The X Transputer Configuration Environment

XTrace Version 2.0

Mike Jipping
Hope College
Department of Computer Science
Holland, MI 49423

Introduction

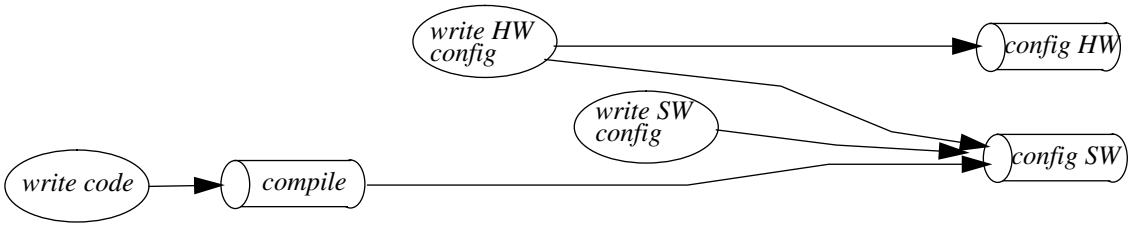
This document explains the "xtrace" program and the way it can be used to provide a graphical, visual method for configuring hardware and software for INMOS Transputer Modules ("TRAMs"). XTrace requires the accessibility of the "check" (or "ispy") program and the INMOS toolset family of programs and is simply a front-end to these.

The user is assumed to have (and be familiar with) the following (in addition to xtrace):

- an IMS B014 VME or IMS B008 Development motherboard
- the check family of programs, version 2.2 or later (this includes check, ftest, mtest, and load, although xtrace does not use load)
- an INMOS toolset: C or Occam
- Unix make facilities
- the X window system

An Overview

In order to run one or more programs on a Transputer network, a programmer must not only deal with programming languages and compilers, but with Transputer configuration tools and languages. There are several parts to the configuration process, and each part uses a different tool -- which uses a different language. The entire process is depicted below:

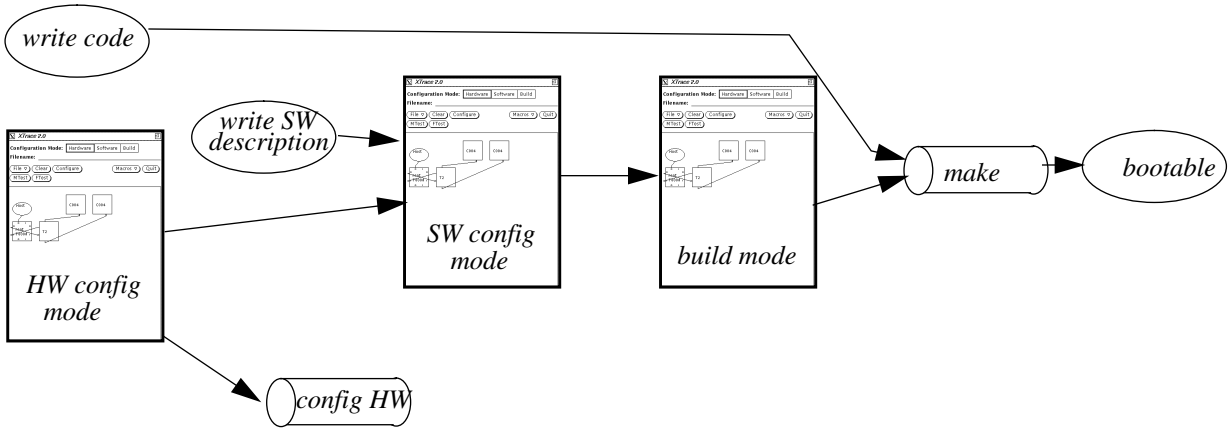


First, the algorithms are developed and coded. Then, the hardware must be configured to fit your problem. Then the software must be mapped to the hardware configuration. And finally, the entire process must be joined together and downloaded onto the Transputer network.

For programmers who are unfamiliar with the whole process, this can be a real morass. And for students who might even be unfamiliar with the programming languages, this process becomes almost impossible.

`XTrace` is designed to help in these configuration problems. In order to use `XTrace`, a programmer needs two items: the set of algorithm modules in a programming language, and a description of the parallel processes in the concurrent system. `XTrace` takes these and produces hardware and software configuration files and a bootable program makefile, and allows the programmer to configure the hardware and generate the bootable. Instead of writing each of these files, the user now specifies using the mouse and drawing.

