WETICS

Web-Enabled Texas Instruments TMS320C30 Simulator

Dogu Arifler, Chi Duong,
Brian L. Evans, and Srikanth Gummadi

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712-1084

October 31, 1997

Contents

List of Figures

List of Tables

Web-Enabled Texas Instruments TMS320C30 Simulator

ABSTRACT

We present a Web-based client-server framework for interactive simulation and debugging of software for programmable digital signal processors (DSPs). The framework, which is accessible by any Java-enabled Web browser, consists of (1) a graphical user interface (in Java), (2) a multithreaded Internet server (in Java), and (3) a simulator (in C/C++) for the Texas Instruments TMS320C30 (C30) DSP processor. The C30 simulator is a standalone application that has been extensively validated using the C30 DSP Starter Kit board and tools. Based on feedback from the C30 simulator, the user interface configures itself. The same user interface can adapt to different simulators and debuggers. The source code for the framework is portable, extensible, and freely distributable. Our framework can support Web-based university education, distance learning, design space exploration, and software validation, for a wide variety of TI DSP processors and boards.

1. Introduction

Many of today's electronics products consist of many dedicated, configurable, and programmable processors that perform a variety of signal processing, control, and communications algorithms in real-time. These products include digital cellular phones, voiceband data modems, compact disc players, disk drives, and engine control systems. In order to perform design tradeoffs of candidate architectures and validate designs at the physical level, companies have to buy and maintain many different design tools due the heterogeneity in implementation technologies. Even for a given technology, wide diversity often exists; e.g., Texas Instrument sells at least nine different families programmable digital signal processors (DSPs) [1]. Furthermore, fast simulation technology is critical for validating physical layer descriptions because simulation is often several orders of magnitude slower than the physical operation of the system. Designers have to wait for the next annual release of CAD software to use advances in simulation technology.

Web-based system simulation frameworks offer a solution [2]. In a Web-based framework, a user would connect to one or

more servers that coordinate access to a variety of design tools such as fast simulators or interfaces to physical devices. The framework would save users from having to buy and maintain the tools and allow them to concentrate resources on finding hardware/software solutions for the design task at hand.

This report describes a client-server framework to support Web interfaces for simulating and debugging software on the Texas Instruments TMS320C30 (C30) processor. The framework, which is shown in Fig. 1, consists of

- a graphical user interface (GUI) written as Java applets.
- a multithreaded Internet server written as a Java application.
- a C30 simulator, written in C and C++.

The GUI applets communicate with the Internet server over a TCP/IP socket, and the server communicates with the simulator using pipes. The server ensures security because the user cannot directly run, copy, or otherwise access the simulator---all user requests must go through the server. We have written and validated the C30 simulator to be instruction, cycle, and bit accurate. The simulator reports implementation costs.

The significance of our framework is that it is

- *configurable*--- the GUI customizes its appearance based on feedback from the DSP simulator or debugger being used;
- *portable*--- the GUI, server and C30 simulator work across multiple platforms;
- *extensible*--- any command-line simulator, debugger, and simulator can be plugged into it; and
- *freely distributable*--- all source code has been released by anonymous FTP.

It can support Web-based university education, distance learning, design space exploration, and software validation, for a wide variety of TI DSP processors and boards.

2. The Texas Instruments TMS320C30 (C30) Simulator

In this section, we discuss our standalone C30 simulator which currently runs on PCs and Unix workstations. We have validated the C30 simulator using several application programs and several random programs. We have designed the C30 simulator with a pure command-line interface so that it can be easily controlled by a parent program such as a Internet server or a system-level design tool. The initial release (version 0.1) of the C30 emulator was on April 19, 1997, which to the best of our knowledge made it the first freely distributable C30 simulator. The most recent release (version 1.0.5) was on October 31, 1997.

The TMS320C30 processor is a 32-bit floating-point DSP introduced by Texas Instruments in 1988. It has a modified Harvard architecture, a multiplier-accumulator, extended precision accumulator, zero-overhead hardware looping, special addressing modes, and a four-stage pipeline with hardware support for interlocking. It uses a non-IEEE floating-point format, which complicates emulation, and supports integer arithmetic. It also offers a conventional set of arithmetic logic unit functions (or, and, xor, etc.).

