Table of Contents

DD4hep-based Simulation and Reconstruction

Local build areas with the last working version of DD4hep and lcgeo are provided. Due to the intense development activity in DD4hep and to interface all simulation and reconstruction packages with DD4hep, new build areas with ILCSOFT installations are often updated and old build directories are canceled.

Current path to the ILCSOFT installation (hereafter: _path_to_ilcsoft):

- /cvmfs/clicdp.cern.ch/iLCSoft/builds/2016-11-22/x86_64-slc6-gcc48-opt/

Current CLIC detector model (hereafter: _clic_model):

- CLIC_o2_v04_p1 (old tracker and calo to finalise tracking studies)
- CLIC_o3_v07 (40L ECal + engineering tracker model)

Getting Started

Note: CVMFS is now a prerequisite to use releases provided by CERN or DESY. On `lxplus` it should be accessible. On your desktop make sure you have CVMFS installed by following the procedure outlined here.

If you are using a full ILCSoft release (either provided by CERN/DESY or built by yourself), it is sufficient to do:

```
source _path_to_ilcsoft/init_ilcsoft.sh
```

By default in this case the environment variables $DD4HEP, $DD4hepINSTALL and $lcgeo_DIR should point to the relevant packages.

Running the simulation

After initializing the ILCSOFT, the simulation program is accessible from everywhere as an executable named `ddsim`. For simulating particles one needs to provide: input file (e.g. mcparticles.slcio/.stdhep) / number of events / compact.xml file . Many more additional options available (physics list, etc), use ddsim --help to see all of them. Default values for all these options are set in $ILCSOFT/ClicPerformance/HEAD/examples/clic_steer.py. Please use this file for your simulation, options set by the command line will overwrite the ones set in the file.

- Use:

```
# init ilc software
source _path_to_ilcsoft/init_ilcsoft.sh

# run simulation using a stdhep/lcio generator file
ddsim --steeringFile $ILCSOFT/ClicPerformance/HEAD/examples/clic_steer.py --compactFile $ILCSOFT/
```

run generation and simulation using a particle gun of single muons

```
ddsim --steeringFile $ILCSOFT/ClicPerformance/HEAD/examples/clic_steer.py  --compactFile $ILCSOFT
```

- See DDSimTips for use cases and tips on running ddsim, including instructions on material scan.

Expert Tip: command line argument and parameter tab completion! (A. Sailer)

- You will need `argcomplete` (install it by running `sudo pip install argcomplete`) and have to be running `bash` (or `zsh`) with `bash-completion` (e.g install by doing `sudo yum install bash-completion`).
- Execute this on every new session (or put it in your `.bashrc`): `eval "$(register-python-argcomplete ddsim)"`
- Enjoy listing and completion of all `ddsim` options by starting with `ddsim - < press tab >`
- Also supports listing of possible options to arguments:
    - For example typing `ddsim --runType < press tab >` will list " `batch run shell vis` " as options.

If you need to produce a large sample of simulated events, you probably would like to run on the grid. Here a couple of concrete examples to run ddsim on the grid (you can see the complete documentation at this link: DiracDDSimDoc)

- Simulate events given a generator file with the physics process

Show... Hide

```
..... your usual imports here ....

from ILCDIRAC.Interfaces.API.NewInterface.Applications import DDSim

..... your usual initialisation of the job here ....

ddsim = DDSim()
ddsim.setSteeringFile(path_to_your_clic_steer.py)
ddsim.setVersion(ILCSoft-2016-11-22_gcc48)
ddsim.setDetectorModel(CLIC_o3_v07)
ddsim.setInputFile(LFN:/ilc/user/s/simoniel/stdhep_files/ttbar_3TeV/whizard.001.stdhep)
ddsim.setOutputFile(sim_ttbar_3TeV_1.slcio)
ddsim.setNumberOfEvents(100)
res = job.append(ddsim)

if not res['OK']:
    print res['Message']
    sys.exit(2)

..... if you want to run the reconstruction - put marlin here ....

..... your usual finalisation and submission of the job here ....
```