The new process looks like this:



Invoking `xTrace`

`xtrace` is invoked by the following command line syntax:

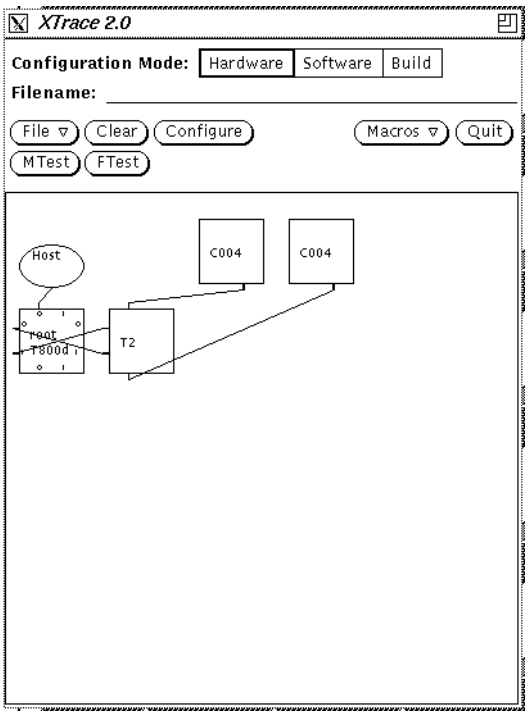
`xtrace <`*`options`*`>`

where <*options*> can be chosen from

`-B <`*`filename`*`>`	use <*filename*> as the build configuration file
`-H <`*`filename`*`>`	use <*filename*> as the hardware configuration file
-S <*filename*>	use <*filename*> as the software configuration file
`-b`	turn on build mode
`-h`	turn on hardware mode (default mode)
`-s`	turn on software mode

Simply typing "`xtrace`" at the command prompt is sufficient, however, to start the program. The default startup options are an empty (i.e., no) filename as a configuration file and to be in hardware mode. The TRAM configuration presented will be a minimum configuration -- a root TRAM. a T2, and two C004s. The configuration is like that shown on the slate of Figure 1.

Figure 1: Opening the "simple" Example



Using `XTrace` for Hardware Configuration

`Check` vs. `XTrace`

To configure a B014 or B008 motherboard, two options are available: use the MMS2 software that accompanies the motherboard device driver from INMOS or the "check" suite of programs. Apparently, the MMS2 software is inferior to the "check" suite when used with boards containing more than one C004 (e.g., a B014), and the "checkout" suite of programs appears more flexible.

The "check" family contains four programs: `check`, `ftest`, `mtest`, and `load`. The `check` program is the program used for configuration. It takes an ASCII configuration file and attempts to configure the TRAMs on a motherboard so that they are linked in the same way as specified. `Check` also generates a configuration file as a report of the current configuration.

The ASCII configuration file generated by `check` is used a "sneaky" way. It reports on the links between the TRAMs on a motherboard, but only uses the C004 link specification when changing the configuration of TRAMs. This C004 specification can be tedious and frustrating, and the `check` configuration is easy to read, but difficult to visualize. Hence, "`xtrace`" is born.

The hardware mode of `xtrace` program provides a visual, window/mouse-based front-end to the `check` family of programs. While still based in the ASCII configuration files of `check`, `xtrace` separates the user from manipulating those configuration files and provides easy, mouse-based writing of configuration files and easy configuration of the TRAMs on a motherboard.

The user is warned, however, that `xtrace` is no substitution for `check`. It is only a front-end for `check` and does not do any actual TRAM configuration on its own. Also, any error checking of the configuration file must be done by `check` and is not performed by `xtrace`.

`XTrace` Hardware Mode Controls

`xtrace` is laid out in two parts: the control panel at the top of the tool window and the "slate" on the lower part of the tool window. The control panel provides command functionality; the slate provides the drawing interface.

Throughout this section, consider a very simple TRAM configuration, consisting of a single T800 root TRAM on a B014 board. The `check` configuration specification would look like this:

```
Using /dev/bxiv0 check 2.53
# Part rate Mb Bt [ Link0 Link1 Link2 Link3 ]
0 T800d-25 0.45 0 [ HOST 1:1 ... ... ]
1 T2 -20 0.95 1 [ 2:C 0:1 ... 3:C ]
2 C004b ()
3 C004b ()
```

Let's say that this configuration is stored in the file "simple". Invoking `xtrace` using "simple" as the input configuration would look like this:

```
xtrace -H simple
```

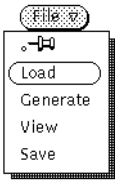
and would result in the tool appearance shown in Figure 1. Note the following:

- the "`Configuration:`" indicator indicates we are configuring the hardware

- the "`Filename:`" field displays the name of the file we are using
- the buttons are displayed and give us choices as to what to do with the configuration
- the slate displays the configuration by depicting the TRAMs and their connections.

The File Button

Manipulating configuration files is done through the "`File`" button. This button is a "menu button", that is, there is a menu associated with it. Simply pressing the button the left hand mouse button invokes the first item on the menu. Pressing the right hand mouse button on the button brings up the menu below:



Note the "pushpin" -- by pressing any mouse button on this pin, the menu will stay up after using it. This is convenient for repeated file manipulations. The various options are outlined below:

- **`Load:`** This is the default. The file specified on the "`Filename:`" field is opened and parsed. If the configuration specification can be parsed without errors, the specification is drawn on the slate; otherwise, the errors are noted and no change is done on the slate. Other errors with the specification file (e.g., an empty file name or a non-existent file) are also noted with no change to the slate.
- **`Generate:`** This option allows the specification to be read from the actual TRAM configuration on the motherboard. This is done using `check` and piping the specification to the file named in the "`Filename:`" field. If that file already exists, or if the file name is empty, these conditions are noted and appropriate action is taken (e.g., the generate function is aborted if an empty file name is given). Once it is cleared to proceed, `xtrace` will invoke `check` to generate the current configuration of the TRAMs, parse that specification, and display it on the slate.
- **`View:`** This option allows the user to view the text of the configuration file being displayed on the slate. A window pops up that displays the test of the configuration file. Whenever the actual configuration file changes, this view will change. (Note that the file changes only by save, generate, and load operations, not by redrawing the configuration on the slate.)
- **`Save:`** This option saves the configuration on the slate to the file given in the "`Filename:`" field. Errors with the name of the file (i.e., empty or already existing) are noted and appropriate action is taken.

