GRAPE-7 Installation Guide

for GRAPE-7 software package version 2.1

K & F Computing Research Co.
E-mail: support@kfcr.jp

Abstract

In this document, we describe installation procedure of GRAPE-7 system.

Contents

1 GRAPE-7 Overview

1.1 Architecture

GRAPE-7 is an FPGA-based reconfigurable add-in card. The card is attached to PCI-X or PCI Express I/O slot of the host computer (i.e. your own PC). It provides additional calculation resource to the host computer.

In the following we overview the architecuture of GRAPE-7. There exist four different models of GRAPE-7, namely, Model 100, Model 300, Model 600, and Model 800 (see figure 1).

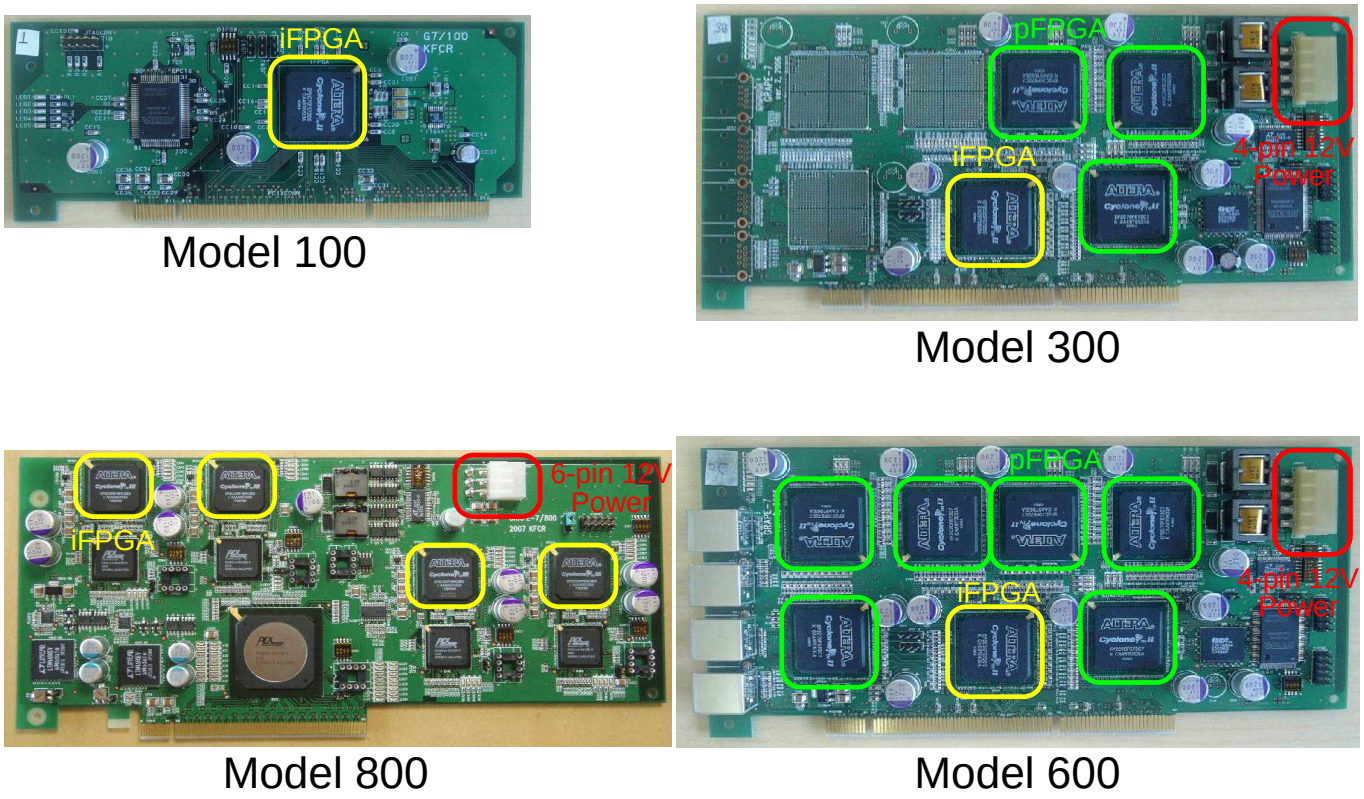

Figure 1: Photograph of GRAPE-7 add-in cards.

1.1.1 Model 100

Figure 2 shows a block diagram of GRAPE-7 Model 100. The add-in card houses a single FPGA, that we call **iFPGA**. The iFPGA contains an arbitrary calculation resource (backend logic) and an interface logic to the host computer (HIB). The backend logic is preconfigured to a pipeline named G5PIPE (described later) before shipping. User can overwrite this logic using *USB Blaster* download cable, which can be obtained from Altera Co.

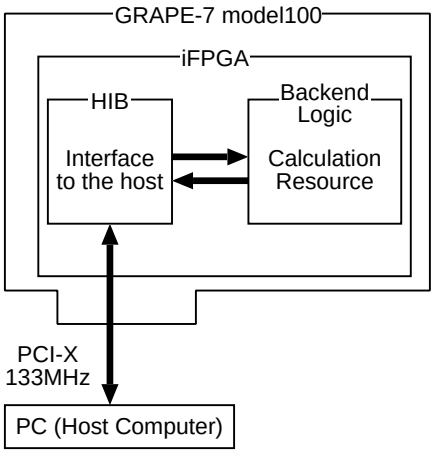

Figure 2: A schematic of GRAPE-7 Model 100.

