ISBN 978-89-6211-502-4

ClearSpeed 설치 및 활용 가이드

부 서: KISTI 슈퍼컴퓨팅본부
작성자: 조혜영

한국과학기술정보연구원

Korea Institute of Science and Technology Information

Korea Institute of
Science and Technology Information

목 차

제 1 장 서 론

현재 프로세서의 한계는 컴퓨터 칩 제조업체들인 인텔과 AMD를 비롯해 선마이크로시스템스와 IBM 등이 두 개 이상의 CPU를 하나의 실리콘 다이에 탑재한 칩을 개발해 발열 연산능력 및 속도 향상 문제에 대응해 왔다. 그러나 이러한 대안은 많은 프로세서들이 메모리를 두고 경쟁하기 때문에, 각각 메모리에서 데이터를 더욱 오래 기다려야 하며, 이에 따라 병목 현상이 발생하고 성능이 저하될 가능성이 있다. 더욱이 현재 많은 응용프로그램들이 멀티코어 칩에 최적화되어 있지 않고 있다는 것이 가장 큰 문제일 것이다.

최근 연산 수행 능력을 향상시키기 위한 가속기술의 다른 접근 방법으로, 재설정가능반도체(FPGA)나 주문형 반도체(ASIC) 등의 전용 칩을 사용하거나 Cell 프로세서, 그래픽 프로세서(GPU)와 같이 비디오 게임용으로 개발된 게임 프로세서(Game Processor)를 가속화 기술에 이용하려는 노력이 대두되고 있다.

이에 본 연구에서는 ASIC 프로세서를 이용한 대표적인 가속 프로세서인 Clearspeed Advanced e620을 대상으로 하드웨어 장착에서부터 소프웨어 설치 과정 및 활용 방법을 자세히 설명하였다. 주로 CPU 기반 환경에서 프로그래밍하는 것이 일반화되어 있는 현실에서 계산에 특화되어 있는 하드웨어를 이용하는 기술을 제공함으로써 멀티코어/매니코어, 이기종(Heterogeneous) 기술로 예상되는 차세대 슈퍼컴퓨터 시스템에 대한 최신을 이해하는데 도움이 되고자 한다.

본 보고서의 구성은 다음과 같다. 2장에서는 CSX600 프로세서의 구조 및 프로그램 구조를 간단히 설명하고, 4장에서는 Clearspeed의 하드웨어, 소프트웨어 설치 과정 및 활용 방법을 자세히 설명하고, 마지막으로 5장에서는 자주 발행하는 문제에 대한 해결책 등을 기술한다.

제 2 장 ClearSpeed 가속 기술

컴퓨팅 환경의 연산 가속 능력을 향상시키기 위한 다른 방법으로 주문형 직접 회로인 ASIC(Application Specific Integrated Circuit)을 사용하는 방법도 활용되고 있다. 2006 년 11 월 기준 슈퍼컴퓨터 Top500 에서 9 위를 차지한 일본의 도쿄공업대학교(Tokyo Institute of Technology)의 TSEBAME Grid Cluster 는 Top500 에서 최초로 ASIC 하드웨어 가속화 칩을 이용한 클러스터 시스템이다. 이 시스템은 계속 시스템을 보강하여 2009 년 6 월 기준 Top500 에서는 41 위를 차지하고 있다.

본 연구에서는 ASIC 을 이용한 가속 기술의 선두 업체인 Clearspeed 의 가속 보드 Advance e620(CSX620)을 사용하여 성능을 분석하였다. CSX620 보드는 기존의 호스트 CPU 와 함께 보조 프로세서(co-processor)로 동작하면서, 전력, 냉각설비 또는 공간의 부담 없이 응용프로그램을 가속 시킬 수 있다.

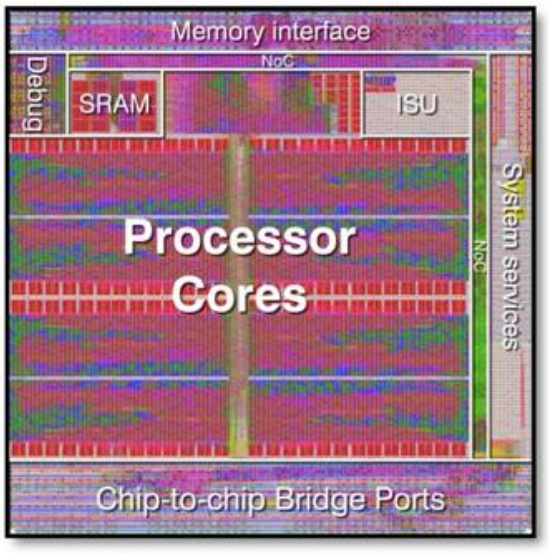

그림 1. CSX620 칩 구조

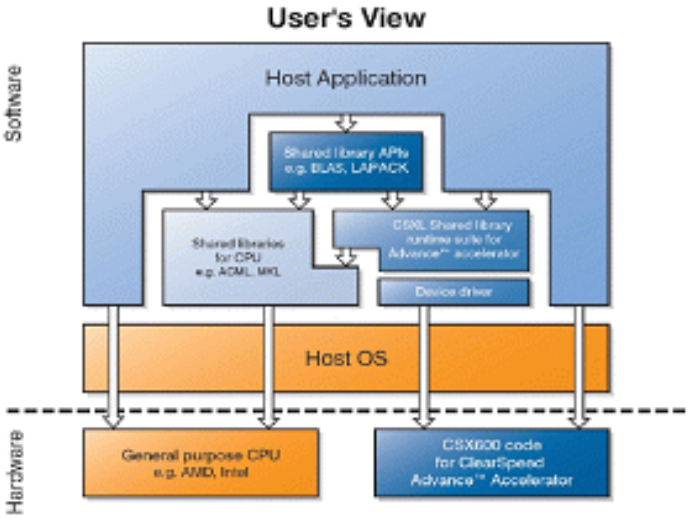

그림 2. CSX620 보드 응용프로그램 구조

그림 1 은 CSX620 보드의 칩 구조를 나타내었다. MTAP(Muti-Thread Array Processor core), 외장 DRAM 인터페이스, high-speed 인터프로세서 I/O port, 내장된 SRAM 이 하나의 칩에 통합된 형태로 구성되어 있다. MTAP 코어들은 내부 명령과 데이터의 병렬 처리 기능을 포함하며, 다중처리 유닛을 가지고 있는 PE(Processing Element) 96 개가 병렬로 배열되어 있다.

그림 2 에 CSX620 응용프로그램의 구조를 나타내었다. CSX600 보드는 호스트 프로세서와 함께 보조프로세서(coprocessor)로 동작한다. CSX620 의 표준수학 라이브러리(CSXL)가 불려지고 가속이 가능하다고 판단되면 필요한 데이터는 PCI-E 를 통해 보드로 이동되고 계산 결과는 호스트 CPU 로 전달된다.

제 3 장 Clearspeed 설치 및 성능 분석

3.1 하드웨어 장착

현재 KISTI가 보유하고 있는 보드는 ClearSpeed Advance e620으로 PCI-Express에 장착 가능

하다. 보드가 두께가 있어 서버의 크기가 2U이상이어야 순조롭게 장착 가능하다.

아래에 실제 보드에 모습이다.