The Clear Button

The "`Clear`" button clears the current configuration by erasing the slate and initializing all internal data structures to an initial configuration. Note that any "reset" motherboard has at least its C004s, a root TRAM, and a T2 TRAM. Hence, pressing the "`Clear`" button will initialize everything to this configuration. The "simple" example, displayed in Figure 1 is the initial configuration upon pressing "`Clear`".

The Configure Button

The "Configure" button allows the user to use the configuration on the slate to configure the TRAMs on the motherboard. Pressing "Configure" starts a sequence of events:

1. Check to make sure the configuration has been saved and handle any file problems that may develop (e.g., perhaps by aborting or by a warning to the user).
2. Pop up a window so the user can "watch" the configuration sequence.
3. Issue a "check -cr" command to clear/reset the TRAM configuration.
4. Issue a "check -cs <*filename*" where *filename* is the name of the file containing the configuration from the slate (i.e., off of the "Filename:" field of xtrace).

The popup window has two purposes: to show the user the configuration sequence and to show the user any error that might have resulted from the configuration. The reader is reminded that xtrace does *no error checking on the configuration specification* except for a few very obvious flaws (e.g., trying to connect a link 0 or 3 to a link 1 or 2). So this window is very important.

Also, it is important to note that the resulting configuration file that is generated in step 4 above -- i.e., from the invocation of check -- is discarded. The configuration depicted on the slate remains the current configuration until the user issues a load, save, or generate command. Thus, errors can be safely corrected by watching the configuration sequence and iterating over the redraw and reconfiguration steps.

The Macros ▽ Button

The xtrace program uses external programs, i.e., from the check family. There may other external programs that a user may have written that are useful in configuring or testing TRAMs. Programs such as "load", which comes with the check programs but which is unused by xtrace, or specialized shell scripts are examples of useful programs. Access to these external features are provided by the "Macros" button.

If a file called ".xtracemacs" exists in the user's home directory, its contents are used to define selections in a menu that is called up from the "Macros" button. Like the "File" button, the "Macros" button is a menu button, where the default selection is the first item on its menu and its menu can be displayed by pressing the right hand mouse button over the "Macros" button. If the ".xtracemacs" file does not exist when xtrace is invoked, the "Macros" button is "grayed out", i.e., unusable, but still there.

The syntax of the ".xtracemacs" file is a set of lines, each defining a menu selection. Each line must be in the following format:

<*label*><tab><*definition*>

where <*label*> is phrase that becomes a menu entry label and can be made up of any characters. The <*definition*> is a string of characters that make up the command that is issued to the system when the selection labelled <*label*> is selected. The command string can be made up of any characters, including those in two special classes. First, if a "$" is encountered, it and one character after the "$" are assumed to be a specification for a filename in the xtrace program. The following sequences are recognized:

`$b`	build makefile name
`$h`	hardware configuration file name
`$s`	software input description file name
`$t`	software output configuration file name

The second class of special characters is comprised of the following special "backslash sequences":

`\b`	blank
`\d`	`DEL` character
`\e`	escape
`\f`	form feed
`\n`	new line
`\r`	return
`\s`	space
`\t`	tab
`\\`	`\`
`\0`	NULL character (ASCII 0)
`\`*`xxx`*	where *xxx* is a three digit hexadecimal number, the character represented by the three-digit ASCII code

Selecting an item in the button's menu causes the defined command string -- with the appropriate substitutions make for special sequences -- to be issued to the system. A popup window opens to display the results of the command.

The FTest and MTest Buttons

The `check` tools include two testing programs that are used in conjunction with the `check` specification file. "`Ftest`" tests the functionality of the TRAMs in the specification; "`mtest`" tests and reports on the memory in each TRAM in the specification. The user may perform each of these tests using the "`Ftest`" and "`Mtest`" buttons.

Pressing one of these will attempt to execute the appropriate test, opening a window to display the results. Checks are made to see if the configuration on the slate has been saved and then uses that configuration. Note that if the user has not configured the motherboard using the current slate configuration, but issues a test, *the results are undefined and will quite probably be anomalous.*

The Quit Button

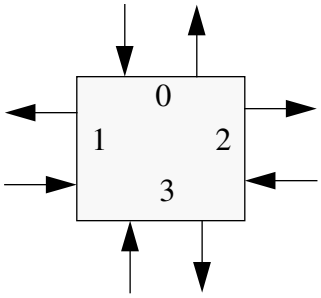
This button allows the user to halt and exit from `xtrace`. If the current configuration needs to be saved, the user is given that option (that is, saving a configuration is *not* automatic upon exit).