We wrote and validated a simulator for the TMS320C30 processor. The C30 simulator is instruction, cycle, and bit accurate, and executes about 40,000 instructions per second on a 167 MHz UltraSparc II workstation. The C30 simulator runs on the PC under Windows 95/NT and on more than twelve different Unix architectures including Solaris and Linux. The simulator is based on the source code for the C30 DSP Starter Kit (DSK) disassembler written by Keith Larson in the Texas Instruments Applications Group and available on the TI Web site at ftp://www.ti.com/pub/tms320bbs/c3xdskfiles/. The disassembler has been extensively validated, so it offered a reliable start for a C30 simulator. In developing the simulator, we modeled the visible architectural elements such as precision registers, auxiliary registers, index registers, program counter, status register, stack pointer, and repeat mode registers. We implemented the fetch, decode, read, and execute stages of the

pipeline. We modeled the stages, memory access counters, an interlock flag, and data passed between stages of the pipeline.

The C30 simulator takes both machine code and hex files as input. The hex files can be generated using the freely distributable DSK tools. Our simulator working in conjunction with the DSK tools provides a zero-cost entry for designers to use the C30 processor in their designs. The C30 simulator emulates the four stages of the C30 pipeline [3]. The four pipeline stages occur simultaneously in a C30 DSP, but they occur sequentially in the C30 simulator in the following order: execute, read, decode, and fetch. Pipeline conflicts can occur from control hazards (e.g., flush the pipeline in a repeat instruction) or data hazards (e.g., an operand cannot be read yet). The simulator properly mimics C30 hardware interlocking [4] in which the processor halts the stage responsible for the conflict and all stages before it. On the C30, no more than two cycles are lost during interlocking. If the user has enabled disassembling of instructions during simulation, then a pipeline hazard will be reported as a NOP instruction with a comment describing the source of the interlocking. Detecting interlocking is crucial in time-critical loops.

We validated the C30 simulator by comparing its behavior with a C30 DSK board for randomly generated opcodes and example programs. A test program was loaded into the C30 DSK board and the C30 simulator. We ran the test program for a few instructions and then generated a random opcode for both the DSK board and the simulator. We then compare the values of all CPU registers and report any differences. We have been able to validate all instructions C30 simulator supports, which is 90 out of 99 instructions of the C30 instruction set. The C30 simulator does not support external—pin XF0, XF1—interlock operations, as well as lopower, maxspeed operations. All these instructions function well in general cases and for typical addressing mode sequences. Some assembly programs were also validated on the C30 simulator. The same programs were run on both the simulator and the DSK debugger, and the CPU register values were checked after each cycle. In all the cases, a complete match was found. A list of some of the programs is shown in Table 1.

Program Name	Program Length
LOOPAIC.HEX	84 words
SCAN.HEX	176 words
DSKSG.HEX	261 words
DSKOSC.HEX	197 words
FIR.HEX	96 words

Table 1. Validated TMS320C30 Programs.

