Chapter 4

Simulation

In a real particle physics experiment, a particle collider generates interactions of bunches of particles at high energy, yielding showers of particles which are then caught by the detectors. The data acquisition system chooses the interesting ones to be written and recorded. After that, the events may be reconstructed, i.e. the electronics signals are translated into a deduced setup of charges' tracks or neutral energy depositions. The full chain of simulation before data analysis work is thus involved event generation, detector simulation, digitization and event reconstruction. Fig. 4.1 shows an example of typical LHC event generated in the CMS detector.

For the CMS project, many softwares are involved in the full simulation chain. The diagram showing an order of simulation task in the CMS experiment and the corresponding software are illustrated in Fig. 4.2. Explanations on each chain of simulation are given in the following sections.

4.1 Event Generation

According to Sjöstrand[30], one of the creator of event generating program named PYTHIA, the main application of event generator are the followings:

- To give physicists a feeling for the kind of events one may expect and hope to find, and at what rates.
- As a help in the planning of a new detector, so that detector performance is optimized, within other constraints, for the study of interesting physics scenarios.

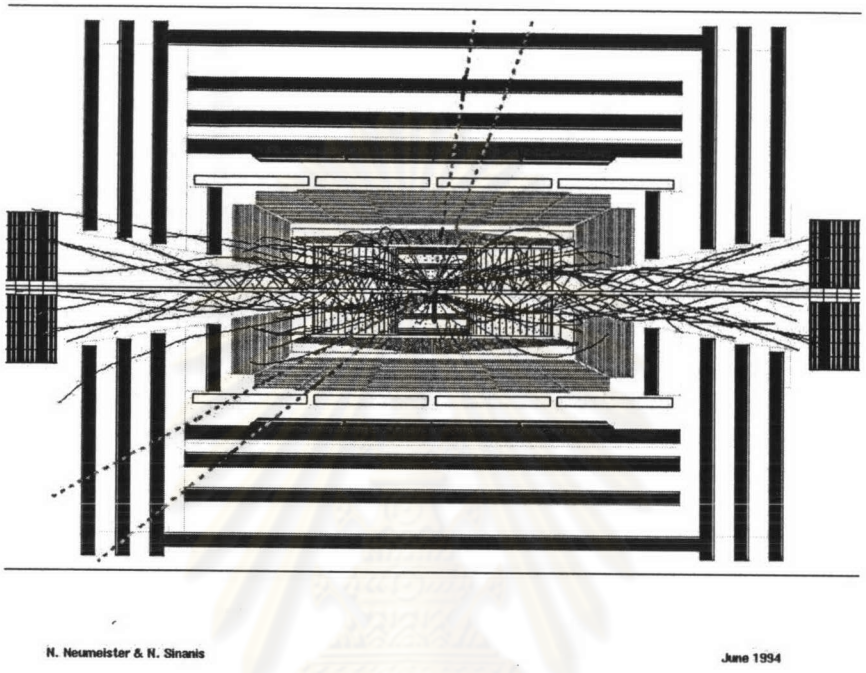

Figure 4.1: An illustration of the complex of typical event at the LHC $p^+p^+ \rightarrow H \rightarrow Z^0Z^0 \rightarrow 4\mu^{\pm}$

- As a tool for devising the analysis strategies that should be used on real data, so that signal-to-background conditions are optimized.
- As a method for estimating detector acceptance corrections that have to be applied to raw data, in order to extract the "true" physics signal.
- As a convenient framework within which to interpret the observed phenomena in terms of a more fundamental underlying theory (usually Standard Model).

In our work, we generated events from proton-proton collision using CMS event KINetic generation (CMKIN). CMKIN is built under Fortran code and is interfaced with many kinds of high energy physics event generators, such as