Configuring the TRAMs: The `XTrace` Slate

Functionality on the slate is based on the mouse buttons. This functionality allows the user to draw/define and delete links between TRAMs, to define TRAMs and delete existing ones, to specify the bootlink of a TRAM, and to obtain information on TRAMs displayed on the slate.

Links

Before defining the functionality concerning links, it is important to define the `xtrace` link model. `Xtrace` views a TRAM as below:



Note the arrangements of links and the link number assignment. Note also the position of the input link and the output link.

Links are defined by the left mouse button. To define a link from output link *a* on a TRAM to input link *b* on another TRAM, the user places the mouse on the side of the TRAM for link *a*, depresses the left mouse button, then positions the mouse pointer on link *b*, and releases the mouse button. If the link just defined is legal, i.e., it is not a link between a 0/3 link and a 1/2 link, and a link does not already exist, then the link is defined and `xtrace` draws a line between the two points. If the link is illegal, the error will be noted and the link will not be defined.

Links are deleted by the middle mouse button. Depressing the middle mouse button on the *source* (i.e., output side) of the link deletes the link, if it is legal to delete it. Illegal situations are noted and the link is not deleted.

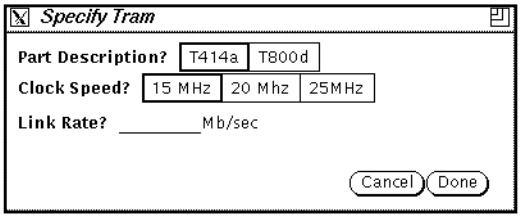
TRAMs

TRAMs are defined and deleted by holding the shift keys down while using mouse buttons. TRAMs may be defined by pressing the left mouse button while holding the shift key down. Doing so opens the window shown in Figure 2 to get information about the newly created TRAM. The user should define the information then press either the "`Cancel`" or "`Done`" buttons. The former cancels the create operation; the latter will complete the definition by redrawing the slate with a new TRAM defined.

The user should note that specification of bootlink speed and actual bootlink are actually superfluous. When `check` configures, it ignores this information and it generates the actual information when requested. So, it is probably preferable to define bogus information for these parts

---

Figure 2: The TRAM Specification Window



and to press the "Configure" button followed by the "Generate" selection from the "File" menu if precise information is desired.

TRAMs are deleted by holding the shift key down and pressing the middle mouse button a TRAM. If it is legal to delete the TRAM (i.e., it is not the root TRAM or a C004 or a T2), the TRAM and all links from and to the TRAM are deleted. Error conditions are noted and result in no change to the configuration.

Specifying the Bootlink of a TRAM

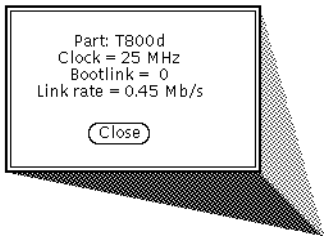
The link a TRAM boots from is assumed to be link 0 when the TRAM is created. To change this specification, the user must hold the Control key down and indicate the link with the right mouse button.

Error conditions are noted and result in no change to the configuration.

TRAM Information

A `check` specification contains information about a TRAM such as part name and speed of the links it has. This information is collected by `xtrace` and can be displayed by positioning the mouse pointer over a TRAM and pressing the right hand mouse button. A "notice" window will appear and display all information `xtrace` knows about the TRAM.

For example, pressing the right hand mouse button over the root TRAM in Figure 1 gives the information below:



This information is gleaned from all sources available. When an memory test is run, information on the memory is gleaned and displayed when requested.

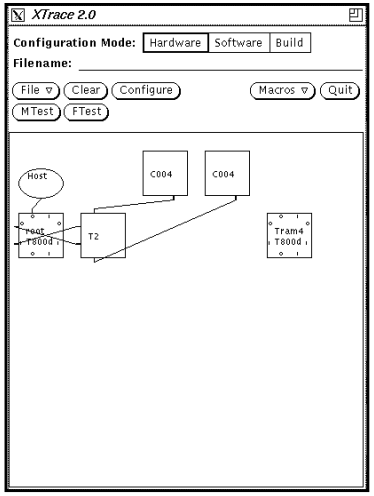
Currently, the gleaning of information from an memory test is not implemented.