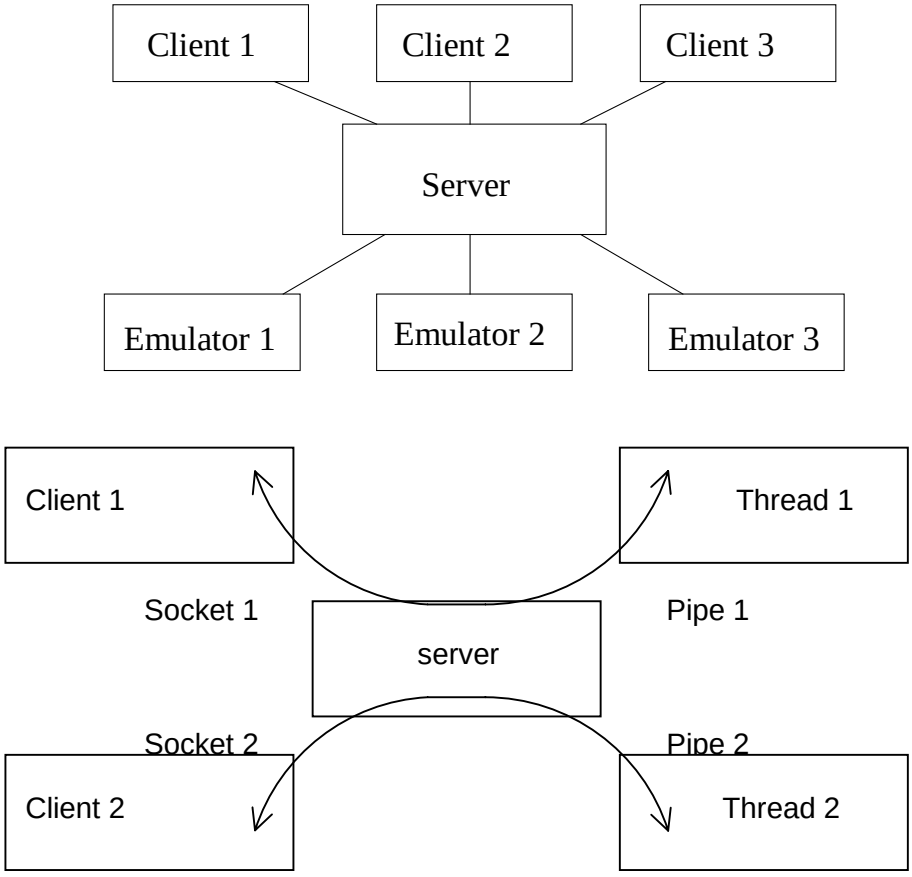

Figure 1. Two views of the WETICS client-server architecture.

3. Internet Server

The Internet server acts as a mediator between the graphical user interface applets and the simulators/debuggers, as shown in Fig. 1. The server is a multithreaded Java application that runs continuously on a host machine [5]. When connection requests are made to the server, a thread is created for each connection. Each thread runs an instance of the C30 simulator.

The primary challenge in the server program is to maintain separate communication channels for each client. WETICS uses TCP connections to handle multiple users. A pair of end-points (one at the client and the other at the server) identifies a connection. An end-point is defined as the integer pair *(host address, port*). The server program listens for connection requests continuously at a previously assigned port (4321 in our case). When a client makes a connection request, it uses an unused random port for sending and receiving data [6]. Therefore, TCP automatically handles the following two cases:

- Clients from different Internet host addresses. Since every host has a different Internet address, we will have a different (*host address, port*) at the client side for each client.

- Clients using the same Internet host but running multiple user interfaces. Since TCP assigns an unused port number to each client, even if the host address is the same, the port number will be different in the (*host address, port*) at the client side. Hence, we will have a unique connection.

When a connection is accepted, we create sockets to enable communication between the clients and the server over the network. We also associate input and output stream to the established socket to send and receive data.

The server program, which creates separate threads for each connection, also passes the reference to the socket it created for a particular connection. The created thread, which executes an simulator or a debugger as a process, is responsible for identifying the input, output and error streams of the process.

The thread chains the input stream of the process to the socket's output stream so that the data coming from the user interface is directed into the simulator/debugger's input stream. Similarly, the output and the error streams of the process are chained to the socket's input stream so that the results produced by the simulator/debugger process can be return to applets to be displayed to the user.

Several security restrictions imposed by Java were overcome by using the server application. For example, downloading a user program into the simulator/debugger can be achieved as follows. The user specifies the location of his/her program as an URL. The user interface reads the contents of this URL byte by byte and sends them to the server, which in turn creates a user file at the host running the server. Then, the server can issue a load command to the simulator/debugger to load the created user file. We overcome security issues by limiting the size of each file and the number of files per user to one.

4. Applets

When a user connects to the Web page

http://anchovy.ece.utexas.edu/~tidesign/wetics/

the applets query the server as to which simulators and debuggers are available and presents the list to the user. Once the user selects a simulator or debugger, the applets interrogate the server to find out what commands the simulator/debugger supports, and what commands to associate with the menu commands to display the processor and simulator state, retrieve help on a command, and step and run the program. Based on responses from the server, the applets configure and display the WETICS GUI. We wrote the WETICS GUI using the Graphic Java Toolkit [7], which provides a set of abstracted classes on top of the Java AWT Windowing Toolkit.

