Freescale Semiconductor
Application Note

Document Number: AN3359
Rev. 0, 05/2007

Performance Monitor on PowerQUICC™ II Pro Processors

by Harinder Rai
Network Computing Systems Group
Freescale Semiconductor, Inc.

The performance monitor is a module on PowerQUICC™ MPC83xx processors that counts predefined events and processor clocks associated with operations such as cache misses, mispredicted branches, and so on. The count of such events can be used to trigger a performance monitor interrupt. The performance monitor helps to identify system bottlenecks and can improve system performance by monitoring the software execution and recording the algorithms for more efficiency. This profiling tool must be used with a debugger to debug applications.

This application note describes the resources, components, and operating modes of the performance monitor and tells you how to get started using this tool.

The resources of performance monitor are as follows:

- Mask bit in the machine state register (MSR), which selects the programs to be monitored.
- Move to/from performance monitor instructions, **mtpmr** and **mfpmr**
- Performance monitor registers (PMRs)
  - — Performance monitor global control register, PMGC0
  - — Performance monitor counter registers, PMC[0–3]
  - — Performance monitor local control registers PMLC[a0–a3]

Contents

1 Performance Monitor Using Oprofile

Oprofile is a statistical continuous profiler that takes regular samples of program counter (PC) values from within the interrupt handler. It can be used for performance monitoring. Oprofile can be used to minimize the overhead of profiling and perform the following tasks:

- Profile interrupt handlers
- Profile an application and its shared libraries
- Capture the performance behavior of the entire system
- Examine hardware effects such as cache misses

1.1 Modes of Oprofile

Oprofile operates in the following modes:

- Hardware performance counters mode. Uses special-purpose hardware counter registers.
- Timer interrupt mode. Uses the timer interrupt for profiling. Disabled interrupts cannot be used to profile the code.
- Real-time clock. Uses only 2.2/2.4 kernels.

1.2 Components of Oprofile

Oprofile is divided into following parts:

- Architecture-specific code. The platform-specific code goes into the main kernel source tree under the `arch` directory.
- Oprofilefs. A pseudo file system mounted on `/dev/oprofile`.
- Generic kernel driver. Resides in the `drivers/oprofile` directory that contains the code for buffer management.
- Oprofile daemon. Takes the raw data from the kernel and places it into disk files.
- Post profiling tools. Along with Oprofile, Opreport and Opannonate are the user-level tools to convert the raw data in binary form.

2 Changing Kernel Source Files for Oprofile

During Linux® kernel boot time, oprofile looks for three entries by searching the cpu_spec_table in `cputable.c`. These entries are oprofile_cpu_type, num_pmcs and oprofile_type. If Oprofile does not detect these entries, it assumes by default that it should run in timer interrupt mode.The kernel-related changes are in the `arch/powerpc/kernel` and `arch/ppc/kernel` directories

The cpu_spec structure is defined in the `cputable.h` header file in the include/asm-powerpc directory.

The enum powerpc_oprofile_type is located around line number 41 in `cputable.h`. Add `PPC_OPROFILE_E300C3 = 5`, to the end of the enum. When Oprofile performs architecture-specific initialization, it looks for oprofile_type. The `cputable.c` file contains the instantiation of the cpu_spec structure. In this file, around line 302, is #if CLASSIC_PPC. Place the following lines under this hash define:

```
{       /* e300c3 (a 603e core, plus some) on 831X  */
                 .pvr_mask           = 0x7fff0000,
                 .pvr_value          = 0x00850000,
                 .cpu_name           = "e300",
                 .cpu_features       = CPU_FTRS_E300,
                 .cpu_user_features= COMMON_USER,
                 .icache_bsize       = 32,
                 .dcache_bsize       = 32,
                 .cpu_setup          = __setup_cpu_603,
                 .num_pmcs           = 4,
                 .oprofile_cpu_type= "ppc/e300",
                 .oprofile_type      = PPC_OPROFILE_E300C3,
                 .platform           = "ppc603",
},
```

In the `head.S` file under `arch/ppc/kernel`, around line 408, there is a trap handler, as follows:

```
                 . = 0xf00
                 b       Trap_0f
                 . = 0xf20
                 b       AltiVecUnavailable
Trap_0f:

         EXCEPTION_PROLOG
         addi    r3,r1,STACK_FRAME_OVERHEAD
         EXC_XFER_EE(0xf00, unknown_exception)
```

At this point, install the performance monitor exception handler. Replace the preceding code with the following lines:

```
                 . = 0xf00
                 b       PerformanceMonitor
                 . = 0xf20
                 b       AltiVecUnavailable

//Trap_0f:

//       EXCEPTION_PROLOG
//       addi    r3,r1,STACK_FRAME_OVERHEAD
//       EXC_XFER_EE(0xf00, unknown_exception)
```

After line 660, add the following performance monitor code:

```
         EXCEPTION_PROLOG
         addi    r3,r1,STACK_FRAME_OVERHEAD
         EXC_XFER_STD(0xf00, performance_monitor_exception)
```

3 Code Changes to Oprofile

To use the performance monitor hardware, two kinds of changes are required in the oprofile source code:

- Change in the oprofile kernel code and makefile. Changes are required in the oprofile source code of FSL MPC8313 board support package (BSP). This BSP has support for using oprofile on e500 core. The code for BSP support can be found in arch/powerpc/oprofile directory. The performance monitor hardware on e500 core is same as e300c3 core, copy *op_model_fsl_booke.c* file from the arc/power/oprofile directory to *op_model_e300c3.c* file and make few changes in the file. Add the following function in *op_model_e300c3.c* file

```
static void e300c3_cpu_setup(void * a)
> {
>       //FIXME: Do nothing
>       return;
> }
```

This function is a placeholder to satisfy the generic code design of oprofile. It requires setup/initialization code for other PowerPC architectures, such as rs64. However, for the e300c3 core, no setup is required. In the makefile, add `oprofile-y += op_model_e300c3.o` after `oprofile-$(CONFIG_FSL_BOOKE) += op_model_fsl_booke.o`.

