Freescale Semiconductor
Application Note

AN2749
Rev. 1, 01/2005

Using sim_G4plus on Genesi Pegasos II

by *Maurie Ommerman, Top Changwatchai, Jack Chen*
CPD Applications
Freescale Semiconductor, Inc.
Austin, TX

This application note, the ninth in a series describing the Genesi Pegasos II system and its applications, addresses the cycle-accurate simulator, sim_G4plus. The sim_G4plus has many features to aid in the performance analysis of software. Although it was specifically developed for the MPC7447A, the simulator may also be useful with the MPC745x and MPC744x processors. This document provides examples that help the reader use the simulator as a learning tool on the Genesi Pegasos II computer.

Contents

freescale™ semiconductor

1 Introduction

At the time of publication of this application note, the document *sim_G4plus v0.8 Cycle-Accurate Simulator User's Guide* was available. However, the Genesi Pegasos II computers were shipped with v0.7 of the simulator. This document addresses only version v0.7, while later documentation will be included in any simulator package available from the PowerPC™ processors page at the Freescale web site. For further instructions see Section 9, "References."

Because the cycle-accurate sim_G4plus has multiple uses in software performance analysis, it is suggested that the reader also consult the *sim_G4plus v0.7 Cycle-Accurate Simulator User's Guide*, which is supplied in the directory, /home/guest/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/doc.

An overview of the sections included in this application note follows:

- Section 2, "Terminology," defines the terminology used in this application note.
- Section 3, "Directory Layout," describes the organization and contents of the directory.
- Section 4, "Compile, Simulate, and View Results," gives the instructions for generating the pipeline output.
- Section 6, "Compile," describes the markers used to generate the pipeline output and the compiler static flag necessary to use the simulator.
- Section 7, "Run the Simulator," describes the options available and the commands used to generate the pipeline information.
- Section 8, "View Pipeline," describes the pipeline viewer display.
- Section 9, "References," lists the reference material used in preparing this application note and the other application notes in this series.
- Section 10, "Document Revision History," describes the history of this application note.

This paper will explore the execution and results of the following commands:

```
Preparation
$which sim_G4plus
$cd ~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples
$less runexample_accel_exemode.sh
$less forloop_lptrace.c

1. Compile source code
$gcc -o forloop_lptrace forloop_lptrace.c -static
$file forloop_lptrace

2. Run simulator
$sim_G4plus -a -p forloop.pipeout  forloop_lptrace  > forloop.stats
$less forloop.stats

3. View pipeline
$sim_G4plus_vp -h
$sim_G4plus_vp forloop.pipeout &
$../bin/spftools/tkvp ../fmts/lsu.fmt forloop.pipeout &
$../bin/spftools/tkvp ../fmts/bpu.fmt forloop.pipeout &
```

The user is assumed to be logged in as `guest` with password `guest`, and all the examples discussed in this paper are in the directory, /home/guest/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples.

The sim_G4plus is characterized as follows:

- It is cycle-accurate; hardware is simulated at a cycle-by-cycle granularity.
- It is an execution-driven timing simulator of the MPC7447A. That is, simulation is driven by an executable program.
- It is useful for the following purposes:
  - application tuning
  - library development and tuning
  - compiler optimization

2 Terminology

The following terms are used in this document:

OS	Operating system including dynamic libraries
Linux OS	Linux operating system
GNU	GNU's not Unix (a recursive acronym)
gcc	GNU compiler collection
Simulation	Using software to simulate the PowerPC hardware behavior

3 Directory Layout

This paper corresponds to the files in the /home/guest/fae-training-04/sim_G4plus_v0_7_1_linux_ppc directory shown in Figure 1.

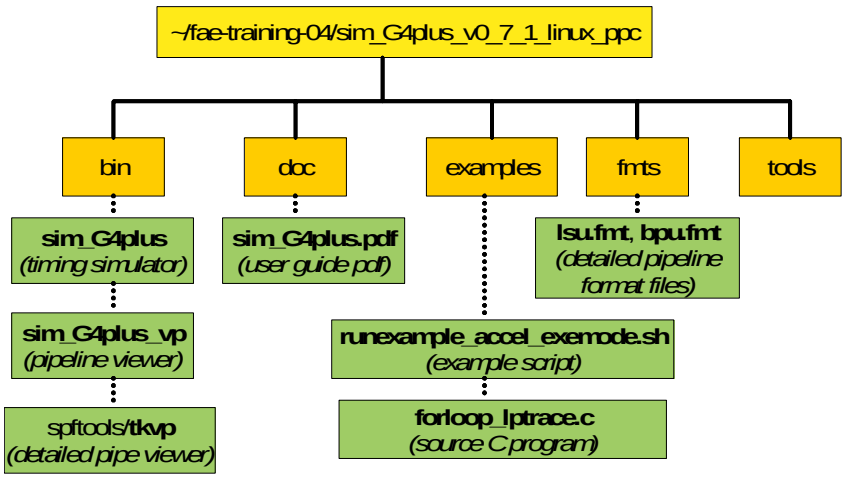


Figure 1. Directory Structure

The bin directory contains the two executables:

- sim_G4plus is the simulator executable
- sim_G4plus_vp is the pipeline viewer
- spftools/tkvp is a directory containing the detailed pipeline viewer, tkvp

The doc directory contains the user guide in PDF format.

The examples directory contains an example that will be discussed in this paper:

- runexample_accel_exemode.sh is a shell script to compile, run the simulator, and view the pipeline
- forloop_lptrace.c is a C program to simulate

The fmts directory contains several format files that are used by the spftools pipeline viewers, sim_G4plus_vp or tkvp, to format the display of the pipeline output:

- bpu.fmt, for the branch processing unit.
- lsu.fmt, for the load store unit
- scoreboards.fmt, for register scoreboards
- fet.fmt, for the fetch unit.
- mss.fmt, for the memory sub-system.
- snapshot.def, for the overall pipeline view
- shortsnapshot.def, for a shorter width overall pipeline view
- snapshot_no_vmx.def, for a shorter width overall pipeline view without AltiVec (VMX) pipeline display

The tools directory contains compressed tar files of the spftools pipeline viewing tools, lptrace trace generation tools, and the TTE trace analysis tools.

4 Compile, Simulate, and View Results

The process to compile, simulate, and view results is carried out by the set of commands below. Running this set of commands allows the user to create an executable from the test program (forloop_lptrace.c), generate the resulting pipeline output, and view the results in a visual display of the pipeline execution. More detailed instructions are supplied in subsequent sections.

4.1 Step 1: Logging In

Login to the Genesi Pegasos II system as `guest` with password `guest`.

Ensure that the path setting includes the sim_G4plus simulator by executing the `which` command.

```
guest@debian:~$ which sim_G4plus

/usr/bin/sim_G4plus

guest@debian:~$
```

As shown above, the result of the `which` command should be the location of the sim_G4plus simulator. If no output is received, then either the sim_G4plus executable is not in the path or it is not installed in the /usr/bin directory.

4.2 Step 2: Viewing Examples

Go to the examples directory.

```
guest@debian:~$ cd fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples

guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$
```

NOTE

Debian Linux supports command and filename completion; type the first few characters of a name and then press <tab> for auto completion.

If this directory cannot be accessed, then either the path was incorrectly typed, or the examples or the sim_G4plus directory were not installed on your machine. Please contact risc10@freescale.com if the latter is the case.

Look at the files in this directory.

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ ls
GNUmakefile               forloop.state      runexample_accel_exemode.sh
README.examples           forloop_exemode.c  runexample_exemode.sh
forloop.pipeout           forloop_lptrace.c  runexample_lptrace.sh
forloop.reference.output  hello.tte
```

4.3 Step 3: Cleaning the Work Directory

If the listing includes many more files, then run the `make clean` command to clean the directory of generated files:

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ make clean
rm -f forloop_exemode
rm -f forloop_lptrace
rm -f forloop_lptrace.tte
rm -f forloop_exemode.txt
rm -f forloop_lptrace.txt
rm -f forloop_accel_exemode.txt
rm -f forloop_exemode.pipeout
rm -f forloop_lptrace.pipeout
rm -f forloop_accel_exemode.pipeout
rm -f forloop_exemode.stats
rm -f forloop_lptrace.stats
rm -f forloop_accel_exemode.stats
```

4.4 Step 4: Compiling with GNU Make

At this point, the directory should be clean with only the files described below. If this directory or these files are missing, please contact risc10@freescale.com.

- The GNU make tool has the ability to find and run the following make files by default:
  - GNUmakefile
  - Makefile
  - makefile
- The GNUmakefile runs three scripts
  - sh ./runexample_lptrace.sh
  - sh ./runexample_exemode.sh
  - sh ./runexample_accel_exemode.sh

- To verify the availability of the GNU make tool, the following or similar output should be seen when executing the `make -v` command:

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ make -v
GNU Make 3.80
```

4.5 Step 5: Running the Example Tool Flows

- forloop_lptrace.c

  Source code program that demonstrates how to instrument a program for tracing

- runexample_lptrace.sh

  Script that demonstrates the tool sequence for generating a pipeline output from a C program. The trace is then fed to sim_G4plus, which in turn generates a pipeout and a statistics file.

- forloop_exemode.c

  Source code program without trace instrumentation for running through execution driven mode

- runexample_exemode.sh

  Script that demonstrates the tool sequence for generating a pipeline output from a C (.c) program. The file that is loaded by the simulator is the same forloop example used by the script runexample_lptrace.sh used to generate a trace file. sim_G4plus is given the forloop executable to run, but is instructed to start execution of the binary in accelerated mode. Once the first lptrace start marker is encountered, sim_G4plus will begin profiling the executable until the lptrace stop marker is encountered. The pipeout represents this subsection between lptrace markers and should be identical to the pipeout generated by the runexample_lptrace.sh script. Stop and start markers are described in Section 7, “Run the Simulator.”

- runexample_accel_exemode.sh

  Script that demonstrates the tool sequence for generating a pipeline output from an lptrace instrumented C program using the accelerated execution-driven mode of sim_G4plus. No trace files are generated in this sequence.

- forloop.pipeout

  Pipeline output file resulting from the simulation.

- forloop.reference.output

  Directory that contains all the output that is generated by the simulator. It is a reference that can be used to compare and verify the output generated by running the shell scripts.

5 The Tool Flow

The listing of the shell script, runexample_accel_exemode.sh, shows the entire tool flow for compiling, simulating, and viewing the results.

