

## The Cmpware CMP-DK Cell B.E. SPU Scheduling Tool (Version 1.0 for Eclipse 3.2)

Cmpware, Inc.

### Introduction

The *Cmpware Configurable Multiprocessor Development Kit (CMP-DK) for the Cell B.E.* is a multicore simulation and software development environment for the Sony / Toshiba / I.B.M. Cell Broadband Engine (B.E.) processor. It provides fast and efficient simulation of the Cell B.E. combined with an interactive, display-rich environment that permits large amounts of information to be displayed in a fast, simple and intuitive format. Such capabilities are essential in analyzing the complex behavior of multicore systems.

One component of the *Cmpware CMP-DK for the Cell B.E.* is a scheduling tool for the Synergistic Processing Unit (SPU). The Cell B.E. contains eight SPU processors which are used to do the bulk of the computation in most Cell B.E. applications. The SPU processor is a powerful Digital Signal Processor (DSP) like device which contains four floating point Arithmetic and Logic Units (ALUs) operating in Single Instruction, Multiple Data (SIMD) fashion.

This collection of processors can provide significant improvements in performance over traditional single processor approaches. Because of their DSP-like architecture, however, the ordering of the instructions executed by the SPUs can have a significant effect on performance.

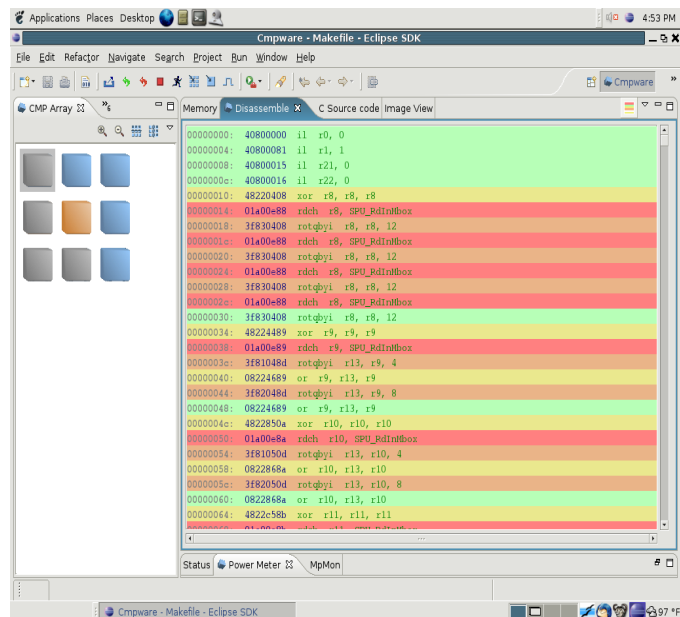


Figure 1: The *Cmpware CMP-DK for the Cell B.E.*



It is not unusual to see a 5x or more improvement in performance from hand-tuning an SPU program. In order to achieve the maximum performance from the Cell B.E., an understanding of SPU scheduling is essential.

The *Cmpware CMP-DK for the Cell B.E.* contains an SPU scheduling tool as a part of the overall software development suite. *Cmpware* has made this part of the *CMP-DK* available to Cell B.E. programmers as a free Eclipse plug-in. This tool permits fast and easy analysis of SPU code in a highly intuitive, interactive, graphical fashion. If you are seeking to get the most of out your Cell B.E. applications, the *Cmpware SPU Scheduling Tool* is for you.