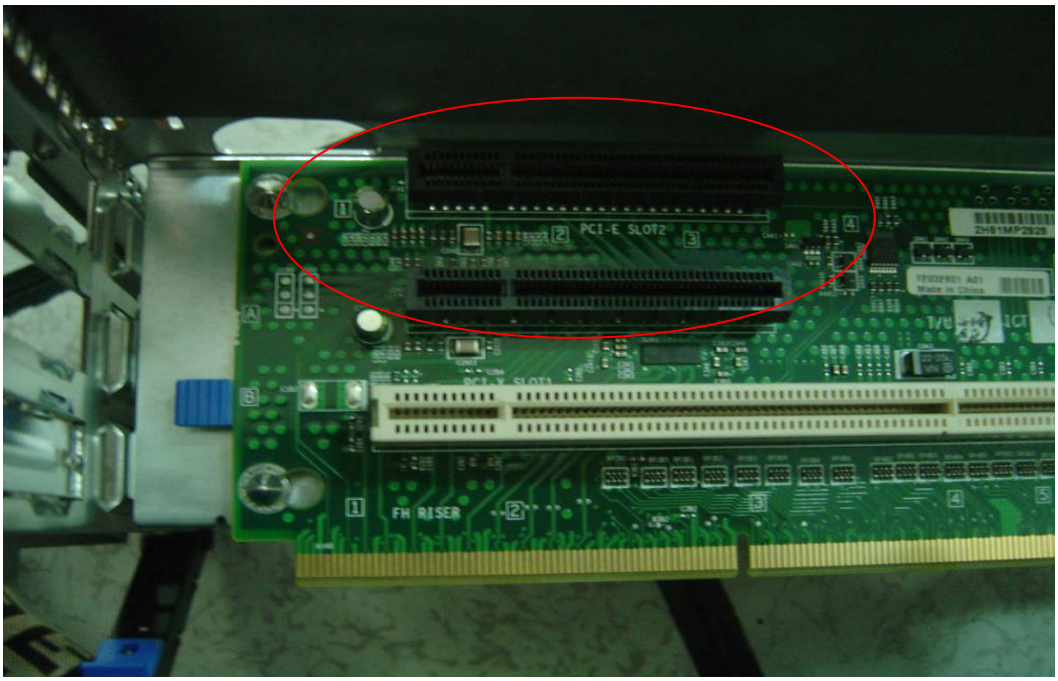

그림 3. 서버의 PCI-E Slot

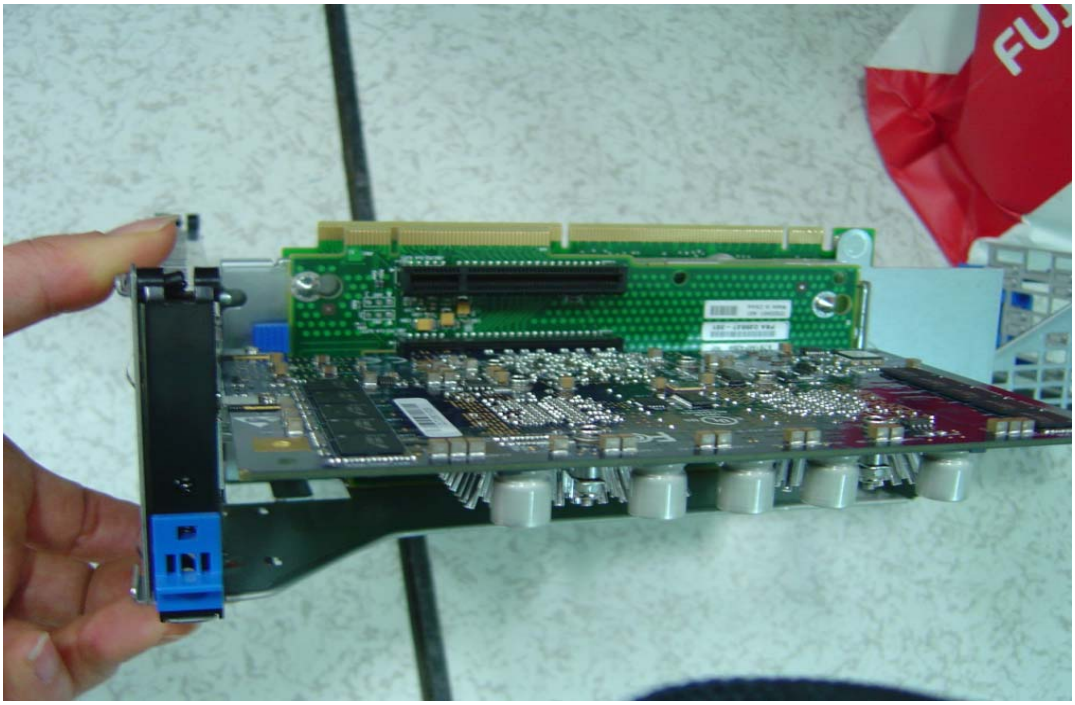

그림 4. ClearSpeed Advance e620 장착 모습

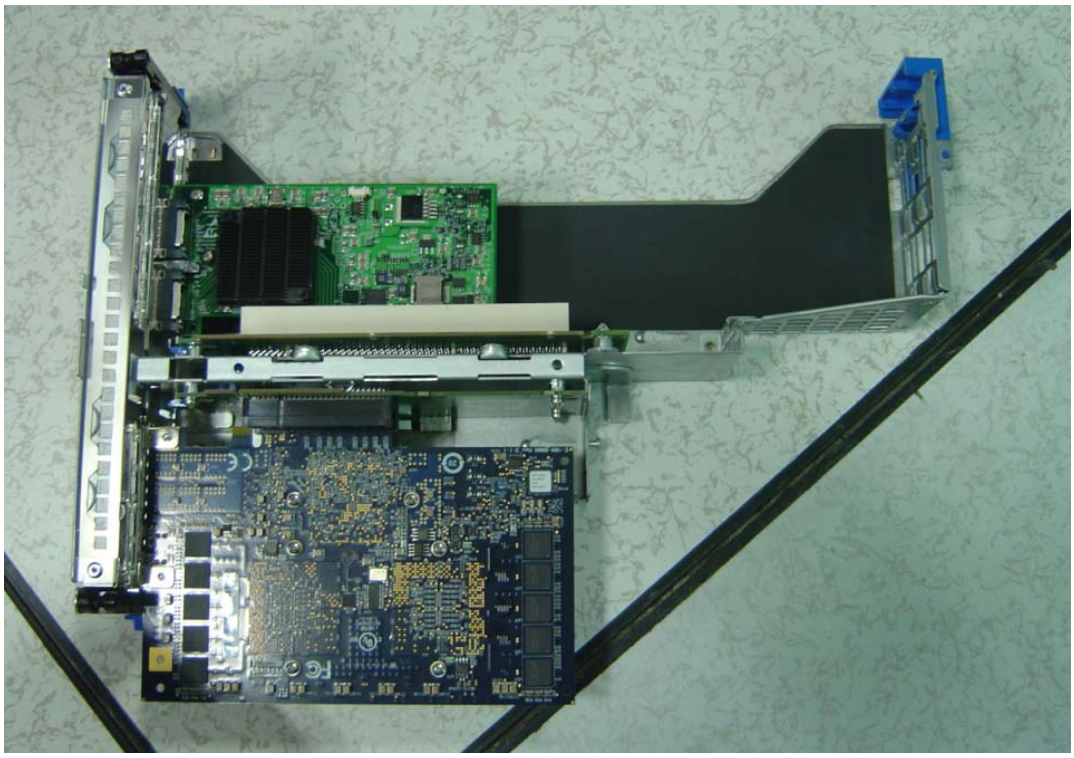

그림 5. ClearSpeed Advance e620 장착 모습

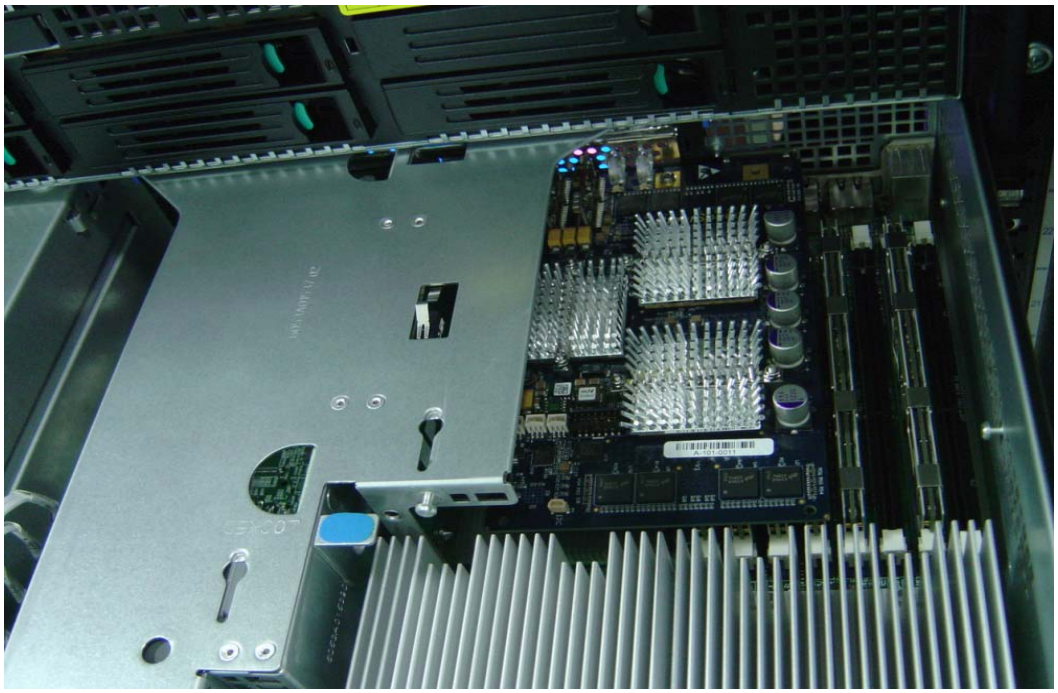

그림 6. ClearSpeed Advance e620 서버에 장착된 모습

ClearSpeed Advance e620는 Low-Profile 카드로도 지원한다.
메인 보드의 PCI-Express 성능이 슬롯마다 다르거나 동시 지원 성능은 낮아질 수 있기 때문에 메인 보드의 PCI-Express 성능 확인에 주의해야 한다.

3.2 소프트웨어 설치

1. 설치 환경
CPU : Intel(R) Xeon(R) CPU X5450 @ 3.00GHz

2. 설치 과정
Software download
http://support.clearspeed.com/downloads/software
프로그램을 다운로드 한다.

```
[root@minerva ~/Down/clearspeed]# ls
clearspeed-rhel5-base-3.11-x86_64.tgz clearspeed-rhel5-developer-3.11-x86_64.tgz
[root@minerva ~/Down/clearspeed]# tar xzvf clearspeed-rhel5-base-3.11-x86_64.tgz
3.11_base/
3.11_base/LICENSE.txt
3.11_base/index.html
3.11_base/clearspeed-rhel5-runtime-3.11-1.339.1.51.x86_64.rpm
3.11_base/clearspeed-rhel5-diagnostics-3.11-1.4.1.18.x86_64.rpm
3.11_base/clearspeed-rhel5-csxl-3.11-1.220.1.45.x86_64.rpm
3.11_base/lgpl.html
3.11_base/gpl.html
3.11_base/README.txt
3.11_base/Base_Software_Installation_Guide.pdf
[root@minerva ~/Down/clearspeed]# ls
3.11_base/ clearspeed-rhel5-base-3.11-x86_64.tgz clearspeed-rhel5-developer-3.11-x86_64.tgz
[root@minerva ~/Down/clearspeed]# tar xzvf clearspeed-rhel5-developer-3.11-
```

```
x86_64.tgz
3.11_developer/
3.11_developer/LICENSE.txt
3.11_developer/index.html
3.11_developer/Developer_Software_Installation_Guide.pdf
3.11_developer/clearspeed-rhel5-simulators-3.11-1.261.1.42.x86_64.rpm
3.11_developer/flexlm_enduser.pdf
3.11_developer/lgpl.html
3.11_developer/clearspeed-rhel5-csvprof-3.11-1.134.1.23.x86_64.rpm
3.11_developer/gpl.html
3.11_developer/README.txt
3.11_developer/clearspeed-rhel5-cs_sdk-3.11-1.418.1.49.x86_64.rpm
3.11_developer/clearspeed-rhel5-clearsp_flexlm-3.11-1.70.1.11.x86_64.rpm
3.11_developer/clearspeed-rhel5-csdft-3.11-1.146.1.46.x86_64.rpm
[root@minerva ~/Down/clearspeed]# ls
```

Base 패키지 설치

- Step1 : install the runtime components

```
[root@minerva ~/Down/clearspeed/3.11_base]# rpm -i clearspeed-rhel5-runtime-
3.11-1.339.1.51.x86_64.rpm
```

- Step2 : install the kernel modules

```
[root@minerva /opt/clearspeed/drivers]# cd csx
/opt/clearspeed/drivers/csx
[root@minerva /opt/clearspeed/drivers/csx]# ls
Makefile  csx*  csx_driver.c  csx_driver.h  install-csx*  uninstall-csx*
[root@minerva /opt/clearspeed/drivers/csx]# sh install-csx
