

# How to get the power values on the Intel SCC during program execution

Yu-Liang Chou

Department of Electrical Engineering and  
Computer Science, University of California, Irvine, California

Email: [yulianc@uci.edu](mailto:yulianc@uci.edu)

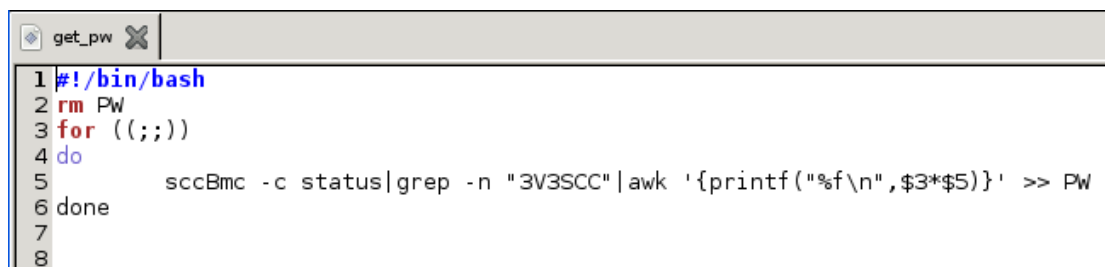
September 18, 2012

## 1. Introduction

This document provides two methods for obtaining the power values and averaged power value on the Intel SCC during program execution. The first method is an easier way to get the power values than the second one but the power sampling rate of the first method is lower than that of the second one.

## 2. Method 1

Method 1 is basically based on the solution proposed by radudavid [1]. First, we write a shell script as shown in Figure 1, named **get\_power**, that continuously calls "sccBmc -c status", and looks for the "3V3SCC" line to extract the current and voltage using the grep command, and computes the instant power by multiplying the current and voltage using the awk command. The power values are stored in a temporary file, PW.

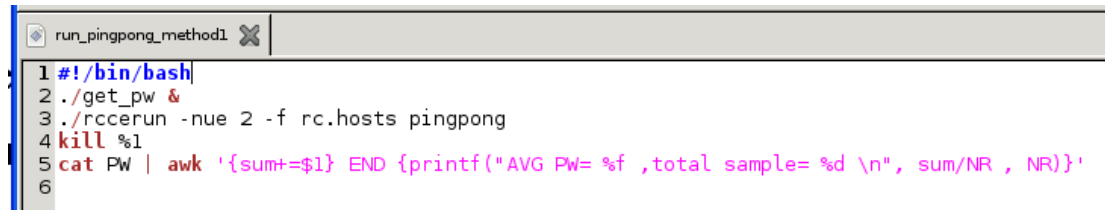
A screenshot of a terminal window titled 'get\_pw'. The terminal shows a shell script with the following content:

```
1 #!/bin/bash
2 rm PW
3 for ((;;))
4 do
5     sccBmc -c status|grep -n "3V3SCC"|awk '{printf("%f\n", $3*$5)}' >> PW
6 done
7
8
```

Figure 1. The get\_power script

Second, we write another shell script as shown in Figure 2, named **run\_pingpong\_method1**. This script first runs the get\_power script in the background (line 2), and then runs the program we want to measure (line 3, we run the pingpong program in this example), and terminates the get\_power script once the program finishes execution (line 4). Finally, we use the awk command to average all the power values stored in PW (line 5). By running this script, we can monitor the power variations during a program execution (all the sampled power values are stored in the

PW). Also, we can get the average power consumption during the program execution by averaging these power values.

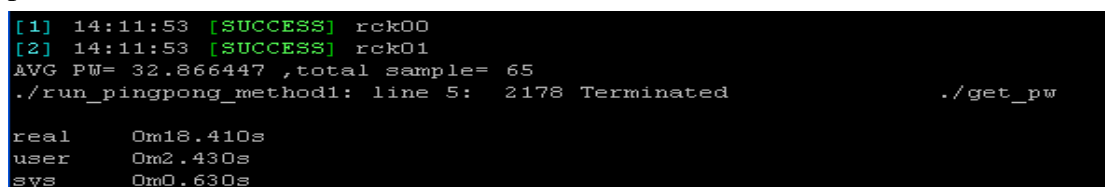


```
run_pingpong_method1
1 #!/bin/bash
2 ./get_pw &
3 ./rccerun -nue 2 -f rc.hosts pingpong
4 kill %1
5 cat PW | awk '{sum+=$1} END {printf("AVG PW= %f ,total sample= %d \n", sum/NR , NR)}'
6
```