The initial prototype supports the downloading of files using the http protocol for use by the simulator. We overcome security issues by limiting the size of each file and the number of files per user to one. Since only one user can use the one simulator, we do not have any open security or privacy issues.

Fig. 2 shows an example of the GUI after it configures itself for the TMS320C30 simulator. The four windows are entitled *Command Window*, *Response Window*, *Previous Registers*, and *Current Registers*. The *Command Window* accepts simulator commands, whereas the other windows only display output. The visual cue is that the command window has a white background and the other windows have a gray background. The icons from left to right are shortcuts for the following operations: evaluate the commands in the Command Window, single step the program, report the number of instruction cycles executed so far, enable debugging (disassembly) of instructions as they are evaluated, and display the history of commands given by textual and graphical input. The command menu in the left column lists the simulator. A user can double click on any of these commands, and the command will appear in the *Command Window*. These commands are retrieved from the simulator itself via the Internet server.

The two pull-down menus from left to right are File and Help. The File menu supports Load, Info, and Exit. The File-Load command allows users to download files to our server from an HTTP address. Files are downloaded to a `userprograms` subdirectory, which users can access using the simulator's load command. Files are limited to 308 KB in size, and are deleted when the user exits the GUI. The File-Info command displays information about WETICS and the C30 simulator. The File-Exit command exits the applet. The Help menu offers help on each of the simulator commands. At the bottom of the GUI, the Register Values button will display the current register values in the *Current Registers* window, and the previous contents of the *Current Registers* window will be moved the *Previous Registers* window. Any values that have changed since the last time that the *Register Values* button has been pressed is shown with an asterisk * in the *Current Registers* window. When the Register Values button is first hit, register values are compared with the register values when the simulator/debugger first started. Fig. 2 also shows interaction with the TMS320C30 simulator, which is running the program,

```
LOAD 1,R0
RPTS 32
ADD 1,R0
```

The *Simulator Response* window shows the disassembled instructions as they are executed. The last instruction shown is `ADDI 1,R0` which is the instruction executed when the *Single Step* icon was pressed. Before and after the last instruction was executed, we hit the Register Values button. By comparing *Previous Registers* and *Current Registers*, we notice that register r0 has changed value from `0x00000003` to `0x00000022`.