An Example

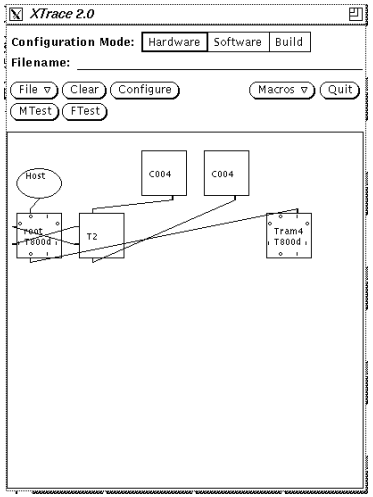
Let's say that Joe User starts with the initial configuration from Figure 1 and wants to expand it. He has a B014 with 3 T800 TRAMs and wants to use all 3. He begins by defining a new TRAM -- by holding the shift key down while pressing the left mouse button. The window in Figure 2 pops up and he defines his TRAM as a T800d with a 20 MHz clock and link rate of 99. He presses the "Done" button and a new TRAM is drawn as in Figure 3(a). Now, he wants to add links from the root TRAM to the new TRAM, and draws these in -- linking slot 0, link 3 to slot 1, link 0, as in Figure 3(b).

Now, Joe wants to configure the B014 as is depicted on the slate. He presses the "Configure" button. First, `xtrace` warns him that he has not saved his configuration (Figure 3(c)). So, he cancels the configuration command and saves his configuration to the file "simple2". Then he presses the "Configure" button, and he gets the results shown in the "results window" in Fig-

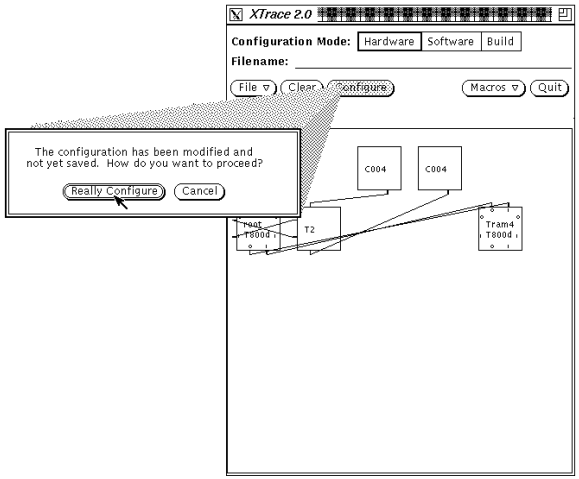
Figure 3: Joe User's Examples



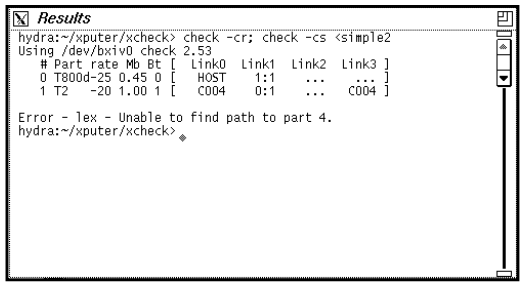
(a)



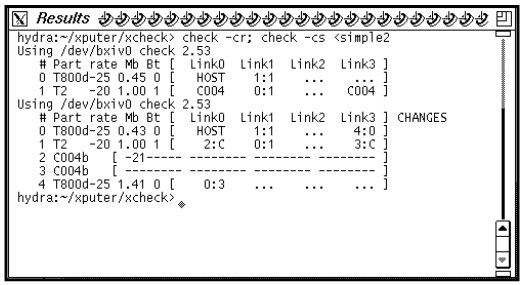
(b)



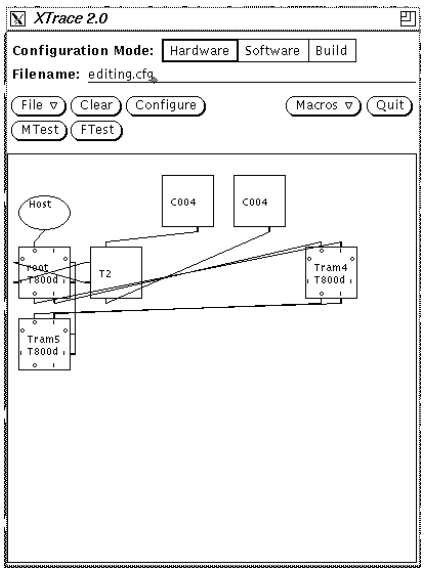
(c)



(d)



(e)



(f)

ure 3(d) -- telling him he has an error. So Joe adds a link back from slot 1, link 0 to slot 0 link 3, and reconfigures. The results window, shown in Figure 3(e) shows that all went well. He repeats this procedure for the remaining two TRAMs, getting the configuration shown in Figure 3(f).

Configuring a Parallel Program

Once the hardware has been configured properly, the software must be mapped to the hardware. Using the INMOS toolsets (C and Occam), this is done using a program called "`icconf`" and a software configuration file. This software configuration file gives the hardware configuration (again), specifies the interface requirements of each process in the system, and maps the processes and channels in the software to hardware processors and links.

