Use of WHIZARD for the mass production of CLIC CDR volume 2

S. Poss for the PH-LCD group

CERN

November 22nd, 2011

Outline

Outline

Introduction

CLIC detector studies use ILC concepts as baseline: use common tools.
For generation, linear collider common generator tools group: use WHIZARD 1.95 as proved to be good for ILC LOIs.

CLIC detector studies as presented in the Conceptual Design Report (CDR): study 6 physics channels to assess detector performance.

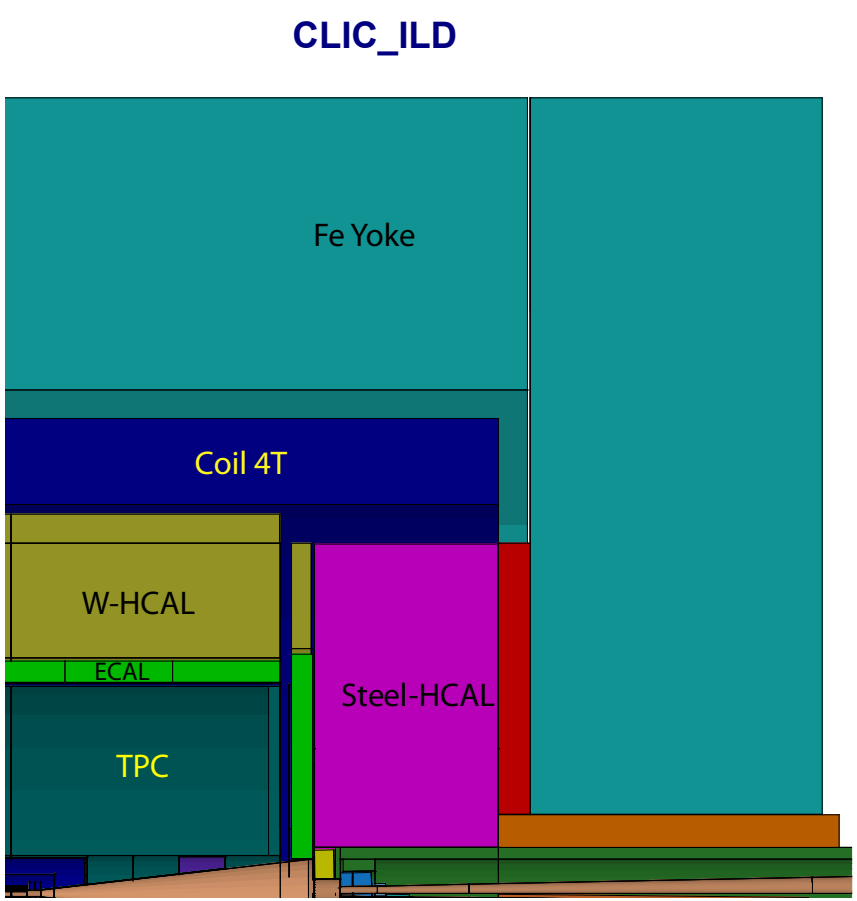
This presentation:

