LuckyDoll, an analysis package for the AIDA detector

Vi Ho Phong

A preliminary manual
February 10, 2017

Contents

1 Changelog

1.1 February 9, 2017

LuckyDoll now can read the GZIP file directly. Use the new option "-gz 1" to read gz file! New Version requires an installed BOOST library $\geq$ 1.42.0

1.2 October 1, 2016

New event builder algorithm using sucessive ADC word is added. Use the excutable ./aidanew or ./aidafullnew

1.3 November 7, 2016

Add the FEE number and channel number to the full tree. Changed default (clear values) of the channel identification (ffee, fch, fxy, fz fid) to -1 to avoid potential problem.

1.4 November 5, 2016

Change the help menu in ./aidafull For ./aidafull and ./aida: fix bug that force users to enter threshold and calibration file even if they don't want to.

1.5 October 28, 2016

A faster clustering function has been added. About 25% improvement in the overall perfomance.

1.6 October 26, 2016

An almost final version released

2 Introduction

2.1 What This is About?

LuckyDoll is a data analysis package which is used for analyzing the data from AIDA detector.

The AIDA detector is a new generation implantation detector which is specificly designed for decay experiment at fast RI beam facilities. More information about AIDA can be found at [2]

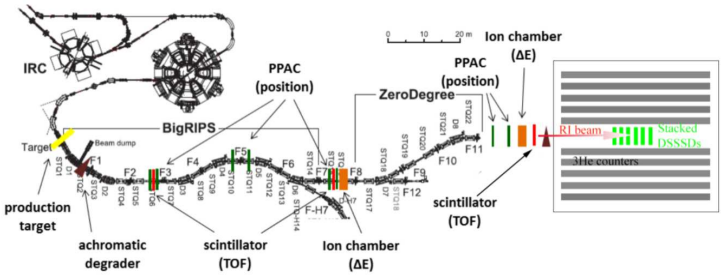

Figure 1: A schematic view of the BRIKEN setup

The developement of the LuckyDoll package is gorverned by a number of principles:

- Capatable to analyze a large amount of data from both implantation and decay events within a reasonable computing time so that it can be used for semi-online analysis during the experiment
- Comprehensive data structure
- Easy to understand, easy to use
- Produce output data with a mimized the number of the background and unwanted events.

2.2 LuckyDoll main classes and what are they good for

LuckyDoll package is divded into several classes, each is programed to perform a particluar task

AIDAUnpacker class handles the decoding of the MIDAS data and convert it into the "hits" seen by AIDA ADC

BuildAIDAEvent class is used to reconstruct the decay and implantation events from the energy deposition in the strips of the sillicon detector. The reconstruction is based on timing and spatial correlation. The implementation of the class is as follow:

- Open files, initialize the Event Builder object

```
TFile* ofile = new TFile(OutFile,"recreate");
 ofile->cd();
```

```
//! Book tree and histograms
TTree* treeion=new TTree("ion","tree_ion");
TTree* treebeta=new TTree("beta","tree_beta");
TTree* treepulser=new TTree("pulser","tree_pulser");


BuildAIDAEvents* evts=new BuildAIDAEvents;
evts->SetVerbose(Verbose);
evts->SetFillData(true);
evts->BookTree(treeion,treebeta,treepulser);
evts->SetMappingFile(MappingFile);
evts->SetThresholdFile(ThresholdFile);
evts->SetCalibFile(CalibrationFile);
evts->SetAIDATransientTime(TransientTime);
evts->SetEventWindowION(WindowIon);
evts->SetEventWindowBETA(WindowBeta);
evts->Init(R10_0);
```

- Event loop (similiar with analoop)

```
while(evts->GetNextEvent()){
if(evts->IsBETA())
evts->GetAIDABeta()->GetCluster(0)->GetHitPositionX();
else
evts->GetAIDAIon()->GetCluster(0)->GetHitPositionX();
evts->GetAIDAIon()->GetCluster(0)->GetEnergy();
evts->GetAIDAIon()->GetCluster(0)->GetTimestamp();
}
```

- Write tree and close files

```
    treeion->Write();
    treebeta->Write();
    treepulser->Write();
    ofile->Close();
```

2.3 LuckyDoll main programs

The Lucky main programs are builted on top of the main classes to perform some paticular analysis jobs.

