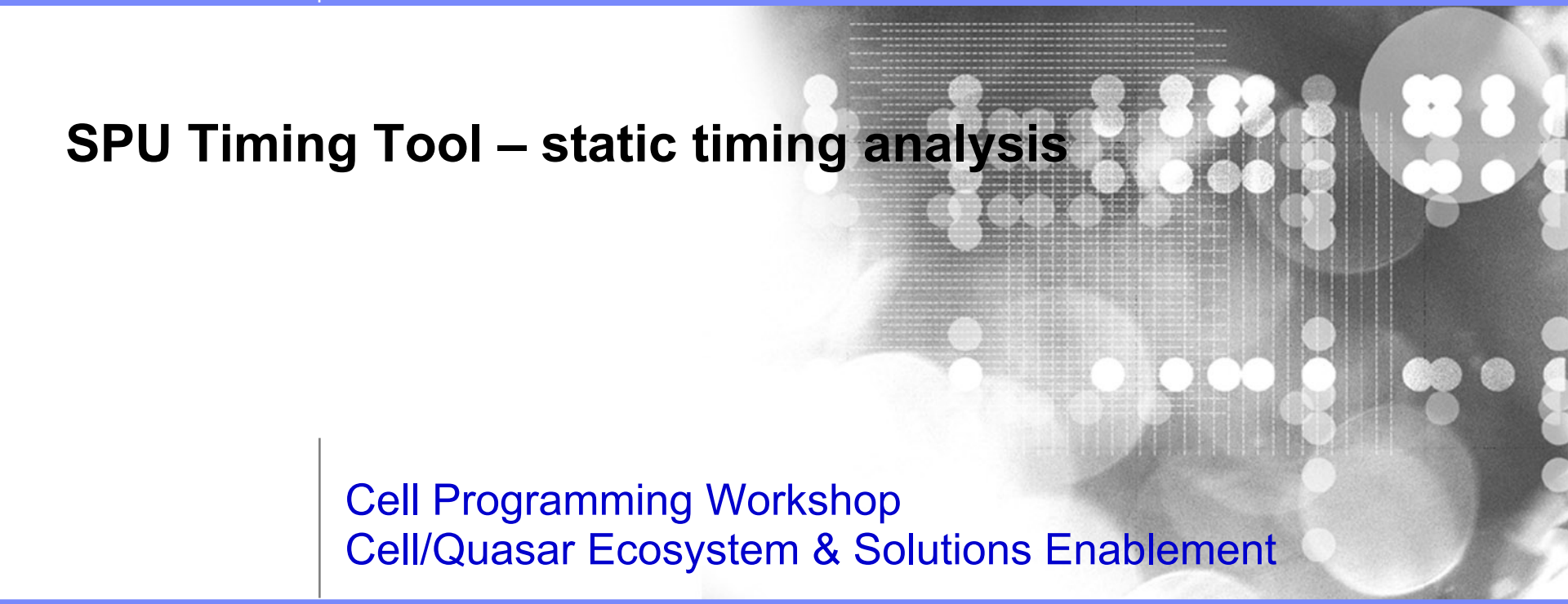


SPU Timing Tool – static timing analysis

Cell Programming Workshop
Cell/Quasar Ecosystem & Solutions Enablement

Class Objectives

- Learn how to use the static spu timing tool and analysis

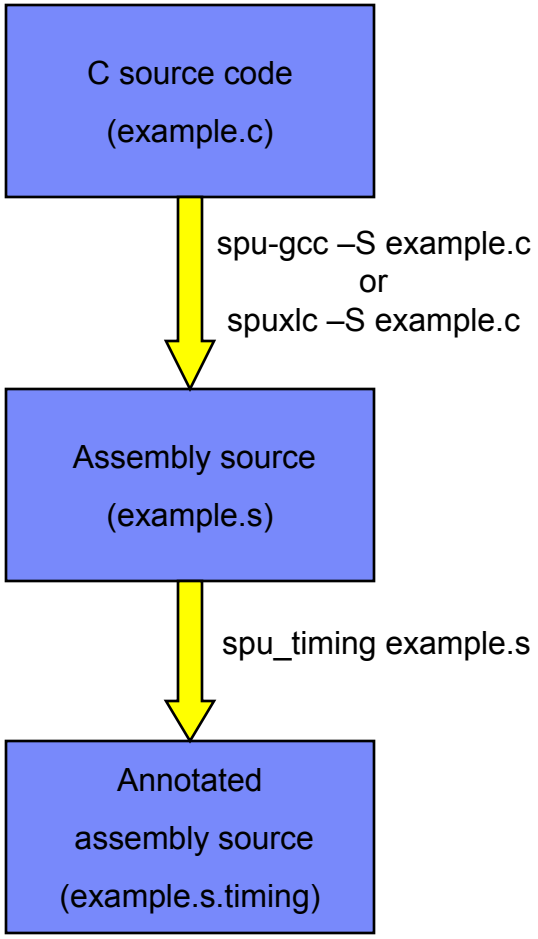
Class Agenda

- **What is the SPU timing tool?**
  - Features
  - Syntax
- **Sample output**
  - Interpreting the output
- **Useful Techniques**
  - Profile markers for locating code sections of interest
- **Functional Limitations**

- **References:**
  - Dan Brokenshire, Quasar Design Center

What is the SPU timing tool

- **Annotates an assembly source file with static analysis of instruction timing assuming linear (branchless) execution.**
  - Simplified pipeline model
  - Does not account for:
    - instruction fetch stalls
    - local store contention
    - branching
  - Supports Cell SDK 2.0 SPU models



Invocation Syntax

spu_timing [options …] [input_file]

- **options:**

  --help displays a verbose help screen.

  -march=<cpu> specifies the target architecture. <cpu> is either *cell* or *cell_edp*

  -o <file> specifies the output file. Default is <input_file>.*timing* or stdout if no input file is specified.

  -running-count include column of running cycles counts for start of each instruction.

- **<input_file> specifies the assembly input file. If not specified, *spu_timing* sources its input from stdin.**

Sample - C source (example.c)

```
#include <spu_intrinsics.h>

// Compute y = alpha * x + y, where alpha is a scalar and x
and
// y are 4*n element vectors.

void saxpy(int n, float alpha, vec_float4 x[], vec_float4 y[])
{
  int i;
  vec_float4 a;

  a = spu_splats(alpha);

  for (i=0; i<n; i++) {
    y[i] = spu_madd(a, x[i], y[i]);
  }
}
```

Sample - assembly source (example.s)

```
            .file     "example.c"
.text
            .align    3
            .global   saxpy
            .type     saxpy, @function