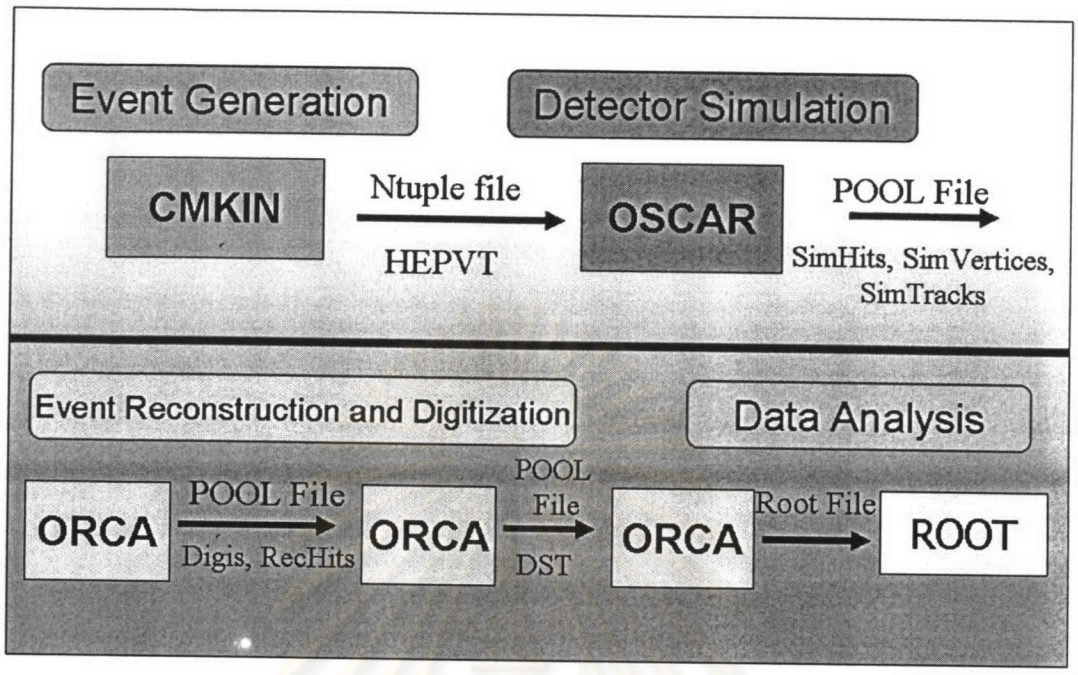

Figure 4.2: The full chain of the CMS simulation softwares.

PYTHIA, HERWIG, ISAJET/ISASUSY, etc. We select PYTHIA as CMKIN interface because it is easily obtainable frequently updated and well documented.

In the early day of doing simulation in particle physics, a program called JETSET, created by the Lund Theory in 1978, was used to generate e^+e^- to do physics study. Then, about 1983, PYTHIA was added to handle the job of generating hadronic physics at very high energies. PYTHIA generates the collisions between leptons; hadrons and gammas based on QCD model. Most of the time, the two programs are used together and their distinction gradually faded. Both programs are now maintained and referred under the same name PYTHIA.

For all particles, CMKIN will provide information on the number of the events, the value of four-momentum and angular distribution.

4.1.1 Physics in CMKIN

The typical collision in high-energy physics event can be described in the following steps, in time order:

1. Initially, the two beams are coming towards each other. Normally, each particle is characterized by a set of parton distribution functions, which defines the partonic substructure in terms of flavor composition and energy sharing.

2. One shower initiator parton from each beam starts off a sequence of branchings such as $q \rightarrow qg$ which build up an initial-state shower.

3. One incoming parton from each of the two showers enters the hard processes, where then a number of outgoing partons are produced, usually two. It is the nature of this process that determines the main characteristics of the event.

4. The hard process may produce a set of short-lived resonances, like the Z^0 and $W^{\pm}$ gauge bosons, whose decay to normal partons and has to be considered in close association with the hard process itself.

5. Also, the outgoing partons may branch (multiple interactions), to build up final-state showers.

6. In addition to the hard process considered above, further semihard interactions may occur between the other partons of two incoming hadrons.

7. When a shower initiator is taken out of a beam particle, a beam remnant is left behind. This remnant may have an internal structure, and a net color charge that relates it to the rest of the final states.

8. The QCD confinement mechanism ensures that the outgoing quarks and gluons are not observable, but instead fragment to color neutral hadrons.

9. Normally the fragmentation mechanism can be seen as occurring in a set of separate color singlet subsystems, but interconnection effects such as color rearrangement or Bose-Einstein may complicate the picture.

10. Many of the produced hadrons are unstable and decay further.

More detail on each step can be found in reference [30].

4.1.2 Generating Events using CMKIN

The CMKIN version we used in our work is 4.0.0. To generate events of interest using CMKIN, we first select the datacards of the type of process we want. The datacard consists of two parts: 1) channel independent cards in the job script and 2) a separate datacards *.txt file which contains the physics channel dependent control cards. The datacards used for $W^{\pm}$ and Z^0 production and the background processes can be found in Appendix B.1.