- **./aida(new)** : Unpack, calibrate, build aida events and produce a root file with minimum information which later can be used for the BRIKEN merger program.
- **./aidafull(new)** : Unpack, calibrate, build aida events and produce a root file with full information.
- **./aida2tree** : Unpack the raw data and produce a root file containing infomation about hits in FEE channels (uncalibrated)

- **./aida2histbkg** : Make histograms of ADC value versus strip number for the purpuse of determining the individual threshold for each AIDA strip.
- **./aida2histcalib** : Make histograms of ADC value versus strip number considering the pulser events (high multiplicity) or low energy(and low multiplicity) events for the purpuse of energy calibration using pulser or radioactive source.

3 LuckyDoll basic: How to run it and what is the data structure?

3.1 Installation

....

Prerequites

- **gcc**
- **cmake** version ≥ 3
- CERN **root** version ≥ 5.34
- BOOST version $\geq 1.42.0$

Installation Steps

```
git clone https://github.com/vihophong/LuckyDoll.git
cd LuckyDoll
cd ../
mkdir build_LuckyDoll
cd build_LuckyDoll
cmake ../LuckyDoll
make
```

3.2 Convert AIDA full data

The help menu will be shown when there is no input argument specified by user:

```
$ ./aidafull
AIDA event builder
use ./aidafull with following flags:
        [-a    <char*     >: AIDA input list of files]
        [-o    <char*     >: output file]
        [-wi   <long long>: Ion event building window (default: 2500*10ns)]
        [-wb   <long long>: Beta event building window (default: 2500*10ns)]
        [-wd   <long long>: Fast Discriminator Scan window
      #(default: 0 i.e no scan for fast discrimination)]
        [-v    <int       >: verbose level]
        [-map  <char*     >: mapping file]
        [-cal  <char*     >: calibration file]
        [-thr  <char*     >: threshold file]
        [-f    <int       >: fill data or not: 1 fill data
```

```
0 no fill (default: fill data)]
[-tt   <long long>: aida transient time (default: 20000*10ns)]
[-ecut <char*     >: specify energy cut file]
[-gz   <int       >: input data from gz file: 1 enable 0 disable (defaul
```

Note:

- The event building window, fast discriminator scan window and transient time have been set by default (see above). Therefore, there is no need to input the -wi -wb -wd -tt if user don't want to change those value.
- If there is no calibration table specified by user the energy calibration will not be made, which is not good for the event correlation, especially the clustering algorithm.
- If there is no threshold table specified by user, the program will consider getting all information from AIDA (threshold = -10000) for all the channels.

The resulting root file from **./aidafull** contains 3 trees: ion, beta and pulsers. The Branches and Leafs structure of those trees are identical. They are defined in the source code "AIDA.h" as a TObject named "AIDA". The "AIDA" object stores STL Vectosr of the so called "hits" and "clusters" within a time window (used for event builder). The explanation for each leafs inside "AIDA" object can be found in the source code:

```
//! aida time stamp (ealiest timestamp within event)
    unsigned long long faidats;
    //! type of event: 0 beta  1 ion
    short ftype;
    //! total multiplicity
    unsigned short fmult;
    //! x multiplicity
    unsigned short fmultx[NumDSSD];
    //! y multiplicity
    unsigned short fmulty[NumDSSD];
    //! x sum all energy
    double fsumx[NumDSSD];
    //! y sum all energy
    double fsumy[NumDSSD];
    //! total clusters
    unsigned short fnclusters;
    //! total clusters
    unsigned short fnclustersz[NumDSSD];
    //! max hit position
    unsigned short fmaxz;
    //!Threshold table
    Double_t fdssd_thr[NumDSSD][NumStrXY];
    //!Calibration table
    Double_t fdssd_cal[NumDSSD][NumStrXY][2];
    //! vector with the hits
    vector<AIDAHit*> fhits;
```

```
    //! vector with the clusters
    vector<AIDACluster*> fclusters;
```