- Change the oprofile scripts and user-level parsing binaries. Download oprofile version 0.9.2 from `www.oprofile.sourceforge.net`. The `op_cpu_type.c` file is located in the `oprofile-0.9.2/libop` directory. In static struct cpu_descr const cpu_descr[MAX_CPU_TYPE] = {, an array of static structures cpu_descrs[MAX_TYPE] can be found. At the end of the array, add `{"e300", "ppc/e300", CPU_PPC_E300, 4}`. The `op_cpu_type.h` file is in the `oprofile-0.9.2/libop` directory. Around line 20, there is an op_cpu enum. Add `CPU_PPC_E300` after line 62.

4 Build the Kernel

After making changes in the oprofile as described in Section 3, “Code Changes to Oprofile, build the kernel image:

1. Configure the kernel

   ```
   make menuconfig
   ```

   Choose the following configuration items:

   ```
   ---  Profiling support (EXPERIMENTAL)
   --- OProfile system profiling (EXPERIMENTAL)
   ```

2. Save the configuration.
3. Make the kernel image:

   ```
   make -j2 uImage
   ```

4. Ensure that the build was successful. Verify that all the `.o` files are built correctly in the `arch/powerpc/oprofile` directory.

5 Compile and Install Oprofile

The command to configure oprofile is as follows, although other options can be given with this command, depending upon the requirements of the application:

```
./configure    '--with-kernel-support'   '--prefix=/tmp/install_binaries'
'CC=powerpc-e300c3-linux-gcc' 'CXX=powerpc-e300c3-linux-g++'
```

After configuration, make the file:

```
make
```

After making a file, install make install, as follows:

```
make install `--prefix=/tmp/install_binaries`
```

6 Application Profiling Using Oprofile

The following code shows the profiled sample code/application.

```
#include <stdio.h>
void a();
void b();
void c();
void d();
void e();
void f();
void g();
void h();
int main()
{
a();
b();
c();
}
void a()
{
long long i=0;
for(i=0; i< 1000000; i++)
;
d();
e();
}
void b()
{
long long i=0;
for(i=0; i< 5000000; i++)
;
e();
g();
}
void c()
{
long long i=0;
for(i=0; i< 20000000; i++)
;
h();
}
void d()
{
f();
}
void e()
{
g();
}
void f()
{
long long i=0;
for(i=0; i< 1000000; i++)
;
}
void g()
```

```
{
long long i=0;
for(i=0; i< 10000000; i++)
;
}
void h()
{
long long i=0;
for(i=0; i< 100000000; i++)
;
}
```

The setup commands for profiling are:

```
./opcontrol --init
/opcontrol --no-vmlinux
/opcontrol --event=COMPLETED_BRANCHES:100000
../opcontrol -c 8
./opcontrol --image=a.out
./opcontrol --start-daemon -V
./opcontrol --start
```

Run the application.

```
./a.out
/opcontrol --dump
```

For the profile output run the following:

```
./opreport -c
```

Count COMPLETED_BRANCHES events (branch instructions completed) with a unit mask of 0x00 (no unit mask).

```
      coun

      t 100000

      samples  %        symbol name

-------------------------------------------------------------------------------

        6576     100.000   b

         130      58.0357   h

         130      100.000   h [self]

-------------------------------------------------------------------------------

        800      100.000   b

         49       21.8750   g

         49       100.000   g [self]

-------------------------------------------------------------------------------

        1600     100.000   main

         32       14.2857   c

         32       100.000   c [self]
```

```
-----------------------------------------------------------------------------
        8573     100.000  main
         9         4.0179  b
         6576     89.0454  h
          800      10.8328  g
         9         0.1219  b [self]
-----------------------------------------------------------------------------
        961      100.000  main
         3         1.3393  a
         3        100.000  a [self]
-----------------------------------------------------------------------------
         1         0.4464  f
         1        100.000  f [self]
-----------------------------------------------------------------------------
         0              0  main
         8573     76.9984  b
         1600     14.3704  c
         961       8.6312  a
         0              0  main [self]
-----------------------------------------------------------------------------
```

7 References

- e500 support, available at the Freescale web site listed on the back cover of this document. Includes the *PowerPC™ e500 Core Family Reference Manual*.
- Oprofile manual, available at `www.oprofile.sourceforge.net`
- Oprofile internal manual, available at `www.oprofile.sourceforge.net`

8 Revision History

Table 1 presents a revision history for this application note.

Table 1. Revision History for AN3359

Rev. Number	Date	Substantive Change(s)
0	05/2007	Initial draft.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
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
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3359
Rev. 0
05/2007