1.1.2 Model 300 and Model 600

Figure 3 shows block diagrams of GRAPE-7 Model 300 and Model 600. These add-in cards houses three (Model 300) or six (Model 600) **pFPGA**s and a single iFPGA.

The pFPGAs are used as arbitrary calculation resources. The internal logic need to be reconfigured by the user, every time the card is powered on. See *GRAPE-7 Installation Guide* section 3.1 for the reconfiguration procedure.

The iFPGA handles data transfer from/to the host computer. Its internal logic is preconfigured and is not reconfigurable by the user.

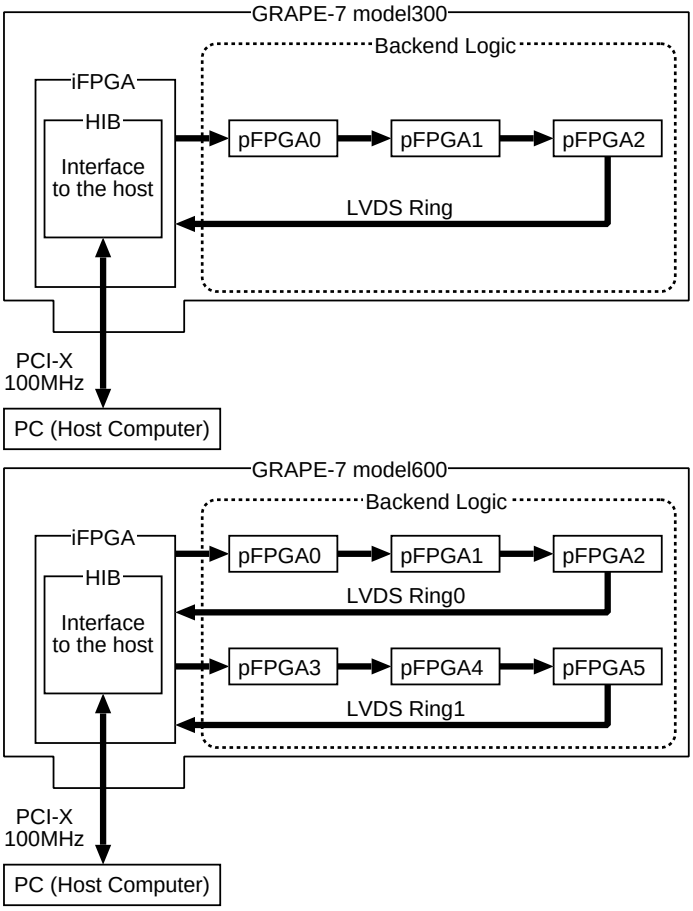

Figure 3: Schematics of GRAPE-7 Model 300 and 600.

1.1.3 Model 800

Figure 4 shows a block diagram of GRAPE-7 Model 800. The add-in card houses four iFPGAs. The iFPGA contains an arbitrary calculation resource (backend logic) and an interface logic to the host computer (HIB). The backend logic is preconfigured to a pipeline named G5PIPE (described later) before shipping. User can overwrite this logic from the host computer.

The host computer communicates with the four iFPGAs via a transparent PCIe switch and four PCIe/PCI-X bridges. The device driver on the host computer recognize these iFPGAs as four independent PCI-X devices. Via the user library, a user can handle these FPGAs as four independent devices, or as a single device.