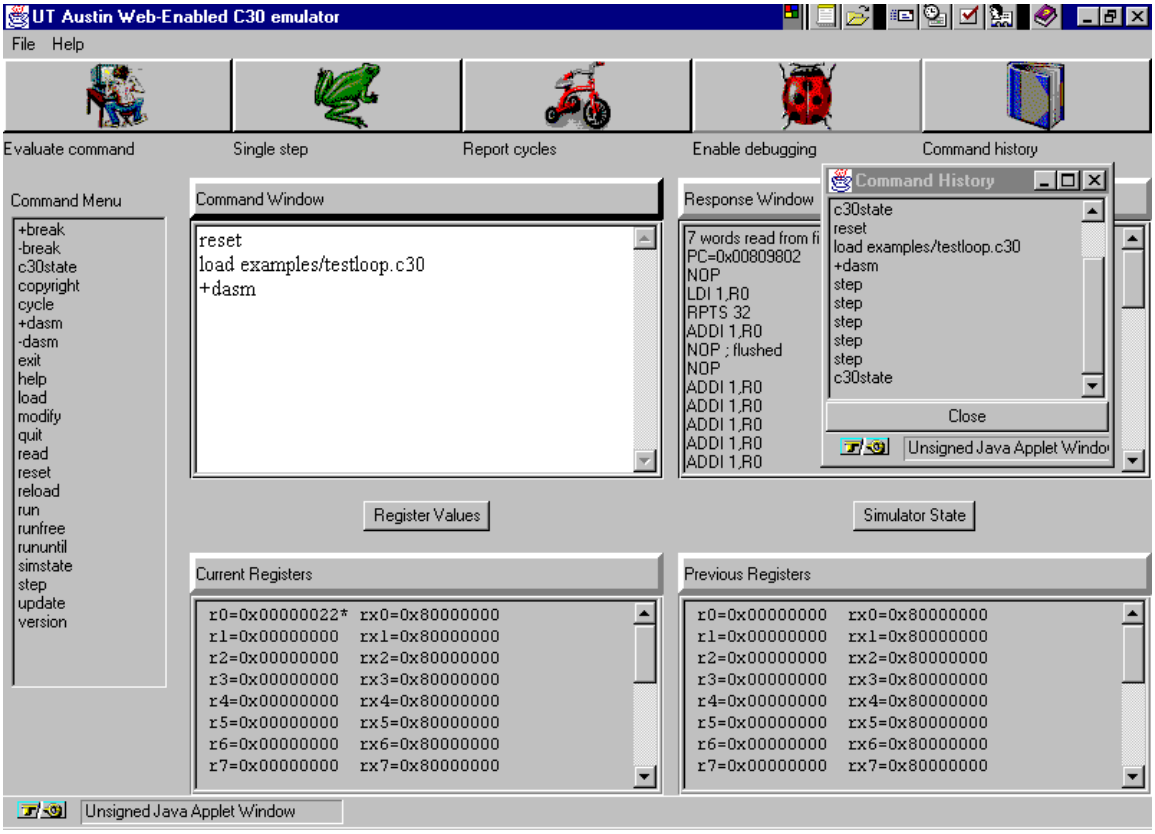

Figure 2. WETICS in action using a Java enabled browser.

5. Summary

Web-based CAD tools can provide designers instant access to the latest advances in simulation and synthesis without the huge maintenance and computation cost associated with modern CAD tools. In this context, we present a framework for Web-Enabled Simulation (WETICS) of software for the Texas Instruments DSPs. The WETICS applets configure themselves according to feedback from the simulator through the TCP/IP server. The Java code underlying the applets and TCP/IP server is portable, and the source code for the TMS320C30 simulators has been ported to Windows and Unix operating systems. The source code for WETICS is freely distributable and frequently released.

Adding new simulators and debuggers to the framework is as simple as adding data about the commands that the simulator supports. So, new simulators can be easily added to WETICS without adding any new program code. Clearly, then, WETICS is a configurable, portable, extensible, freely distributable framework for Web-enabled simulation of embedded software.

References

[1] "Pocket guide to popular DSP processors and cores." Berkeley Design Technology, Inc., Freemont, CA, 1997. http://www.bdti.com/pocket/dsp_guide.htm.

[2] "Web-based Electronic Design (WELD) Project." http://www.cad.eecs.berkeley.edu/Respep/Reaserch/weld/, Directed by Richard A. Newton. Dept. of Electrical Eng. and Comp. Sciences, The University of California, Berkeley, CA. 94720.

[3] Texas Instruments, Inc., Dallas, Texas, *TMS320C3x User's Guide*, 1994.

[4] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals: Architectures and Features*. Fremont, CA: Berkeley Design Technology, Inc., 1996.

[5] D. Flanagan, *Java in a Nutshell*. O'Reilly and Associates, 1996.

[6] C. Comer, *Internetworking with TCP/IP*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

[7] D. Geary and A. McClellan, *Graphic Java*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1997.

Appendix A
Advantages of WETICS over TechOnLine

WETICS is a valuable complement the on-line access provided by TechOnLine http://www.techonline.com to two C30 Evaluation Module (EVM) boards:

- WETICS provides a fast, robust, Java-based Web interface to C30 tools as compared to a sluggish command-line telnet interface.
- WETICS allows instant access to the C30 emulator while TechOnLine users are waiting for their turn to access one of the two C30 EVM boards.
- WETICS allow users to download programs to our framework from a Web address instead of forcing the user to cut and paste code in a Web browser window.

Other Advantages of our framework are that

- It allows any command line simulator or debugger to be added easily and thus we could give access to more DSP boards as the need arises
- It provides zero cost entry for companies to develop C30 applications because our freely distributable C30 simulator work seamlessly with the freely distributable C30 DSK tools developed by TI.
- It is released by FTP with all source code (under the GNU license) so that it can be easily installed on other sites and in other Web-based CAD frameworks.