```
1  #!/bin/sh
2
3
4  # This script demonstrates the tool sequence for generating a pipeline
5  # output from an lptrace instrumented .c program using the accelerated
```

```
6  # execution-driven mode of sim_G4plus.  No trace files are generated
7  # in this sequence.
8  #
9  # This script expects to be run on a Linux PowerPC platform because
10  # the binary needs to be built using gcc for PowerPC.
11  #
12
13  if [ "`uname -sm`" != "Linux ppc" ]; then
14  echo
15  echo This script must be run on a Linux PowerPC platform
16  echo Please examine this script to understand how to invoke the tools
17  echo for performance analysis.
18  echo
19  exit 1
20  fi
21
22
23  # Compile instrumented program (.c -> executable)
24
25  echo "Compiling forloop_lptrace.c..."
26
27  # The generated executable must be linked statically
28  gcc -O -o forloop_lptrace forloop_lptrace.c -static
29
30
31  if [ ! -x ../bin/sim_G4plus ]; then
32  echo "ERROR: Expected sim_G4plus to be here: ../bin/sim_G4plus"
33  exit 1
34  fi
35
36  echo "Running on the simulator..."
37  # Run through simulator (.tte -> .pipeout) (.tte -> .stats)
38  ../bin/sim_G4plus -a -p forloop_accel_exemode.pipeout \
39              forloop_lptrace > forloop_accel_exemode.stats
40
```

```
41  if [ ! -x ../bin/sim_G4plus_vp ]; then
42  echo "ERROR: Expected sim_G4plus_vp to be here: ../bin/sim_G4plus_vp"
43  exit 1
44  fi
45
46  echo "Generaing pipeout dump..."
47  # View pipeout (.pipeout -> .txt)
48  ../bin/sim_G4plus_vp -r 60 -n forloop_accel_exemode.pipeout > forloop_accel_exemode.txt
49
50  echo
51  echo
52  echo If no errors occurred, the file forloop_accel_exemode.txt should now contain the
53  echo pipeline output.  It is viewable using a text viewer. For example:
54  echo
55  echo "   less -S forloop_accel_exemode.txt"
56  echo
```

Table 1 shows the phases of the tool flow by line number for the above script.

Table 1. Phases of Tool Flow by Line Number

Line Numbers	Phase Description
1	Indicates run as a Bourne shell, sh.
2–12	Comments
13–21	Tests whether the Linux OS is running on a PowerPC processor; if not, it just exits. Pipeline output is only available on Linux running on a PowerPC processor.
23–29	The compile phase. See Section 6, “Compile.”
31–35	A test to ensure that sim_G4plus tool—the simulator that creates the pipeline output—is available. If failure then quit.
36–40	The simulator run. See Section 7, “Run the Simulator.”
41–45	A test to ensure that sim_G4plus_vp tool—the pipeline viewer—is available
46–49	Generates the pipeline dump. See Section 8, “View Pipeline.”
50–56	Comments explaining how to read the pipeline output. This shell script does not invoke the tkvp detailed pipeline output. However, see Section 8, “View Pipeline,” for more information on the pipeline output.

The tool flow for this script is shown in Figure 2.

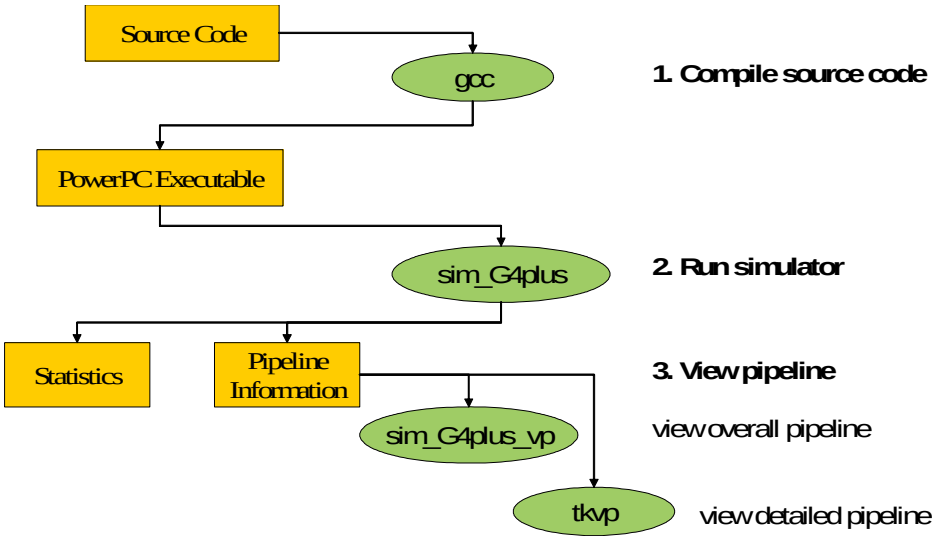


Figure 2. The Tool Flow

6 Compile

Compiling program source code for use with the sim_G4plus simulator may require some additional steps. These special considerations are discussed in the sections that follow.

6.1 Markers

The simulator uses two special markers to determine when detailed pipeline information is generated. Because the process of generating and writing the pipeline information is very slow relative to the simulation speed, using the markers only on the area of code where pipeline results are needed allows the simulation results to show only the results of interest, reducing the overhead of generating and writing pipeline information results to a file.

The two markers are:

- Start code: 0x14000001
- Stop code: 0x14000002

The markers, as shown in Figure 3, are 32-bit data codes inserted into the instruction stream, that is, within the instruction code. These markers are illegal instructions and, if compiled and run on the hardware, will generate an illegal instruction exception. However, when run on the simulator, these markers are used by the simulator to control the starting and stopping of generated pipeline results.

The markers are inserted with the special compiler construct (assembler directive), asm, which inserts the instruction or data designated within quotes into the object code. Thus, the resultant executable will only run on the simulator.

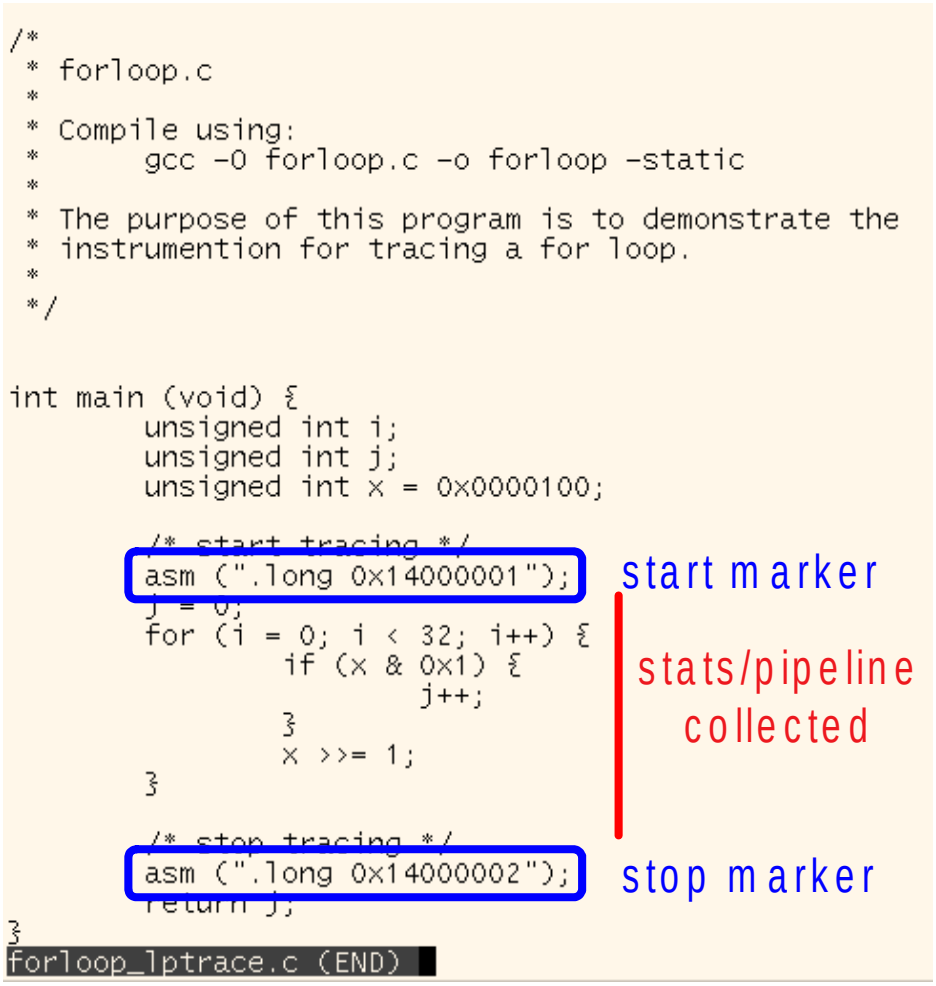


Figure 3. Start and Stop Markers

6.2 Static Libraries

The simulator cannot make calls to library code dynamically (it is not an OS). Therefore, all programs consumed by the simulator must be self-contained, with any needed libraries included statically. Thus, it is necessary to use the -static flag on the GCC compiler command line.

As Figure 4 indicates, the GCC compiler is invoked with the following options:

```
gcc -o forloop_lptrace forloop_lptrace.c -static
```

- gcc calls the compiler tool chain
- -o forloop_lptrace generates an ELF executable with the name forloop_lptrace
- forloop_lptrace.c is the C source file
- -static indicates that all library calls are to be statically loaded into the resultant ELF file forloop_lptrace

As the file command shows below, this is an ELF file that is statically linked:

```
file forloop_lptrace

forloop_lptrace: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV), for
GNU/Linux 2.2.0, statically linked, not stripped
```

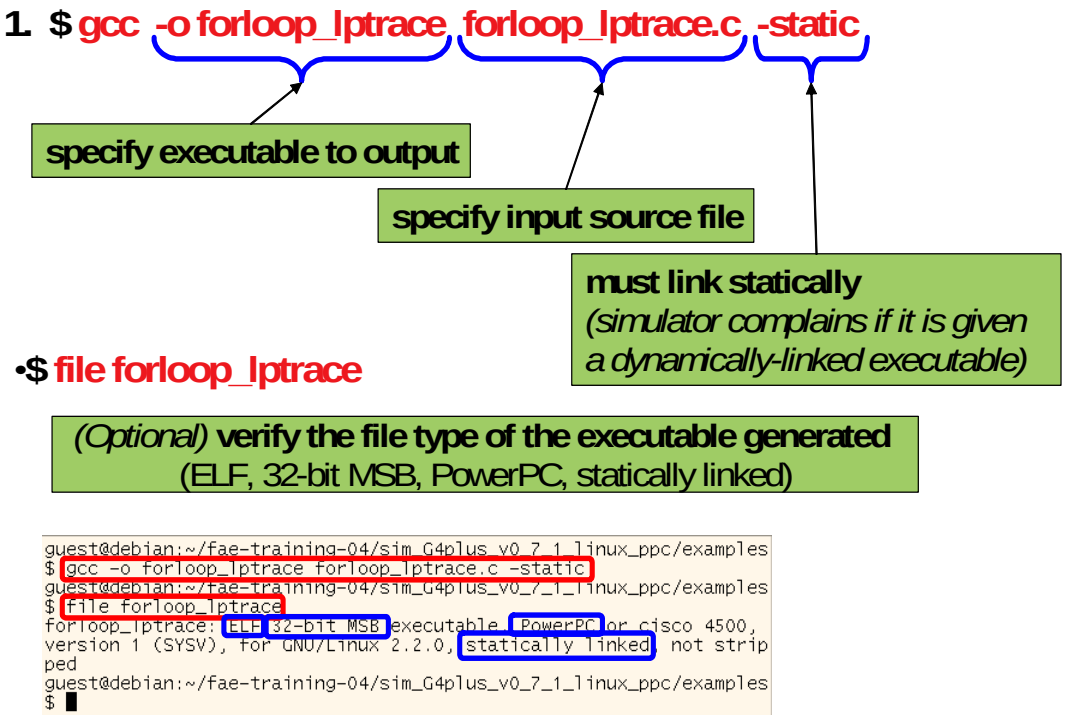


Figure 4. Compiler Options

This completes the first part of the tool flow, the compile phase, shown in Figure 5.

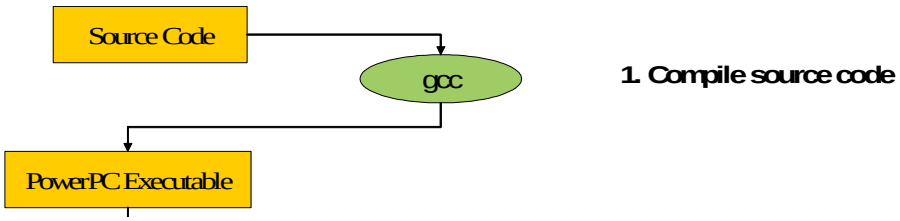


Figure 5. The Source Code Compile Phase

7 Run the Simulator

The simulator has many options; a quick way to view all the available command line options is with the `-h`, help option, as shown in Figure 6.

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ sim_G4plus -h
sim_G4plus expiration date: Sun Nov 14 23:59:59 2004


sim_G4plus v0_7_1

The performance model for the MPC7447A
Copyright 2003 Motorola.
Motorola Confidential Proprietary.

Usage: sim_G4plus [options] [input_tracefile]

Options for controlling simulation runs and output:
  -c <clocks>    How many clock cycles to run (default is seq.max_cycles)
  -d path1:path2 Search path for parameter files
  -e <eventfile> Set the param seq.pevents_file to <eventfile>
  -f <filename>  Use a parameter file
  -p <filename>  Write pipeline info for post-processing to <filename>
  -s param=value Set the runtime parameter 'param' to 'value'
  -a             Run a PowerPC ELF binary in accelerated mode

Options for displaying status (no simulation done):
  -E             Print the registered events, then exit
  -r             Print runtime parameters then exit
  -R             Print runtime parameters with descriptions then exit
  -v             Print version message then exit
  -h             Print help message then exit

guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$
```

Usage

Figure 6. Help Display

The `-v` option will display the version information, as shown in Figure 7.

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ sim_G4plus -v
sim_G4plus expiration date: Sun Nov 14 23:59:59 2004


sim_G4plus v0_7_1

The performance model for the MPC7447A
Copyright 2003 Motorola.
Motorola Confidential Proprietary.

Run on host   : debian
Current Time  : Mon Jun  7 22:49:51 2004

guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$
```

Expiration Date

Version

Figure 7. Version Display

NOTE

There is an expiration date for each version. New versions are released periodically and can be obtained from the Freescale models web page. See Section 9, "References." As new versions are released, old versions are no longer supported. To encourage users to upgrade the simulator, each version will stop running after its expiration date.

Two command line options are available to describe all of the parameters; `-r` provides a summary listing, while `-R` provides a listing that contains more detail about each parameter.

As shown in Figure 8, the parameters and their default or current values are displayed. For example, the bpu.enable_link_stack parameter is true, which will generate the detailed branch and link stack data that can be viewed by tkvp, the detailed pipeline viewer described in Section 8.2, "Detailed Pipeline Viewer."

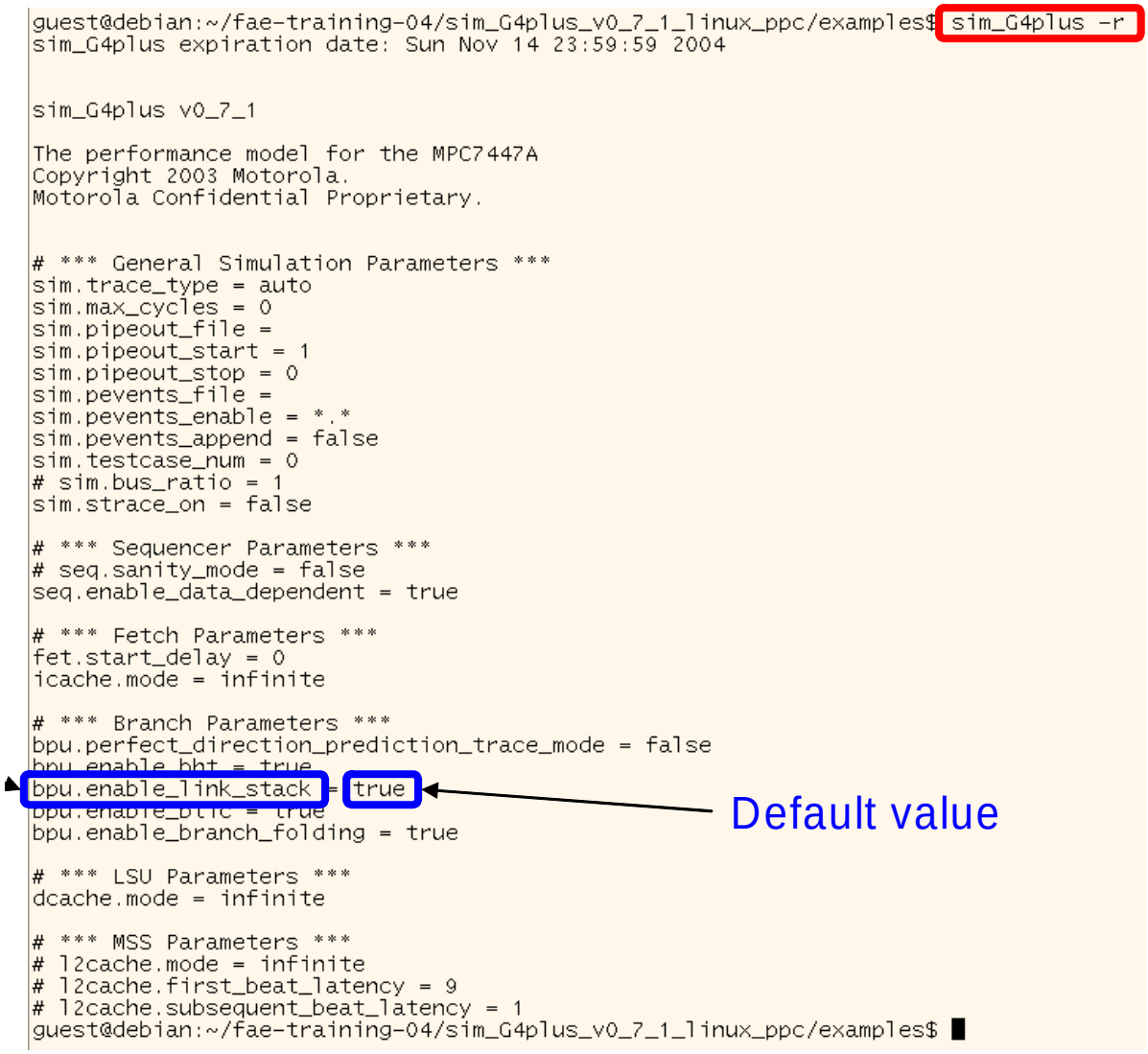


Figure 8. Simulation Parameters

The example in Figure 9 will run the simulator in accelerated mode, -a, and generate a pipeline output file, -p. The use of the two parameters, -a and -p, is described in Figure 6 and below:

- -a Run a PowerPC ELF binary in accelerated mode
- -p <filename> Write pipeline info for post-processing to <filename>

As shown in Figure 9, the simulator can run the forloop program in accelerated mode. No pipeline statistics are gathered until the start marker is encountered. Between the start marker and the stop marker, pipeline statistics are gathered. After the stop marker, no more pipeline statistics are gathered until, and unless, another start marker is encountered.

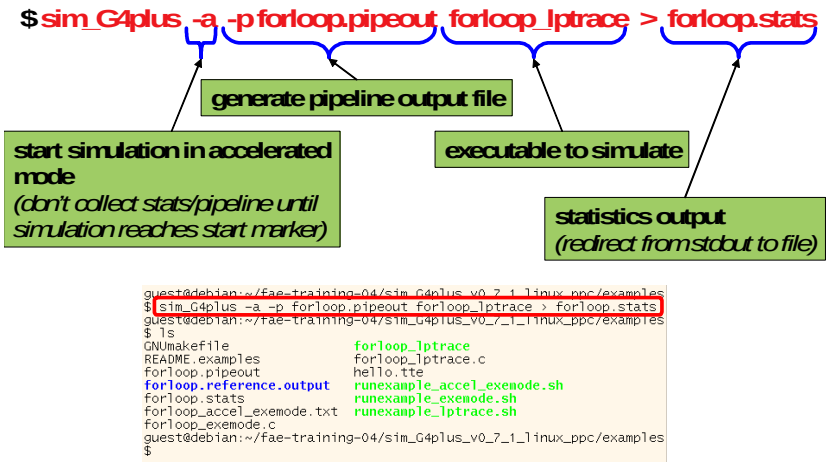


Figure 9. Simulate in Accelerated Mode

Although the `ls` command shown in Figure 9 lists all the files in the work directory, by using the full list command, `ls -l`, and the date, the user can determine which files were newly created.

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ ls -l
total 1080
-rw-r--r--    1 guest    guest         460 Mar  8 13:16 GNUmakefile
-rw-r--r--    1 guest    guest        3057 Mar  8 13:16 README.examples
-rw-r--r--    1 guest    guest      960812 Jul 21 17:59 forloop.pipeout
drwxr-xr-x    2 guest    guest        4096 Apr 21 18:59 forloop.reference.output
-rw-r--r--    1 guest    guest       26700 Jul 19 22:49 forloop.state
-rw-r--r--    1 guest    guest       26700 Jul 21 17:59 forloop.stats
-rw-r--r--    1 guest    guest         365 Mar  8 13:16 forloop_exemode.c
-rw-r--r--    1 guest    guest         460 Mar  8 13:16 forloop_lptrace.c
-rw-r--r--    1 guest    guest       48196 Mar  8 13:16 hello.tte
-rwxr-xr-x    1 guest    guest        1656 Apr 20 20:46 runexample_accel_exemode.sh
-rwxr-xr-x    1 guest    guest        1574 Apr 20 20:46 runexample_exemode.sh
-rwxr-xr-x    1 guest    guest        1816 Apr 21 19:59 runexample_lptrace.sh
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$
```

This listing was made on July 21, so the only two files created on this day are forloop.stats and forloop.pipeout.