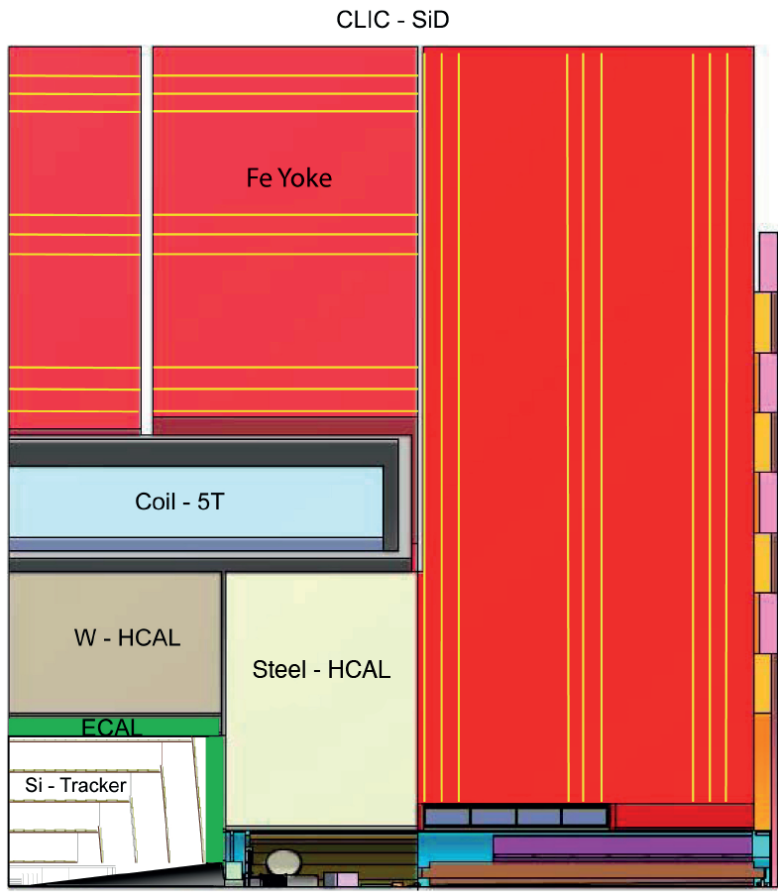
- Channels studied
- Tools developed for mass production
- Limitations

A short word on CLIC

CLIC is

- an e^+e^- linear collider
- operating nominally at 3TeV cme
- 2 detector concepts (ILD and SiD) based on ILC concepts





Outline

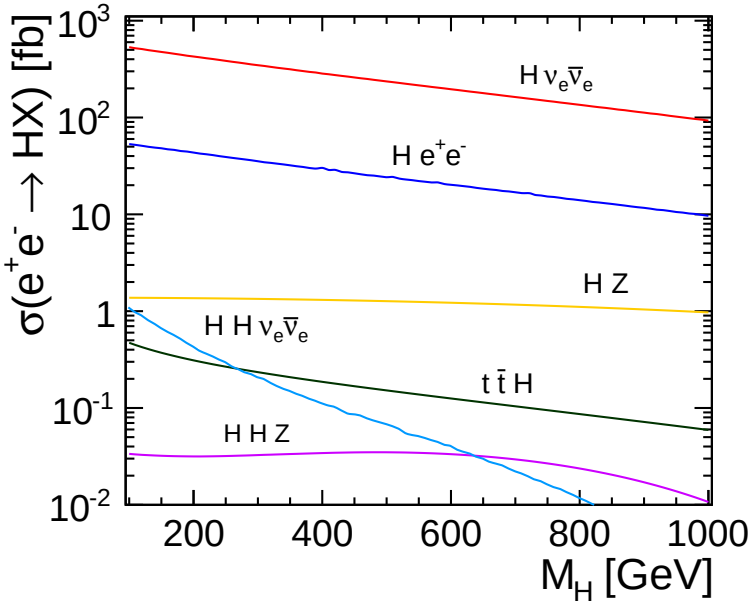
Physics channels studied

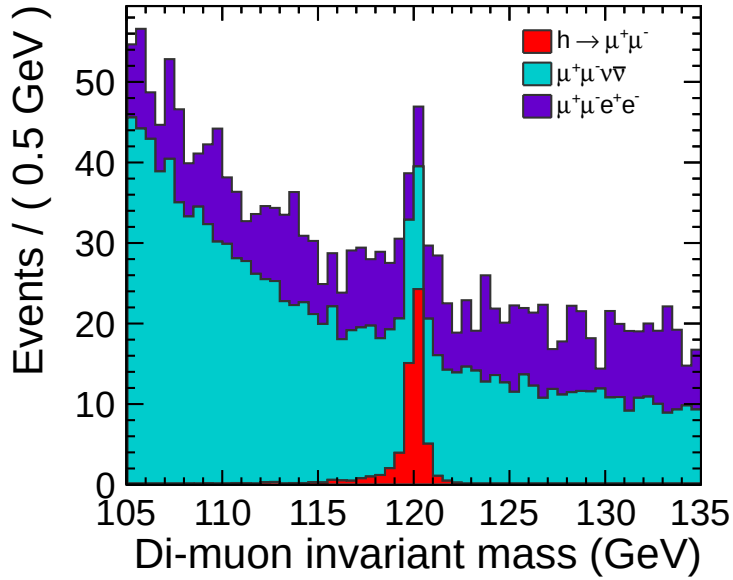
6 benchmark channels to assess detector performance:

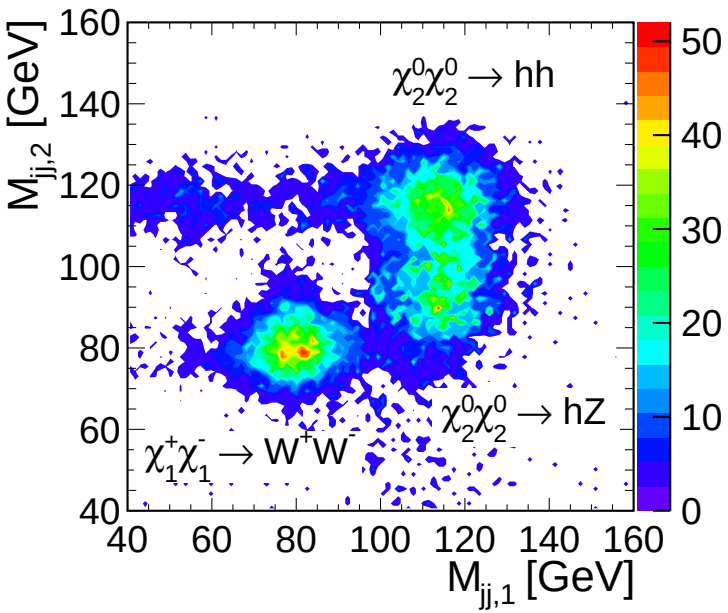
- $e^+e^- \to h\nu_e\overline{\nu}_e, h \to \mu^+\mu^-, h \to b\overline{b}$,
- $e^+e^- \to H^+H^-$, $e^+e^- \to H^0A$,
- $e^+e^- \to \widetilde{q}_R\overline{\widetilde{q}}_R$,
- $e^+e^- \to \widetilde{\ell}\overline{\widetilde{\ell}}\,(\ell = e, \mu, \nu_e)$,
- $e^+e^- \to \widetilde{\chi}^{\pm}_i\widetilde{\chi}^{\mp}_j$, $e^+e^- \to \widetilde{\chi}^0_i\widetilde{\chi}^0_j$,
- $e^+e^- \to t\bar{t}$ (500 GeV).

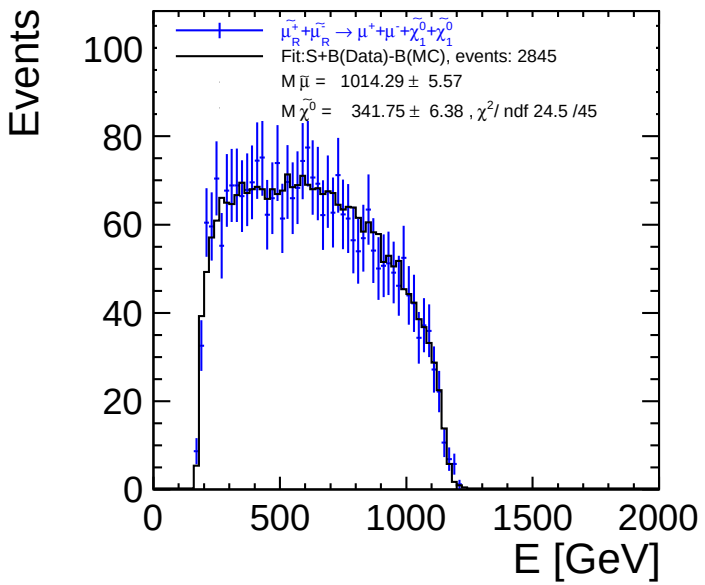
2 different SUSY models, 2 different energies (luminosity spectra), many background samples (SUSY and SM), millions of events needed