saxpy:
            ila       $2,66051
            shlqbyi   $7,$3,0
            cgti      $3,$3,0
            shufb     $8,$4,$4,$2
            nop       $127
            biz       $3,$lr
            ori       $4,$7,0
            hbra      .L8,.L4
            il        $7,0
            lnop
.L4:
            ai        $4,$4,-1
            lqx       $2,$7,$5
            lqx       $3,$7,$6
            fma       $2,$8,$2,$3
            stqx      $2,$7,$6
            ai        $7,$7,16
.L8:
            brnz      $4,.L4
            bi        $lr
```

Sample – annotated source (example.s.timing)

```
000000 0D 01
000000 1D 0123
000001 0d  12
000002 1d  -2345
000003 0D    3
000003 1D    3456
000004 0D     45
000004 1D     456789
000005 0D      56
000005 1D      5

000006 0d       67
000007 1d       -789012
000008 1          890123
000014 0           -----456789
000020 1                -----012345

000022 1                      2345
000023 1                       3456
```

```
                .file     "example.c"
.text
                .align    3
                .global   saxpy
                .type     saxpy, @function
saxpy:
                ila       $2,66051
                shlqbyi   $7,$3,0
                cgti      $3,$3,0
                shufb     $8,$4,$4,$2
                nop       $127
                biz       $3,$lr
                ori       $4,$7,0
                hbra      .L8,.L4
                il        $7,0
                lnop
.L4:
                ai        $4,$4,-1
                lqx       $2,$7,$5
                lqx       $3,$7,$6
                fma       $2,$8,$2,$3
                stqx      $2,$7,$6
.L8:
                brnz      $4,.L4
                bi        $lr
```

Sample – annotated source (example.s.timing)

running-count – cycle count for which each instruction starts. Useful for determining the cycles in a loop. For example, our loop is 17 cycles (23-6).

