TOF detector simulation tutorial and next steps

Det1 TOF Meeting
June 13, 2022

Nicolas Schmidt

Oak Ridge National Laboratory

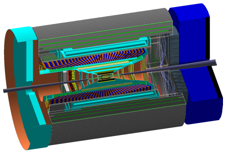
EIC

Tutorial - Fun4All Code

- **ECCE Singularity:** [github.com/ECCE-EIC/Singularity]
  → contains nightly build of ECCE-EIC code
  → recommended to use when running EIC Fun4All simulations
- **Fun4All base classes:** [github.com/ECCE-EIC/coresoftware]
  → contains classes for tracking, DST tree, physics lists, HepMC, hits, event generators, ...
- **Detector base classes:** [github.com/eic/fun4all_eicdetectors]
  → contains full detector implementations (geometry, active/passive volumes, visualization, stepping action, ...)
  → TTL detector class located here: PHG4TTLDetector.cc
  → also contains special reconstruction classes (e.g. TowerBuilder, Digitizer, ...)
  → EventEvaluator class for analysis output located here
- **Detector (configuration) macros:** [github.com/ECCE-EIC/macros]
  → contains steering code for different detector setups (e.g. loading of different geometries, calibrations, exclusion of detector systems from simulation, ...)
- **Detector geometry inputs:** [github.com/ECCE-EIC/calibrations]
  → contains geometry input files (loaded in "macros"), e.g. tower position files for calorimeters, support input files from CAD, field map, ...)

Tutorial - Running simulations

- Use ECCE **Singularity**:
  `singularity shell -B cvmfs:/cvmfs cvmfs/eic.opensciencegrid.org/singularity/rhic_sl7_ext.simg`
- **Source environment** ([LOCALLIBS] is path to locally compiled code, e.g. $HOME/install):
  `source /cvmfs/eic.opensciencegrid.org/ecce/default/opt/fun4all/core/bin/ecce_setup.sh -n`
  `source /cvmfs/eic.opensciencegrid.org/ecce/gcc-8.3/opt/fun4all/core/bin/setup_local.sh [LOCALLIBS]`
- Move to detector steering macro e.g. **ECCEModular folder** and run code: `root.exe Fun4All_G4_ECCEModular.C(NEVT,PTLOW,PTHIGH,"DETSETTING","MCSETTING","INPUTFILE")`
  - **NEVT** = number of events
  - **PTLOW/PTHIGH** = p_{T} or p range for single particles (-1 when using event generators)
  - **DETSETTING** = special detector setups as a single string e.g. "STANDALONE_CTTL_ETTL_PIPE_display" for default Det1 setup use empty string ""
    → "STANDALONE" to only simulate detectors listed in DETSETTING string
    → "NOFIELD" to deactivate magnetic field
    → "display" to use GEANT geometry visualizer (for less complex geometry also interactive "displayviewer" can be used)
  - **MCSETTING** = single particle or event generator switch, e.g. "SimpleElectron"
    → "Single(Multi)Pion(fwd)" various options available for single particles (single particle, multiple particles, focuses on certain region, ...)
    → all basic particles can be selected in macro (add more if you want)
    → "PYTHIA6" or "PYTHIA8" when using generator, requires INPUTFILE to be set e.g. "phpythia6_ep18x275_q2_100.cfg"
- **To show current detector setup:**
  `root.exe Fun4All_G4_ECCEModular.C(1,0.3,30,"display","SimplePion","")`



Tutorial - Modifying and Compiling Code

How to compile code changes in coresoftware or fun4all_eicdetectors:

- Use ECCE **Singularity**:
  `singularity shell -B cvmfs:/cvmfs cvmfs/eic.opensciencegrid.org/singularity/rhic_sl7_ext.simg`
- **Source environment** ([LOCALLIBS] is path to locally compiled code, e.g. $HOME/install):
  ```
  source /cvmfs/eic.opensciencegrid.org/ecce/default/opt/fun4all/core/bin/ecce_setup.sh -n
  source /cvmfs/eic.opensciencegrid.org/ecce/gcc-8.3/opt/fun4all/core/bin/setup_local.sh [LOCALLIBS]
  ```
- **Prepare build and compile** code (e.g. for TTL):
  ```
  cd fun4all_eicdetectors/simulation/g4simulation/g4ttl
  mkdir build
  cd build
  ../autogen.sh --prefix=[LOCALLIBS]
  make install
  ```
- Changes will now be used if the **[LOCALLIBS] path is sourced before running the detector macro**, via:
  `source /cvmfs/eic.opensciencegrid.org/ecce/gcc-8.3/opt/fun4all/core/bin/setup_local.sh [LOCALLIBS]`

