

Comparison of the Hybrid and Wafer Data

V 1.0

Vitaliy Fadeyev

September 16, 2002

Introduction

In the course of testing the hybrids and modules, it is sometimes useful to compare the results with the wafer-level tests of the corresponding chips. When the ASIC part of the SCT database becomes fully functional, that would be the primary, and most efficient way to retrieve the information. Right now we are left with the "backdoor" way of obtaining the relevant information from the "raw" data. This note is meant to be an outline of a step-by-step procedure of getting the information. The procedure is admittedly involved, but the backdoor way never was meant or designed to be proper, nice and convenient.

The procedure

The procedure of getting the information consists of three actions:

1. fetching the file with data,
2. processing the file,
3. looking at the data, usually with a help of ROOT macros (scripts).

All the wafer measurement data are kept at CERN. We process them locally, at *vostok-lnx.lbl.gov* machine running *linux*, under *lblatlas* account.

Fetching the data file

So, the steps to get the file are:

1. Log in on *atlas.cern.ch* (have to have an account there; I recommend to use *ssh* to log in)
2. Find the relevant *zip* file. The data files are at the so-called CASTOR storage. The storage space is transparent for a casual user, but you have to use special commands to look at and to get the data. The command

```
prompt> nsls /castor/cern.ch/user/s/sctwafer/data/
```

would give you the list of directories with data files. It is somewhat analogous to the unix "*ls*" command. Each directory corresponds to a given tested wafer. The format of the directory names is

```
<Test Site/Machine Name>-Z<lot #>-W<wafer #>-D<Test Date>
```

Please note that there may be more than one test/directory for a given wafer. I recommend "*grep*"ing the "*nsls*" command output for a known lot number when you are looking for a wafer data. Once you know the

directory name, you can look inside with the same "ns/s" command. Typically you'd see a *zip* file with the "raw" data and a root file with the processed results. In principle, one wants the *root* data file. However, there is a problem that the root data file is larger than one's allowed home directory disk space, so it is not possible to copy it over.

3. So, the next step is to copy the zip file to the home directory of the login account you are using:

```
prompt> rfcop /castor/cern.ch/user/s/sctwafer/data/<dir name>/<zip file>.zip ./
```

At this point the system may seem to hang up for a few seconds to a few minutes. This is due to the nature of the CASTOR storage.

4. Transfer the file to *vostok-lnx* machine:

```
prompt> scp -pr <zip file> lblatlas@vostok-lnx.lbl.gov:/datafiles/
```

The *scp* command will prompt you for the *lblatlas* account password. It may take a few minutes to transfer the file. The *scp* is nice enough to report on the transfer progress continuously.

Processing the data file

Processing the data is as easy and straightforward procedure as getting them. The following steps are involved:

1. Log in to *vostok-lnx* as *lblatlas*. You have to set up the right environment for the ROOT-related data processing. The command is

```
prompt> . ~/set_LD_LIBRARY_PATH_v7_7
```

Please not the dot at the beginning of the command.

2. Change to the data storage directory.

```
prompt > cd /datafiles
```

"/datafiles" is a dedicated large disk for the data storage. Therefore, we keep the zip and root files on that disk, and analyse them from the *lblatlas* home area.

3. Make a proper directory for the zip file and move the file to it. "cd" to that directory. You can also copy the file to the "/datafiles/zipfiles" directory, if you care to save it.

4. Unpack the *zip* file, i.e.

```
prompt> unzip <zip-file>
```

Remove the *zip* file, you do not need it anymore.

Depending on the directory structure of the host machine where the *zip* file was generated, there may be a directory tree unpacked. What you want is just "*Chip*.dat*" data files. If there is a tree, please move the data files to the current directory and remove the extraneous directories.

5. Now you are in the position to really process the data. There are two steps involved:

- Making the *root* file (the bulk of the processing).

Copy "wafer.map" file over, i.e.

```
prompt>cp ../wafer.map ./
```

Run the executable making the *root* file, i.e.

```
prompt> ../tstWafer >& run_tstWafer.log &
```

The program will run in the background, and the log file with its output will be created. You do not necessarily want to look at it, but it's a good idea to leave traces of the activity in case something strange happens later on.

This is the bulk of the data processing. It takes 1-2 hours to get thru.