make  -C  /lib/modules/2.6.18-92.el5/build  O=/lib/modules/2.6.18-92.el5/build
M=`pwd` ₩
                clean
make[1]: Entering directory `/usr/src/kernels/2.6.18-92.el5-x86_64'
make[1]: Leaving directory `/usr/src/kernels/2.6.18-92.el5-x86_64'
```

```
make -C /lib/modules/2.6.18-92.el5/build O=/lib/modules/2.6.18-92.el5/build
M=`pwd` ₩

make[1]: Entering directory `/usr/src/kernels/2.6.18-92.el5-x86_64'
  LD      /opt/clearspeed/drivers/csx/built-in.o
  CC [M]  /opt/clearspeed/drivers/csx/csx_driver.o
  LD [M]  /opt/clearspeed/drivers/csx/csx.o
  Building modules, stage 2.
  MODPOST
  CC      /opt/clearspeed/drivers/csx/csx.mod.o
  LD [M]  /opt/clearspeed/drivers/csx/csx.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.18-92.el5-x86_64'
Starting csx driver csx                                    [  OK  ]
[root@minerva /opt/clearspeed/drivers/csx]#
```

드라이버를 시작 하기

/etc/init.d/csx start

드라이버 Stop

/etc/init.d/csx stop

```
[root@minerva /opt/clearspeed/drivers]# /etc/init.d/csx start
Starting csx driver csx                                    [  OK  ]
[root@minerva /opt/clearspeed/drivers]# /etc/init.d/csx stop
Shutting down csx driver csx                               [  OK  ]
[root@minerva /opt/clearspeed/drivers]#
```

● Step3 : install diagnostics and validate installation

```
[root@minerva ~/Down/clearspeed/3.11_base]# pwd
/root/Down/clearspeed/3.11_base
[root@minerva ~/Down/clearspeed/3.11_base]# rpm -i clearspeed-rhel5-
diagnostics-3.11-1.4.1.18.x86_64.rpm
```

보드의 상태를 다음과 같이 체크할 수 있다.

```
[root@minerva /opt/clearspeed/bin]# ./csdiag --all
```

● Step4 : Install CSXL library

```
[root@minerva ~/Down/clearspeed]# cd 3.11_base/
```

```
/root/Down/clearspeed/3.11_base
[root@minerva ~/Down/clearspeed/3.11_base]# ls
Base_Software_Installation_Guide.pdf                clearspeed-rhel5-csxl-3.11-
1.220.1.45.x86_64.rpm          gpl.html
LICENSE.txt                                         clearspeed-rhel5-diagnostics-3.11-
1.4.1.18.x86_64.rpm    index.html
README.txt                                          clearspeed-rhel5-runtime-3.11-
1.339.1.51.x86_64.rpm      lgpl.html
[root@minerva  ~/Down/clearspeed/3.11_base]#  rpm  -i  clearspeed-rhel5-csxl-
3.11-1.220.1.45.x86_64.rpm
[root@minerva ~/Down/clearspeed/3.11_base]#
```

```
[root@minerva /opt/clearspeed/bin]# ./csdiag --all
```

● Uninstalling the componets

[주의] 삭제시 꼭 커널 모듈을 먼저 삭제 후 각 RPM을 삭제해야 한다.