After selecting the datacard, we define other necessary parameters:

- total CMS energy
- kinematic limits for individual particles
- total number of generated events
- a number of different PYTHIA parameters
- the output file name
- the ntuple format (in our work a HEPEVT ntuple format)
- the Parton Distribution Function (PDF sets)

We are now ready to compile and execute event generation jobs through the KINE module. The CMKIN code on KINE module offers a standard way to interface kinematics generators. Below is a flow chart of the KINE module for PYTHIA ntuples generation program in CMKIN. The source code of the main program of the KINE module can be found in Appendix B.1.

The output events are stored in HEPEVT ntuple format. The kind of information stored in each entry of HEPEVT ntuple can be found in Appendix A.7.

In our work, we have generated $W^{\pm}$ and Z^0 in channels $q\bar{q} \rightarrow W^+W^- \rightarrow \mu^{\pm}\nu_{\mu}(\bar{\nu}_{\mu})$ and $q\bar{q} \rightarrow Z^0Z^0 \rightarrow 2\mu^+2\mu^-$ which we referred them as "signal" processes. Four of our "background" processes are 1) the Drell-Yan process, 2) $q\bar{q} \rightarrow \gamma^*\gamma^* \rightarrow$

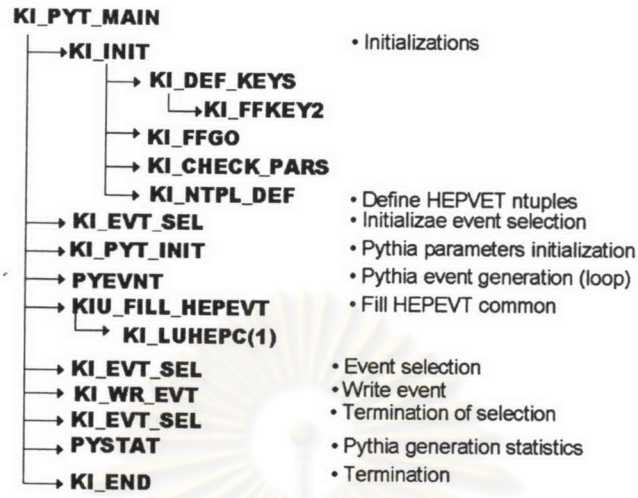

Figure 4.3: The flow chart of KINE module for PYTHIA ntuple generation in CMKIN.

$q\bar{q} \rightarrow \gamma^*\gamma^* \rightarrow \mu^+\mu^-$, 3) $q\bar{q} \rightarrow \gamma\gamma \rightarrow \mu^+\mu^-$ and 4) minimum bias process. The processes are considered as background if they give muons in their final decay product. For descriptions of the Drell-Yan process and the minimum bias events, see Appendix A.4 and A.5.

All physics processes were generated under similar kinematic and geometric configurations as the followings:

- center-of-mass energy: 14 TeV
- pseudorapidity: $|\eta| < 2.4$ (corresponds to the full coverage of CMS Muon Chamber)
- transverse momentum: $0 < P_T < 200$ GeV/c
- acoplanarity angle: $0\ ^\circ < |\phi| < 360\ ^\circ$
- Parton Distribution Function (PDF) set: CTEQ5L [8]

There are 10,000 events being generated for each process, except for the processes $q\bar{q} \rightarrow \gamma^*\gamma^* \rightarrow \mu^+\mu^-$ and $q\bar{q} \rightarrow \gamma\gamma \rightarrow \mu^+\mu^-$ which contain 40,000 events. We need more events for the two processes because, from our preliminary analysis, they yielded very few numbers of muons in the detector simulation.

Table 4.1: The number of events being generated by CMKIN for each process.

Event	Number of events being generated
$q\bar{q} \rightarrow W^+W^- \rightarrow \mu^{\pm}\nu_{\mu}(\bar{\nu}_{\mu})$	10,000
$q\bar{q} \rightarrow Z^0Z^0 \rightarrow 2\mu^+2\mu^-$	10,000
$q\bar{q} \rightarrow \gamma^*\gamma^* \rightarrow \mu^+\mu^-$	40,000
$q\bar{q} \rightarrow \gamma\gamma \rightarrow \mu^+\mu^-$	40,000
Drell-Yan Process	10,000
minimum bias	10,000

4.2 Detector Simulation

Detector simulation is of a fundamental importance in many aspects and phases of particle physics experiment. For example, it provides a preliminary investigation to the design of the experimental set-up, an evaluation of the potential physics output and risks involving the detector construction. In addition, it posts a good assessment on the detector development, tests and optimizes of reconstruction and physics analysis, and helps in estimation and validation of physics outcome.