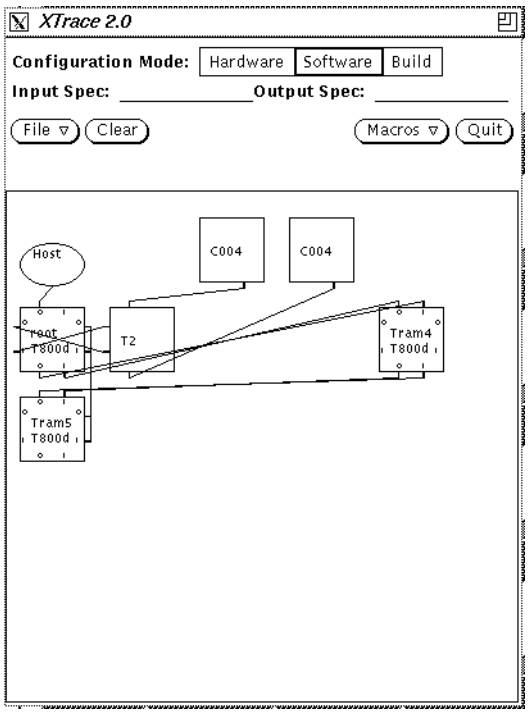
`XTrace` removes the necessity of writing a new configuration file and re-specifying the hardware configuration. By using the mouse, the user can place processes and channels on a hardware configuration, and the software will write the configuration.

Using `XTrace` to Configure a Parallel Program

To configure a parallel program, the user must place `xtrace` in software mode. This is done by invoking with a "`-s`" option or by choosing the "`Software`" selection in the "`Configuration Mode:`" item. An example of `xtrace` in software mode is shown in Figure 4. If the user has a special hardware configuration upon which the software is to be placed, that hardware configuration must have been loaded into `xtrace`. Failing to load this hardware configuration file will map the software onto a one-processor Transputer network.

The user must have a *description file* to load into the software configuration portion of `xtrace`. This file is comprised of three kinds of specifications, each coming from those in a complete INMOS toolset software configuration specification.

Figure 4: XTrace in Software Mode



- *Process specifications:* The “process” specifications in the INMOS toolset configuration syntax that describe the requirements of each process.
- *Connection specifications:* The “connect” statements that describe the channel connections between processes.
- *Program code usage specifications:* The “use” statements that specify what code to use for each process.

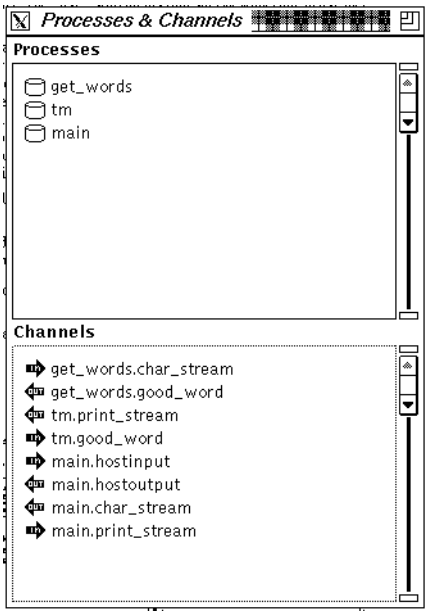
> **NOTE:** In its version 2.0 release, the syntax that `xtrace` will accept is restricted from the complete syntax as specified by INMOS. Basically, arrays, expressions, and iterated constructs are disallowed. Also, only channel specifications are allowed in the `interface` section of a `process` specification.
>
> These restrictions will go away in the next release of `xtrace`.

This specification is parsed by `xtrace` and a list of processes and channels are presented to the user. An example is shown in Figure 5. Now, processes and channels may be placed onto the hardware by first selecting the component to be placed from the list -- by clicking the left mouse button the item’s list text -- and then selecting the corresponding TRAM or link on the slate -- again by pressing the left mouse button on the corresponding slate item. Upon placement, the text of the selected process or channel will be amended to include information as to where the item has been placed. In addition, if the item is a channel, the channel to which the item is connected will also automatically be placed, and its list text will be changed accordingly.

Items may be placed in the manner described above repeatedly until the user is satisfied with the configuration.

When all the items have been placed satisfactorily, the “`Save`” item from the file menu should be chosen to save the software configuration in the specified output file.

Figure 5: The Process and Channel Lists

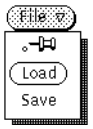


`XTrace` Software Configuration Controls

This section describes the controls specific to software configuration. The "`Macros`" and "`Quit`" button were covered in the hardware configuration section.

The File Button

As with hardware configuration, file manipulation is done through the "`File`" button. The menu in this case is very straightforward; pressing the right mouse button on the "`File`" button gives the menu below:



The options are as follows:

- **`Load`**: Choosing this option is the default when the user simply presses the "`File`" button with the left mouse button. This causes `xtrace` to open and parse through the file whose name appears on the "`Input Spec:`" line. Any problems with opening the file (bad names or no specification, etc.) are dealt with appropriately. The process and channel lists are then filled in.
- **`Save`**: This option writes a complete software configuration into the file name named on the "`Output Spec:`" line. Problems with file naming are dealt with in the usual way.

Some brief notes should be made about the way the configuration file is written.