Kernel Module 삭제

```
#cd /opt/clerspeed/drives/csx
#sh uninstall-csx
```

설치된 RPM 삭제

아래와 같이 설치된 리스트를 확인하고 하나씩 삭제한다.

```
[root@minerva ~/Down/clearspeed/3.11_base]# rpm -qa | grep clearspeed
clearspeed-rhel5-csxl-3.11-1.220.1.45
clearspeed-rhel5-runtime-3.11-1.339.1.51
clearspeed-rhel5-diagnostics-3.11-1.4.1.18
```

● 기본 패키지의 설치 상태 확인

```
[root@minerva ~/Down/clearspeed/3.11_base]# /etc/init.d/csx stop
Shutting down csx driver csx                                    [   OK   ]
[root@minerva ~/Down/clearspeed/3.11_base]# /etc/init.d/csx start
Starting csx driver csx                                         [   OK   ]
[root@minerva ~/Down/clearspeed/3.11_base]# csdiag --all

Time: Fri Sep 25 11:01:22 2009
Host: minerva.kisti.re.kr Linux x86_64 2.6.18-92.el5
```

```
SVer: 3.11 (1.404.1.39 at Thu Aug 21 16:50:25 BST 2008 on linux_x86_64)
Card: Instance 0, Bus 160, Device 0 (1 card present)
        e620 (PCIe x8), FPGA 0x6f02c000, SN CLTJ08010035

Basic checks: FAIL (Register access OK, Status OK, Memory access check failed.)

Overall result: FAIL
[root@minerva ~/Down/clearspeed/3.11_base]# csreset
[root@minerva ~/Down/clearspeed/3.11_base]# csdiag --all

Time: Fri Sep 25 11:02:16 2009
Host: minerva.kisti.re.kr Linux x86_64 2.6.18-92.el5
SVer: 3.11 (1.404.1.39 at Thu Aug 21 16:50:25 BST 2008 on linux_x86_64)
Card: Instance 0, Bus 160, Device 0 (1 card present)
        e620 (PCIe x8), FPGA 0x6f02c000, SN CLTJ08010035

Basic checks: PASS
Reset card:     PASS
Temperature:    PASS
Frequency:      PASS
PCI Read BW:    PASS
PCI Write BW: PASS
Combined BW:    PASS
Memory tests: PASS
Semaphores:     PASS
PIO tests:      PASS
PE memory:      PASS
Compute test: PASS
Temperature:    PASS
Frequency:      PASS

Overall result: PASS
[root@minerva ~/Down/clearspeed/3.11_base]# rpm -qa | grep clearspeed
clearspeed-rhel5-csxl-3.11-1.220.1.45
clearspeed-rhel5-runtime-3.11-1.339.1.51
```

```
clearspeed-rhel5-diagnostics-3.11-1.4.1.18
[root@minerva ~/Down/clearspeed/3.11_base]#
```

[참고]

csdiag 실행시 아래와 같이 Error가 발생할 경우

```
[root@minerva /opt/clearspeed/bin]# csdiag --all
csdiag: error while loading shared libraries: libc_buildversion.so: cannot open shared object file: No such file or directory
```

다음과 같이 환경파일에 디렉토리 경로는 작성하고 적용한다.

```
[root@minerva /opt/clearspeed/bin]# pwd
/opt/clearspeed/bin
[root@minerva /opt/clearspeed/bin]# ls
adbgt*  bashrc*  csdiag*  csgdb*  cshrc*  csinfo*  csled*  csreset*  csrun*
env/  power_mon*  temp_mon*  xsvfplayer*  zkey*
[root@minerva /opt/clearspeed/bin]# vi bashrc
# ClearSpeed runtime bash resource file - please source this before
# running any ClearSpeed tools.

export CSHOME=/opt/clearspeed
export CSPLATFORM=linux
if [ -z $CSPATH ]
then
export CSPATH=.:$CSHOME/csx:$CSHOME/config
else
export CSPATH=.:$CSHOME/csx:$CSHOME/config:$CSPATH
fi
export PATH=$CSHOME/bin:$PATH
if [ -z $LD_LIBRARY_PATH ]
then
export LD_LIBRARY_PATH=$CSHOME/lib
else
export LD_LIBRARY_PATH=$CSHOME/lib:$LD_LIBRARY_PATH
fi
```

```
# Source any additional ClearSpeed bash resource files

if [ -d $CSHOME/bin/env ]
then
FILES=`/bin/ls -1 $CSHOME/bin/env/*.bash 2>/dev/null`
for file in $FILES
do
source $file
done
fi
[root@minerva /opt/clearspeed/bin]# source bashrc
```

Development 패키지 설치

● Step1 : install rpm components

http://support/clearspeed.com/downloads/software로 부터 다운받는 clearspeed-rhel5-developer-3.11-x86_64.tgz를 압축해제하고 rpm들을 설치한다.

```
[root@minerva ~/Down/clearspeed/3.11_developer]# ls
Developer_Software_Installation_Guide.pdf                    clearspeed-rhel5-csvprof-3.11-1.134.1.23.x86_64.rpm
LICENSE.txt                                                  clearspeed-rhel5-simulators-3.11-1.261.1.42.x86_64.rpm
README.txt                                                   flexlm_enduser.pdf
clearspeed-rhel5-clearsp_flexlm-3.11-1.70.1.11.x86_64.rpm    gpl.html
clearspeed-rhel5-cs_sdk-3.11-1.418.1.49.x86_64.rpm           index.html
clearspeed-rhel5-csdft-3.11-1.146.1.46.x86_64.rpm            lgpl.html

[root@minerva ~/Down/clearspeed/3.11_developer]# rpm -Uvh clearspeed-rhel5-cs_sdk-3.11-1.418.1.49.x86_64.rpm
준비 중...                ########################################### [100%]
   1:clearspeed-rhel5-cs_sdk ########################################### [100%]

[root@minerva ~/Down/clearspeed/3.11_developer]# rpm -Uvh clearspeed-rhel5-csdft-3.11-1.146.1.46.x86_64.rpm
준비 중...                ########################################### [100%]
```

```
   1:clearspeed-rhel5-csdft ########################################### [100%]
[root@minerva ~/Down/clearspeed/3.11_developer]# rpm -Uvh clearspeed-rhel5-clearsp_flexlm-3.11-1.70.1.11.x86_64.rpm
준비 중...                ########################################### [100%]
   1:clearspeed-rhel5-clears########################################### [100%]
[root@minerva ~/Down/clearspeed/3.11_developer]# rpm -Uvh clearspeed-rhel5-csvprof-3.11-1.134.1.23.x86_64.rpm
준비 중...                ########################################### [100%]
   1:clearspeed-rhel5-csvpro########################################### [100%]
[root@minerva ~/Down/clearspeed/3.11_developer]# rpm -Uvh clearspeed-rhel5-simulators-3.11-1.261.1.42.x86_64.rpm
준비 중...                ########################################### [100%]
   1:clearspeed-rhel5-simula########################################### [100%]
```

설치된 rpm들을 확인한다.

```
[root@minerva /opt/clearspeed/lib]# rpm -qa | grep clearspeed
clearspeed-rhel5-csvprof-3.11-1.134.1.23
clearspeed-rhel5-csxl-3.11-1.220.1.45
clearspeed-rhel5-cs_sdk-3.11-1.418.1.49
clearspeed-rhel5-csdft-3.11-1.146.1.46
clearspeed-rhel5-runtime-3.11-1.339.1.51
clearspeed-rhel5-clearsp_flexlm-3.11-1.70.1.11
clearspeed-rhel5-diagnostics-3.11-1.4.1.18
clearspeed-rhel5-simulators-3.11-1.261.1.42
[root@minerva /opt/clearspeed/lib]#
```

