CLIC Detector and Full Simulation

André Sailer

CERN-EP-SFT

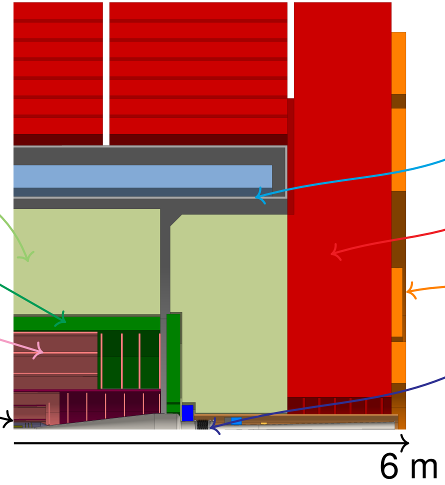
ECFA Higgs Factories: 1st Topical Meeting on Simulation
February 1, 2022

Table of Contents

Detector for CLIC

General purpose detector for Particle Flow reconstruction [1]

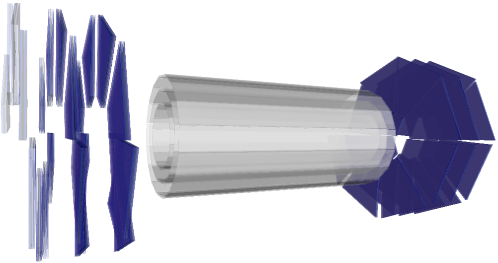
- Steel–Scintillator HCal with 3 cm cell-size
- Silicon–Tungsten ECal with 5 mm cell-size
- Silicon Tracker, mostly 50 μm pitch strips
- Vertex Detector with 25 μm pixels

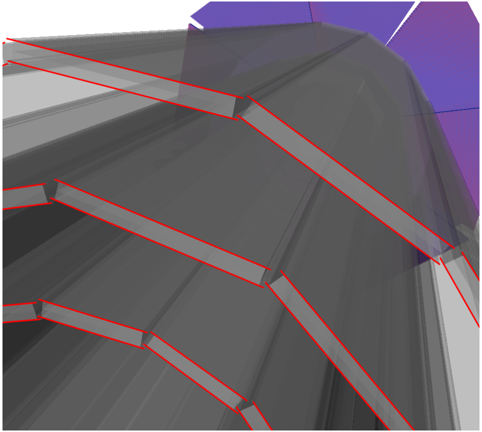


- Superconducting Solenoid of 4 T
- Iron Yoke with RPCs for Muon ID
- End-coils
- Forward calorimeters for EM coverage down to 10 mrad

Vertex Detector

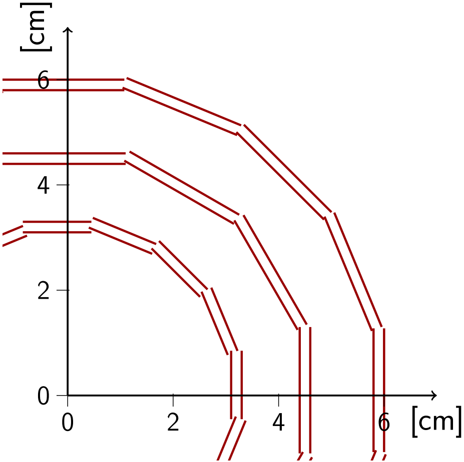
- Silicon vertex detector: precise vertex reconstruction
- Double layers (0.2% X_0 per detection layer)
- $R_{\text{in}} = 31$ mm
- Spiral geometry in endcaps for air cooling



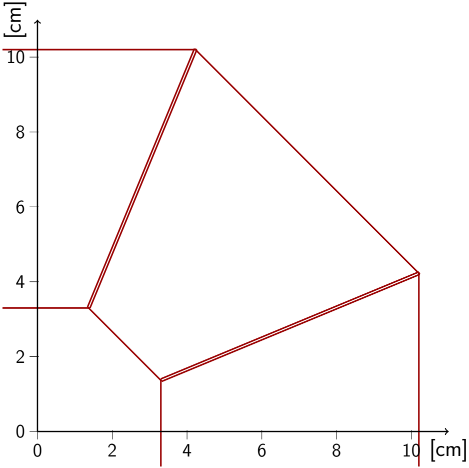


Vertex Detector Sensor Layout

Vertex Detector Barrel



Vertex Detector Endcap



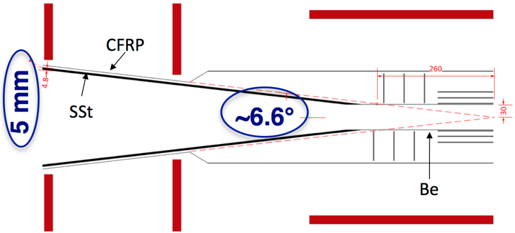
Air Cooling of the Vertex Detector

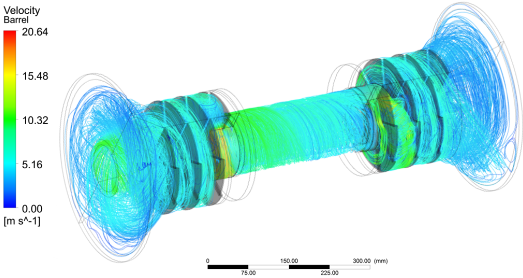
Rationale:

- Need a way to get cool air into and hot air out of the vertex detector region
- Use double walled beam-pipe as air-duct
- Need spiral vertex endcaps to get air to vertex barrel

Studies

- Simulations and full scale mock-up of vertex region

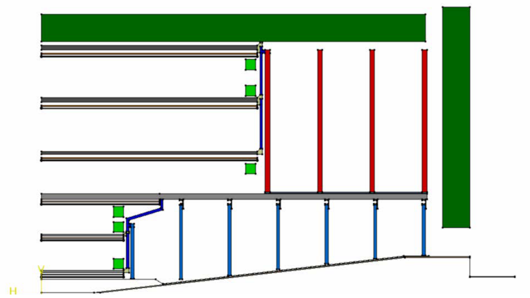




Tracker Layout

- Inner and Outer Tracker
  - Support tube for extraction with beam-pipe assembly
- 3 short and 3 long barrel layers
- 7 inner and 4 outer endcaps
- **Engineering design**

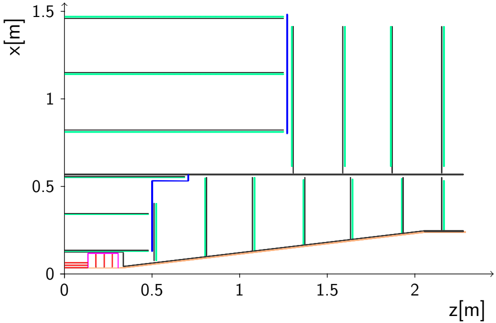
- At least 8 hits for $\theta > 8^{\circ}$



Tracker Layout

- Inner and Outer Tracker
  - Support tube for extraction with beam-pipe assembly
- 3 short and 3 long barrel layers
- 7 inner and 4 outer endcaps

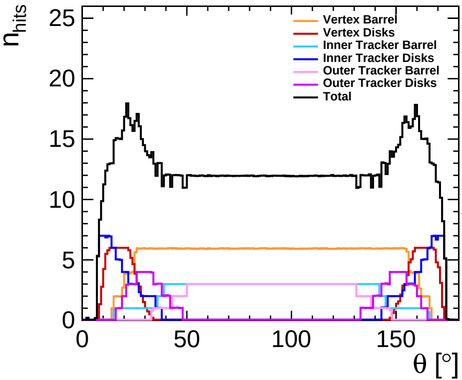
- **Full simulation implementation**
- At least 8 hits for $\theta > 8^\circ$



Tracker Layout

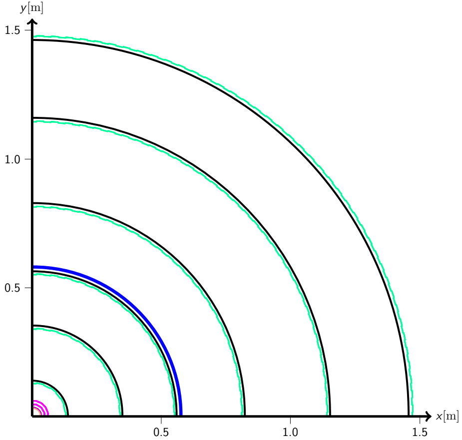
- Inner and Outer Tracker
  - Support tube for extraction with beam-pipe assembly
- 3 short and 3 long barrel layers
- 7 inner and 4 outer endcaps