Examples of results









Outline

Modifications needed

ILC LOIs used WHIZARD.

Several tweaks implemented (T. Barklow, M. Berggren, A. Miyamoto):

- Installation tools
- Specific support for luminosity spectra: chose between ILC, CLIC, etc.
- TAUOLA support
- Stdhep output
- 6 fermions final state handling (for ILC LOIs): $t\bar{t}$ reconstruction
- Color flow (for DBD studies)

Production framework

CLIC detector studies use the GRID for mass production.
Need to run many applications in heterogeneous context:
ILCDIRAC

- Based on DIRAC: grid solution developed initially for the LHCb experiment. PYTHON based.
- Comes with production system: define task and let the system create and monitor the jobs.

Needed to interface WHIZARD in ILCDIRAC:

- Run configuration: ILCDIRAC configuration is in PYTHON $\rightarrow$ convert to whizard.in
- Process selection: Make sure all relevant files (e.g. LesHouches files) are available

Example of job definition

```
from ILCDIRAC.Interfaces.API.NewInterface.Applications import *
from ILCDIRAC.Interfaces.API.NewInterface.UserJob import *
from ILCDIRAC.Interfaces.API.DiracILC import *
d = DiracILC()
j = UserJob()
wh = Whizard(processlist=d.givePro cessList())
pdict = {'process_input':{'process_id':"h_n1n1","sqrts":3000.,
                          "polarized_beams":"F"},
         "simulation_input":{"n_events":10000}}
wh.setFullParameterDict(pdict)
res = j.append(wh)
if not res['OK']:
    print res['Message']
d.submit(j)
```

Defines the WHIZARD executable to install.

What happens on the grid node

1. Install software if not available
2. Get files: user whizard.in, LesHouches file
3. Set whizard.in as needed
4. Set the environment: read the lumi spectra, set the library path
5. Run WHIZARD
6. Treat output: rename according to user convention and upload to Storage Element

Running WHIZARD in ILCDIRAC

Problem: How to prevent users from trying non-existent channels?

- Store the content of the whizard.prc in a file that can be used at submission time by DIRAC
- Keep relation between WHIZARD version and channels (SM vs SUSY)

Problem: How to reduce configuration issues?

- Catch all errors before the job is submitted
- New functionality: all WHIZARD options are wrapped in a XML file, also holds default values and types

Running WHIZARD in the production for the CLIC CDR

- Framework is global: generation, simulation and reconstruction of events done using same tool (ILCDIRAC)
- Account for simulation/reconstruction CPU time constraints on the GRID (24 hours max on average)
- Account for optimal file sizes: Storage Element access is the most problematic

$\Rightarrow$ need to generate $10 - 1000$ events per job depending on the channel.