- forloop.pipeout is generated via the -p parameter
- forloop.stats will always be generated

The next series of screen shots (Figure 10, Figure 11, and Figure 12), captured on November 14, displays the contents of the forloop.stats file. The statistics file includes the following information:

- Simulation parameters
- Simulation messages
- Output statistics
- Number of instructions
- Number of cycles
- Instructions per cycle

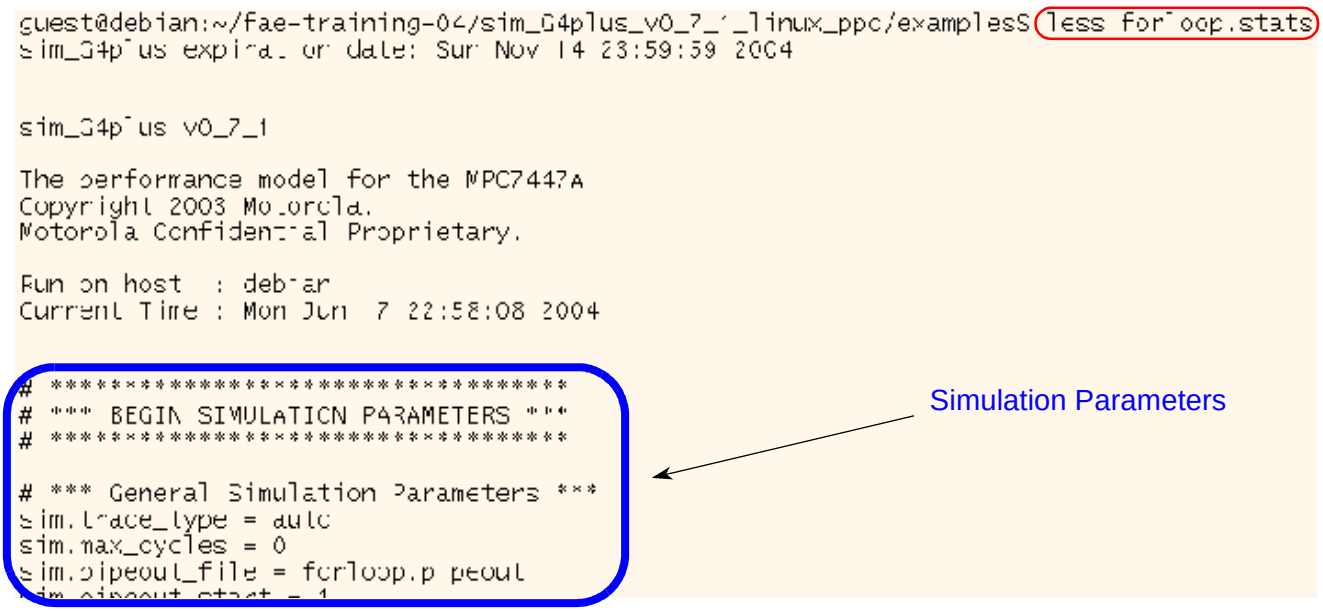


Figure 10. Pipeline.stats Display: Part 1 of 3

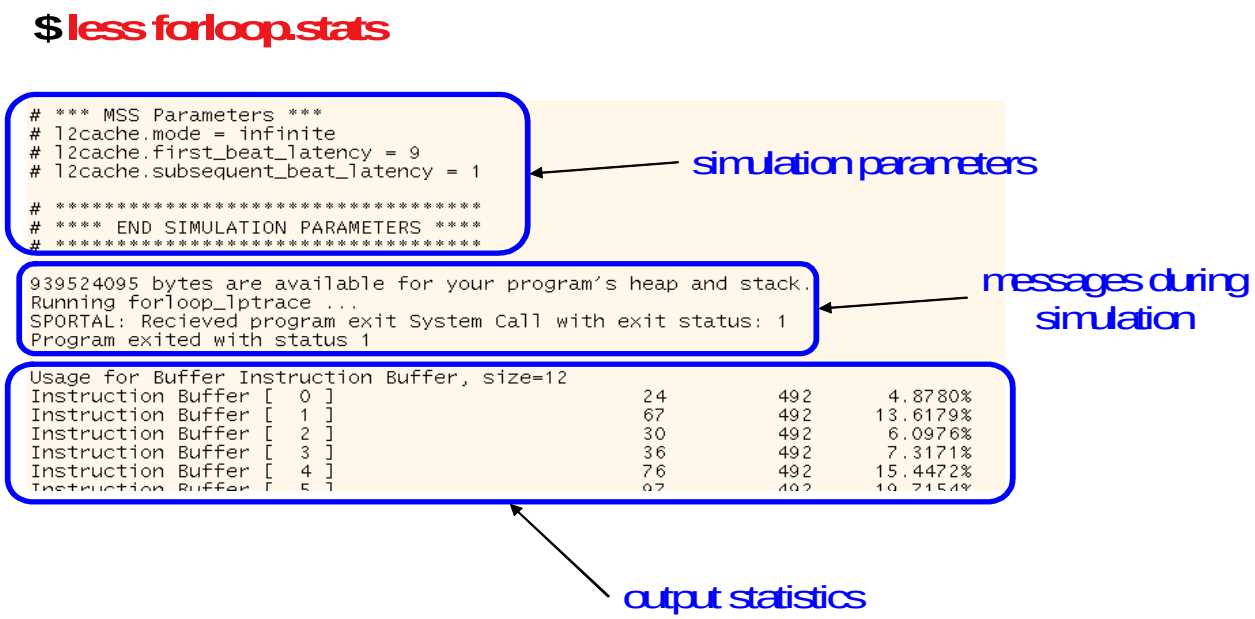


Figure 11. Pipeline.stats Display: Part 2 of 3

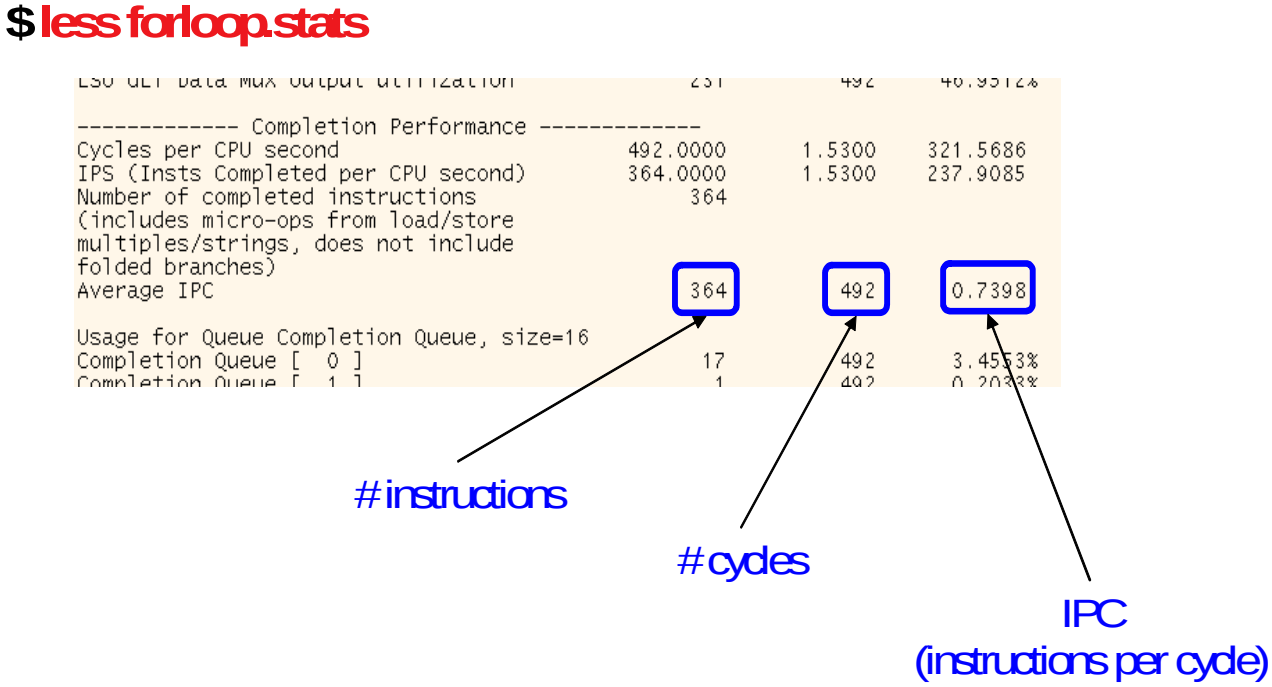


Figure 12. Pipeline.stats Display: Part 3 of 3

Figure 13 summarizes this section, in which the compiled executable is simulated, resulting in two files: the statistics captured and the pipeline information generated.

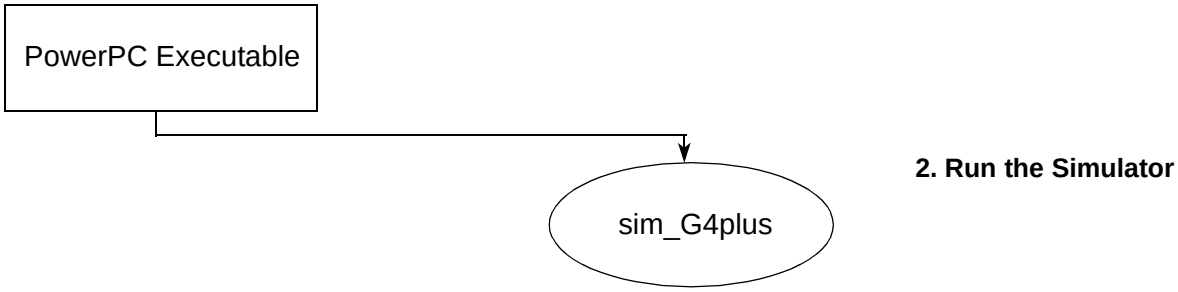


Figure 13. Simulate the Executable

8 View Pipeline

The pipeline output file is a text file. It can record the status of the processor for each clock cycle that is simulated.