- **At least 8 hits for** $\theta > 8^\circ$

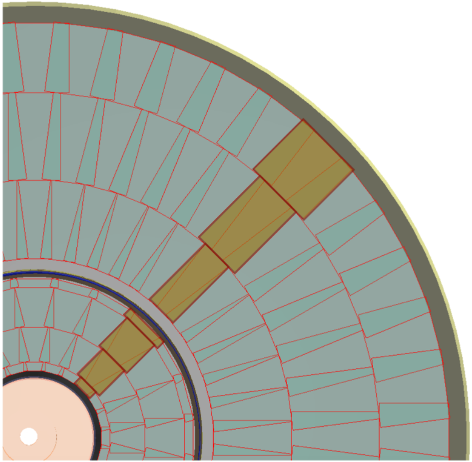


Silicon Tracker Sensor Layout

Barrel



Endcap



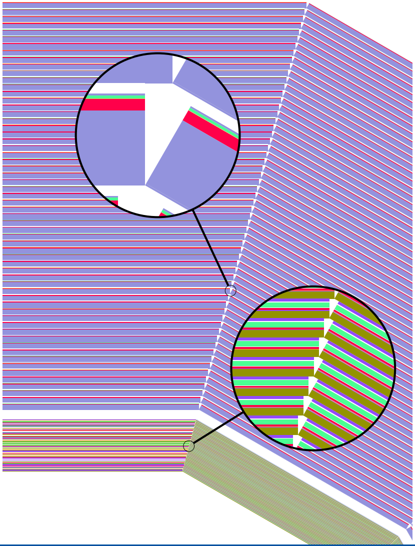
ECAL/HCAL

ECal

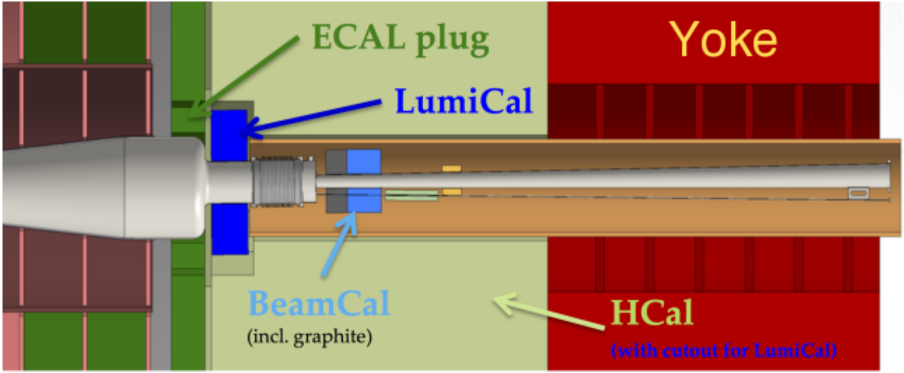
- 40 layers, 1.9 mm tungsten absorber, 22 X_0,
- silicon sensors with 5×5 mm^2 granularity

HCal

- 60 layers, 19 mm steel absorber, 7.5 (+1) λ_I
- scintillator tiles with 3×3 cm^2 granularity



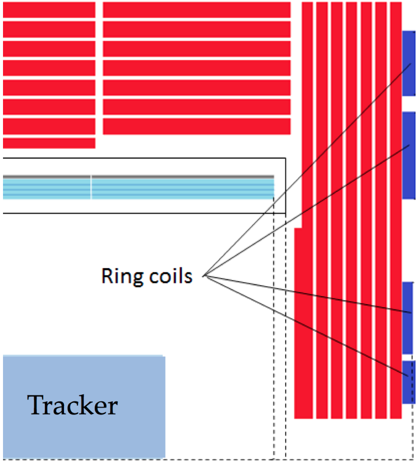
Forward Region



- Two detectors aligned on the outgoing beam axis complementing e.m. coverage
- LumiCal for luminosity measurements, BeamCal suffering from incoherent pairs
- 40 layers of 3.5 mm tungsten and sensors (silicon for the LumiCal, something radiation hard for the BeamCal)
- support tube radius 25 cm, Conically opening beampipe

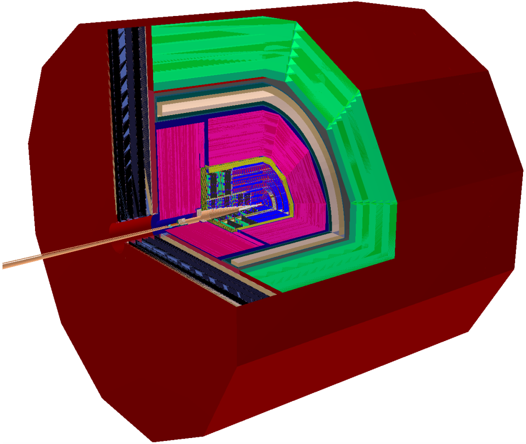
Magnetic Field and the Yoke

- 4 Tesla Solenoid Field
- Use End Coils to allow for reduced thickness of Return Yoke endcaps
- No need for an anti-solenoid (according to beam simulations)
- Return yoke contains muon system with 6 equidistant layers