## Installation

The *Cmpware SPU Scheduling Tool* is a standard *Eclipse* plugin and may be installed using the standard procedure for installing Eclipse plugins. The Eclipse update site URL for the *Cmpware SPU Scheduling Tool* is:

**<http://www.cmpware.com/sputool/>**

Select the menu item **Eclipse Help --> Software Updates --> Find and Install ...** from the main Eclipse menu and following the prompts to use this URL to install the *Cmpware SPU Scheduling Tool*. More information on installing Eclipse plugins can be found in the *Cmpware CMP-DK for the Cell B.E.* tutorial at:

**[http://www.cmpware.com/Docs/Cmpware\\_3\\_CellBE.pdf](http://www.cmpware.com/Docs/Cmpware_3_CellBE.pdf)**

More detailed information on Eclipse, Eclipse plugins and using update sites to install Eclipse plugins can be found at the Eclipse web site at: **<http://www.eclipse.org/>**

## Scheduling the Cell B.E. SPU

Perhaps the most important issue in improving SPU performance through instruction scheduling involves *data dependencies*. Because of the pipelining of the SPU, results from one operation may not be available for several cycles after an instruction executes. Instructions attempting to use these results will *stall*, often for six or as many as thirteen cycles. This can have a dramatic impact on performance.

One problem with attempting to write code for the SPU is that the numbers of cycles an instruction will stall in such a data dependency situation will depend on the instructions being executed. Some instructions produce no stalling at all, some two cycles, others



six. The other factor is the register usage. Keeping track of registers across as many as a dozen instructions can be challenging, even when writing small programs. How to rearrange code, or even how to implement a particular algorithm can depend heavily on the particular mix of instructions and their ordering.