- Simulate events generated with a particle gun

Show... Hide

```
ddsim = DDSim()
ddsim.setSteeringFile(path_to_your_clic_steer.py)
ddsim.setVersion(ILCSoft-2016-11-22_gcc48)
ddsim.setDetectorModel(CLIC_o3_v07)
ddsim.setOutputFile(sim_mu.slcio)
ddsim.setNumberOfEvents(100)
ddsim.setExtraCLIArguments(" --enableGun --gun.particle mu- --gun.energy 500*GeV --gun.distri
res = job.append(ddsim)

if not res['OK']:
    print res['Message']
    sys.exit(2)
```

Running the Reconstruction with or without performance monitoring

The reconstruction is performed using Marlin and an associated steering xml file in a way similar to the previous Mokka/Gear based simulation and reconstruction chain. An example steering file is being implemented in a new package called `ClicPerformance` in iLCSoft:

```
$ILCSOFT/ClicPerformance/HEAD/examples/clicReconstruction.xml
```

The package includes code used for performance studies, but crucially includes example xml files to run the full reconstruction chain using DD4hep: tracking (full tracking or with cheated pattern recognition) and PandoraPFA (via a bridging package and Marlin processor called DDMarlinPandora). The steering file instantiates the Geometry using DD4hep and calls the relevant processors to run the reconstruction and any higher level processing. The examples may also include some performance processors which the user can comment out. The steering files (e.g. `clicReconstruction.xml`) have hard-coded references to DD4hep compact file (the geometry) and the input LCIO files. All are either included in the package or accessible over AFS. Feel free to edit your local xml copy or overwrite the parameters via the Marlin command line arguments:

```
Marlin --InitDD4hep.DD4hepXMLFile="$lcgeo_DIR/CLIC/compact/CLIC_o2_v03/CLIC_o2_v03.xml"  clicReco
```

You can also modify the input file or other parameters. Please consult `Marlin -h` for more options.

If you would like to turn off the monitoring processors, please comment MyClicEfficiencyCalculator, MyTrackChecker and MyHitResiduals in the execute.

Access geometry information

- Code snapshots for accessing geometry information like B field or subdetector/layer/module information associated to a hit are available at this page: GeoInfo

Files location

- Generator files (stdhep):
  - ttbar 3 TeV: /eos/experiment/clicdp/grid/ilc/user/s/simoniel/stdhep_files/ttbar_3TeV/
    - 100 events per file
  - ee --> bb 3 TeV (no ISR, no luminosity spectrum): /eos/experiment/clicdp/grid/ilc/user/s/simoniel/stdhep_files/bb_3TeV/
    - 100 events per file
  - Zuds (several energies): /eos/experiment/clicdp/grid/ilc/user/w/webermat/stdhep/
    - 1000 events per file split jobs when running on the grid
- Simulated events:
  - UPDATED ttbar (also with overlay) and bb for tracking studies: /eos/experiment/clicdp/grid/ilc/user/s/simoniel/sim_files_v5/CLIC_o2_v04_p1/
  - Zuds for calorimeter studies: /eos/experiment/clicdp/grid/ilc/user/s/simoniel/sim_files_v4/CLIC_o3_v07

Using CED viewer with DD4hep to create publication-quality event displays

The CED event display can now visualize the geometry using the DD4hep instead of GEAR. You can use it to visualize the detector geometry with less detail (faster) and draw the event data on top of it.

Geometry model CLIC_o3_v06 (and followers) are badly visualised in CEDViewer. CLIC_o2_v04 should be fine. The reason of the bad visualisation is a known problem. To get around it, a "dummy" geometry has been produced for visualisation-only purposes (NOT FOR SIMULATION, NOT FOR RECONSTRUCTION). The geometry is called CLIC_o3_v06_CED but it will be valid also for versions > v06. You can find it in:

```
ClicPerformance/Visualisation/CLIC_o3_v06_CED/CLIC_o3_v06_CED.xml
```

The simplest way to use CEDViewer is the following:

```
ced2go -d $lcgeo_DIR/CLIC/compact/CLIC_o2_v03/CLIC_o2_v03.xml <input_file.slcio>
```

A template can be provided with more specific drawing options. In addition, several options are available within the application window.

