown and then discarded.

The *RAMPROM* system allows a Xilinx Teaching Board (XTB)[1], by means of a special cable, to be plugged into the socket normally occupied by a Xilinx serial prom. An emulator configuration is loaded into the XTB. The RAM on the XTB can then be loaded with the data that would normally be blown into the serial prom.

Here are some details of important files:

`rammer.hdl` The HDL for the emulator itself.

`rammerout.xil` The resulting configuration file for the teaching board.

`xidump` A program for sending data to the teaching board. It opens a connection to the specified host and service, and sends the XTB commands given on stdin to the teaching board. The returning codes are checked and then sent to stdout. Xidump is written in perl.

`ramprom` The script which does the setting up and loading of the prom emulator. See below.

`promMakefile` A makefile used by *RAMPROM* .

2 The ramprom script

This creates a suitable file to be downloaded to the XTB and downloads it and the emulator. Here it is:

[1]For more information on the Xilinx Teaching Board, see Steve Temple's documentation elsewhere in this Snapshot.

```
#! /usr/bin/perl
#
# ramprom - 'creates' a prom using a xilinx teaching board
#
# This script:
# Converts .mcs to .xil
# Converts .xil to .dat (replaces M with K & N with W)
# Appends the .dat file to rammerout.xil (the prom emulator)
# Downloads the result to XTB on specified host & port
#
$Syntax= <<end;

RAMPROM - Serial prom emulator for the Xilinx Teaching Board

Syntax:

   ramprom -h host [-s service] [-p port] <mcs_file_without_extension>

The default is -s telnet.

end

$servport = "telnet";

# process switches
while ($_ = $ARGV[0], /^-/ ) {
    shift;
    last if /^--/;
    if (/^-h/) {
$host = $ARGV[0];
shift;
    }
    if (/^-s/) {
$servport =  $ARGV[0];
shift;
    }

    if (/^-p/) {
$servport =  $ARGV[0];
shift;
    }
}

if (!$ARGV[0]) {
    print STDERR $Syntax;
    exit(1);
}

if (!$host) {
    print STDERR "$Syntax\n*** Please add host name using -h option.\n";
    exit(1);
}

# Where is the makefile?
$mf="/usr/groups/fairisle/ramprom/promMakefile";

# Where is xidump?
$xidump="/usr/groups/fairisle/ramprom/xidump";

# Produce needed files
system("make -f $mf CHIP=$ARGV[0]");

# and send result to XTB.
print "Sending to $host...\n";
system("$xidump $host $servport < $ARGV[0].prm");
print "done\n";
```

To use the script, create a `.mcs` version of your PROM data using the standard tools (`makebits` & `makeprom`). Most makefiles will do this for you. Plug the XTB into the socket where the PROM would go. Then, supposing that your output is `freddyout.mcs` and that the teaching board is connected to the telnet port of grosvenor8-r, run `ramprom -h grosvenor8-r freddyout`. That's it.

This will:

1. Create a `.xil` file from the `.mcs` file. A `.xil` file contains the data in a form that the teaching board understands. It starts with an 'M' and then contains the data with each byte preceeded by 'N'. See the XTB documentation for more info. This would normally be used for configuring the Xilinx chip on the teaching board. We don't actually do that, because this is the data for some other Xilinx chip. The XTB itself is configured by `rammerout.xil`, which is the code for the PROM emulator.

2. Create a `.dat` file from the `.xil` file. A `.dat` file is the format that the XTB understands once the PROM emulator is loaded into it. It consists of a `.xil` file with the Ms replaced with Ks and the Ns replaced with Ws.

3. Append the `.dat` file to `rammerout.xil`. We then have a file which consists of both the PROM emulator and the PROM data.

4. Download this into the XTB.

Strictly speaking, there is no need for `rammerout.xil` to be downloaded each time. Once the emulator is there, it stays there, but this is simpler.

RAMPROM can be used for proms which initialise daisy-chained Xilinx chips, provided that the memory on the XTB is sufficient. For simplicity's sake, the current version of the emulator just uses the bottom byte of each memory word for storage. This means that the board will actually need four times as much memory as there is Xilinx data. This has not proved to be a problem so far, but volunteers to fix it are welcomed.

3 The cable

IMPORTANT: Note that *RAMPROM* monitors the Vcc line of the serial PROM. If an uninitialised XTB is connected to a PROM socket which then has power applied, there will be a 5-volt supply to a pin of the unconfigured Xilinx chip on the XTB. This may damage it. Always make sure that the XTB is configured before you power-up your circuit, and that you remove the power from the circuit if the XTB should lose its configuration or be programmed with another one. There should really be a resistor in the line to protect against this.

Here are the connections for a *RAMPROM* cable:

XTB connector J3 pin	Xilinx Signal	Prom pin	PROM signal
6	XTL1	2	promCLK
35	X0	8	promVCC
33	X1	3	promRST_OE
31	X2	4	promCE
29	X3	1	promDATA
39/40	0v	5	Gnd

4 Miscellaneous Notes & Future possibilities

There is also a program called `xiload` which is menu-driven and is possibly more useful than `xidump` for general XTB experiments. It allows individual commands, or files containing commands, to be sent to the XTB and the returned values to be seen. Because of this, the XTB 'W' command is not buffered, and sending `.dat` files takes rather longer than with `xictrl`.

Work is being done to allow more general access to the serial ports on the back of Unix machines. This generally involves running a special daemon under inetd.

Serial PROMs are more complex than many other memories. It should be easy now to emulate various other chips.

The HDL should really be rewritten to use more of a state-machine model, rather than being so clock-based. This version seems to work reliably, but I'm not sure how good it would be at different clock frequencies.

Quentin Stafford-Fraser
February 1992