BLAS 라이브러리 설치

- http://www.netlib.org/blas/ 로부터 ATLAS를 다운한다.

```
[root@minerva ~/Down]# cd BLAS/
/root/Down/BLAS
[root@minerva ~/Down/BLAS]# ls
```

```
ATLAS/   atlas3.9.14.tar.bz2   ssgemmbased.tgz
[root@minerva ~/Down/BLAS]# mv ATLAS/ ATLAS3.8.14
[root@minerva ~/Down/BLAS]# cd ATLAS3.8.14/
/root/Down/BLAS/ATLAS3.8.14
```

● Turn off CPU throttling

```
[root@minerva ~/Down/BLAS/ATLAS]# /usr/bin/cpufreq-selector -g performance
```

[주의] cpufreq-selector 명령은 첫번째 CPU에만 적용된다. 따라서 CPU가 여러 개일 경우 아래와 같이 설정 파일을 복사해야 한다.

```
[root@minerva ~/Down/BLAS/ATLAS]# cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
performance
[root@minerva ~/Down/BLAS/ATLAS]# cat
/sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
ondemand

[root@minerva ~/Down/BLAS/ATLAS]# cp -f
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu2/cpufreq/scaling_governor
cp: overwrite `/sys/devices/system/cpu/cpu2/cpufreq/scaling_governor'? y
[root@minerva ~/Down/BLAS/ATLAS]# cp -f
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu3/cpufreq/scaling_governor
cp: overwrite `/sys/devices/system/cpu/cpu3/cpufreq/scaling_governor'? y
[root@minerva ~/Down/BLAS/ATLAS]# cp -f
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu4/cpufreq/scaling_governor
cp: overwrite `/sys/devices/system/cpu/cpu4/cpufreq/scaling_governor'? y
[root@minerva ~/Down/BLAS/ATLAS]# cp -f
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu5/cpufreq/scaling_governor
cp: overwrite `/sys/devices/system/cpu/cpu5/cpufreq/scaling_governor'? y
[root@minerva ~/Down/BLAS/ATLAS]# cp -f
```

```
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu6/cpufreq/scaling_governor
cp: overwrite `/sys/devices/system/cpu/cpu6/cpufreq/scaling_governor'? y
[root@minerva ~/Down/BLAS/ATLAS]# cp -f
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
/sys/devices/system/cpu/cpu7/cpufreq/scaling_governor
cp: overwrite `/sys/devices/system/cpu/cpu7/cpufreq/scaling_governor'? y
```

● Basic Steps of an ATLAS install

```
[root@minerva ~/Down/BLAS/ATLAS3.8.14]# mkdir LINUX_ATLAS3.8.14/
[root@minerva ~/Down/BLAS/ATLAS3.8.14]# cd LINUX_ATLAS3.8.14/
/root/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14
[root@minerva ~/Down/BLAS/ATLAS3.8.14]# pwd
/root/Down/BLAS/ATLAS3.8.14

[root@minerva ~/Down/BLAS/ATLAS3.8.14]../configure -b 64 --
prefix=/opt/lib.gnu/ATLAS
.....
[root@minerva ~/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14]# make build
[root@minerva ~/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14]# make



[root@minerva ~/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14]# make check
[root@minerva ~/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14]# make time
.......
res+ off= 4882.0 1.00    ---

BIG_MM N=1600, mf=4882.00,4869.00!

The times labeled Reference are for ATLAS as installed by the authors.
NAMING ABBREVIATIONS:
    kSelMM : selected matmul kernel (may be hand-tuned)
    kGenMM : generated matmul kernel
    kMM_NT : worst no-copy kernel
```

```
    kMM_TN : best no-copy kernel
    BIG_MM : large GEMM timing (usually N=1600); estimate of asymptotic peak
    kMV_N  : NoTranspose matvec kernel
    kMV_T  : Transpose matvec kernel
    kGER   : GER (rank-1 update) kernel
Kernel routines are not called by the user directly, and their
performance is often somewhat different than the total
algorithm (eg, dGER perf may differ from dkGER)


Reference clock rate=2394Mhz, new rate=3000Mhz
    Refrenc : % of clock rate achieved by reference install
    Present : % of clock rate achieved by present ATLAS install

                 single precision                  double precision
            ******************************    ******************************
                 real            complex           real            complex
            ---------------  ---------------  ---------------  ------------
---
Benchmark    Refrenc Present   Refrenc Present   Refrenc Present   Refrenc Present
=========    ======= =======   ======= =======   ======= =======   ======= =======
=======
  kSelMM       575.1   162.4     498.1   169.7     365.6   166.7     353.1   143.3
  kGenMM       171.4   162.4     177.1   169.7     167.2   166.7     162.5   143.3
  kMM_NT       134.8   152.0     138.6   149.9     128.4   138.6     132.0   138.9
  kMM_TN       159.7   167.3     166.5   162.4     149.2   152.4     158.6   156.3
  BIG_MM       562.3   172.1     560.2   158.9     355.7   171.0     349.9   162.7
   kMV_N       111.4    55.0     212.4    98.4      58.1    33.4      94.7    43.0
   kMV_T        89.6    58.4      96.1    85.3      47.3    32.3      76.7    42.7
    kGER        57.2    37.5     123.1    83.0      29.2    20.8      59.5    42.9
make[1]: Leaving directory `/root/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14'
[root@minerva ~/Down/BLAS/ATLAS3.8.14/LINUX_ATLAS3.8.14]# make install
```

3.3 라이센스 등록

Clearspeed 보드에서 수행할 프로그램을 컴파일하기 위해서는 라이센스 등록이 필요하다.라이센스 파일을 얻기 위해서는 미리 설치할 서버의 hostname과 Flexlm Host ID, Max Address 등을 clearspeed 본사에 보내 license file을 받아 lmgrd를 구동시켜야 컴파일 작업을 진행할 수 있다.

```
[root@harper01 bin]# hostname
harper01
[root@harper01 bin]# cd /opt/clearspeed/
bin/                    csa/                    docs/                   include/
ClearSpeed.hpl.tar.gz   csx/                    drivers/                lib/
config/                 csx600_m512_le/         examples/               src/
[root@harper01 bin]# cd /opt/clearspeed/csx
csx/            csx600_m512_le/
[root@harper01 bin]# cd /opt/clearspeed/csx
csx/            csx600_m512_le/
[root@harper01 bin]# cd /opt/clearspeed/csx/
bootstrip.csx                   csdiag.csx                      dgemm96.csx
fft_cs_processor_0.csx
[root@harper01 bin]# cd /opt/clearspeed/csx/bin
-bash: cd: /opt/clearspeed/csx/bin: No such file or directory
[root@harper01 bin]# ./lmhostid
lmhostid - Copyright (c) 1989-2005 Macrovision Europe Ltd. and/or Macrovision
Corporation. All Rights Reserved.
The FLEXlm host ID of this machine is "00151762d7e8"
```

위의 붉게 표시한 부분은 Hostname과 함께 메일로 보내면 hostname.lic에 해당하는 파일을 보내준다.

라이센스 파일은 다음과 같은 포맷이다.