Thus, this file can become quite large, consisting of 16,578 lines for this small example.

```
wc forloop.pipeout

 16578  75175 960812 forloop.pipeout
```

The pipeline results file is a log of detailed state information and is difficult to read and understand. A pipeline viewer should be used to display the information in a more useful and visually meaningful way.

8.1 Standard Pipeline Viewer

The standard pipeline viewer is the sim_G4plus_vp program. The GUI version of this program requires the Perl/Tk module to be installed. If Perl/Tk is not available to the user, the -n (--notk) option can be used to output a text display for a view of the pipeline results.

This program has many options available and the -h parameter will display them as shown in Figure 14.

```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples$ sim_G4plus_vp -h

   Usage /home/guest/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/bin/spftools/tksnp.pl [options
] <snapshot_def_file> <spf_file>

   Where options are one of:
      -f | --firstcycle <num> /* First cycle to display (Non-TK mode only) */
      -e | --endcycle <num>   /* Last cycle to display */
      -s | --showcols <list>  /* Colon-seperated list columns to display */
      -c | --cycwidth <num>   /* Number of characters for the cycle column */
      -m | --mapwidth <num>   /* Number of characters for the map column */
      -r | --rows <num>       /* Number of lines per page */
      -d | --deffile <file>   /* Used the deffile (instead of on cmd line) */
      -i | --ignoresize       /* In non-Tk mode, ignore the size of the
                                 terminal in which we were invoked */
      -n | --notk             /* Don't use the TK interface */
      -p | --printall         /* Print all cycles on a page in Non-TK mode.
                                 Default is to try to only print the number
                                 of rows */
      -D | --Delimiter <s>    /* Output using the specified
                                 string as the delimiter between columns. */
      -w | --warnings         /* Give warnings if we can't find a symbol
                                 in the snapshot file */
      -l | --list             /* List all known columns, then exit
                                 NOTE: This option requires both a
                                 snapshot_def_file AND an spf_file.  This
                                 is the only way that the columns for
                                 a specific spf can be properly displayed */
      -F | --Font <font>      /* Use the specified font. */
      -P | --P <Program>      /* Run Program on the input file */
      -h | --help             /* Print this message, then exit */

   The snapshot_def_file should be provided when this script
   is delivered.  The spf_file is the pipline output of the simulation.
```

Figure 14. sim_G4plus_vp Options

The simplest command to view the pipeline results is as follows:

```
sim_G4plus_vp forloop.pipeout &
```

The '&' character allows the viewer to run in the background, while more commands run in the initiating window. See Figure 15.

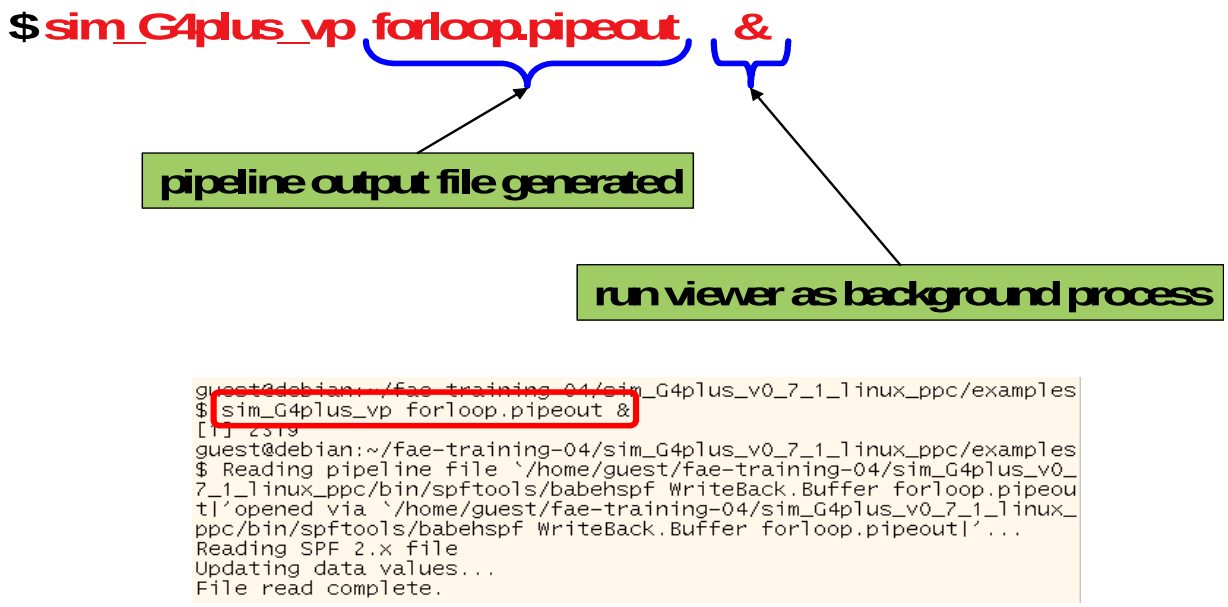


Figure 15. Start the Pipeline Viewer without Arguments

A separate display window will open, displaying the pipeline as shown in Figure 16.

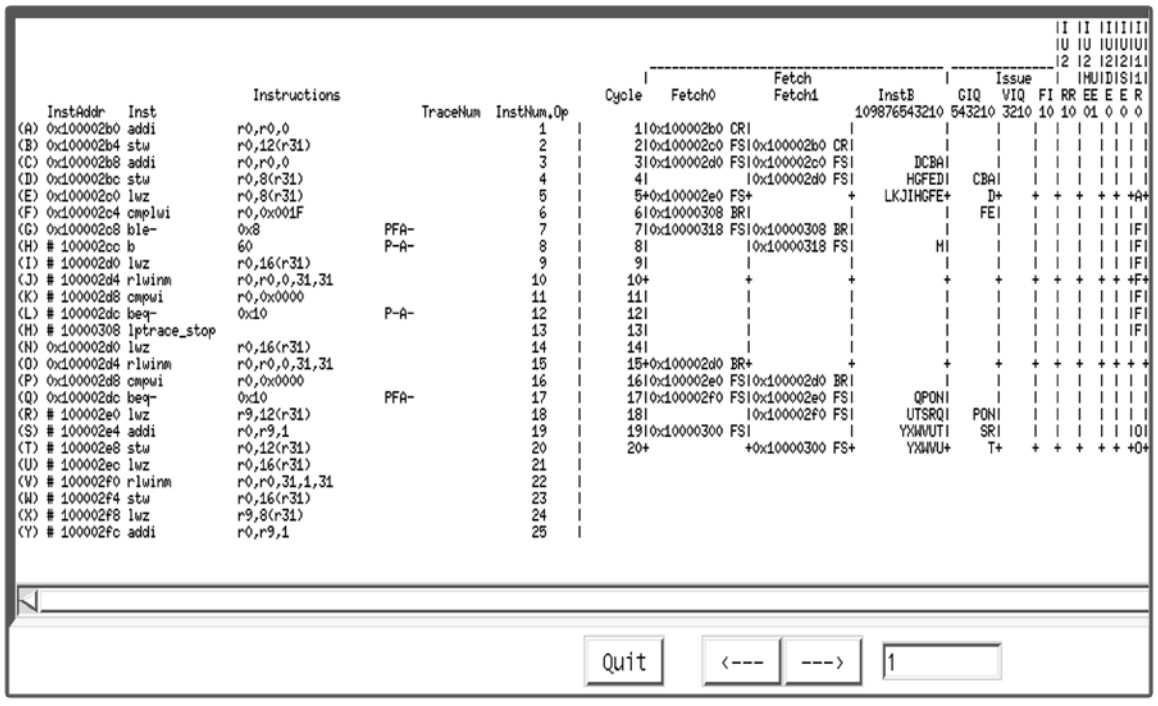


Figure 16. Pipeline Display Window

The pipeline display window is very wide. As seen above, the right section of the window continues on, and scrolls to the right and left with the slide bar on the bottom of the window. The window displays the pipeline execution history and is used to follow pipeline activity with the left and right arrow buttons. The number field, the box displaying 1 at the bottom right hand part of the screen shown in Figure 16, indicates the current cycle number in the pipeline execution history. The ‘Quit’ button will end the pipeline display.

Each line represents one cycle of activity.

In order to understand the pipeline output effectively, it is useful to review the MPC7447A block diagram as shown in Figure 17.