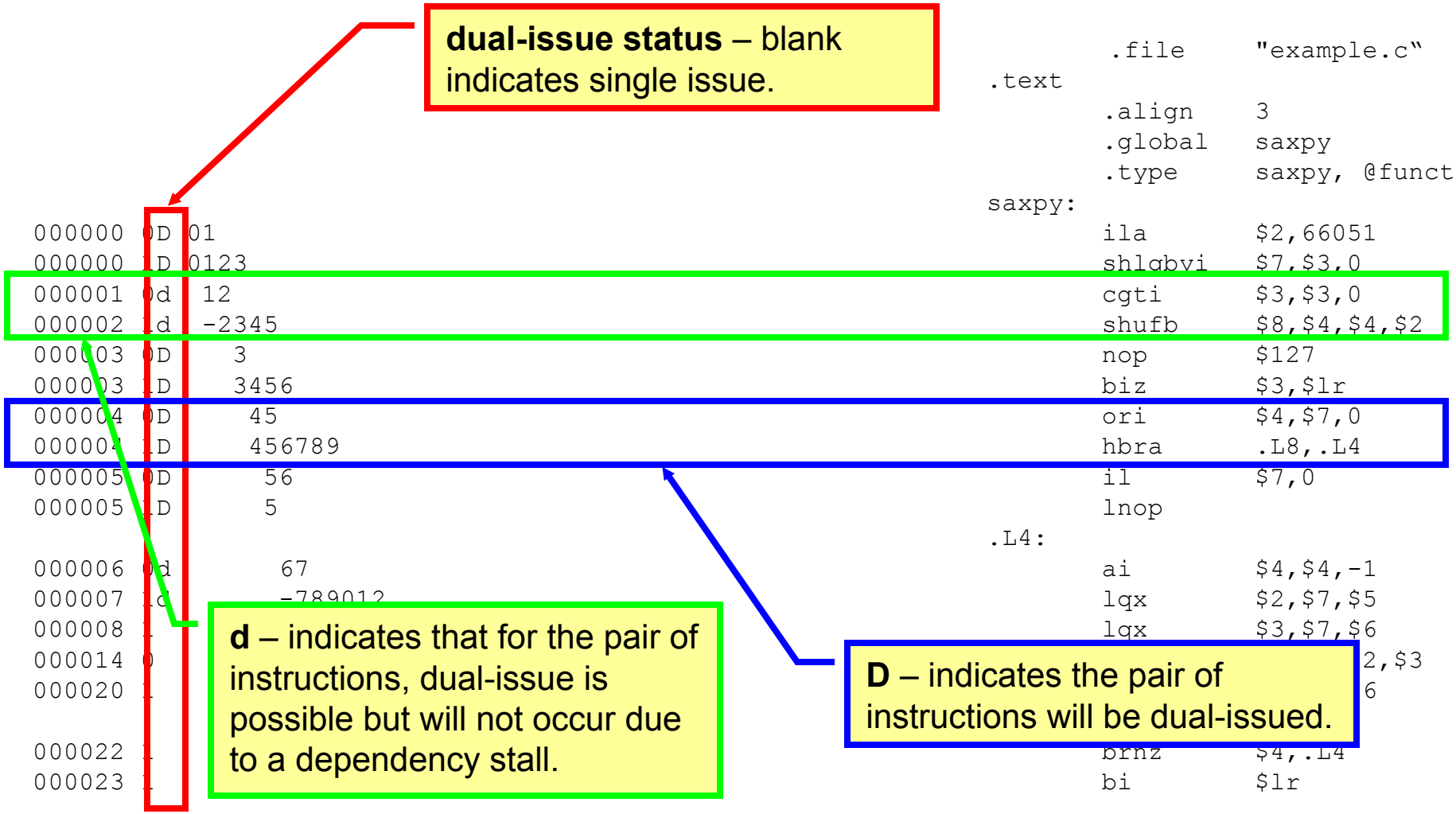
```
000000 0D 01                                        .file     "example.c"
000000 1D 0123                                      .text
000001 0d  12                                               .align    3
000002 1d  -2345                                            .global   saxpy
000003 0D    3                                              .type     saxpy, @function
000003 1D    3456                                   saxpy:
000004 0D     45                                            ila       $2,66051
000004 1D     456789                                        shlqbyi   $7,$3,0
000005 0D      56                                           cgti      $3,$3,0
000005 1D      5                                            shufb     $8,$4,$4,$2
                                                            nop       $127
000006 0d       67                                          biz       $3,$lr
000007 1d       -789012                                     ori       $4,$7,0
000008 1          890123                                    hbra      .L8,.L4
000014 0           -----456789                              il        $7,0
000020 1                -----012345                         lnop
                                                    .L4:
000022 1                      2345                          ai        $4,$4,-1
000023 1                       3456                         lqx       $2,$7,$5
                                                            lqx       $3,$7,$6
                                                            fma       $2,$8,$2,$3
                                                            stqx      $2,$7,$6
                                                    .L8:
                                                            brnz      $4,.L4
                                                            bi        $lr
```