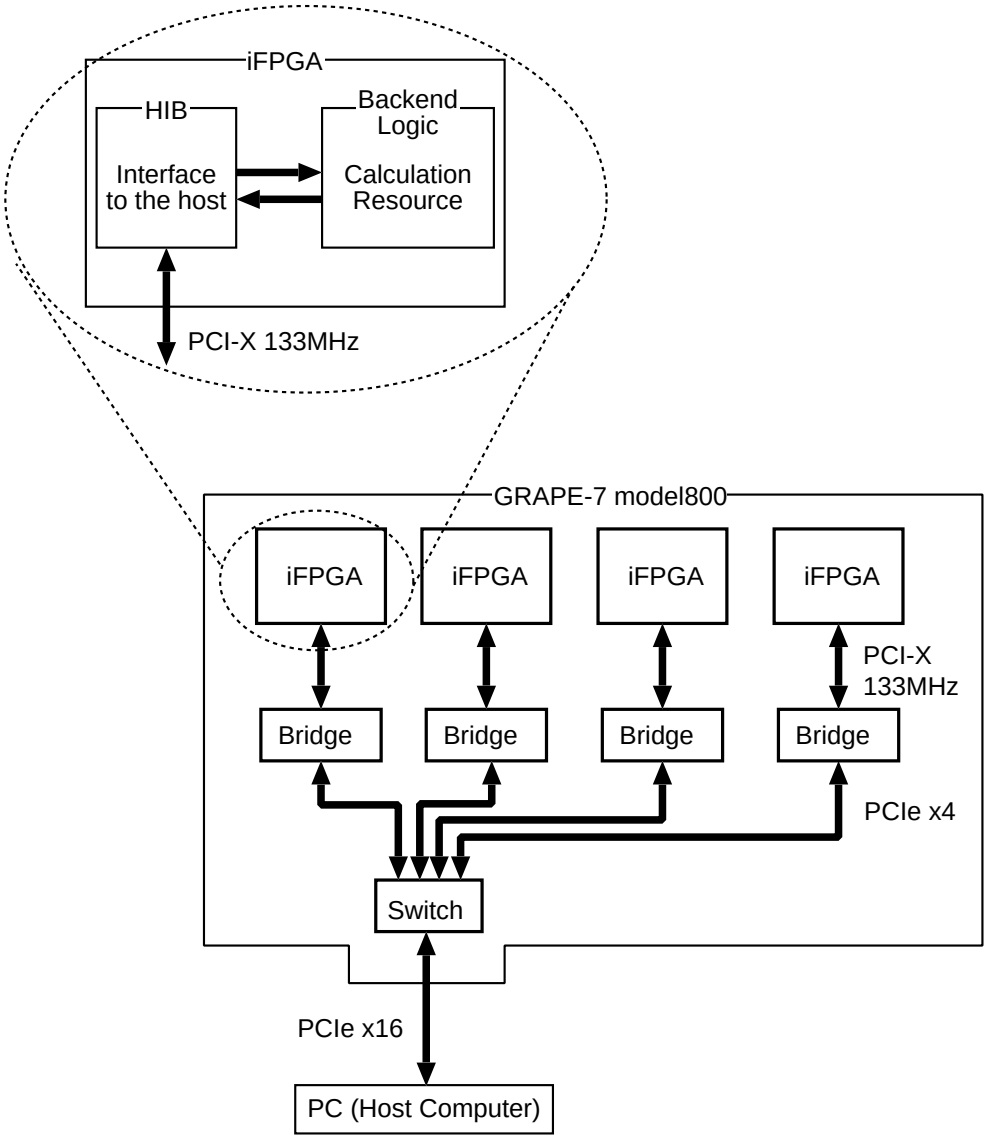

Figure 4: A schematic of GRAPE-7 Model 800.

1.2 Backend Logic

Two backend logics, **G5PIPE** and **G5nbPIPE** are provided with GRAPE-7. G5PIPE calculates gravitational forces among particles, just as traditional GRAPEs do. G5nbPIPE is an extention of G5PIPE which has an additional function to store neighbor-particle lists.

Backend logics other than G5PIPE and G5nbPIPE will be released at our company's Web site[1]. Also the user can reconfigure the backend logic with a pipeline designed by himself/herself. See section 4 for the reconfiguration procedure.

In the following, we overview the two backend logics, G5PIPE and G5nbPIPE.

1.2.1 G5PIPE

G5PIPE calculates gravitational forces among particles using hardwired pipelines (figure 5). All other calculations, such as time integration of particle orbits, are performed on the host computer. The pipeline is basically equivalent to that of GRAPE-5[1], except that it does not calculate gravitational potential nor does neighbor particles.

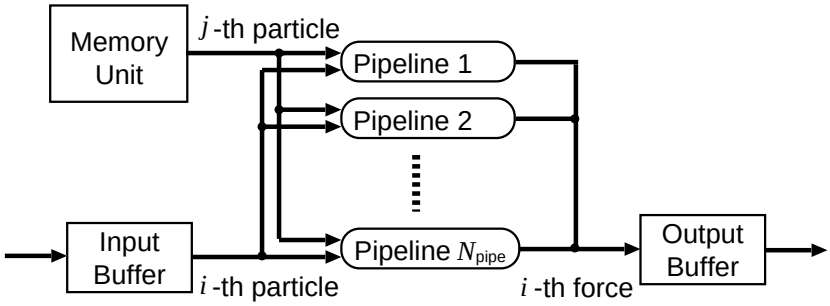

Figure 5: A schematic of G5PIPE.

In the following, we describe a force-calculation procedure using G5PIPE. Here and hereafter, we denote a particle at which force is evaluated as an i-particle, while a particle that exerts force as a j-particle.

step 1. The host computer sends a set of j-particles to the memory unit of G5PIPE. The number of j-particles sent to each G5PIPE should not exceed its memory size.

step 2. The host computer sends a set of i-particles to the input buffer of G5PIPE. The number of i-particles should not exceed the buffer size.

step 3. The host computer sends a command to indicate start of the runs.

step 4. From the input buffer, $N_{\rm pipe}$ particles are retrieved. Each of them is sent to one of $N_{\rm pipe}$ pipelines. Here, $N_{\rm pipe}$ is the number of pipelines integrated into one G5PIPE.

step 5. The pipelines start a run. Each pipeline calculates forces from all j-particles stored in the memory unit to its own i-particle. During the run, a j-particles is broadcasted to all $N_{\rm pipe}$ pipelines in every clock cycle. Each pipeline accumulates one pairwise

[1] `http://www.kfcr.jp`

force from a j-particle to its own i-particle into its internal register, in every clock cycle.

step 6. When a pipeline complete the run, i.e., it finishes accumulation of forces from all j-particles to its own i-particle, it sends the accumulated force to the output buffer of G5PIPE. Contents of the buffer are automatically sent back to the host (In the case of Model 300 and Model 600, outputs from all pFPGAs are accumulated before they are sent back to the host).

step 7. Step 4–6 is repeated until all i-particles in the input buffer are processed.

step 8. Now forces from all j-particles sent to the memory unit at step 1, to all i-particles sent to the input buffer at step 2, are calculated. The host computer repeats step 2–7 for different sets of i-particles, until all i-particles in the simulation are processed.

step 9. Now forces from all j-particles sent to the memory unit at step 1, to all i-particles in the simulation, are calculated. The host computer stores these forces to the main memory, and then process step 1–8 for a different set of j-particles. Obtained forces are added to the stored forces. The host computer repeats this procedure for yet another set of j-particles, until all j-particles in the simulation are processed.