Figure 2. The run\_pingpong\_method1 script

Figure 3 shows the execution results by executing the command `time ./run_pingpong_method1`

The average power during program execution is 32.866447 Watt, the total amount of sampled power value is 65, and the total execution time of pingpong is 18.41 Seconds. Therefore, the power sampling rate of method 1 is about 3.53 power measurements per second.



```
[1] 14:11:53 [SUCCESS] rck00
[2] 14:11:53 [SUCCESS] rck01
AVG PW= 32.866447 ,total sample= 65
./run_pingpong_method1: line 5: 2178 Terminated ./get_pw

real    0m18.410s
user    0m2.430s
sys     0m0.630s
```

Figure 3. The execution results of method 1

### 3. Method 2

First, we need to modify the source code of sccPerf from sccKit and then compile it so that we can make the sccPerf print the power values on the screen. The corresponding steps are described as follows:

1. Install qmake tools (libqt4-dev and qt4-dev-tools) so that you can build your own custom sccKit. If you are using the Data Center, you need to request the package installation through the product “Marc Administration Needed” on <http://marcbug.scc-dc.com/bugzilla3/>
2. Check your compiler. Since the sccPerf runs on the MCPC not the SCC chip, you should use the native compiler, not the cross-compiler. Execute the command `which gcc`, if the screen shows “/usr/bin/gcc”, your default compiler is native compiler. On the other hand, if the screen shows “/opt/i386-unknown-linux-gnu/bin/gcc”, your default compiler is cross-compiler.



```
jcdc@marc018:~$ which gcc
/usr/bin/gcc
jcdc@marc018:~$
```

- Download the sccKit source. Execute the command  
`svn co http://marcbug.scc-dc.com/svn/repository/tags/sccKit\_V1.4.1.2/`
- After downloading the sccKit source, there will be a directory called sccKit\_V1.4.1.2 in your working directory. Enter the directory /sccKit\_V1.4.1.2/sccPerf/src and edit **sccPerf.cpp** as Figure 4 shows. In Figure 4, an infinite loop that continuously reads the power value and prints it out is placed after the line 65 of sccPerf.cpp.

```

57 //endif
58 }
59 #else
60 printf("Didn't load PCIe driver! This is a binary for the power management demo...");
61 #endif
62
63 // Don't open GUI if initialization failed. Otherwise open it...
64 if (!initFailed) {
65     printf("Welcome to sccPerf ", VERSION_TAG, (QString)((BETA_FEATURES!=0)?"beta":""), " (build date ", __T
66
67     /***** edited by Chou Yu-Liang 20120918*****/
68     int i;
69     double power;
70     for (;;) {
71         power = sccAccess->getCurrentPowerconsumption();
72         cout << power << endl;
73     }
74     /*****/
75
76 // Create PerfMeterWidget...
77 PerfMeterWidget = new PerfMeterWidget(NUM_COLS, NUM_ROWS, NUM_CORES, sccAccess);

```

Figure 4. The modified sccPerf.cpp

- Enter the directory /sccKit\_V1.4.1.2 and compile the **sccPerf.cpp**. Enter  
`touch */src/main.cpp`  
`make -f Makefile.release sccPerf`

```

jcdc@marc018:/shared/jcdc/run_pingpong$ cd sccKit_V1.4.1.2/
jcdc@marc018:/shared/jcdc/run_pingpong/sccKit_V1.4.1.2$ touch */src/main.cpp
jcdc@marc018:/shared/jcdc/run_pingpong/sccKit_V1.4.1.2$ make -f Makefile.release sccPerf
cd sccPerf; qmake -recursive CONFIG+=debug_and_release; make release && cp bin/sccPerf ../rel
ease/development/bin/
make[1]: Entering directory `/shared/jcdc/run_pingpong/sccKit_V1.4.1.2/sccPerf'
make -f Makefile.Release
make[2]: Entering directory `/shared/jcdc/run_pingpong/sccKit_V1.4.1.2/sccPerf'
g++ -c -pipe -fno-strict-aliasing -O2 -Wall -W -D REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_N
ETWORK_LIB -DQT_CORE_LIB -DQT_SHARED -I/usr/share/qt4/mkspecs/linux-g++ -I. -I/usr/include/qt
4/QtCore -I/usr/include/qt4/QtNetwork -I/usr/include/qt4/QtGui -I/usr/include/qt4 -Iinclude -
I../sccCommon/include -I../sccGui/include -I../sccApi/include -Irelease -o release/sccPerf.o