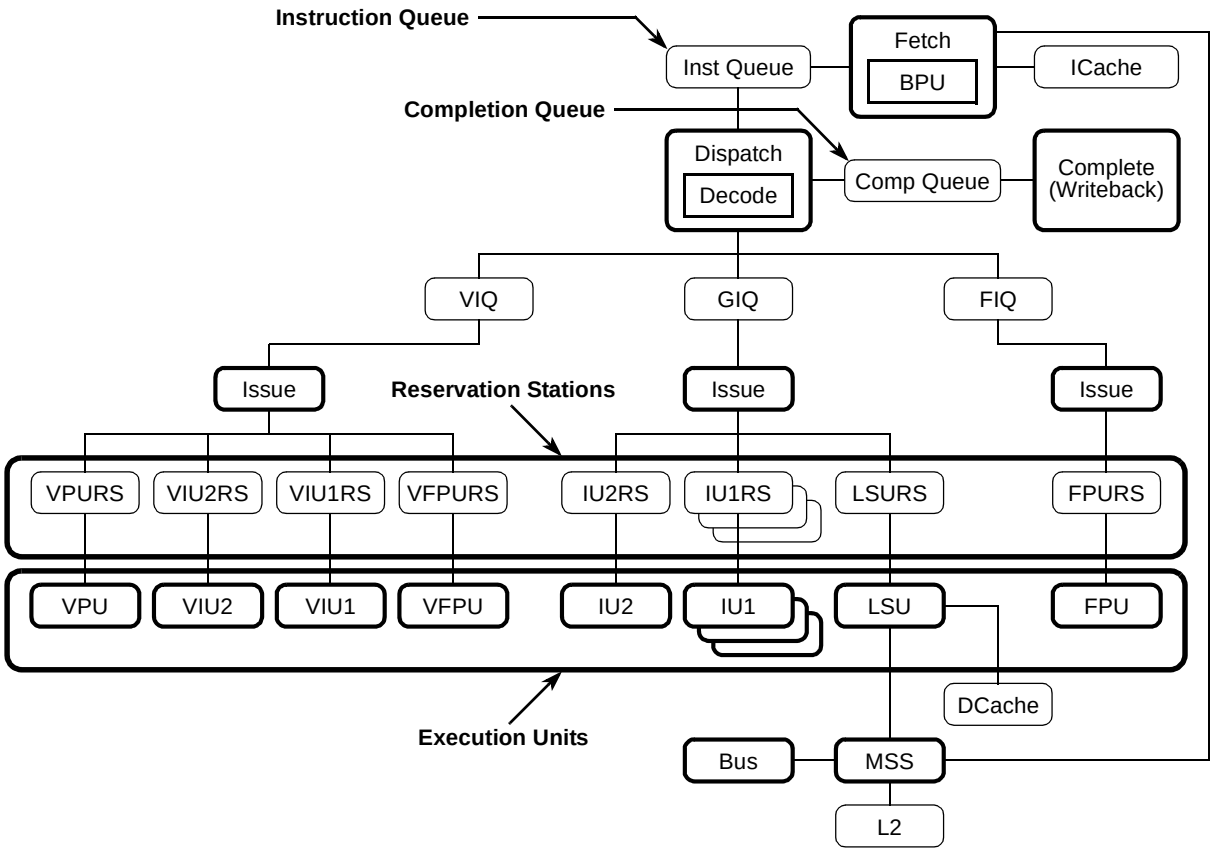


Figure 17. MPC7447A Block Diagram

A more detailed description of the pipeline display is given below and illustrated in Figure 18 through Figure 24. Processor units include a fetch unit, an instruction queue, reservation stations, and various execution units. Each major unit corresponds to a section of the pipeline viewer display. In Figure 18 the legend shows the instructions as they are encountered in the code, the pipeline shows the flow of these instructions through the various units from Figure 17, and the controls are as described after Figure 16.

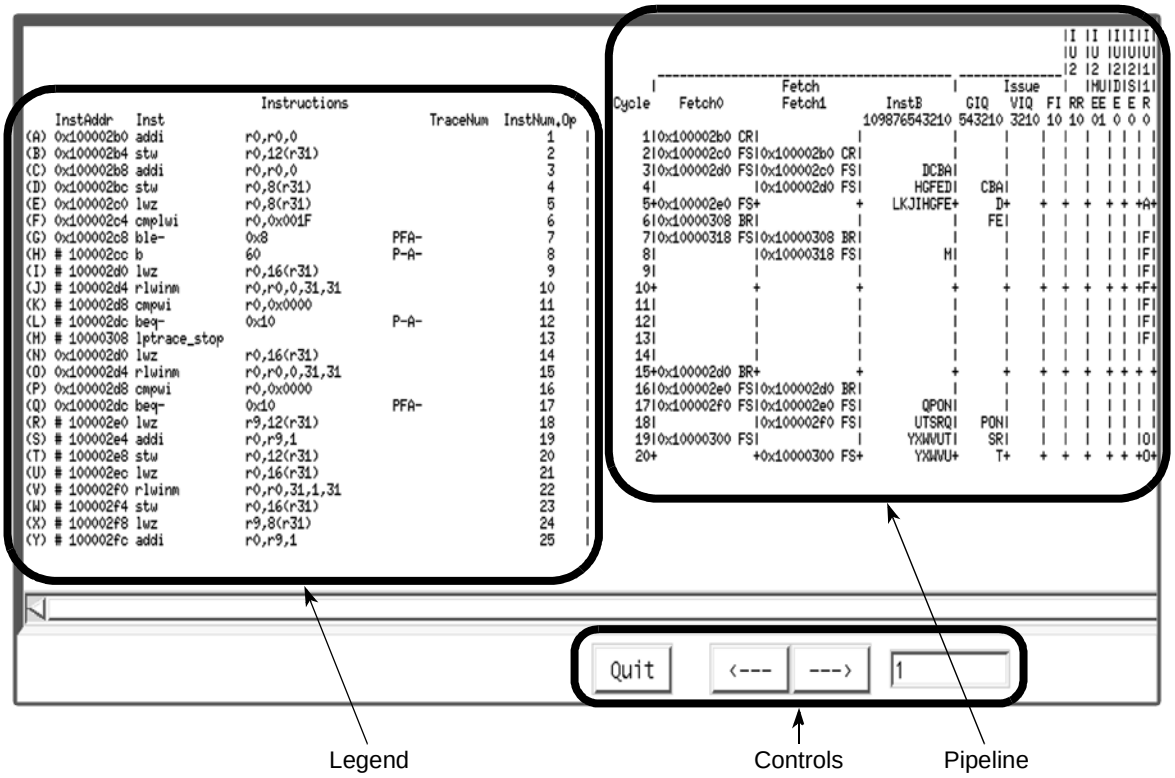


Figure 18. Overall Pipeline Viewer Display

The next figures (Figure 19 through Figure 24) show the details of this pipeline display.

Figure 19 is the detail of the legend section. Each instruction, as it is encountered, is assigned a sequential one letter designator. This designator follows this instruction during its life in the pipeline. Thus, there are 26 times 2 (capital and small letters), or 52 unique instructions at any time in the pipeline display. If there are more than 52 instructions in the pipeline at one time, then there will be duplicate designators; however, it should be obvious which one is being referenced. The hex value of the instructions and the disassembled mnemonic and arguments are shown.

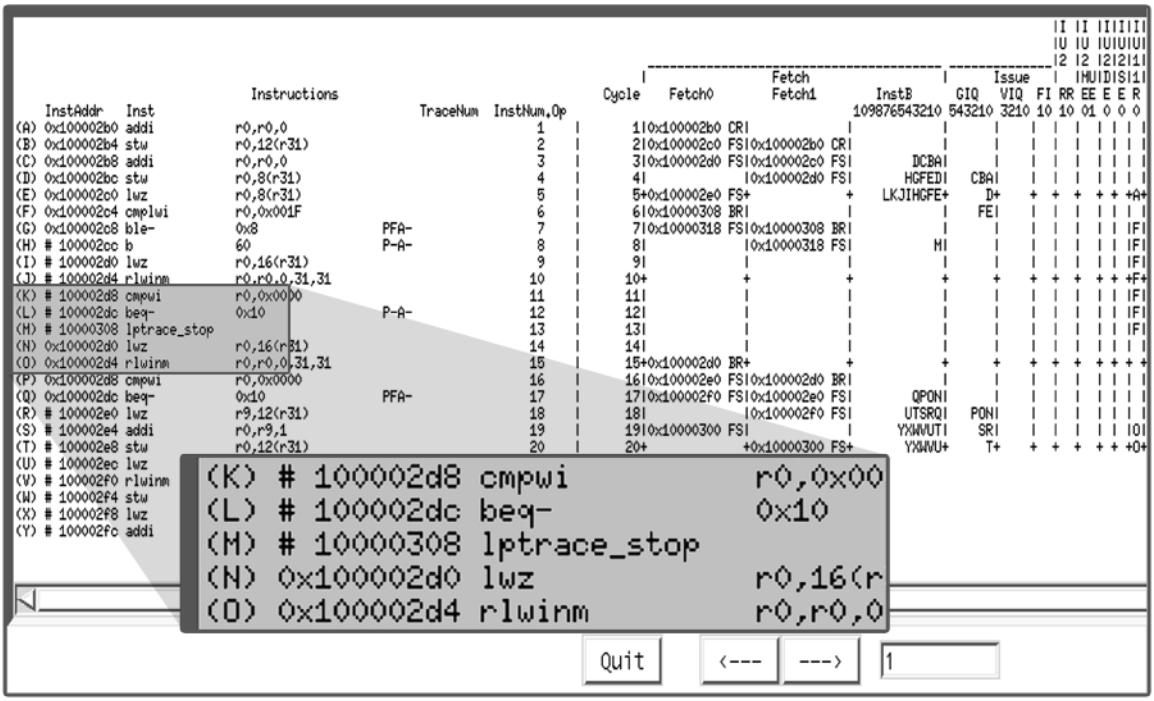


Figure 19. Pipeline, Instruction Fetch

Figure 20 shows the cycle count. As each cycle passes, the fetch queues will begin to fill and instructions will march through the various execution states.

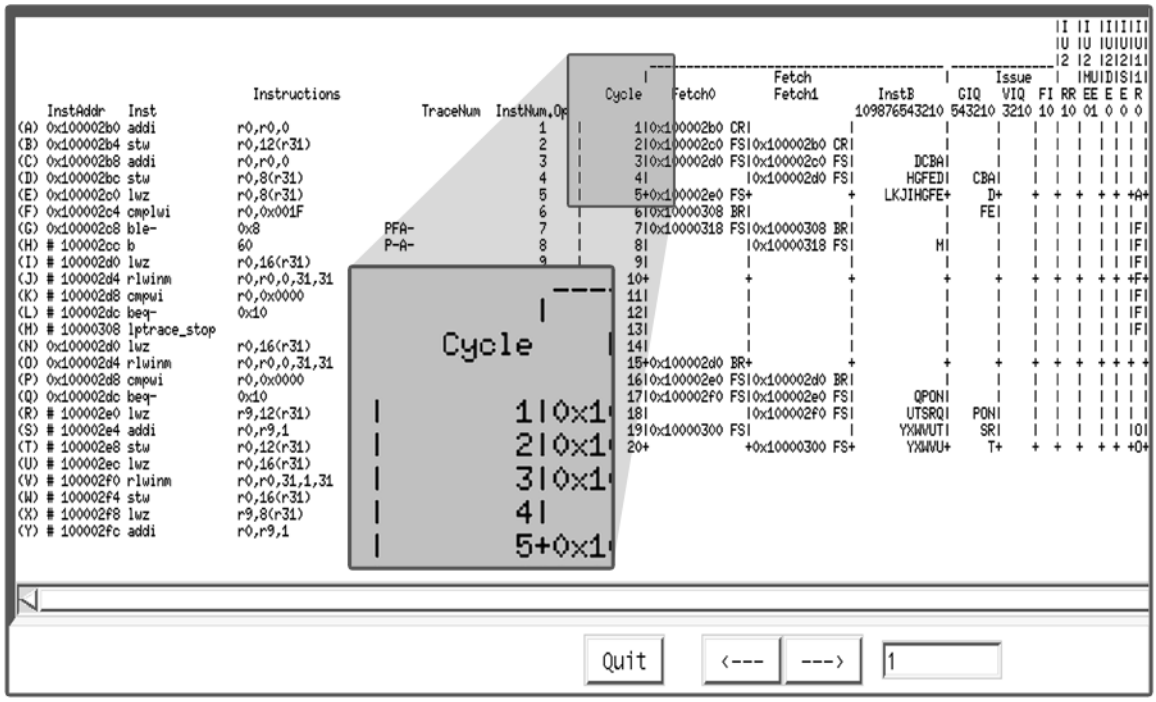


Figure 20. Pipeline Cycle

Figure 21 shows the one-character designators for the instructions as they are allocated to the various queues.