In the GRAPE-7 software package, a function library to handle G5PIPE is included as `/usr/g7pkg/lib/libg75.a`. The library has backward compatibility with GRAPE-5 library. Its usage is documented in *G5PIPE User's Guide* (`/usr/g7pkg/doc/g5user.pdf`).

1.2.2 G5nbPIPE

G5nbPIPE is an extention of G5PIPE. It calculates gravitational forces among particles as well as G5PIPE does. In addition, it creates lists of neghbor particles and stores them into its internal memory, (shown in figure 6 as *NB Memory*). The peak performance of G5nbPIPE is lower than that of G5PIPE. This is because hardware resources required to implement one pipeline logic increased due to the additional function, resulting decrease of the number of pipelines, $N_{\rm pipe}$, integrated into the FPGA.

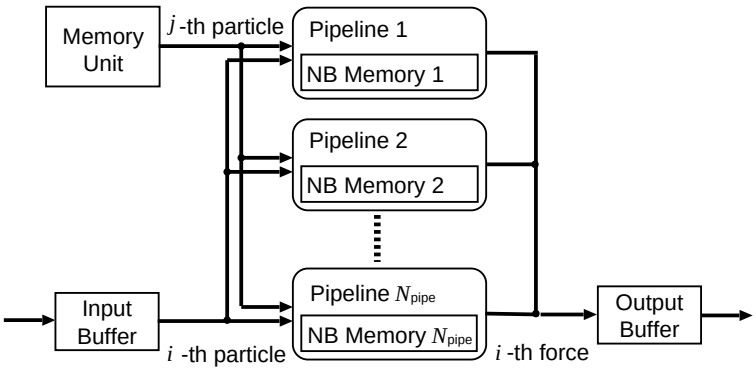

Figure 6: A schematic of G5nbPIPE.

In the GRAPE-7 software package, a function library to handle G5nbPIPE is included as `/usr/g7pkg/lib/libg75nb.a`. The library has backward compatibility with GRAPE-5 library. As for calculation of gravitational force, the usage is the same as that of G5PIPE, which is documented in *G5PIPE User's Guide* (`/usr/g7pkg/doc/g5user.pdf`). As for creation of neighbor-particle lists, the usage is documented in *G5nbPIPE User's Guide* (`/usr/g7pkg/doc/g5nbuser.pdf`).

1.3 HIB

Host Interface Bridge (HIB) is a logic that handles data transfer between GRAPE-7 and the host computer.

In the GRAPE-7 software package, a function library to directly handle HIB is included as `/usr/g7pkg/lib/libhib.a`. Its usage is documented in *HIB Library Functions Reference Manual* (`/usr/g7pkg/doc/hibref.pdf`).

A user of G5PIPE or G5nbPIPE needs no knowledge about this library. It is wrapped by G5PIPE/G5nbPIPE library functions, and is completely hidden to the user.

A user of a backend logic which is designed by her/himself, the user need to control the logic directly using the HIB library.

2 Installation

2.1 Hardware Installation

2.1.1 Model 100

Insert GRAPE-7 add-in card into the PCI-X 133MHz slot of the host computer. Although the card operates at arbitrary clock cycle lower than or equal to 133MHz, we recommend to use 133MHz slot if available, so that it offers the highest performance. Connect the 4-pin 12V power cable of the power supply unit to the connector, as shown in figure 7. **Maximum care should be taken for cooling.**

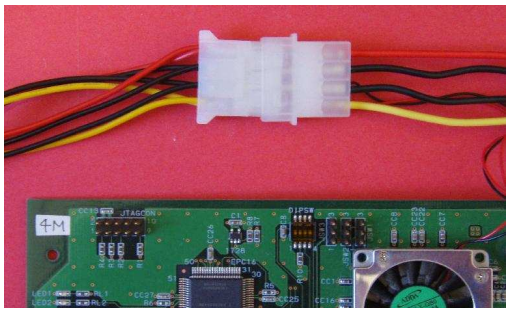

Figure 7: Power cable connection (Model 100).

2.1.2 Model 300 and 600

Insert GRAPE-7 add-in card into the PCI-X **100MHz** slot of the host computer. Model 300 and Model 600 operates at 100MHz only. The clock cycle should not be lower nor higher than 100MHz. Connect the 4-pin 12V power cable of the power supply unit to the connector, as shown in figure 8. **Maximum care should be taken for cooling.**

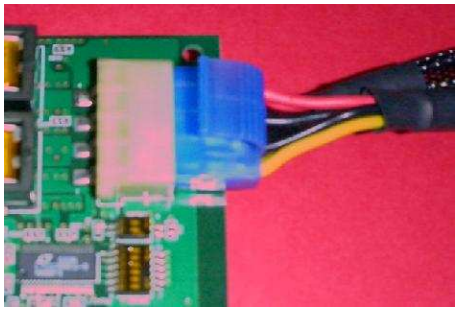

Figure 8: Power cable connection (Model 300 and Model 600).