* 현재 KISTI에서는 Harper01의 Eth0 Mac Address에 대해서 라이센스 파일을 확보한 상태이다.

```
[root@harper01 bin]# cat Harper01_3.999.lic
SERVER Harper01 00151762D7E8
VENDOR clearsp
```

```
FEATURE cs_sdk_arcs clearsp 3.999 31-oct-2009 1 SIGN="1769 99CD 6627 ₩
        53EA C4C5 62CD D5DF CBFE A721 5456 B4C9 2985 6984 F687 78A4 ₩
        0934 ACFA AF8B 52E2 241C C1CC E0E6 3393 554C 45D3 1A28 5421 ₩
        8E14 9432 2858"
FEATURE cs_sdk_csdump clearsp 3.999 31-oct-2009 1 SIGN="0010 B8C9 ₩
        000D C543 43BF A7A0 A98B 7D62 E47A 7E98 EEB5 6D92 84B5 A27A ₩
        BA2E 13FC 7F5E 0B5D 8F3F 008F 57BE 2D7F 3A53 508A 2D28 B805 ₩
        7F23 9563 FF9D 73F4"
FEATURE cs_sdk_cld clearsp 3.999 31-oct-2009 1 SIGN="10ED 293A C7F9 ₩
        B5C4 7EAE AAE5 4DF9 46CC 9AE6 453D CBDD FB71 4051 DB7A 930B ₩
        15BA 356A E9CF 92E8 C5D7 417E 4E6E 8FBE 2D2D 96CF C6A8 967D ₩
        C477 3C06 2136"
FEATURE cs_sdk_mass clearsp 3.999 31-oct-2009 1 SIGN="165C 968D BBE7 ₩
        57B9 AF03 B082 4519 65AC 641D FDC0 9318 6D2A FC2A C683 C0C3 ₩
        12B7 FFC6 98FC E195 41F9 7758 B275 5D25 0C84 36C6 4EAD 4088 ₩
        227C 6596 E74D"
FEATURE cs_sdk_cosycn clearsp 3.999 31-oct-2009 1 SIGN="1CE8 BD5E ₩
        DA94 92DD 8C02 0107 F18F 7D25 DA45 D115 3594 7502 D92C 313A ₩
        1255 131E C632 E983 332A 0A62 25A9 DDF1 5837 3402 C123 253A ₩
        AD6E F10A F2A8 7680"
```

라이센스 서버 구동

라이센스 파일을 가지고 license server를 구동하는 방법은 다름과 같다.

```
[root@harper01 bin]# cd /opt/clearspeed/bin
[root@harper01 bin]# ./lmgrd -c Harper01_3.999.lic -l flexlm.log
[root@harper01 bin]# ps -ef | grep lmgrd
root      3598  2866  0 02:34 pts/2    00:00:00 grep lmgrd
root     11239 20366  0 Sep21 ?        00:00:00 clearsp -T Harper01 10.8 4 -c
Harper01_3.999.lic -lmgrd_port 6978 --lmgrd_start 4ab111b8
root     20366     1  0 Sep17 ?        00:04:25 ./lmgrd -c Harper01_3.999.lic -
l flexlm.log
[root@harper01 bin]#
```

3.4 컴파일 및 실행

● HelloWorld 실행 구현

HelloWorld.cn 파일을 작성한다.

Clearspeed 예제 중 HelloWorld를 실행 해 보자.

CSX600에서 실행되는 프로그램 확장자는 .cn으로 작성된다.

```
[root@harper01 hello_world]# pwd
/opt/clearspeed/examples/sdk/hello_world
[root@harper01 hello_world]# cat hello_world.cn
#include <lib_ext.h>
#include <dprint.h>
#include <stdiop.h>

int main(void) {

   // The standard libraries provide standard printf and a poly printfp function

   printf("Hello world₩n");
   printfp("Hello world from PE %d₩n",get_penum());



   // Also provided are some debugging print funtions which use less memory
   // overhead and easier to call from assembly

   dprint_mono(STRING,"Hello World (dprint)₩n");
   dprint_poly(INT,get_penum());

   return 0;

}
[root@harper01 hello_world]#
```

● 컴파일과 Link 및 실행 파일을 만들기

아래와 같이 컴파일 한다.

```
[root@harper01 hello_world]# cscn –o helloworld.csx helloworld.cn
```

.

● 실행하기

아래와 같이 csrun 명령어를 이용하여 실행한다.

```
[root@harper01 hello_world]# csrun ./helloworld.csx
Hello World 0
Hello World 1
Hello World 2
Hello World 3
Hello World 4
Hello World 5
Hello World 6
Hello World 7
Hello World 8
```

3.5 Benchmark를 위한 프로그램

HPL 및 DGEMM Benchmark를 위해 아래와 같은 프로그램 설치가 필요하다.

● MPICH 설치

```
[root@harper02 Down]# tar xzvf ClearSpeed.mpich.tar.gz
[root@harper02 Down]# cd mpich-1.2.5.2/
[root@harper02 mpich-1.2.5.2]# ls
acconfig.h  buildmsg  configure.in  f90modules  Makefile.in
MPI-2-C++  mpid  util
aclocal.m4  ccbugs  COPYRIGHT  include  makelinks
mpichconf.h.in  README  www
aclocal_tcl.m4  ClearSpeedLicense.txt  doc  installtest  man
mpich.dsw  romio  www.index
bin  configure  examples  KnownBugs  mpe
mpich.static.dsw  src
[root@harper02 mpich-1.2.5.2]# ./configure --prefix=/usr/local/mpich/
Configuring with args --prefix=/usr/local/mpich
Configuring MPICH Version 1.2.5 (release) of : 2003/01/13 16:21:53
checking whether filesystem respects case in file names... yes
```

```
checking for current directory name... /root/Down/mpich-1.2.5.2
checking for architecture... LINUX
checking device... ch_p4
checking for install
checking for ranlib
checking gnumake... yes using --no-print-directory
checking whether make supports include... yes
checking OSF V3 make... no
checking for virtual path format... VPATH
checking for xlC
checking for g++
checking if g++ returns correct error code... yes
Compiling C++ interface with g++
checking that selected C++ compiler can compile iostream.h... yes

Include C++ bindings for MPI from http://www.osl.iu.edu/research/mpi2c++
Send bug reports about the C++ to mpi2cpp-devel@osl.iu.edu

checking for g++ compiler exception flags... -fexceptions
checking for cc... found /usr/bin/cc (cc)
checking that the compiler cc accepts ANSI prototypes... yes
checking for f77... found /usr/bin/f77 (f77)
checking for f77... found /usr/bin/f77 (1)
checking for ar... found /usr/bin/ar (ar)
.......
[root@harper02 mpich-1.2.5.2]# make
[root@harper02 mpich-1.2.5.2]# make install
```

● HPL Make file 수정

```
[root@harper02 Down]# pwd
/root/Down
[root@harper02 Down]#
[root@harper02 Down]# tar xzvf ClearSpeed.hpl.tar.gz
```

```
[root@harper02 hpl]# pwd
/root/Down/hpl
```

```
TOPdir       = /root/Down/hpl

LAdir        = /opt/clearspeed

#LAdir        = /opt/clearspeed/csx600_m512_le/
LAdir        = /opt/clearspeed
LAinc        =
#LAlib        = -L$(LAdir)/lib/ -lcsxl -lm -lg2c -lpthread -lcsthreads
LAlib        = -L$(LAdir)/lib/ -lcsxl_mkl -lm -/usr/lib64/lg2c -lpthread -lcsthreads
```

```
 1069  make arch=ClearSpeed.no.profiler.support
 1070  cd /root/Down/hpl/src/panel/ClearSpeed.no.profiler.support
 1071  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1072  cd -; make arch=ClearSpeed.no.profiler.support
 1073  cd /root/Down/hpl/src/pauxil/ClearSpeed.no.profiler.support
 1074  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1075  cd -; make arch=ClearSpeed.no.profiler.support
 1076  cd /root/Down/hpl/src/pfact/ClearSpeed.no.profiler.support
 1077  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1078  cd -; make arch=ClearSpeed.no.profiler.support
 1079  cd /root/Down/hpl/src/pgesv/ClearSpeed.no.profiler.support
 1080  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1081  cd -; make arch=ClearSpeed.no.profiler.support
 1082  cd /root/Down/hpl/testing/matgen/ClearSpeed.no.profiler.support
 1083  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1084  cd -; make arch=ClearSpeed.no.profiler.support
```

```
 1085  cd /root/Down/hpl/testing/timer/ClearSpeed.no.profiler.support
 1086  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1087  cd -; make arch=ClearSpeed.no.profiler.support
 1088  cd /root/Down/hpl/testing/pmatgen/ClearSpeed.no.profiler.support
 1089  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1090  cd -; make arch=ClearSpeed.no.profiler.support
 1091  cd /root/Down/hpl/testing/ptimer/ClearSpeed.no.profiler.support
 1092  mv Make.inc Make.inc.old; ln -s ../../../Make.ClearSpeed.no.profiler.support
Make.inc
 1093  cd -; make arch=ClearSpeed.no.profiler.support
```

3.6 성능 측정

아래에서는 Haper01에서 hpcc의 DGEMM 성능과 SCX620의 DGEMM 성능을 비교 분석한다.