- *The software configuration file must have hardware configuration information in it.* For `xtrace` to write this file, it must have hardware configuration. It will write a software configuration file regardless of how much hardware information it has, but the file is useless unless the user has started by loading or specifying a hardware configuration.
- *The user must explicitly place the channel connections to the host.* It is not obvious in the description file which channels off of the processes on the root must be connected to the host. This must be explicitly placed.
- *The link to the host from the root TRAM is given by a special constant.* The constant "`hostlink`" is used to represent the link through which the root TRAM is connected to the host. This is so debugging can be done using the INMOS debugger "`idebug`" -- which connects the configuration through link 1, not link 0 as is normal without debugging.
- *The description file is incorporated in the final software configuration file.* The description file is copied verbatim into the final configuration specification, so place specifications in there carefully.

The Clear Button

The "`Clear`" button will clear out the software configuration while retaining the software description. This has the same effect as reloading the software description file.

Configuring Our Example System

Now Joe Software wants to configure his software. He has written the following configuration file:

```
process (stacksize = 20K, heapsize = 20K,
 interface (input char_stream, output good_word)) get_words;

process (stacksize = 20K, heapsize = 20K,
 interface (output print_stream, input good_word)) tm;

process (stacksize = 20K, heapsize = 20K,
interface (input hostinput, output hostoutput,
 output char_stream, input print_stream)) main;

connect from_server to main.hostinput;
connect to_server to main.hostoutput;

connect main.char_stream to get_words.char_stream;
connect get_words.good_word to tm.good_word;
connect tm.print_stream to main.print_stream;

use "main.c8x" for main;
use "get_words.c8x" for get_words;
use "tm.c8x" for tm;
```

Joe now starts `xtrace` using the hardware configuration from the last section. After entering the software configuration mode and loading the above file, Joe gets the windows in Figure 6(a). He starts by placing the processes -- selecting each process and then clicking the left mouse button on a TRAM. Then he places the channels in the same manner. When he has completed this placement, `xtrace` looks like it does in Figure 6(b).