2.1.3 Model 800

Insert GRAPE-7 add-in card into the 16-lane PCI Express slot of the host computer. Connect the 6-pin 12V power cable of the power supply unit to the connector, as shown in figure 9. **Maximum care should be taken for cooling.**

Figure 9: Power cable connection (Model 800).

2.2 Software Installation

All softwares are packed into a single package `g7pkg.tar.gz` which you can find at `http://www.kfcr.jp/support-e/`.

2.2.1 Supported environment

The GRAPE-7 software currently supports Linux system (kernel 2.6 x86_64) only. Note that **a complete source tree of the Linux kernel is required** for successful installation. The package is tested on the following platforms:

- Fedora Core 4, 5
- WhiteBox Enterprise Linux 4
- CentOS 5
- Red Hat Enterprise Lunux 4, 5

It is expected to work also on CentOS 4. It is known that the current version does not work on Fedora Core 6, though it will be supported soon.

2.2.2 Unpacking Software

Download the package `g7pkg.tar.gz` into your local disk system, and type

```
tar xvfz g7pkg.tar.gz
```

to unpack it. This will creates a directory named `g7pkg`, and retrieves all softwares in that directory. In the following we assume the path you unpacked the package is `/usr/g7pkg`. If it is not, replace `/usr/g7pkg` below with appropriate path name.

The package contains following items. Please identify each item before you start installation.

```
./00readme    --  a brief instruction of the package.
./00readme-j  --  00readme in Japanese.
./doc/        --  user's guide, reference manual, and other documents.
./scripts/    --  install & backup scripts.
./include     --  header files.
./lib         --  libraries.
./driver/     --  device driver.
./g5util/     --  G5PIPE user library and utility. source codes and pFPGA
                  configuration files (.ttf) are included.
./hibutil/    --  Host Interface Bridge (HIB) user library and utility.
                  source codes are included.
./direct      --  a sample code (direct-summation algorithm in C).
./directf77   --  a sample code (direct-summation algorithm in Fortran77).
./vtc         --  a sample code (Barnes-Hut tree algorithm in C).
```

2.2.3 Installing Software

In order to start an installation script, change working directory to `/usr/g7pkg`, and type `./scripts/install.csh`

The script asks you several questions, such as the number of the cards you have and some miscellaneous stuffs. After that, it will install everything automatically.

```
localhost>./scripts/install.csh
-----------------------------------------------
GRAPE-7 software package installation program
version 0.3 22-Jan-2007
-----------------------------------------------
Using Linux kernel version 2.6 or later? (y/n): y
How many GRAPE-7 cards are you installing?: 1
.................
done
-------------------------------------------------------------------------------
All installation process has been completed.
Now you can check the functions of GRAPE-7 card(s)
following the procedure below:
.................
NOTE THAT STEP (1)-(3) ARE NECESSARY EVERYTIME YOU RESTART THE HOST COMPUTER.
-------------------------------------------------------------------------------
```

When the installation completes, you will see the libraries at:

```
/usr/g7pkg/lib/libhib.a
/usr/g7pkg/lib/libg75.a
/usr/g7pkg/lib/libg75nb.a
```

Here, `libhib.a`, `libg75.a` and `libg75nb.a` are libraries for the HIB, G5PIPE and G5nbPIPE, respectively. Header files should be found at:

```
/usr/g7pkg/include/grape7x.h
/usr/g7pkg/include/hibutil.h
/usr/g7pkg/include/g5util.h
/usr/g7pkg/include/g5nbutil.h
/usr/g7pkg/include/typedef.h
/usr/g7pkg/include/gp5util.h
```

Here, `hibutil.h`, `g5util.h`, and `g5nbutil.h` are headers for the HIB, G5PIPE and G5nbPIPE, respectively. `grape7x.h` is a header for the device driver, and `typedef.h` is a header included from other headers. `gp5util.h` is a symbolic link to `g5util.h`. It exists just for backward compatibility.

> **Need for re-installation:** You need to rerun `install.csh`, when you add new cards into a preconfigured GRAPE-7 system. You also need to rerun the command when you remove some of the cards from the system.

3 Setting up the System

3.1 Startup Utilities

Now all the installation process is completed. Next, you need to set up the device driver and MTRR registers on the host computer, and then reconfigure the pFPGAs on the

add-in card. Follow the procedure below to complete the set up. **Note that steps 1–3 below are necessary everytime you restart the system.**

Step 1 (Root Permission Required): Here we describe a procedure to plug-in the GRAPE-7 device driver into the Linux kernel. Change directory to `/usr/g7pkg/driver`, and then type

```
make installmodule
```

This will plug-in the driver into the kernel.

```
root@localhost# make installmodule
./install.csh grape7x
-- install module grape7x --
1 GRAPE-7(s) found.
rm -f /dev/grape7x[0-9]
/sbin/insmod -f grape7x.ko
mknod /dev/grape7x0 c 253 0
chgrp wheel /dev/grape7x0
chmod 666 /dev/grape7x0
ls -l /dev/grape7x0
crw-rw-rw-  1 root wheel 253, 0 Feb 12 21:14 /dev/grape7x0
-- done --
```