Sample – annotated source (example.s.timing)

Execution pipeline – the pipeline in which the instruction is issued. Either 0 (even pipeline) or 1 (odd pipeline).

```
000000 00 01                                .file   "example.c"
000000 10 0123                      .text
000000 10                                   .align  3
000000 10                                   .global saxpy
000000 10                                   .type   saxpy, @function
000000 10                           saxpy:
000000 10                                   ila     $2,66051
000000 10                                   shlqbyi $7,$3,0
000000 10                                   cgti    $3,$3,0
000000 10                                   shufb   $8,$4,$4,$2
000000 10                                   nop     $127
000000 10                                   biz     $3,$lr
000000 10                                   ori     $4,$7,0
000000 10                                   hbra    .L8,.L4
000000 10                                   il      $7,0
000000 10                                   lnop
000000 10                           .L4:
000000 10                                   ai      $4,$4,-1
000000 10                                   lqx     $2,$7,$5
000000 10                                   lqx     $3,$7,$6
000000 10                                   fma     $2,$8,$2,$3
000000 10                                   stqx    $2,$7,$6
000000 10                           .L8:
000000 10                                   brnz    $4,.L4
000000 10                                   bi      $lr
```

Sample – annotated source (example.s.timing)



Sample – annotated source (example.s.timing)

```
000000 0D 01
000000 1D 0123
000001 0d  12
000002 1d  -2345
000003 0D    3
000003 1D    3456
000004 0D     45
000004 1D     456789
000005 0D      56
000005 1D      5

000006 0d       67
000007 1d       -789012
000008 1          890123
000014 0           -----456789
000020 1                -----012345

000022 1                      2345
000023 1                       3456
```

```
mple.c"


y
y, @function

6051
3,0
3,0
4,$4,$2

lr
7,0
.L4


4,-1
7,$5
7,$6
ima       $2,$8,$2,$3
stqx      $2,$7,$6
.L8:
brnz      $4,.L4
bi        $lr
```

Instruction clock cycle occupancy – A digit (0-9) is displayed for every clock cycle the instruction executes. Operand dependency stalls are flagged by a dash (“**-**”) for every clock cycle the instruction is expect to stall.

Steeply sloping cascading numbers signify good scheduling.

Shallow sloping (horizontal) numbers signify poor scheduling.

Sample – annotated source (example.s.timing)

Inner Loop – contains lots of dependency stalls.

The load of **y** stalls 1 cycle for address increment. The **fma** stalls 5 cycles waiting for the load to complete. The store of the resulting **y** stalls 5 cycles waiting for the fma to complete.

Dependency stalls could be eliminated by unrolling the loop. Loop unrolling could also result in moderate dual issue because the instruction mix is 1/3 pipe 0 and 2/3 pipe 1.

```
                                                                                    "example.c"

                                                                                    3
                                                                                    saxpy
                                                                                    saxpy, @function

000000 0D 01                                                                        $2,66051
000000 1D 0123                                                                      $7,$3,0
000001 0d  12                                                                       $3,$3,0
000002 1d  -2345                                                                    $8,$4,$4,$2
000003 0D    3                                                                      $127
000003 1D    3456                                                                   $3,$lr
000004 0D     45                                                                    $4,$7,0
000004 1D     456789                                                    nbra        .L8,.L4
000005 0D      56                                                       il          $7,0
000005 1D      5                                                        lnop
                                                                  .L4:
000006 0d       67                                                      ai          $4,$4,-1
000007 1d       -789012                                                 lqx         $2,$7,$5
000008 1          890123                                                lqx         $3,$7,$6
000014 0           -----456789                                          fma         $2,$8,$2,$3
000020 1                 -----012345                                    stqx        $2,$7,$6
                                                                  .L8:
000022 1                          2345                                  brnz        $4,.L4
000023 1                           3456                                 bi          $lr
```

Useful Technique – profile markers

- **For complex source code, insert profile checkpoint markers to locate specific code sections.**
  - #include <profile.h>
  - place prof_cp#() function in desired locations.
  - use unique # for improved identification.
  - prof_cp# results in "and $#, $#, $#" instructions, where # is 0 – 31.
  - When using spuxlc, the profile checkpoints are coded as mc_funcs. Therefore, to locate them, search for a ".word 0x########", where ######## corresponds to the encode and instruction.