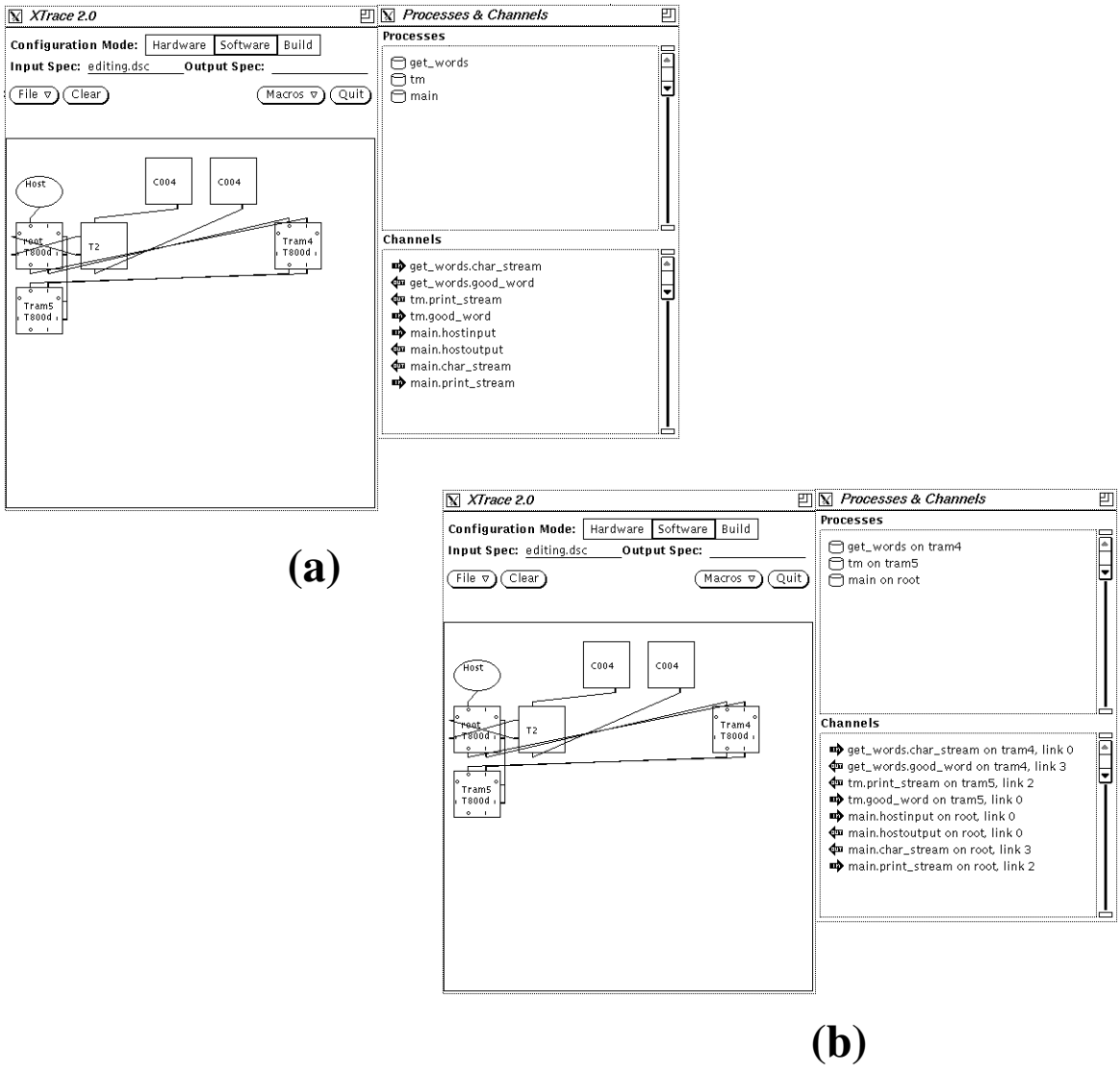
Configuring a Program Build

The combination of parallel programming requirements, proper software design techniques, and INMOS tools and configuration files results in the use of many, many programs and files to produce one bootable program that is run through the `iserver` and downloaded to the Transputer network. Through its build mode, `xtrace` helps this process by allowing the user to build a Makefile that can be used to produce the bootable end result.

Using XTrace to Generate a Makefile

In order for `xtrace` to know what files are involved in the build process, the description must have been loaded. This provides information -- through the “use” statements -- on the program code used to produce the bootable. Once this description file has been loaded, the build mode is

Figure 6: Joe Software’s Software Configuration



meaningful. In build mode, a new panel -- called the “build panel” -- replaces the configuration slate. This panel lists information for each program module used in the bootable build process. This panel has a scrollbar, and this should be used to scroll through the information specifications for each module. The (educated) guesses that `xtrace` has made pertaining to module specifics may be altered. When the user is satisfied with the information provided, the information should be saved through the “`File`” button -- thus generating a makefile to be used with the Unix “`make`” facility.

An example of `xtrace` in build mode is given in Figure 7.

Please note that the makefile generated is quite standard, and will not take into account any special needs or facilities you want represented. *The user should always review the makefile before attempting a bootable build for the first time.*

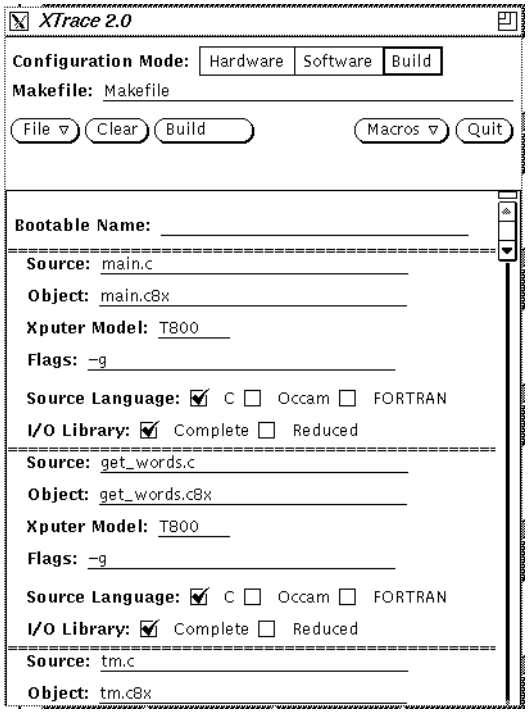
Build Controls

This section describes the controls specific to build configuration. The “`Macros`” and “`Quit`” button were covered in the hardware configuration section.

The File Button

As before, the “`File`” button us used to manipulate files. However, in build mode, the only file manipulation that goes on is the saving of the build information into a makefile. Thus, the menu connected with the button is simple:

Figure 7: XTrace in Build Mode



As one might expect, the "`Save`" feature deals appropriately with errors in file handling.

The Clear Button

This button resets any information not set up by the parsing the software configuration file. That is, the makefile name and the bootable program name are cleared, and the parameters for each module that `xtrace` guessed at are reset.

The Build Button

This button initiates the build process. A command (shell) window is opened and the command

```
make -f <makefile name>
```

is executed. The specified makefile name is substituted for *<makefile name>* above. The results of the build process are recorded in the command window.

Finishing the Example

Once Joe Software loaded the software description, and switched to build mode, the window displayed in Figure 7. After inspecting the compiler options and libraries -- and changing the I/O libraries used in the processes not on the root TRAM -- Joe can go ahead and save the makefile.

Conclusion: Bugs, Warnings, and Future Enhancements

`XTrace`, while in its 2.0 release, is in its first release to the public. This version has moved from 1.0 to 2.0 by being tested by a class of 10 students madly parallel programming on a set of Transputer B014 boards -- each with eight TRAMs. However, there are bound to be bugs in the implementation, so stand warned.

Please send comments, bug notices, and suggestions to Mike Jipping, Hope College Department of Computer Science at `jipping@cs.hope.edu`.

Some future enhancements that are planned to the 3.0 release:

- *Better hardware information gathering:* Currently very little information is gathered on the hardware except for link connections. Parsing the results from `mtest` and `ftest` invocations would give more information about memory and link speed, for example.
- *More complete syntax on the software description file:* The current parser restricts the syntax of the software description file. A complete syntax would be very desirable.
- *Different -- and OpenLook compliant -- interface for the build mode:* The current scrollable panel is just sufficient as an interface. A different -- compliant -- interface is definitely needed.