TTL Layers in Geant4

Support:

Layer	material	thickness
Top plate	aluminum	1mm
air gap	air	5mm
bottom plate	aluminum	1mm
cooling	aluminum	5mm diam. tube 1mm wall

Services:

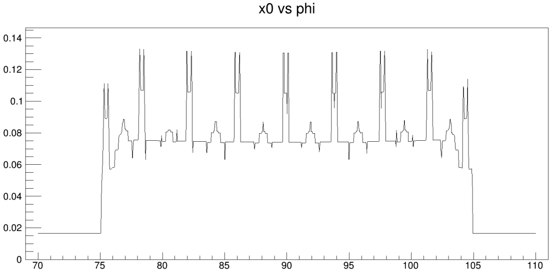
Layer	material	thickness
Thermal pad	graphite	0.25mm
High Speed Board	polystyrene	1mm
Power board	polystyrene	3.1 mm

Sensor:

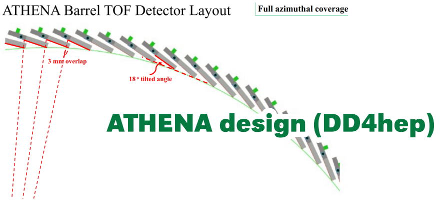
Layer	material	thickness
Thermal pad	graphite	0.25mm
AlN	AlN	0.79mm
Laird Film	graphite	0.08mm
ROC	plastic	0.25mm
Solder (Tin)	tin	0.03mm
Sensor	silicium	0.3mm
Epoxy	epoxy	0.08mm
AlN	AlN	0.51mm

More infos in CMS ETL TDR [[Link]]

TTL Layers in Geant4



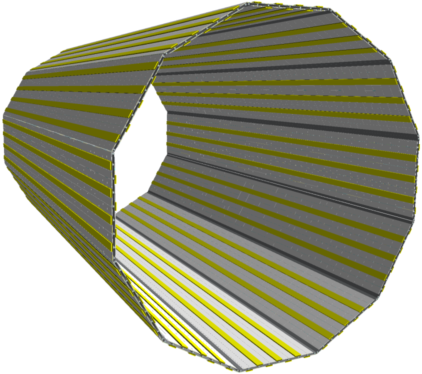
- Material budget $\sim 8\% X/X_0$ dominated by Al plates
  $\rightarrow$ cooling pipes with substantial material
- ATHENA barrel TOF $\sim 1\% X/X_0$
  $\rightarrow$ carbon foam/comb stave design

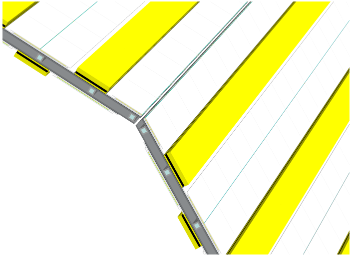


Oak Ridge National Laboratory

TTL Layers in Geant

- Barrel made of 12 modules in azimuth and multiple modules along z-axis
- Forward layers mounted on both sides of large disk

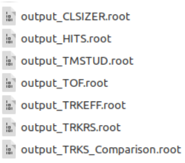


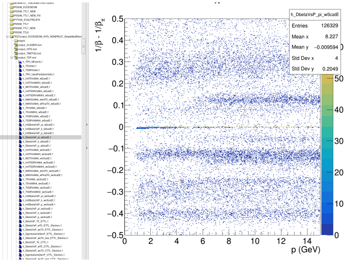


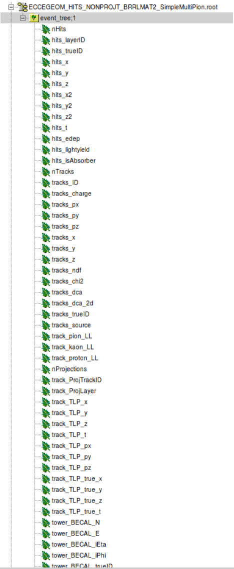
Analyzing TTL simulation output

- Recommended to use **Event Evaluator** output: EventEvaluatorEIC.cc
  → automatically enabled when using ECCEModular steering macro
  → produces event-based TTree with all necessary information (vertex, hits, tracks ,track projections , MC info, calo towers, ...)
  → TTL-relevant info (x, y, z, t, E, trueID of every hit)
- Tree post-processing with EIC Analysis Software
  → performs clusterization, MC info association, TOF PID calculation, and more ...
  → requires geometry.root (if calorimeters are used) and certain arguments

```
root -x -l -b -q treeProcessing.C+("G4EICDetector_eventtree.root","geometry.root","OUTPUT_NAME",-1,true,false,false)
```