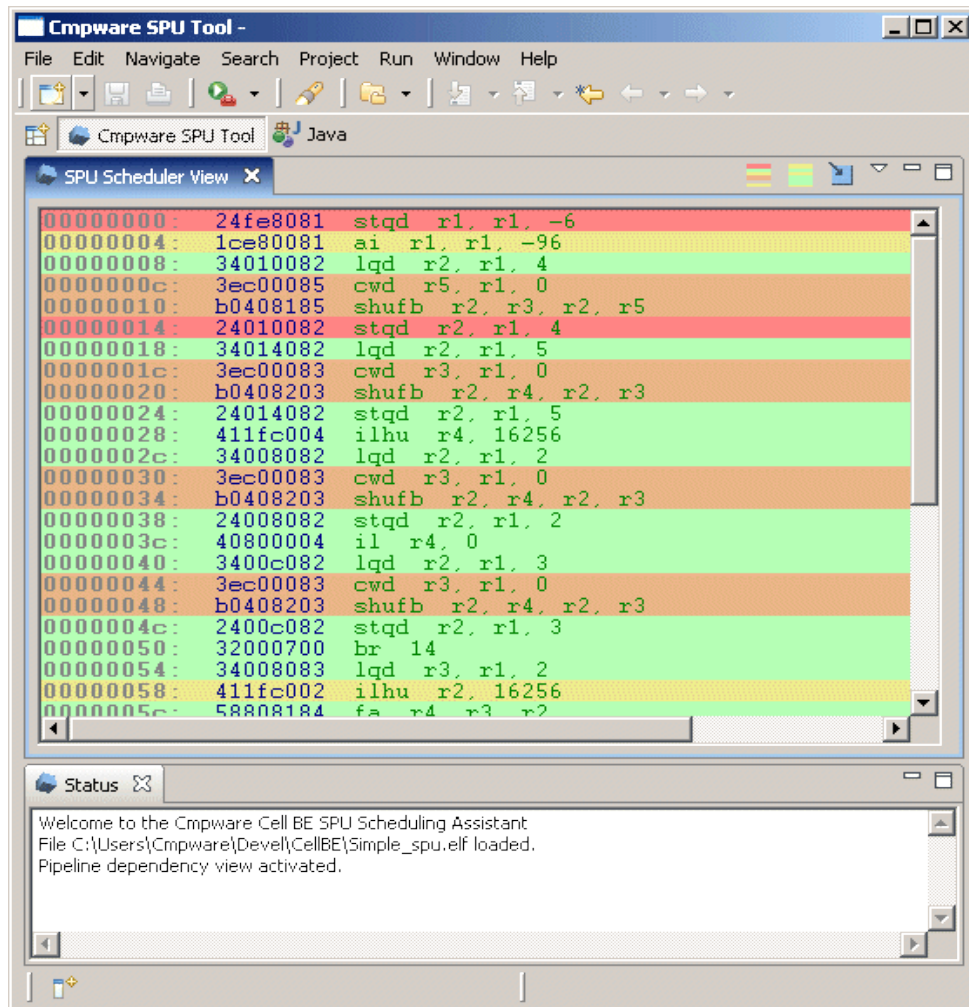


Figure 2: The SPU pipeline data dependency view

In addition to the instruction scheduling for data dependencies, the SPU has a second performance enhancing architectural feature which can double performance in some cases, but also further complicates instruction scheduling. The SPU ALU has two instruction pipelines which run in parallel, permitting two instructions to be executed in each clock cycle. Unfortunately, these two pipelines are not identical. One pipeline handles (in general) memory access functions while the second handles arithmetic and



logical operations.

On each clock cycle, two instructions can be executed, provided they are the correct type of instruction and aligned at the correct address. Keeping track of which instructions go in which pipeline and which instruction is at a given address can be daunting. The instruction pipeline issue scheduling is also particularly sensitive. The addition or removal of a single instruction can, in some cases, cause a 2x change in performance. Combining these two scheduling activities can be extremely difficult, even for experienced programmers.

The *Cmpware SPU Scheduling Tool* addresses the problems associated with scheduling SPU code in several ways. First, the scheduling process is split into two separate tasks: the data dependency stalls are handled separately from the dual pipeline issue scheduling. But because these two activities are interdependent, the *Cmpware SPU Scheduling Tool* features rapid display of data for both scheduling modes. The click of a single button toggles the display from one optimization phase to the other, permitting fast, interactive experimentation of different scheduling approaches. Finally, the displays themselves provide scheduling information in a simple and easy to use format. A quick glance indicates not only exactly where performance bottlenecks, but their severity.

## The Cmpware SPU Scheduling Tool

The *Cmpware SPU Scheduling Tool* is a part of the commercial version of the *Cmpware CMP-DK* and is made available as a free Eclipse plug-in. Figure 2 shows the complete user interface. The only controls are three buttons on the top of the main window. These are used to load data files and display scheduling information as described in Figure 3.

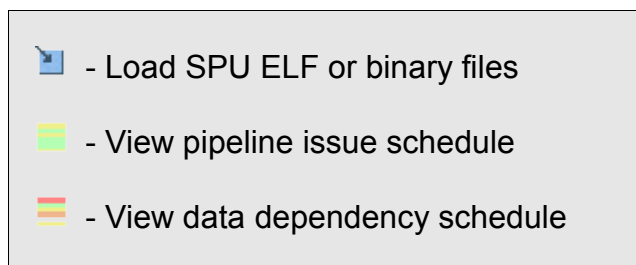





Figure 3: The SPU Scheduling Tool controls.



Once an SPU binary file (either in ELF or raw binary) is loaded into the viewer using the **File Load** button (  ), the disassembly for this SPU code will be displayed in the main window. By clicking on either the **Pipeline Instruction Issue View** button (  ) or the **Data Dependency View** button (  ), each instruction in the disassembly will be colored according to their schedule properties. In general, the coloring goes from green (good) to red (bad). Figure 4 shows the actual colors and cycle times for the **Data Dependency View**.


The goal in optimizing SPU software is to achieve a 'perfect' schedule with all instructions colored in green. Depending on your application, it may not be necessary to achieve the maximum performance. In other cases, some code may not be in critical loops and may not require optimization. Profiling with a tool such as the *Cmpware CMP-DK* will indicate which sections of SPU code are most often used and will help assist in deciding which portion of the SPU code will benefit most from schedule optimizations.