1. 호스트의 DGEMM Test

● Software Down

http://icl.cs.utk.edu/hpcc/overview/index.html

● Makefile 수정

hpcc-1.2.0/hpl/setup아래의 Makefile 중 선택하여 디렉토리 패스를 수정하여 /root/Down/hpcc-1.2.0/hpl로 복사한다.

● 컴파일 한다.

[root@harper01 hpcc-1.2.0]# pwd

/root/Down/hpcc-1.2.0

[root@harper01 hpcc-1.2.0]# make arch=Linux_PII_CBLAS

[참고] HPCC 프로그램에서는 기본적으로 DGEMM 테스트 파라미터를 수정하지 못하게 되어 있다.

1. **Optimized Runs**
    1. Code modification

    Provided that the input and output specification is preserved, the following routines may be substituted:

    - In HPL: HPL_pdgesv(), HPL_pdtrsv() (factorization and substitution functions)
    - *no* changes are allowed in the DGEMM testing harness and the substituted DGEMM routine (if any) should conform to BLAS definition
    - In PTRANS: pdtrans()
    - In STREAM: tuned_STREAM_Copy(), tuned_STREAM_Scale(), tuned_STREAM_Add(), tuned_STREAM_Triad()

소스를 분석했을 때 아래와 같이 /hpl/include/hpl_blash.h에 아래와 같이 소스에 DGEMM 파라미터가 정해져 있으며, 이를 수정해서 컴파일해도 실행되지 않는다.

```
[root@harper01 hpcc-1.2.0]# pwd
/root/Down/hpcc-1.2.0
[root@harper01 hpcc-1.2.0]# find . -name "*.h" -exec grep HplColumnMajor {} ₩; -print
```

```
{   HplRowMajor = 101,   HplColumnMajor   = 102 };
./hpl/include/hpl_blas.h
```

enum HPL_ORDER

{   HplRowMajor = 101,   HplColumnMajor   = 102 };

```
[root@harper01 hpcc-1.2.0]# /usr/local/mpich/bin/mpirun -machinefile   ./machines.HPL -np 1 ./hpcc
Order must be 101 or 102, but is set to 1001Parameter 1 to routine cblas_dgemm was incorrect
[root@harper01 hpcc-1.2.0]#
```

결과

```
Current time (1254407718) is Thu Oct   1 23:35:18 2009

End of HPL section.
Begin of StarDGEMM section.
Scaled residual: 0.002461
Node(s) with error 0
Minimum Gflop/s 7.422929
Average Gflop/s 7.422929
Maximum Gflop/s 7.422929
Current time (1254407725) is Thu Oct   1 23:35:25 2009

End of StarDGEMM section.
Begin of SingleDGEMM section.
Scaled residual: 0.00249844
Node(s) with error 0
Node selected 0
Single DGEMM Gflop/s 7.392672
Current time (1254407731) is Thu Oct   1 23:35:31 2009
```

/root/Down/hpcc-1.2.0/hpccoutf.txt 에서 메트릭스 사이즈 및 성능을 확인 할 수 있다.

2. CSX600의 DGEMM Test

```
Runtime package version: 3.11
CSX kernel version:          3004
Card type:                   e620
Fpga version number:         0x6f02c000
Fpga timestamp:              2007-08-16 10:08:17Z (0x46c42211)
Card serial number:          CLTJ07060076
Card final test date:        2008-03-28 01:47:20Z
Card location:               Bus 9, Device 0, Function 0
PCI Express link:            x4 (for optimum performance use a x8 slot)
Current Temperatures:        Core: P0 36, P1 31   Board: 34, 38, 30 degC
Core clock frequency:        210.0 MHz
ESRAM on each processor: 128KB
DRAM on each processor:   512MB running at 200 MHz
Processor 0 LMI - DRAM init complete.
Read fuses: 0x0, 0x0, 0x0, 0x80000000
Generated pe enable mask: 0xfbffdffe, 0xffefff7f, 0xf7ffbffd, 0x7fffffff
Processor 1 LMI - DRAM init complete.
Read fuses: 0x0, 0x0, 0x0, 0x80000000
Generated pe enable mask: 0xfbffdffe, 0xffefff7f, 0xf7ffbffd, 0x7fffffff
Reset OK for local hardware Instance: 0


--------------------------------
       ClearSpeed Environment
--------------------------------
CS_HOME=/opt/clearspeed
CSLIB=/opt/clearspeed/csa
CSPATH=.:/opt/clearspeed/csx:/opt/clearspeed/config
CSHOSTLIB=/opt/clearspeed/lib
CSPLATFORM=linux
CS_HOST_BLAS=/usr/lib64/libblas.so.3.0.3
CSINC=/opt/clearspeed/include:/opt/clearspeed/include/card/cn:/opt/clearspeed/include
/card/asm:/opt/clearspeed/include/common
CSHOSTINC=/opt/clearspeed/include/common:/opt/clearspeed/include:/opt/clearspeed/
```

```
include/host
CSHOME=/opt/clearspeed
CSSDKHOME=/opt/clearspeed
Iteration 0     C(1920, 1920),k=1152     Time: 0 min   0.194 sec   GF/s: 43.854444
Iteration 1     C(1920, 1920),k=1152     Time: 0 min   0.204 sec   GF/s: 41.704142
Finished.
Iteration 0     C(2880, 2880),k=1152     Time: 0 min   0.357 sec   GF/s: 53.538944
Iteration 1     C(2880, 2880),k=1152     Time: 0 min   0.384 sec   GF/s: 49.725739
Finished.
Iteration 0     C(3840, 3840),k=1152     Time: 0 min   0.621 sec   GF/s: 54.687266
Iteration 1     C(3840, 3840),k=1152     Time: 0 min   0.624 sec   GF/s: 54.450004
Finished.
Iteration 0     C(4800, 4800),k=1152     Time: 0 min   0.950 sec   GF/s: 55.883357
Iteration 1     C(4800, 4800),k=1152     Time: 0 min   0.932 sec   GF/s: 56.975532
Finished.
Iteration 0     C(5760, 5760),k=1152     Time: 0 min   1.313 sec   GF/s: 58.217797
Iteration 1     C(5760, 5760),k=1152     Time: 0 min   1.314 sec   GF/s: 58.169508
Finished.
Iteration 0     C(6720, 6720),k=1152     Time: 0 min   1.746 sec   GF/s: 59.605487
Iteration 1     C(6720, 6720),k=1152     Time: 0 min   1.741 sec   GF/s: 59.752201
Finished.
Iteration 0     C(7680, 7680),k=1152     Time: 0 min   2.250 sec   GF/s: 60.410758
Iteration 1     C(7680, 7680),k=1152     Time: 0 min   2.249 sec   GF/s: 60.436953
Finished.
Iteration 0     C(8640, 8640),k=1152     Time: 0 min   2.769 sec   GF/s: 62.120620
Iteration 1     C(8640, 8640),k=1152     Time: 0 min   2.767 sec   GF/s: 62.152430
Finished.
Iteration 0     C(9600, 9600),k=1152     Time: 0 min   3.368 sec   GF/s: 63.046182
Iteration 1     C(9600, 9600),k=1152     Time: 0 min   3.372 sec   GF/s: 62.964054
Finished.
Iteration 0     C(10560, 10560),k=1152   Time: 0 min   4.067 sec   GF/s: 63.175383
Iteration 1     C(10560, 10560),k=1152   Time: 0 min   4.052 sec   GF/s: 63.406487
Finished.
Iteration 0     C(11520, 11520),k=1152   Time: 0 min   4.824 sec   GF/s: 63.382745
Iteration 1     C(11520, 11520),k=1152   Time: 0 min   4.832 sec   GF/s: 63.281449
```

```
Finished.
Iteration 0     C(12480, 12480),k=1152   Time: 0 min   5.629 sec   GF/s: 63.751605
Iteration 1     C(12480, 12480),k=1152   Time: 0 min   5.623 sec   GF/s: 63.821732
Finished.
Iteration 0     C(13440, 13440),k=1152   Time: 0 min   6.465 sec   GF/s: 64.378158
Iteration 1     C(13440, 13440),k=1152   Time: 0 min   6.460 sec   GF/s: 64.420071
Finished.
Iteration 0     C(14400, 14400),k=1152   Time: 0 min   7.374 sec   GF/s: 64.785387
Iteration 1     C(14400, 14400),k=1152   Time: 0 min   7.373 sec   GF/s: 64.794261
Finished.
```