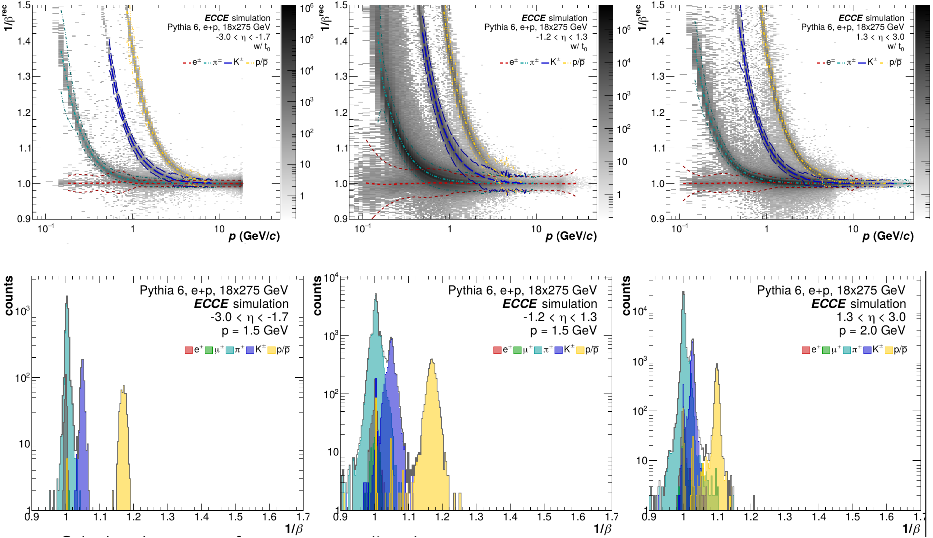
Analyzing TTL simulation output - 2

- TOF afterburner code in tofpid.cxx
  → calculates t_0, β, ...
- Current TOF code from Friederike Bock
- Performance (next slide) meets expectation
  → determine possible improvements and more realistic simulations

```cpp
  for(int it=0; it<nIterT; it++) mT0[it] = -9999;

  h_TPH_NEvents->Fill(0);

  // fixed mass assumptions
  float mElectron = TDatabasePDG::Instance()->GetParticle(11)->Mass();
  float mPion     = TDatabasePDG::Instance()->GetParticle(211)->Mass();
  float mProton   = TDatabasePDG::Instance()->GetParticle(2212)->Mass();
  float mKaon     = TDatabasePDG::Instance()->GetParticle(321)->Mass();
  //------------------------------------------------------------------------
  // determine start time based on scattered electron
  //------------------------------------------------------------------------
  int nScatElect  = 0;            // number of scattered electron candidates
  int scEIdx      = -1;           // what kind of determination
  Int_t nTracksWTime = 0;         // number of track with timing
  Int_t nTracksF     = 0;         // number of track with timing
  for(Int_t itrk=0; itrk<(Int_t)_nTracks; itrk++){
    if (_track_source[itrk] != 0) continue;
    if (_track_trueID[itrk] < 0) continue;
    nTracksF++;
    if (!_track_hasTTL[itrk]) continue;
    // count tracks with timing
    nTracksWTime++; // count tracks with timing

    // abort if not a probable scattered electron
    if (!_track_TOFmeas[itrk].at(0).isProbEScat) continue;

    TVector3 mcpartvec(_mcpart_px[(int)_track_trueID[itrk]],_mcpart_py[(int)_track_trueID[itrk]],_mcpart_pz[(int)_track_trueID[itrk]]);
    float trueP         = mcpartvec.Mag();
    // generated beta for electrons
    float betaGen = trueP/sqrt(trueP*trueP + mElectron*mElectron);
    // calculate average start time for single electron track
    float tstart  = 0;
    for (int tl = 0; tl < int(_track_TOFmeas[itrk].size()); tl++){
      float tflight  = _track_TOFmeas[itrk].at(tl).pathlength/(betaGen*c);
      tstart  += _track_TOFmeas[itrk].at(tl).recTime-tflight;
    }
    tstart  = tstart/_track_TOFmeas[itrk].size();
    if (nScatElect == 0)  mT0[0]  = tstart;
    else                  mT0[0]  += tstart;
    nScatElect++;
  }
//   cout << __LINE__ << endl;
  //------------------------------------------------------------------------
  // Did we find the scattered electron?
  //------------------------------------------------------------------------
  if (nScatElect > 0){
```