Useful Technique – profile markers

```
#include <spu_intrinsics.h>
#include <profile.h>

// Compute y = alpha * x + y, where
alpha is a
// scalar and x and y are 4*n element
vectors.

void saxpy(int n, float alpha,
vec_float4 x[], vec_float4 y[])
{
  int i;
  vec_float4 a;

  a = spu_splats(alpha);

  prof_cp1();
  for (i=0; i<n; i++) {
    y[i] = spu_madd(a, x[i], y[i]);
  }
  prof_cp2();
}
```

```
        .align  3
        .global saxpy
        .type   saxpy, @function
saxpy:
        ila     $2,66051
        shlqbyi $7,$3,0
        and     $1,$1,$1; lnop
        cgti    $3,$3,0
        shufb   $8,$4,$4,$2
        nop     $127
        biz     $3,$lr
        ori     $4,$7,0
        hbra    .L8,.L4
        il      $7,0
        lnop
.L4:
        ai      $4,$4,-1
        lqx     $2,$7,$5
        lqx     $3,$7,$6
        fma     $2,$8,$2,$3
        stqx    $2,$7,$6
        ai      $7,$7,16
.L8:
        brnz    $4,.L4
        and     $2,$2,$2; lnop
        bi      $lr
```

Functional Limitations

- **Does not support multiple assembly instructions per line.**
- **Does not support generalized expressions. An expression will terminate the assembly parser.**
- **Does not support symbols and symbol substitution. Can terminate the assembly parser.**
- **Does not support completely the .repeat assembler directive.**

Works OK with compiled assembly, but often doesn't work for hand written assembly.

Special Notices -- Trademarks

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Many of the features described in this document are operating system dependent and may not be available on Linux. For more information, please check: http://www.ibm.com/systems/p/software/whitepapers/linux_overview.html

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Revised January 19, 2006

Special Notices (Cont.) -- Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States and/or other countries: alphaWorks, BladeCenter, Blue Gene, ClusterProven, developerWorks, e business(logo), e(logo)business, e(logo)server, IBM, IBM(logo), ibm.com, IBM Business Partner (logo), IntelliStation, MediaStreamer, Micro Channel, NUMA-Q, PartnerWorld, PowerPC, PowerPC(logo), pSeries, TotalStorage, xSeries; Advanced Micro-Partitioning, eServer, Micro-Partitioning, NUMACenter, On Demand Business logo, OpenPower, POWER, Power Architecture, Power Everywhere, Power Family, Power PC, PowerPC Architecture, POWER5, POWER5+, POWER6, POWER6+, Redbooks, System p, System p5, System Storage, VideoCharger, Virtualization Engine.

A full list of U.S. trademarks owned by IBM may be found at: http://www.ibm.com/legal/copytrade.shtml.

Cell Broadband Engine and Cell Broadband Engine Architecture are trademarks of Sony Computer Entertainment, Inc. in the United States, other countries, or both.
Rambus is a registered trademark of Rambus, Inc.
XDR and FlexIO are trademarks of Rambus, Inc.
UNIX is a registered trademark in the United States, other countries or both.
Linux is a trademark of Linus Torvalds in the United States, other countries or both.
Fedora is a trademark of Redhat, Inc.
Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries or both.
Intel, Intel Xeon, Itanium and Pentium are trademarks or registered trademarks of Intel Corporation in the United States and/or other countries.
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
AltiVec is a trademark of Freescale Semiconductor, Inc.
PCI-X and PCI Express are registered trademarks of PCI SIG.
InfiniBand™ is a trademark the InfiniBand® Trade Association
Other company, product and service names may be trademarks or service marks of others.

Revised July 23, 2006

Special Notices - Copyrights

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.

IBM IBM Logo Power Architecture

Other company, product and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page is http://www.ibm.com
The IBM Microelectronics Division home page is
http://www.chips.ibm.com