In order to confirm that the driver is properly loaded, type `/sbin/lsmod`. You should see the following lines:

```
Module                  Size   Used by
grape7x                 XXXX   0
```

Step 2 (Root Permission Required): Here we describe a procedure to set up MTRR (memory type range register) of the host computer. Change directory to `/usr/g7pkg/hibutil`, and then type `./setmtrr.csh`

```
root@localhost# cd ../hibutil/
root@localhost# ./setmtrr.csh
Searching for HIB(s)...Found 1 PCI-X HIB(s).
Trying to set MTRR(s)...
    echo "base=0xfeaf8000 size=0x1000 type=write-combining" > /proc/mtrr
Done.
current setting of MTRRs:
.................
regXX: base=0xfeaf8000 (4074MB), size=   4KB: write-combining, count=1
```

This will set MTRR to "write-combining" mode, which improve speed of Programmed I/O Write (PIOW) data transfer. It affects performance of sending data from the host

computer to GRAPE-7.

> **When `setmtrr.csh` failed:** In some case MTRR cannot be set up to "write-combining" mode (For example, all 8 existing MTRR are already assigned to other PCI devices, or, the total size of the main memory exceeds 4GB and the chipset of the mother board does not support I/O mapping to the main memory address higher than 4GB). In such cases, you can continue all installation procedure described below, without MTRR set up. All functions of GRAPE-7 should work without problem, except that the speed of data transfer from the host computer to GRAPE-7 would be reduced by 50% or more.
> If you use G5PIPE or G5nbPIPE as the backend logic, you can set an environment variable `G5_SENDFUNC` to supress the reduction of the tranfer speed. See *G5PIPE User's Guide* section 3.2 for the detail.
> Note that MTRR affects speed of data transfer from the host computer to GRAPE-7 only. It does not affect transfer of inverse direction, that is, transfer from GRAPE-7 to the host.

Step 3 (Model 300 and Model 600 Only): Here we describe a procedure to configure pFPGAs with G5PIPE backend logic. This procedure is not necessary for Model 100 and Model 800.

Change directory to `/usr/g7pkg/g5util/config`, and confirm directories in which backend logics are placed.

```
localhost> ls -F
Makefile  boardinfo@  config.c  g5p16nb24c100MHz/  g5p20c100MHz/
p1_grav.ttf@  p2_grav.ttf@  p6_grav.ttf@  sample.c
```

A G5PIPE backend logic is placed in the directory `g5p20c100MHz`. It integrates 20 pipelines per pFPGA, each of which operates at 100MHz clock cycle. A G5nbPIPE is placed in the directory `g5nbp16nb24c100MHz`. It integrates 16 pipelines per pFPGA, each of which operates at 100MHz clock cycle.

In order to to configure the pFPGAs with G5PIPE, change directory to `g5p20c100MHz` and type

`/usr/g7pkg/scripts/config` [*device_id*]

This will reconfigure the pFPGAs on the *device_id*-th card. If the argument *device_id* is not given, device ID 0 is assumed. If you have only one GRAPE-7 card in the system, its device ID should always be 0, and thus you can omit the argument. If you have multiple cards, you need to confirm the device ID of the card to be configured. Use a command `/usr/g7pkg/scripts/lsgrape` to identify the device ID. Usage of `lsgrape` can be found in section 5.

```
localhost> cd g5p20c100MHz
localhost> /usr/g7pkg/scripts/config 0
```

```
devid 0 model 600 npipe 16
**** Configuration for model600 [1-6] ****
fin0: 1789902
fin1: 1789902
fin2: 1789902
nconf = 0x3f
datacount = 80000 524288
.................
nconf = 0x24
datacount = 80000 524288
datacount = 100000 1048576
datacount = 180000 1572864
configuration finished
```

To see detail of the configured backend logic, give `-v` option to `lsgrape`:

```
localhost> /usr/g7pkg/scripts/lsgrape -v
devid grape(model)              backend-logic
 0    GRAPE-7(Model600)         G5
        number of chips             : 6
        number of pipelines/chip    : 20
        j-particle memory size/chip : 4096 particles
        fout fifo size              : 8192 bytes
        P3M cutoff                  : available
        neighbor search             : available
        number format               : floating-point
        gravitational potential     : not available
```

3.2 Test Programs

Now your GRAPE-7 system is ready to run many-body time integration programs using G5PIPE backend logic. However, before running your own code, we recommend to check its functionality following the procedure below:

Step 1 (Multiple Cards User Only): Before you start the test, you need to set an environment variable `G5_CARDS` to specify a single card. For example, in order to test a card with device ID 2, you should set the variable as:

```
csh> setenv G5_CARDS 2
 sh> export G5_CARDS=2
```

Device IDs of the cards are obtained using `lsgrape` utility. For its usage, see section 5.

Step 2: In order to start the test, change directory to `/usr/g7pkg`, and type

```
./scripts/check.csh
```

The script runs many-body simulations on the specified card with various numbers of particles. The final distribution of all particles for each run is saved to a temporary file. Then the file is compared with its corresponding in `/usr/g7pkg/direct/snapshots`. The test takes several minutes. If you see any error message during the test, please make a contact with us at `support@kfcr.jp`.