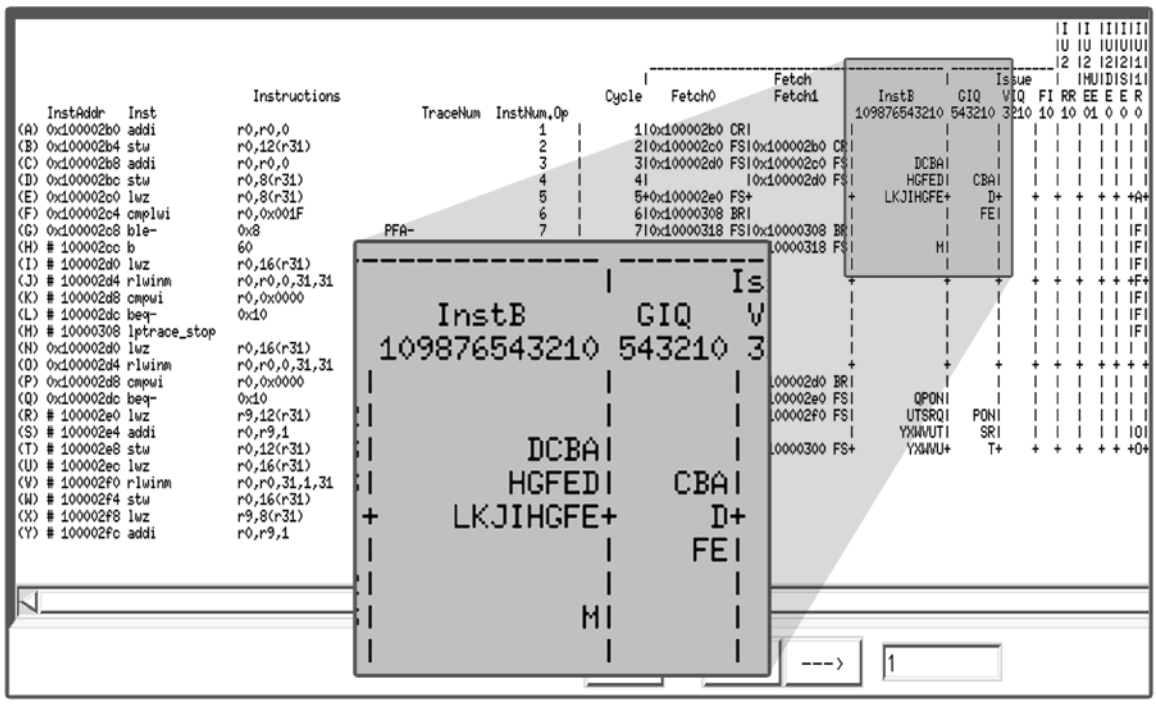


Figure 21. Pipeline Instruction Queue

Figure 22 shows the instruction queue and the issue queue. Not shown are the completion queue and writeback queues.

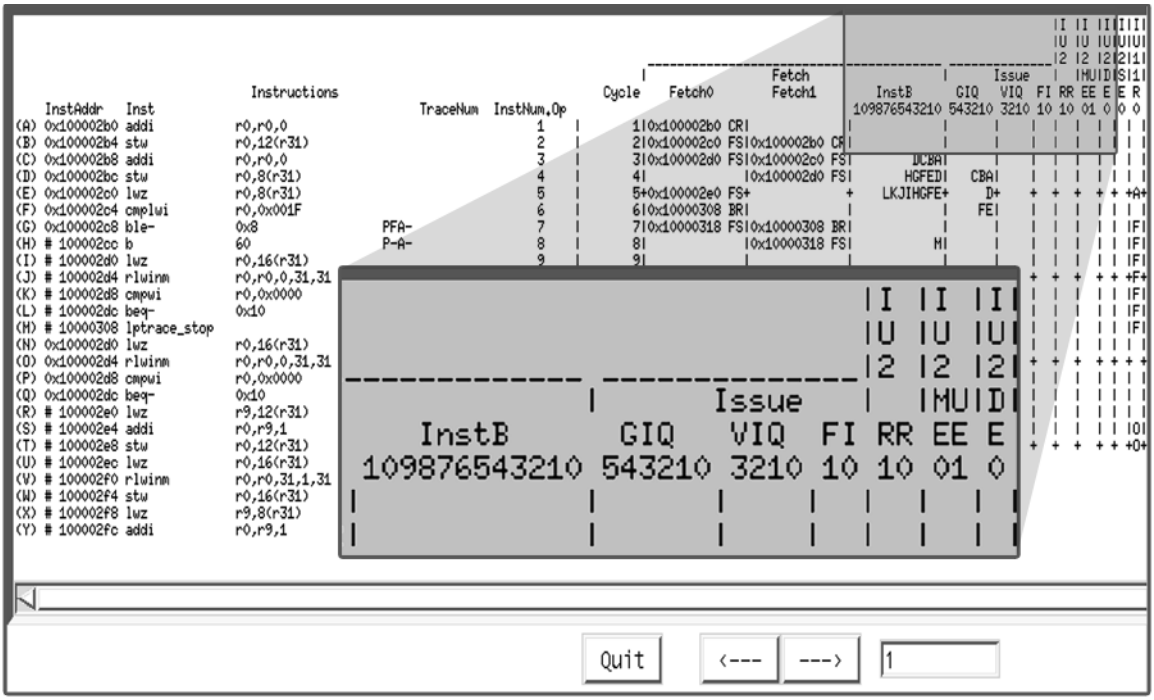


Figure 22. Pipeline Issue Queue

Figure 23 shows the complete pipeline section with each section clearly delineated: the fetch, instruction, issue, execution, completion, and writeback queues. More details describing the sections of the pipeline viewer display are given in the Freescale application note AN2750, *Analysis and Optimization of Code with sim_G4plus on Genesi Pegasos II*.

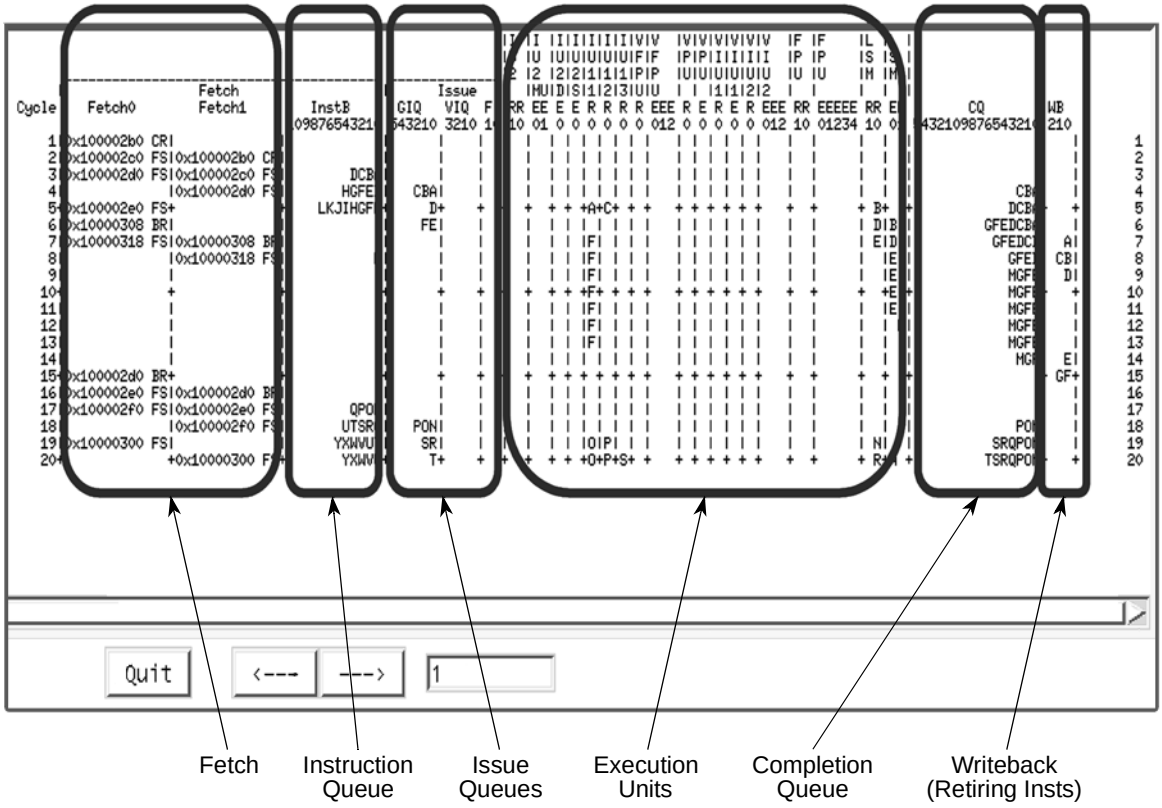


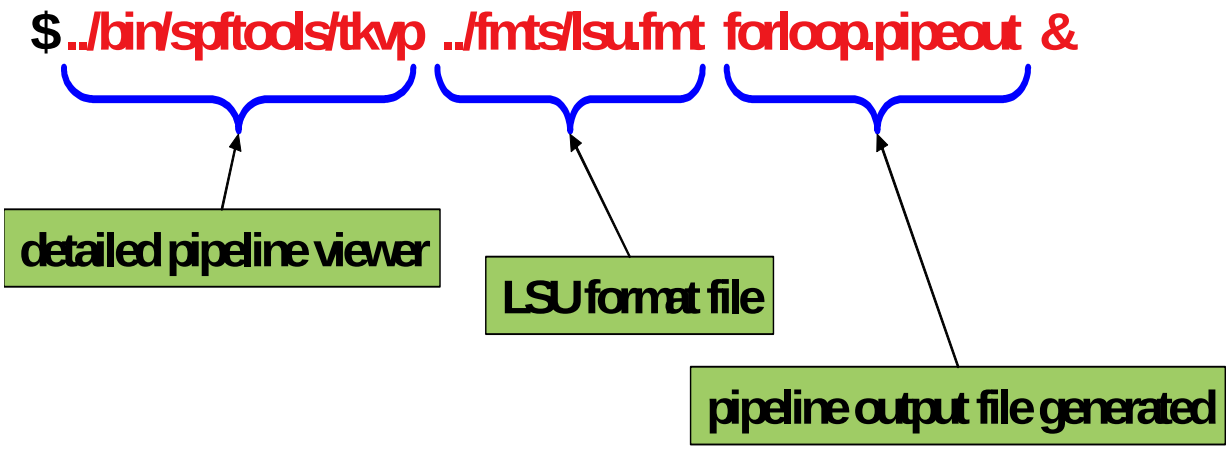
Figure 23. Overall Pipeline Viewer Display

8.2 Detailed Pipeline Viewer

The tkvp pipeline viewing tool provides more detailed information on a per-cycle basis for specific units in the core pipeline. tkvp must be given a format file and a pipeline output file. tkvp can be called with any of these combinations of detailed files.