Outline

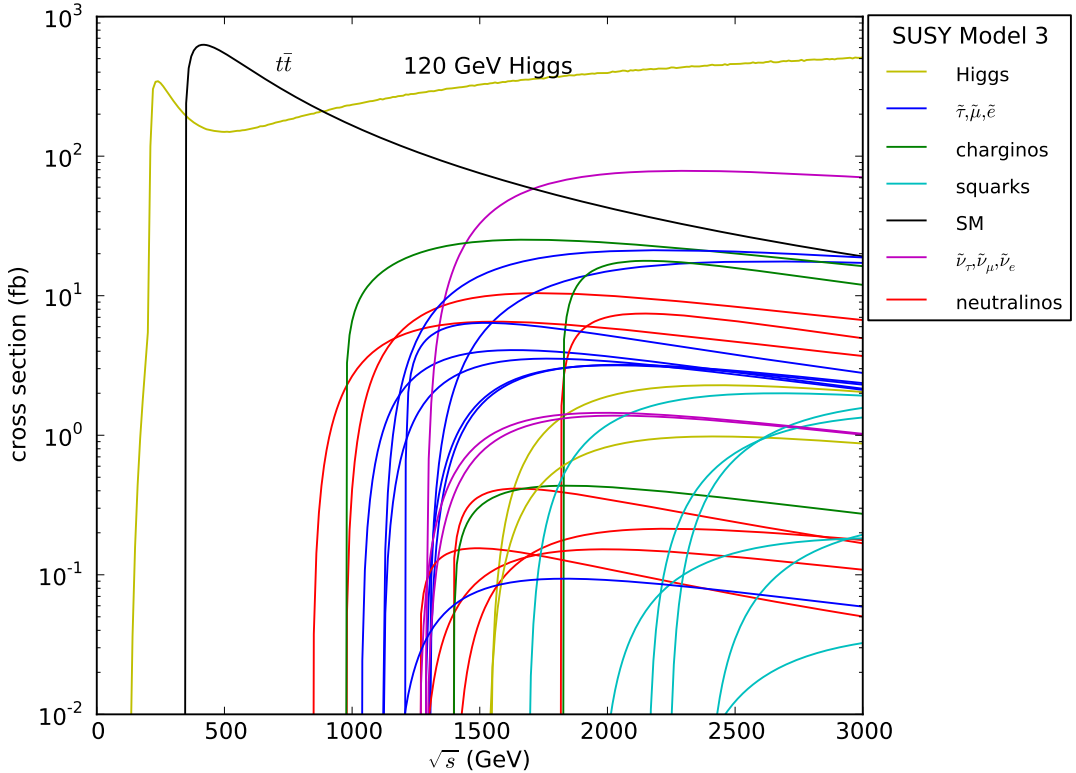
Limitations met

- Impossible to generate specific final states with width: $t\bar{t}$, WW, ZZ, HH, HA. Used PYTHIA standalone for those. For SM Higgs processes assume width to be 0.
- Would have been useful to be able to disable diagrams explicitly to have pure signal samples
- Generator level cut interface not easily generalized: had to add extra filtering programs to run after WHIZARD.
- Process selection for people not compiling WHIZARD: Adding a new process requires compiling a WHIZARD executable, not easy from scratch

Outline

Using WHIZARD 2.0

Using analysis framework of WHIZARD 2:



Includes CLIC 3TeV luminosity spectrum.

Outline

Conclusion

For the CDR:

- Very complete software
- Accurate cross sections
- Efficient event generation
- Lightweight to run on the GRID once compiled
- Compilation is tricky
- Some channels could not be done ($t\bar{t}$, WW, ZZ, HH, HA)

For WHIZARD 2.0:

- Easier process selection
- Work on run configuration
- 8 fermions final states (ILC DBD studies)

Conclusion

CDR has been internally reviewed, visible here:
`https://edms.cern.ch/document/1160419`

Join the signatories here:
`https://indico.cern.ch/conferenceDisplay.py?confId=136364`

Backup slides

Requests for the future

- Mechanism to suppress explicit contributions in processes
- Do not depend on compiler at run time (More or less OK if I understand)
- Interface to other languages (e.g. XML, PYTHON)

Making WHIZARD 2.0 and ILCDIRAC play well with each other

- Install WHIZARD 2.0 on an ILCDIRAC server
- Interface properly in ILCDIRAC (.sin content)
- Make the central installation compile the process on the fly (Request mechanism)
- Register that WHIZARD version as a new application and install it on the sites
- Run the user job against this version when compilation is done

Needs to hold on some database the processes (can use some checksum mechanism) and the versions.