```

- Copy the **systemSettings.ini** from the /opt/sccKit/ directory to your own /sccKit\_V1.4.1.2 directory. Enter  
`cp /opt/sccKit/systemSettings.ini .`

```

jcdc@marc018:/shared/jcdc/run_pingpong/sccKit_V1.4.1.2$ cp /opt/sccKit/systemSettings.ini .
jcdc@marc018:/shared/jcdc/run_pingpong/sccKit_V1.4.1.2$ ls
docs          Makefile.release  sccBmc         sccDisplay     sccKonsole     sccWrite
Makefile      release          sccBoot        sccDump        sccPerf        systemSettings.ini
Makefile.debug sccApi          sccCommon      sccGui         sccReset
jcdc@marc018:/shared/jcdc/run_pingpong/sccKit_V1.4.1.2$

```

(Note that there is a period after /opt/sccKit/systemSettings.ini, followed by one space.)

- Enter the /sccKit\_V1.4.1.2/sccPerf/bin directory and execute the sccPerf. You can see the following outputs.

```
jedc@marc018:/shared/jedc/run_pingpong/sccKit_V1.4.1.2$ cd ./sccPerf/bin
jedc@marc018:/shared/jedc/run_pingpong/sccKit_V1.4.1.2/sccPerf/bin$ ./sccPerf
INFO: openEMCCConnection(10.3.16.138:5010): You are participant #155
INFO: Packet tracing is disabled...
INFO: Initializing System Interface (SCEMI setup)....
INFO: Successfully connected to PCIe driver...
INFO: Welcome to sccPerf 1.4.2 (build date Sep 19 2012 - 20:16:32)...
31.9811
31.9811
31.9811
31.9811
32.3074
32.3074
```

Second, we need a shell script as shown in Figure 5, named **run\_pingpong\_method2**. This scrip first runs the modified sccPerf in the background (line 3, make sure that the path of sccPerf matches your own sccPerf), whose outputs will be stored in the PW, and then runs the program we want to measure (line4, we run the pingpong program in this example), and terminates the sccPerf once the program finishes execution (line 5). Finally, we use the awk command to average the power values stored in PW (line 6).

```
run_pingpong_method2 X
1 #!/bin/bash
2 rm PW
3 /shared/jedc/run_pingpong/sccKit_V1.4.1.2/sccPerf/bin/sccPerf >> PW &
4 ./rccerun -nue 2 -f rc.hosts pingpong
5 kill %1
6 cat PW | awk '{sum+=$1} END {printf("AVG PW= %f ,total sample= %d \n", sum/NR , NR)}'
7
8
9
```

Figure 5. The run\_pingpong\_method2 script

Figure 6 shows the execution results by executing the command time ./run\_pingpong\_method2

The average power during program execution is 31.562049 Watt, the total amount of sampled power value is 115, and total execution time of pingpong is 18.405 seconds. Therefore, the power sampling rate of method 2 is about 6.2483 power measurements per second.

```
2965792 0.180447283
3526944 0.215122120
[1] 16:16:36 [SUCCESS] rck00
[2] 16:16:36 [SUCCESS] rck01
AVG PW= 31.562049 ,total sample= 115
./run_pingpong_method2: line 6: 6179 Terminated /shared/jedc/run_p
ingpong/sccKit_V1.4.1.2/sccPerf/bin/sccPerf >> PW

real    0m18.405s
user    0m3.210s
sys     0m1.010s
jedc@marc018:/shared/jedc/run_pingpong$
```

Figure 6. The execution results of method 2

## Reference

[1] <http://communities.intel.com/thread/19608>