In general, with input of known physics theory, the detector simulation simulates how particles interact with the detector material, how they transverse through the detector, curve in magnetic fields, produce shower in calorimeters and deposit energy, etc.

The software responsible for detector simulation in the CMS experiment is Object-Oriented Simulation for CMS Analysis and Reconstruction (OSCAR). OSCAR is a software package built on the (Geometry and Tracking 4) (GEANT4) tool kit [31], providing a complete set of tools for all domains of detector simulation: Geometry, Tracking, Detector Response, Run, Event and Track management, and User Interface. A large number of set of physics processes in OSCAR gives many possible interactions of particles with matter over a broad range of energy. In addition, OSCAR has implemented abundant set of utilities, including PDF (Parton Distribution Function) information, particle management, sets of

random number generators, physics constants and units and interfaces to event generators. A detail of OSCAR *BuildFile* and *OSCAR run configuration file* can be found in Appendix C.1.

OSCAR (In our work, we used OSCAR version 3.4.0.) works by incorporating four ingredients.

- **Primary Event:** input events from event generator CMKIN. These input events must be in HEPEVT format in order to be processed by OSCAR.
- **Detector Description:** data containing CMS detector information. it is separated in three categories: 1) data of the detector's materials, shapes, geometrical information, 2) data specific to the tasks of particular parts of the detector, or so-called "sensitive detectors," and 3) data describing the CMS magnetic field. This data is stored in a database named Detector Description Database (DDD) .
- **Physical Processes:** set of physics processes (electromagnetic and hadronic process). The processes are incorporated during the simulation job. For available physics processes in OSCAR, see the GEANT4 manual [31].
- **User Action:** adjusting the physical parameters and input-output configuration.

The simulation task in OSCAR consists of 3 steps.

- **Initialization:** construction of detector material and geometry, construction of particles and physics processes and calculation of cross-section.
- **Runs:** close and optimize geometry, prepare one event to be processed.
- **Events Processing:** processing one *track*, *stepping* and generate hits. Note: *Tracks* are traces of particles corresponds to the simulation of a particle from its birth to death. When a particle is tracked in a volume, each step in the volume is signaled to the sensitive detector which extracts hit information. *Steps* is a simulated segment of the track of a particle.

To sum up, OSCAR uses information from the DDD, takes input of generated events from CMKIN and uses GEANT4 to simulate how particles propagate through space, interact with the CMS detector material, deposit energy, etc. From the simulations, OSCAR then produces output stored it in POOL database. The information stored in the database consists of three types of information: *SimVertices*, *SimTracks* and *SimHits*. They will be an input for the next chain of simulation – digitization.

OSCAR simulation is very CPU intensive work. For example, it had taken 10,000 events of $Z^0 \rightarrow \mu^+\mu^-$ more than 2 weeks to run OSCAR on a 1.6 GHz Pentium 4 PC. Under a tight schedule, we cannot simulate as many number of events as we plan to. We were forced to input only 1,000 events in OSCAR, which was expected to yield high statistical errors. A plan to run more number of events is set for the near future. Table 4.2 below displays the types and number of events that were input into OSCAR.

Table 4.2: The type and number of events being input in OSCAR

Event	Number of events input in OSCAR
$q\bar{q} \rightarrow W^+W^- \rightarrow \mu^{\pm}\nu_{\mu}(\bar{\nu}_{\mu})$	1,000
$q\bar{q} \rightarrow Z^0Z^0 \rightarrow 2\mu^+2\mu^-$	1,000
$q\bar{q} \rightarrow \gamma^*\gamma^* \rightarrow \mu^+\mu^-$	1,000
$q\bar{q} \rightarrow \gamma\gamma \rightarrow \mu^+\mu^-$	1,000
Drell-Yan Process	1,000
minimum bias	1,000

For visualization, OSCAR has an associated software, an Interactive Graphics and User ANAlysis for CMS (IGUANACMS), responsible for the part. In our work, we are not concern with graphical aspects of the simulation, and thus have omitted this part for future study. Below, in Fig. 4.4, we show one example of OSCAR visualization in the IGUANCMS program.