Implementation

- CLIC detector (CLIC_o3_v14) implemented with DD4hep in the **lcgeo package**



DDSim

- `ddsim` python executable is part of the DD4hep release [2]
- Get steering file `ddsim --dumpSteeringFile > mySteer.py`
  - Steering file includes documentation for parameters and examples
  - The python file contains a `DD4hepSimulation` object at global scope
  - Configure simulation directly from command-line

```
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV, keV
SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v06.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"
SIM.part.minimalKineticEnergy = 1*MeV
SIM.filter.filters ['edep3kev'] =
dict (name="EnergyDepositMinimumCut/3keV" ,
      parameter={"Cut" : 3.0*keV} )
```

```
$ ddsim
--action.calo                  --filter.tracker      --part.keepAllParticles
--action.mapActions            -G                    --part.minimalKineticEnergy
--action.tracker               --gun.direction       --part.printEndTracking
--compactFile                  --gun.energy          --part.printStartTracking
--crossingAngleBoost           --gun.isotrop         --part.saveProcesses
--dump                         --gun.multiplicity    --physics.decays
--dumpParameter                --gun.particle        --physics.list
--dumpSteeringFile             --gun.position        --physicsList
--enableDetailedShowerMode     -h                    --physics.rangecut
--enableGun                    --help                --printLevel
--field.delta_chord            -I                    --random.file
--field.delta_intersection     --inputFiles          --random.luxury
--field.delta_one_step         -M                    --random.replace_gRandom
--field.eps_max                --macroFile           --random.seed
--field.eps_min                -N                    --random.type
--field.equation               --numberOfEvents      --runType
--field.largest_step           -O                    -S
--field.min_chord_step         --outputFile          --skipNEvents
--field.stepper                --output.inputStage   --steeringFile
--filter.calo                  --output.kernel       -v
--filter.filters               --output.part         --vertexOffset
--filter.mapDetFilter          --output.random       --vertexSigma
```

DDG4 Configuration

DDSim offers access to the usually used plugins for currently supported detector models. For more configurability, the desired plugins can also be directly configured outside of the ddsim eco-system

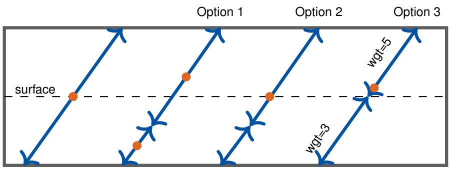
- configurable through python (configure actions, filters, sequences, cuts. . .)

```
#...
part = DDG4.GeneratorAction(kernel, "Geant4ParticleHandler/ParticleHandler")
kernel.generatorAction().adopt(part)
part.SaveProcesses = ['Decay']
part.MinimalKineticEnergy = 1*MeV
part.KeepAllParticles = False
#...
user = DDG4.GeneratorAction(kernel,"Geant4TCUserParticleHandler/UserParticleHandler")
user.TrackingVolume_Zmax = DDG4.tracker_region_zmax
user.TrackingVolume_Rmax = DDG4.tracker_region_rmax
```

DD4hep Plug-in Palettes

- Providing input handlers, sensitive detectors for most cases. . .
- Hard to provide Geant4 Sensitive Detectors for all cases
  - Couples detector 'construction' to reconstruction, MC truth and Hit production
  - Too dependent on technology and user needs

e.g. several possibilities for tracker



  - Providing palette of most 'common' sensitive components for trackers and calorimeters
- Physics lists, Physics/particle constructors etc.
  - Wrapped factory plug-ins directly taken from Geant4
  - Users extend physics list (e.g. QGSP)
- Several IO handlers (LCIO, ROOT, StdHep, HepEvt, HepMC)
- MC truth handling w/ or w/o record reduction

Full Simulation Example

https://key4hep.github.io/key4hep-doc/examples/clic.html

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh
git clone https://github.com/iLCSoft/CLICPerformance
ddsim --compactFile $LCGEO/CLIC/compact/CLIC_o3_v14/CLIC_o3_v14.xml \
      --outputFile ttbar_edm4hep.root \
      --steeringFile clic_steer.py \
      --inputFiles ../Tests/yyxyev_000.stdhep \
      --numberOfEvents 3
```

`clic_steer.py` contains the configuration of sensitive detectors, crossing-angle and physics tuned for CLIC simulations

Summary

- The CLIC detector has been fully implemented in DD4hep, for more details see our comprehensive note [1]
- Full simulation can be controlled with `ddsim` [2]

References

[1] N. Alipour Tehrani et al. "CLICdet: The post-CDR CLIC detector model". In: (Mar. 2017). CLICdp-Note-2017-001. URL: https://cds.cern.ch/record/2254048.

[2] Markus Frank et al. "DDG4: A Simulation Framework using the DD4hep Detector Description Toolkit". In: *J. Phys. Conf. Ser.* 664 (Apr. 2015), p. 072017. DOI: 10.1088/1742-6596/664/7/072017.