Some hints and tips:

- Press "Enter" to move to the next event
- Double click on a drawn event object (Hit, PFO, MC Particle, ...) to dump some relevant information in the console.
- Press "h" to overlay the several keyboard shortcuts to toggle options on and off
- Go to the "Graphics" menu to chose between "Graphics low" (wireframe) and "Graphics high" (nicer views with transparency).
- Press "`" (backtick) to toggle between showing all or none of the LCIO collections
- Press "~" (tilde) to show/hide the geometry

Convert DD4hep geometry to gear file

For backward compatibility and validation tests. Use in case you want to use older GEAR-based reconstruction software. In the future, all new reconstruction code should not depend or use GEAR.

- Use and example:

```
convertToGear   plugin    dd4hep_compact_file   name_of_output_gear_file
convertToGear   GearForCLIC   $ILCSOFT/lcgeo/[HEAD]/CLIC/compact/_clic_model/_clic_model.xml   ge
```

- If the command is not working, probably DDh4hep has been compiled with the option "DD4HEP_USE_GEAR" OFF. In this case:
  - check out the DDh4hep package
  - in file CMakeLists.txt, modify OFF in ON for line: option(DD4HEP_USE_GEAR "Build gear wrapper for backward compatibility" OFF)
  - recompile: cd build/ ; cmake -C $ILCSOFT/ILCSoft.cmake .. ; make install ; cd ../ ;
  - use: path_to_your_local_DDh4ep/bin/convertToGear

Installing local copies of DD4hep and LCGeo

More Advanced! It is suggested to use one of the provided releases (see next section).

1.) Install the CLICCvmfs

2.) There is a build with geant4 10.1, CLHEP 2.1.4.1 here: _path_to_ilcsoft

If you just need dd4hep you can source: source _path_to_ilcsoft/init_ilcsoft.sh

3.) If you are building your own DD4hep or LCGEO adapt the init script, so it points to your own dd4hep or lcgeo

Then to install for the first time:

```
svn co https://svnsrv.desy.de/public/aidasoft/DD4hep/trunk dd4hep
svn co https://svnsrv.desy.de/public/ddsim/lcgeo/trunk lcgeo
cd dd4hep; mkdir build; cd build;
cmake \
-D DD4HEP_USE_GEANT4=1 \
-D DD4HEP_USE_BOOST=ON \
-DBoost_NO_BOOST_CMAKE=ON \
-D DD4HEP_USE_LCIO=ON \
-D BUILD_TESTING=On \
-D CLHEP_DIR=$CLHEP \
-D DD4HEP_USE_GEAR=1 \
-DCMAKE_C_COMPILER=`which gcc` \
-DCMAKE_CXX_COMPILER=`which g++` ..
# optional flags:
# If you want to use xerces-c add above: -D DD4HEP_USE_XERCESC=ON
# If you want to specify the Geant4 version: -D Geant4_DIR=$G4INSTALL/lib[64]/Geant4-<VERSION> \
###cmake -DDD4HEP_USE_BOOST=True -DDD4HEP_USE_GEANT4=True -DDD4HEP_USE_LCIO=True -DGeant4_DIR=$G4
make -j3 install; source ../bin/thisdd4hep.sh
cd ../../lcgeo; mkdir build; cd build; cmake -DBoost_NO_BOOST_CMAKE=ON ..;
make install; source ../bin/thislcgeo.sh
```

After dd4hep was installed once only the environment files need to be sourced.

```
source yourDD4hepConfig.sh
#which should include
source $DD4HEP/bin/thisdd4hep.sh
source $LCGEO/bin/thislcgeo.sh
```

The DD4HEP or LCGEO variables are not set, they have to be replaced with the proper path.

-- AndreSailer - 2014-11-25

This topic: CLIC > CLICDD4hep
Topic revision: r33 - 2016-12-15 - RosaSimoniello