Analyzing TTL simulation output - 3

Plotting of PID studies via `pidreso_Pythia.C`

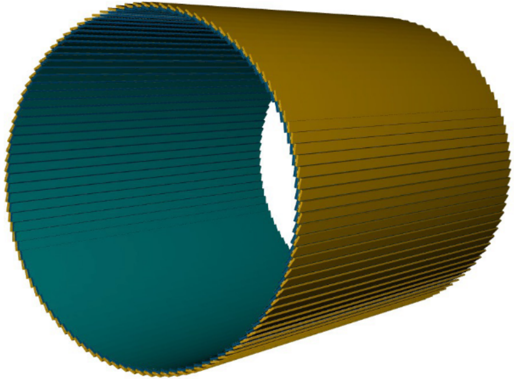


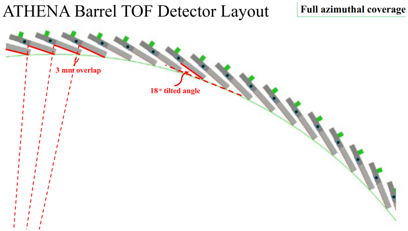
Critical next steps

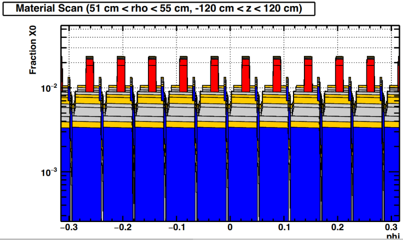
- **Compare implementations and physics performance** between DD4hep and Fun4All
  → ATHENA barrel design to be implemented in Fun4All as cross check
- Determine **further optimizations** of TTL design (barrel and forward!)
  → study impact of pixel or strip sensors
  → determine necessary overlap of sensors for maximal acceptance
  → integration of modules/staves into DIRC frame
  → maximize distance to vertex for all TTL layers
- Check **impact on physics**
  → make TOF information easily available for analyzers (EventEvaluator or Afterburner code?)
- Determine **material optimizations and global integration** with engineers
  → focus on minimizing support and cooling (depending on performance impact of other systems)
  → detailed CAD model needed soon (support, electronics, services)