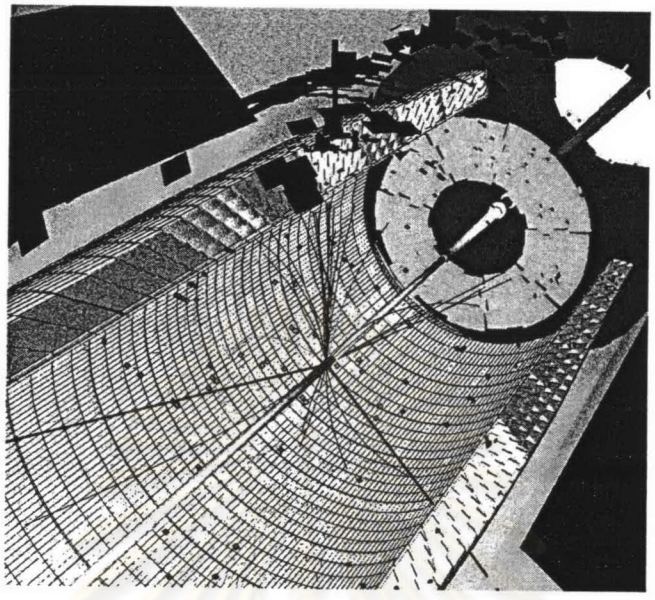

Figure 4.4: An IGUANACMS visualization of SUSY (Super Symmetry) event. (picture from http://iguanacms.web.cern.ch/iguanacms/gallery.html)

4.3 Event Reconstruction and Digitization

From the full detector simulation by OSCAR, the information of every detector readout channel are stored in a POOL database, ready to be reconstructed and digitized. The Object Oriented Reconstruction for CMS Analysis (ORCA), is the software responsible for CMS event reconstruction and digitization task. It organizes and converts information obtained from OSCAR into physics meaning quantities and further performs simulation of the responses of readout electronics.

ORCA utilizes object oriented technology for which C++ has been chosen as programming language. The design of ORCA is based on CMS Analysis and Reconstruction Framework (CARF), which is developed to prototype reconstruction methods.

There are two logical layers in ORCA; the *subsystems* and in each subsystem, *packages*. The packages in a subsystem generate a single library to handle a particular domain of the associated sub-detector (e.g. for Calorimetry: G3EcalDigits, G3HcalDigits, EcalCluster) or reconstruction utility. In standard ORCA package, there are minimum of four packages located in four ORCA sub-directories. Only

the first two of these are exposed to external users of the packages. Those four packages are:

- **doc** The documentation related to the package.
- **interface** The C++ header files that can be referred from other packages in the same or in different subsystems. In case of missing documentation these header files reveal the implemented functionality of the package.
- **src** The actual source code and C++ header files local to the package.
- **test** Source code of test programs for quality control of the package.

Except from the four minimum packages, an ORCA subsystem also contains a directory *domain*, which holds files and documentation shared by all packages of the subsystem.

To run event reconstruction and digitization in ORCA (in our work, we used ORCA version 8.4.0.), we first specified the location of POOL database, the number of event to be processed and the output dataset to be stored after ORCA had finished the job. Then, we run ORCA by executing two operations: first, digitization and, second, Data Summary Tape (DST) recording.

1. **Digitization:** In digitization process, ORCA simulated the response of electronic readout in the CMS detector. It first took input files from OSCAR output, processed and produced *Digis* objects to be stored in POOL database. The output Digis files contained three kinds of information: 1) information from the event generator, 2) information from detector simulation (SimHits, SimTHits, SimTracks, SimVertices) and 3) information from digis (including pile-up and electronics response simulation). The data at this stage was not yet in format to be analyzed. It has to be recorded in DST database first. In Appendix C.2, we provide ORCA *BuildFile* and *run configuration* file of the digitization operation.
2. **DST Recording:** ORCA has implemented a DST feature for recording and retrieving database stored from digitization process. DST is chosen

as data storage technology for the CMS project because it could provide compact information for analysis and homogeneous collections of objects: tracks, vertices, muons, electrons, jets, taus, etc.

In DST recording, ORCA took POOL data file from digitization and write all objects created during reconstruction in DST format. The information from the output of DST recording was to be retrieved by DST reading feature in ORCA before being able to be analyzed. The *BuildFile* and *run configuration for DST recording* file can be found in Appendix C.3.