- The fetch unit—tkvp fet.fmt pipeout
- The branch processing unit—tkvp bpu.fmt pipeout
- The load store unit—tkvp lsu.fmt pipeout
- The memory sub-system—tkvp mss.fmt pipeout

Figure 24 shows the invocation of the detail display of the load store unit, LSU. The two input files are required, ../fmts/lsu.fmt and forloop.pipeout. The format file, lsu.fmt, resides in the fmts directory, which is up one directory from the examples directory.




```
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples
$ ../bin/spftools/tkvp ../fmts/lsu.fmt forloop.pipeout &
[1] 2364
guest@debian:~/fae-training-04/sim_G4plus_v0_7_1_linux_ppc/examples
$ Reading pipeline file `forloop.pipeout' opened via `<forloop.pipeo
ut'...
Reading SPF 2.x file
Updating data values...
File read complete.
```

Figure 24. Command to View the LSU Load Store Unit Pipeline Viewer

Figure 25 shows the LSU Load Store Unit in detail.

```
$ ../bin/spftools/tkvp   ../fmts/lsu.fmt   forloop.pipeout   &
```

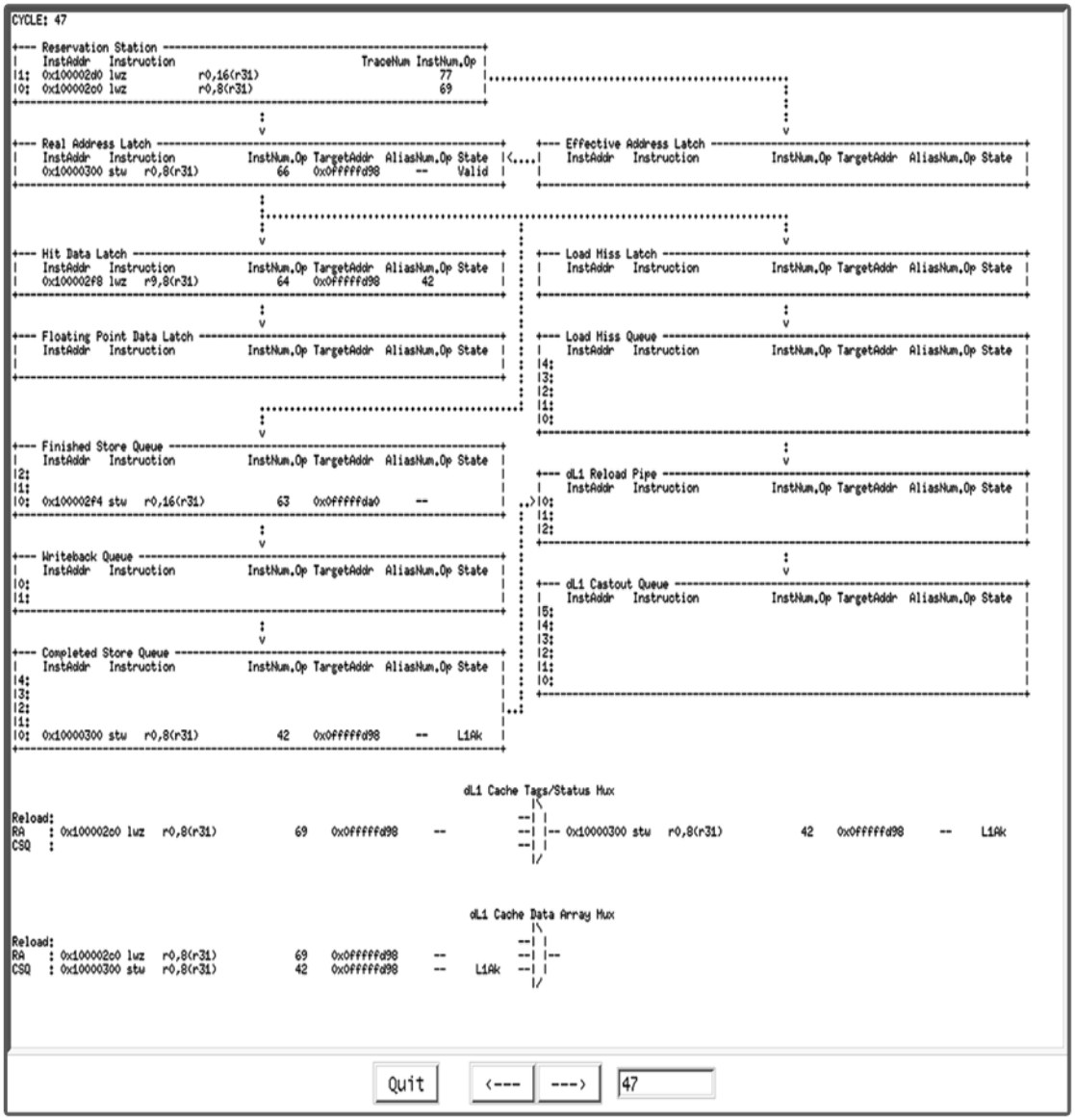


Figure 25. Detailed Pipeline Viewer: LSU Load Store Unit

Figure 26 shows the branch processing unit in detail.

```
$ ../bin/spftools/tkvp   ../fmts/bpu.fmt   forloop.pipeout   &
```

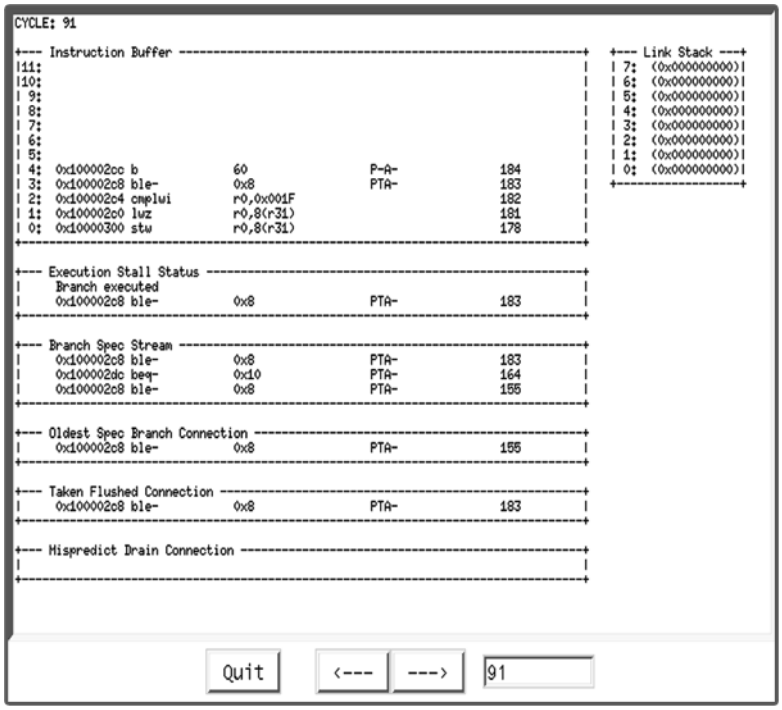


Figure 26. Detailed Pipeline Viewer: BPU Branch Processing Unit

This completes the last phase of this paper as shown in Figure 27.

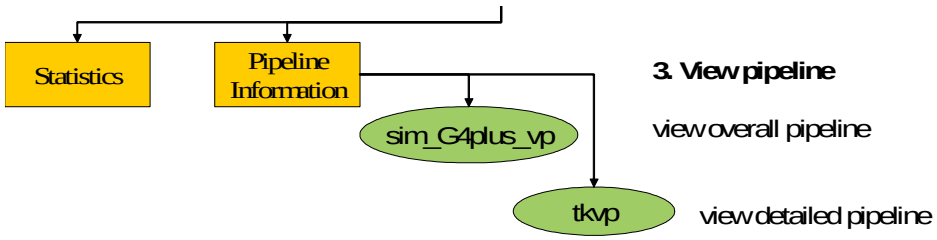


Figure 27. View the Pipeline Output

9 References

The following documents describe the various applications of the Genesi Pegasos II system or are references for the systems.

1. Freescale application note AN2666, *Genesi Pegasos II Setup*
2. Freescale application note AN2736, *Genesi Pegasos II Boot Options*
3. Freescale application note AN2738, *Genesi Pegasos II Firmware*
4. Freescale application note AN2739, *Genesi Pegasos II Debian Linux*
5. Freescale application note AN2751, *Genesi Pegasos II Yellow Dog Linux*
6. Freescale application note AN2743, *Software Analysis on Genesi Pegasos II Using PMON and AltiVec*

7. Freescale application note AN2744, *PMON Module—An Example of Writing Kernel Module Code for Debian 2.6 on Genesi Pegasos II*
8. Freescale application note AN2748, *Genesi Pegasos II Kernel and NFS facility*
9. Freescale application note AN2750, *Analysis and Optimization of Code with sim_G4plus*
10. *sim_G4plus v0.7 Cycle-Accurate Simulator User's Guide* Rev 1.5
11. *RISC Microprocessor Family User's Manual* (MPC7450)

For assistance or answers to any question on the information that is presented in this document, send an e-mail to risc10@freescale.com.

To request a sim_G4plus v0.8 simulator or documentation download, follow these steps:

1. Go to freescale.com/powerpc.
2. Click on the 'More' field of the "Timing & Performance Model" section (middle of the page).
3. Find SimG4+ from the list and choose your platform (currently there are Linux PowerPC and Linux x86 versions with a Solaris version on the way).
4. Make sure to enter valid FAE or Freescale contact information in the form that is presented. Filling in the form and submitting it sends e-mail to Freescale for approval.
5. Assuming you are approved, you will receive an e-mail from your FAE or Freescale contact with further download instructions.

10 Document Revision History

Table 2 provides a revision history for this application note.

Table 2. Document Revision History

Revision Number	Date	Change(s)
1	01/04/2005	Updated Figure 16 through Figure 23, Figure 25, and Figure 26. Minor language edits.
0	09/15/2004	Initial release

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-0047 Japan
0120 191014
+81 3 3440 3569
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@
hibbertgroup.com

AN2749
Rev. 1
01/2005