제 4 장 문제 해결

보드의 상태를 확인 하기 위해서 아래와 같이 Diagnostics 테스트를 한다.

```
[root@harper02 bin]# csdiag --all

Time: Tue Sep 22 00:51:56 2009
Host: harper02 Linux x86_64 2.6.18-92.el5
SVer: 3.11 (1.404.1.39 at Thu Aug 21 16:50:25 BST 2008 on linux_x86_64)
Card: Instance 0, Bus 11, Device 0 (1 card present)
        e620 (PCIe x4), FPGA 0x6f02c000, SN CLTJ08010009

Basic checks: PASS
Reset card:    PASS
Temperature:   PASS
Frequency:     PASS
PCI Read BW:   PASS
PCI Write BW: PASS
Combined BW:   PASS
Memory tests: PASS
Semaphores:    PASS
PIO tests:     PASS
PE memory:     PASS
Compute test: PASS
Temperature:   PASS
Frequency:     PASS

Overall result: PASS
[root@harper02 bin]#
```

csdiag 실행시 아래와 같이 Error가 발생할 경우

```
[root@minerva /opt/clearspeed/bin]# csdiag --all
csdiag: error while loading shared libraries: libc_buildversion.so: cannot open shared object file: No such file or directory
```

다음과 같이 환경파일에 디렉토리 경로는 작성하고 적용한다.

```
[root@minerva /opt/clearspeed/bin]# pwd
/opt/clearspeed/bin
[root@minerva /opt/clearspeed/bin]# ls
adbgt*  bashrc*  csdiag*  csgdb*  cshrc*  csinfo*  csled*  csreset*  csrun*
env/  power_mon*  temp_mon*  xsvfplayer*  zkey*
[root@minerva /opt/clearspeed/bin]# vi bashrc
# ClearSpeed runtime bash resource file - please source this before
# running any ClearSpeed tools.

export CSHOME=/opt/clearspeed
export CSPLATFORM=linux
if [ -z $CSPATH ]
then
export CSPATH=.:$CSHOME/csx:$CSHOME/config
else
export CSPATH=.:$CSHOME/csx:$CSHOME/config:$CSPATH
fi
export PATH=$CSHOME/bin:$PATH
if [ -z $LD_LIBRARY_PATH ]
then
export LD_LIBRARY_PATH=$CSHOME/lib
else
export LD_LIBRARY_PATH=$CSHOME/lib:$LD_LIBRARY_PATH
fi

# Source any additional ClearSpeed bash resource files

if [ -d $CSHOME/bin/env ]
then
FILES=`/bin/ls -1 $CSHOME/bin/env/*.bash 2>/dev/null`
for file in $FILES
do
source $file
done
fi
[root@minerva /opt/clearspeed/bin]# source bashrc
```

제 5 장 결 론

범용 컴퓨팅에서의 하드웨어 가속화는 아직은 보편화된 IT 기술이 아닌 최첨단 연구 분야에 국한되어 있다. 하지만 하드웨어 가속화는 고성능 컴퓨팅과 기업용 데이터 프로세싱의 통합이 증가됨에 따라 머지 않은 미래에 시장으로 확산될 수 있을 것이다.

가속화의 영역은 점점 넓어지고 있는 추세다. 초기의 가속화는 프로그래밍이 어렵고 하드웨어에 맞도록 애플리케이션을 조율해야 하는 어려움이 많았다. 그러나 현재는 가속 영역과 가속화를 위한 기술적인 진보가 점점 그 영역을 넓혀 나가고 있는 것이 새로운 추세다. 주로 사용되는 분야는 전통적인 연산 집중화 애플리케이션이 많은 제조분야의 CAE/CAD/CAM의 해석분야, 디지털 비디오 프로세싱과 렌더링과 같은 컴퓨터 그래픽스, 금융 분석, 정유 업계에서의 지진 이미지 프로세싱, 생명과학분야의 각종 모의실험(in silico experiment) 등이다. 상기의 열거분야는 실제 환경에서의 도입에는 아직 제한적이라고 볼 수 있지만, 기술자들이 프로그래밍과 기타 여러 문제들을 극복한다면 하드웨어 가속화 기술은 80년대의 벡터 칩에서 90년대의 RISC 기반 시스템, 2000년대의 클러스터로 진화한 것처럼 차세대 컴퓨팅을 주도할 수 있을 것으로 예상된다.

결론적으로 아직은 도입단계로 보이는 것이 사실이다. 그러나 그것은 현재 이러한 기술에 대한 인식이 국내에서만 떨어지고 있는 점이다. 세계 유수의 기업과 연구단체들은 점진적으로 사용자 기반을 넓히고 있음은 주목할 필요가 있는 기술이며, 그 활용범위는 날로 커질 수 있는 기술로 주목하고 있으며 또한 개발에 박차를 가하고 있다.

최근 세계 슈퍼컴퓨터의 통계인 Top500에 등장하고 있는 멀티코어/매니코어, 이기종(Heterogeneous) 기술을 적용한 슈퍼컴퓨터의 비중이 높아지는 것이 현실이다. 차세대 페타플롭스(PFlops)을 능가하는 슈퍼컴퓨터를 개발하기 위해서는 CPU 이외의 가속 기술이 공존해야한다는 것이 대세이다. 이러한 시점에서 Clearspeed와 같이 계산에 특화되어 있는 하드

웨어를 이용한 기술 확보는 의미를 가진다고 할 수 있다. 앞으로도 다양한 새로운 가속 기술에 대한 인식과 이해도를 넓히고, 다양한 애플리케이션에 가속화 기술 적용함으로써 가속 기술에 관한 기반 기술의 발전 및 축적에 노력을 기우려야 할 것이다.

참고문헌

[1] Owens, J.D. Houston, M. Luebke, D. Green, S. Stone, J.E. Phillips, J.C. "GPU Computing," Proceedings of the IEEE, Vol 96, pp.879 - 899. 2008.

[2] NVIDIA GUDA web page, http://www.nvidia.com/ object/cuda_home.html#

[3] clearspeed home page, http://www.clearspeed.com

[4] Top500 Supercomputing Sites, http://www.top500.org

[5] IBM web page, http://www-03.ibm.com/press/us/en/ pressrelease/24405.wss

[6] Xilinx homepage, http://www.xilinx.com

[7] DEGEMM Benchmark, http://linux.wareseeker.com/free-dgemm

[8] Titech homepage http://www.gsic.titech.ac.jp/~ccwww/

[9] Toshio Endo and Satoshi Matsuoka, "Massive Supercomputing Coping with Heterogeneity of Modern Accelerators", In Proceedings of IEEE International Parallel & Distributed Processing Symposium, 2008.

[10] clearspeed 기술 문서, "Clearspeed Advance e620 Accelerator," available at http://www.clearspeed.com

Clearspeed 설치 및 활용 가이드

2009 년 12 월 인쇄

2009 년 12 월 발행

발행처: KiSTi www.kisti.re.kr 한국과학기술정보연구원 Korea Institute of Science and Technology Information

대전시 유성구 과학로 335 번지

Tel: (042) 869-1004(代)

ISBN 978-89-6211-502-4 보급가 : 비매품