Backup

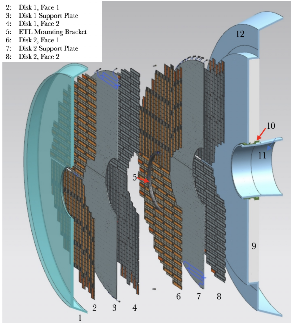
ATHENA barrel TTL

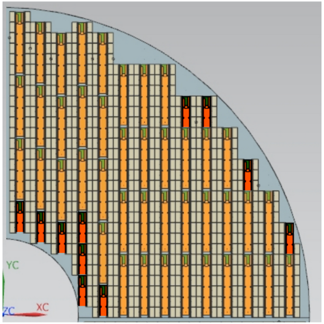




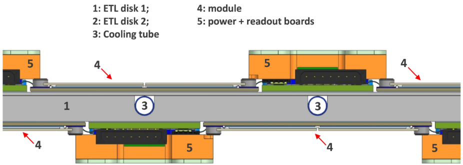


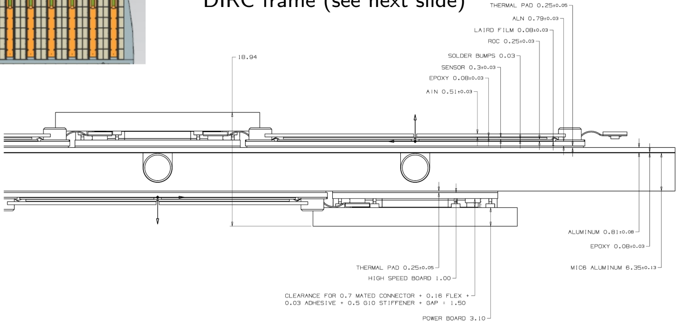
TTL disk design



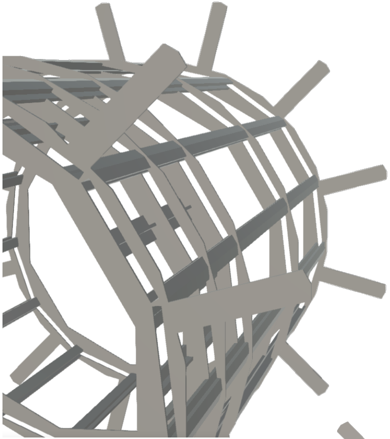


- Design based on the CMS forward upgrade [link]
- Basic elements: ladders of 3 or 6 LGAD sensors with service hybrid (for readout and power)
- Sensors mounted on aluminum plate (currently 6mm thick) and contains cooling
- Sensors on back side of plate shifted to cover service hybrid dead area (see bottom figure)
- Barrel layer to be mounted on inner or outer part of DIRC frame (see next slide)

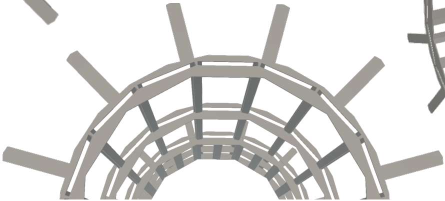


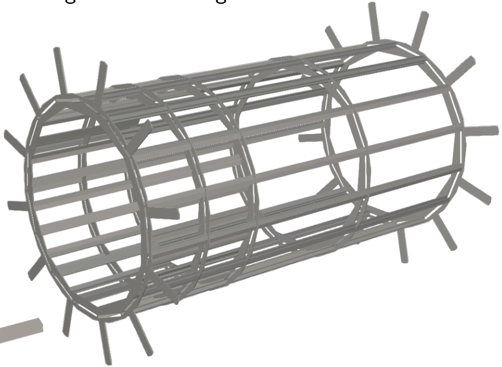


DIRC frame in barrel

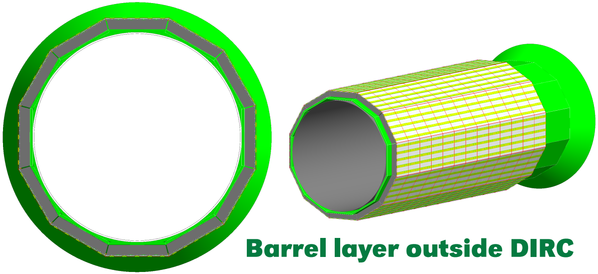


- Currently only stepping files of this frame exist (sent around by Tanja)
  → porting to Fun4All needed
- Frame allows to mount modules on various radial positions
- Considered material is steel at the moment
  → significant material budget in certain regions

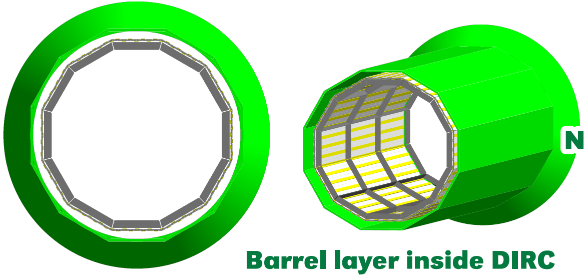




New Layers in Geant4 - 3

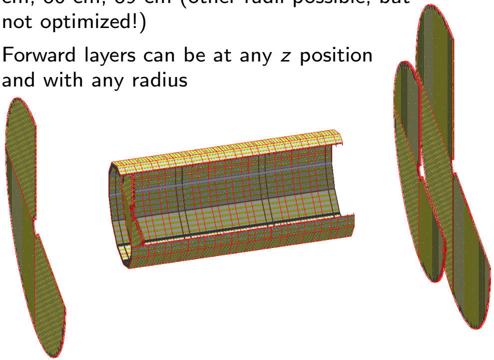


Barrel layer outside DIRC



Barrel layer inside DIRC

- Implemented barrel radial positions: 50 cm, 80 cm, 89 cm (other radii possible, but not optimized!)
- Forward layers can be at any z position and with any radius



New TTL layers in default ECCE configuration