The file "*kk.root*" will be created. It's a good idea to rename it in the same fashion the *zip* file was named, i.e. preserving the test site name, wafer ID numbers and test date.

It is also a good idea to "save" the newly created "bare" *root* file, i.e.

```
prompt> cp <basename>.root ../rootfiles/
```

- Now you need to reprocess the file to collect the physical observables into the *Chip* object status registers. A typical command would be

```
prompt> ../MkDST <basename>.root 0 1 1 800. 1 1 0.85 >& run_mkdst.log &
```

Of the *MkDST* command line arguments, only number "800." is important at this point. It means the noise cut level (noisy chips are "cut off"). In principle, this value can be derived from the chip noise distribution for the wafer by running yet another macro. However, the number "800." is sufficiently large to be very inclusive for our purposes.

It takes a couple of minutes for the program to run. At this point you have a properly made *root* file.

Looking at the data

Now you can run a macro in the *root* session to compare wafer test data with hybrid/module data. Change directory to the proper one.

```
prompt> cd /home/lblatlas/xcheck_chips_hybrid/
```

Make a link to the newly created *root* file, for example,

```
prompt> ln -s /datafiles/SCIPPW-Z40859-W02-D20020503/SCIPPW-Z40859-W02-D20020503.root scippw-z40859-w02-d20020503.root
```

There are several macros in the current directory, designed for different purposes.

Bad channel checks

To check the bad channel list from the wafer data for a chip, use "*lkup_chip_channels.C*". It has the following three arguments:

1. (int) chip number on the wafer, starting from 1 (this is what supplied with the gelpacks),
2. (char *) the name of the root file with the wafer test results,
3. (int, optional) the chip number on the hybrid, starting from 0.

To run the macro, you type

```
prompt> root
```

this starts the *root* session. At the *root* prompt, you type something like

```
root prompt> .x lkup_chip_channels.C( 123, "simple.root", 3)
```

Then the macro runs, and the results are printed on the screen.

To check the bad channel list for the 12 chips on a hybrid, you can do the following:

1. take "*lkup_E12_channels.C*" routine, rename it appropriately (E12 --> your hybrid name)
2. in the code itself
 - rename the routine name,
 - replace the chips numbers with yours
("-1" operation is to correct the numbering from the one supplied with the gelpack info to the one used internally in the root file; the latter starts from 0),
 - replace the root file name in the line "TFile f...." .

Now you can run the new macro. It does not have arguments, because there is quite a bit of information to supply. So, I resolved to multiplying essentially the same routine many times and modifying the internals. This method has the advantage that the information stays in the files, which is useful for re-running the routines or tracing the bugs.

Comparison of the two tests

Another common task is to check the compatibility of the wafer- and hybrid-level results for a chip. One can use macro "*lkup_chip_hybrid.C*" to compare the gain and offset. It has the following arguments:

1. (int) the number of the chip on the wafer (information provided with the gelpacks),
2. (char *) text file with the channel-wise hybrid test results
3. (char *, optional) the base name of the output picture files (*.ps and *.gif)

Note that there is no root file name in the argument. This macro expects that the root file would already be opened in the *root* session:

```
root prompt> .x rootlogon.C
```

```
root prompt> TFile f("simple.root")
```


The printout is done separately for the gain and offset comparisons. Of these, the offset is the most reliable, since it varies much more than the gain. The power of the identification method relies crucially on the variation of a quantity as a function of the channel number. For a hypothetical case of constant function, the verification cannot be made.

In case if you want to do the comparison for the whole hybrid, you can take the macro "*lkup_E12_hybrid.C*", rename it, modify the content appropriately, and run.

Retrieving other information

Once you have the root file, all the relevant information is in principle available. The scripts described above can be a starting point. The *root* file has a very hierarchical structure. It contains object *Wafer*, which contains objects *Chip*, each of which contains object *Channel*. The best source for the detailed description of the objects and their data is, of course, the source code. It is located in the directory *~lbatlas/unix_v7_7_anal/wafer/*.

Other tips

One can easily hang the root session without even trying very hard. If this happened, you can do the following to get rid of the session:

- open another session under *lbatlas* account,
- look for the root process number:

```
prompt> ps aux | grep lbatlas
```

- find the root session and kill it:

```
prompt> kill -9 xxxx
```

here, xxxx is the root session process number.

Be careful not to kill other things, like your own new shell session.