The **"hits"** information is generally defined in the class "AIDAHit" which contains the channel identification (i.e. id, fee number, channel number, strips number and dssd number), energy, ADC value and timestamp for each hits. They are described in the source code:

```
  //! energy range : (0: high gain, 1: low gain)
  short frange;
  //! translate into channel ID number
  short fid;
  //! translate into the DSSDs coordinator
  short fxy;
  short fz;
  //! FEE information
  short ffee;
  short fch;

  //! the energy lab system
  double fen;
  //! the raw adc value
  int fadc;
  //! the timestamp
  unsigned long long fts;
  //! the fast timestamp
  unsigned long long ffastts;

  //! current hits
  unsigned short fhitsadded;
```

Based on the information collected within an event window as a vector of hits, the event is reconstructed by using the spatial correlation. For that purpose, the term **"cluster"** is introduced, which represents the position where the energy deposition of the radiation takes place in the DSSD. It can be energy deposition of the heavy ion which triggers the high gain branch of the AIDA electronics or the energy deposition of the beta particles emitted from the decay of the implanted radiactive isotope. The details on how to reconstruct the event based on spatial correlation (to make "clusters" from "hits") can be found in the section 4.2.

The **"cluster"** is defined by the class "AIDACluster" as follows:

```
  //! translate into the DSSDs coordinator
  TVector3 fpos;

  //! the energy lab system
  double fsumenx;
  double fsumeny;

  //! number of hit in one cluster
  unsigned short fnx;
```

```
  unsigned short fny;

  //! the ealiest timestamp
  unsigned long long fcts;

  //! the ealiest fast timestamp
  unsigned long long fcfastts;

  //! current hits
  unsigned short fclustersadded;
```

3.3 Convert AIDA partial data

The help menu will be shown when there is no input argument specified by user:

```
$ ./aida
AIDA event builder
use ./aida with following flags:
        [-a    <char*    >: AIDA input list of files]
        [-o    <char*    >: output file]
        [-wi   <long long>: Ion event building window (default: 2500*10ns)]
        [-wb   <long long>: Beta event building window (default: 2500*10ns)]
        [-wd   <long long>: Fast Discriminator Scan window
      #(default: 0 i.e no scan for fast discrimination)]
        [-v    <int      >: verbose level]
        [-map  <char*    >: mapping file]
        [-cal  <char*    >: calibration file]
        [-thr  <char*    >: threshold file]
        [-f    <int      >: fill data or not: 1 fill data
        0 no fill (default: fill data)]
        [-tt   <long long>: aida transient time (default: 20000*10ns)]
        [-ecut <char*    >: specify energy cut file]
```

The data structure is as follows:

```
typedef struct {
    unsigned long long T;         // Calibrated timestamp
    unsigned long long Tfast; // calibrated fast timestamp
    double E;     // Energy depostions = (EX+EY)/2
    double EX;// X strips energy deposition
    double EY; // Y strips energy deposition
    double x,y,z;// number of pixel for AIDA,
    int nx, ny, nz;// multiplicity of hits in x
    //and y strips, and dssd planes
    unsigned char ID;     // Detector type:
    //Aida ion = 4, AIDA beta = 5
} datatype;
```

which represents the **"cluster"** information in a simplified structure.

4 Advance topics: Understanding LuckyDoll

4.1 How to read the timestamp in AIDA

It's complicated...

4.2 How the clustering algorithm works

It's complicated...

4.3 How to handle the transient time after the implantation event

It's complicated...

References

[1] A. Tarifeño-Saldivia et al.: arXiv:1606.05544.

[2] http://www2.ph.ed.ac.uk/ td/AIDA/Information/information.html