In the **Data Dependency View**, SPU instructions are colored according to the type of instruction and the registers used by that instruction. If a source register used by an instruction has recently been written with a result from a previous instruction, that register may not yet be ready to be used. The **Data Dependency View** will color an instruction to indicate how many cycles the instruction will have to stall before all source data in input registers is available. Figure 4 gives the cycle stall values associated with each color in the display.

Figure 4: The color codes for the SPU data dependency stall view.

Improving SPU performance involves reducing the number of cycles stalled by re-arranging instructions. Typically, instructions which do not contain registers used by subsequent instructions can be inserted, supplying useful work for the processor to perform while waiting for data to become available. Techniques used to perform this sort of optimization are varied and beyond the scope of this document. Making more liberal use of registers and not re-using registers can greatly improve performance. Loads from memory are often a source of stalls, so loading a group of registers, then using those registers later in the code is one technique used to reduce stalls. But many times simply trial and error and even changes to the underlying algorithm can produce large increases in performance.

Once data dependency stalls have been improved, the **Pipeline Instruction Issue View** can be selected with the  button. As shown in Figure 5, this view indicates the



alignment of instructions, and the ability to use the dual issue pipeline. A green colored instruction indicates a correctly aligned instruction, a yellow colored instruction indicates an incorrectly aligned instruction. The goal is to correctly align all instructions, producing a completely green view. This will permit two instructions to be executed per cycle.

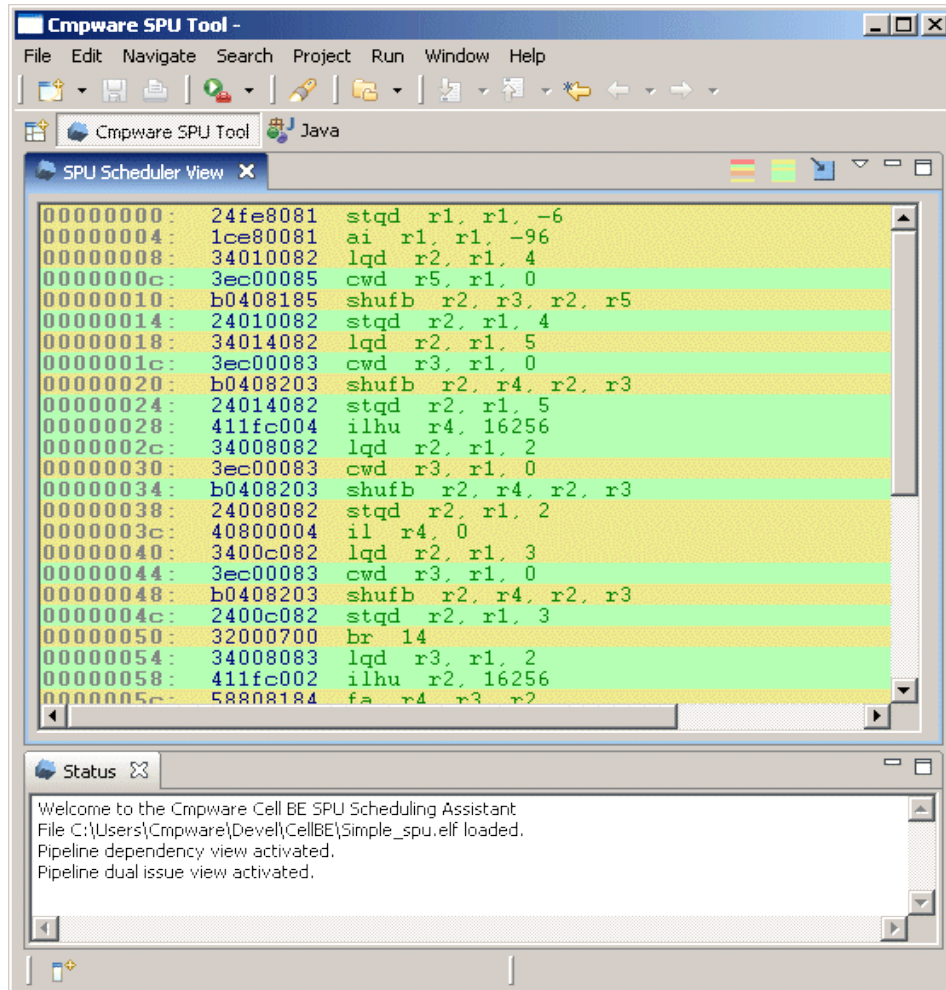


Figure 5: The SPU dual issue pipeline view.