Output Example:

```
localhost>./scripts/check.csh
------------------------------------------------------------------
 GRAPE-7 functionality test program
 version 0.3    08-Feb-2007
.................
---------- in: pl1k out: tmp3052_pl1kT100.snap endt: 100 ----------
nj: 1024
n: 1024 outfile: tmp3052_pl1kT100.snap endtime: 100.000000 n: 0
.................
OK
---------- in: pl12k out: tmp3052_pl12kT100.snap endt: 100 ----------
nj: 12288
n: 12288 outfile: tmp3052_pl12kT100.snap endtime: 100.000000 n: 0
.................
OK
Passed all tests.
```

You will see a message "Passed all tests." at a successful completion of the tests.

Now all the installation procedure is completed. You can use G5PIPE with your own application code. See *G5PIPE User's Guide* (`/usr/g7pkg/doc/g5user.pdf`) for usage of G5PIPE.

4 FPGA Reconfiguration

The FPGAs of GRAPE-7 can be reconfigured to arbitrary backend logics other than G5PIPE. In this section, we describe a procedure for reconfiguration.

4.1 Model 100

There exist two different methods to reconfigure iFPGA of Model 100. For both methods you need a download cable *USB Blaster*[2] and a design software *Quartus II*[3], which are available from Altera Co. You can download a backend logic from a PC to the iFPGA via the download cable connected to the JTAG port of GRAPE-7, using *Programmer* utility

[2]http://www.altera.com/support/devices/tools/altera/cables/tls-altera-cables.html
[3]http://www.altera.com/products/software/products/quartus2/qts-index.html

of Quartus II suite. Before starting the reconfiguration, connect the USB Blaster cable to the JTAG port of Model 100 (figure 10). For usage of Quartus II, see its manual.

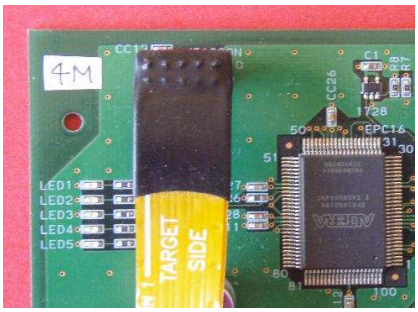

Figure 10: The USB Blaster Download Cable connected to the JTAG port of Model 100.

Method 1 (temporal reconfiguration): You can download a backend logic to the iFPGA using Programmer utility of Quartus II suite. The logic should be prepared in SRAM Object File (.sof) format. When the dowload is completed, you need to reboot the host computer. Note that you should reboot it without powering off (i.e., you need a hot reboot), otherwise the downloaded data would be evaporated.

Method 2 (permanent reconfiguration): You can download a backend logic to a configuration ROM attached to the iFPGA, using Programmer utility of Quartus II suite. The logic should be prepared in Programmer Object File (.pof) format. When the download is completed, you need to restart the host computer and GRAPE-7 add-in card. When the add-in card is powered on, the data downloaded to the configuration ROM is automatically loaded to the iFPGA. Note that you should power it off and on again (i.e., you need a cold reboot). You do not need to reconfigure the ROM everytime you restart the host computer. The data once downloaded to the ROM does not evaporate when it is powered off.

4.2 Model 300 and Model 600

For Model 300 and Model 600, you do not need any special device or additional software for reconfiguration. You can reconfigure pFPGAs of Model 300 and Model 600 following the procedure described in section 3.1 Step 3.

4.3 Model 800

For Model 800, you do not need any special device or additional software for reconfiguration. Using a command `/usr/g7pkg/scripts/config800`, you can download a backend logic directly from the host computer to configuration ROMs attached to iFPGAs. The logic should be prepared in Raw Programming Data file (.rpd) format. When the download is completed, you need to restart the host computer. When the add-in card is booted, the data downloaded to the configuration ROMs are automatically loaded to the iFPGAs.

Note that you should power it off and on again (i.e., you need a cold reboot). You do not need to reconfigure the ROM everytime you restart the host computer. The data once downloaded to the ROMs do not evaporate when they are powered off.

The command `config800` download a given .rpd file to configuration ROMs specified by their device IDs. For example, on a system with one Model 800 add-in card installed, you can type

```
/usr/g7pkg/scripts/config800 g5.rpd 0 1 2 3
```

to download a backend logic `g5.rpd` into all four ROMs attached to the card's iFPGAs (each of which has device ID 0 to 3). For usage of the command `config800`, see section 5.

When the download is completed, you need to restart the host computer and GRAPE-7 add-in card. When the add-in card is powered on, the data downloaded to the configuration ROMs are automatically loaded to the iFPGAs. Note that you should power them off and on again (i.e., you need a cold reboot). You do not need to reconfigure the ROMs everytime you restart the host computer. The data once downloaded to the ROMs do not evaporate when they are powered off.

5 Command Reference

Here we describe usage of commands included in the GRAPE-7 software package. All commands are placed in a directory `/usr/g7pkg/scripts/`. Some of them are placed also in `/usr/g7pkg/hibutil/` or `/usr/g7pkg/g5util/config/` for backward compatibility.

`install.csh` — install the software package

SYNOPSIS: `install.csh`

DESCRIPTION: `install.csh` compiles all source codes included in the GRAPE-7 software package. You need to run this command when you install the package for the first time. You need to run it also when you add new cards into a preconfigured GRAPE-7 system, or when you remove some of the cards from the system.