As with the data dependency scheduling, the amount of optimization required will be application specific. If the code is not in a loop or performance critical section, it may be acceptable to ignore a relatively bad schedule in order to concentrate efforts on parts of the code that will have a greater impact on overall performance.

It should be noted that SPU dual issue instruction scheduling is very sensitive to





instruction ordering and may be somewhat non-intuitive at first. For instance, a section of code with all yellow (unaligned) instructions will only require the shift of a single instruction at the top of the block, often through the insertion of a noop, to completely align the entire block, changing all of the 'yellow' instructions to 'green'. While the worst possible performance comes from an all yellow region, it is the easiest to fix. In general, a 'striped' region of alternating 'yellow' and 'green' instructions will require the most work to optimize.

Finally, the process of optimizing the data dependency stalls and dual issue pipeline optimizations are not completely independent. Some small amount of iteration may be useful to achieve the highest performance. It is useful to check both views and be sure that changes to optimize one mode do not cause setbacks in the other mode.

SPU instruction scheduling can provide dramatic increases in SPU performance. It is, however, a technique that many not be familiar to many programmers. While common in DSP processors, most desktop and server processors do not expose such pipeline issues to the programmer. The *Cmpware SPU Scheduling Tool* is aimed at taking the complex task of scheduling the SPU and not only making these techniques available to less experienced programmers, but providing a 'power tool' for more experienced programmers.

## Conclusions

The *Cmpware SPU Scheduling Tool* makes every effort to present the SPU scheduling information in a intuitive, visually appealing format. The *Cmpware SPU Scheduling Tool* displays the complex SPU scheduling information in a color-coded format that is simple and easy to understand. A quick glance indicates where performance bottlenecks are. Fast, interactive controls permit views for the different optimization modes to be quickly viewed, assisting in globally optimizing the SPU code.

While this tool is described in terms of assembly language coding, it can also be used to view and examine the output of compilers and other high level tools. While many of these tools automatically provide schedule optimizations of various types, they only optimize based on the source code supplied. It is often possible to make changes to high level code implementation and algorithms that may provide dramatically improved SPU performance. The *Cmpware SPU Scheduling Tool* allows a detailed, but easy to understand, view into the complex but important performance aspects of the SPU.



## Resources

- A tutorial on the fully featured *Cmpware CMP-DK for the Cell B.E.* can be found at:

***[http://www.cmpware.com/Docs/Cmpware\\_3\\_CellBE.pdf](http://www.cmpware.com/Docs/Cmpware_3_CellBE.pdf)***

This tutorial includes detailed information on installation of an Eclipse plugin as well as information on the entire *Cmpware CMP-DK for the Cell B.E.*, including the integrated *SPU Scheduling Tool*.

- A more detailed explanation on the Cell B.E. SPU internal pipeline architecture and scheduling issues can be found in the article:

***Maximizing the power of the Cell Broadband Engine processor: 25 tips to optimal application performance*** by Daniel A. Brokenshire (brokensh@us.ibm.com), Senior Technical Staff Member, IBM STI Design Center, 27 Jun 2006:  
***<http://www.ibm.com/developerworks/power/library/pa-celltips1/>***

- For more information on the *Cmpware CMP-DK* see our web site at:

***<http://www.cmpware.com/>***

or send an email to:

***[info@cmpware.com](mailto:info@cmpware.com)***