`bakup.csh` — make a snapshot of the software package

SYNOPSIS: `bakup.csh` [*package_name*]

DESCRIPTION: `bakup.csh` makes your own tarball of the GRAPE-7 software package. Change directory to `/usr/g7pkg` and run `./scripts/bakup.csh` with no argument to generate a package named `g7pkg`. Name of the tarball will be `g7pkg.tar.gz`, and name the top most directory archived into the tarball will be `./g7pkg`. The package name `g7pkg` is replaced with the first argument of the command, if it is given.

```
localhost>./scripts/bakup.csh
-----------------------------------------------
GRAPE-7 software package back up program
version 0.3 22-Jan-2007
-----------------------------------------------
Specify the root name for the package (default g7pkg): g7pkg20070212
./00readme
./g5util/Makefile
./g5util/g5util.h
.................
./g7pkg20070212/scripts/check.csh
./g7pkg20070212/scripts/gensnap.csh
Created ./g7pkg20070212.tar.gz
```

In order to uninstall the software package, just remove everything in `/usr/g7pkg`.

```
localhost> cd /usr
localhost> rm -rf g7pkg
```

`setmtrr.csh` — **set up MTRR**

SYNOPSIS: `setmtrr.csh`

DESCRIPTION: `setmtrr.csh` sets up MTRR (memory type range register) to "write-combining" mode. This improves speed of Programmed I/O Write (PIOW) data transfer.

`lsgrape` — **list GRAPE add-in cards**

SYNOPSIS: `lsgrape [-l] [-v] [-d` *device_id*`] [-h]`

DESCRIPTION: `lsgrape` shows information about GRAPE add-in cards installed in the system.

OPTIONS:

`-l`
shows detailed information about the cards, such as the number of pipelines and the size of memory unit.

`-v`
works as the same as `-l`.

`-d` *device_id*
shows information about only one card specified by the device ID *device_id*. Without this option,
tt lspci shows information about all cards installed.

EXAMPLES:

```
localhost>/usr/g7pkg/scripts/lsgrape
devid grape(model)       backend-logic
  0   GRAPE-7(model100)  G5
  1   GRAPE-7(model300)  G5
  2   GRAPE-7(model300)  empty
```

This output is showing that the system has one Model 100 and two Model 300s. It is also showing that the Model 100 and one of the two Model 300s are configured with G5PIPE backend logic.

```
localhost> /usr/g7pkg/scripts/lsgrape -v -d 0
devid grape(model)              backend-logic
 0    GRAPE-7(Model600)         G5
```

```
number of chips             : 6
number of pipelines/chip    : 20
j-particle memory size/chip : 4096 particles
fout fifo size              : 8192 bytes
P3M cutoff                  : available
neighbor search             : not available
number format               : floating-point
gravitational potential     : not available
```

This output is showing that the system has one Model 600 as a device of ID 0, which is configured with G5PIPE backend logic. It is also showing that the G5PIPE integrates 20 pipelines and a memory unit of size 4096 for each pFPGA.

`config` — configure pFPGAs of Model 300 and Model 600

SYNOPSIS: `config` [*device_id*]

DESCRIPTION: `config` downloads a backend logic to all pFPGAs of Model 300 or Model 600. The logic should be prepared in Tabular Text File (.ttf) format, and should be placed in the current directory. The argument *device_id* specifies device ID of the card to be configured. If the argument is not given, device ID 0 is assumed. Use a command `/usr/g7pkg/scripts/lsgrape` to identify the device ID. Usage of `lsgrape` can be found in section 5.

`config800` — configure iFPGAs of Model 800

SYNOPSIS: `config800` <*rpd_file*> [*ifpga_ids*]

DESCRIPTION: `config800` downloads a backend logic *rpd_file* to the four configuration ROMs attached to iFPGAs of Model 800. The logic should be prepared in Raw Programming Data file (.rpd) format. The second argument *ifpga_ids* is a space-separated list of device IDs, which specifies iFPGAs to be configured. If it is not given, four iFPGAs with device ID 0 to 3 are assumed. Use a command `/usr/g7pkg/scripts/lsgrape` to identify the device ID. Usage of `lsgrape` can be found in section 5.

EXAMPLES:
For example, on a system with one Model 800 add-in card installed, you can type

```
/usr/g7pkg/scripts/config800 g5.rpd 0 1 2 3
```

to download a backend logic `g5.rpd` into all four ROMs attached to the card's iFPGAs (each of which has device ID 0 to 3).

References

[1] Kawai A., Fukushige T., Makino J., and Taiji M.,
GRAPE-5: A Special-Purpose Computer for N-Body Simulations,
Publ. Astron. Soc. Japan (2000), Vol. 52, p. 659,
`http://xxx.lanl.gov/abs/astro-ph/9909116`.

Modification History

version	date	description	author(s)
2.1	09-Feb-2008	Support for Model 800. Support for G5nbPIPE. Section "FPGA Reconfiguration" added. Section "Command Reference" added.	AK, TF
1.0	13-Feb-2007	Description for new features added.	AK
0.0	26-Jan-2007	Created based on the *GRAPE-7 User's Guide*.	AK