GLAST The Gamma Ray Large Area Space Telescope

Concept for a

GEM Configuration Tool

Martin Kocian

SLAC, 16 November 2005

Introduction

- The goal is to create a GEM configuration tool with the following features:
  - Input and output of the configuration should be human readable and convenient.
  - The tool should fully reflect the abilities of the hardware.
  - The configuration should be easy to store in a database.
  - The same software object should be used
    - to create and read configurations.
    - to configure the hardware/produce hardware configuration files
    - to retrieve configuration information in reco/analysis and simulation code.

Trigger parameters

- The “real” configuration is a number of GEM register settings
- These settings include
  - Lookup table for the mapping of conditions and trigger engines
  - Engine definitions
  - Input enables
  - Window open mask and window width
  - Periodic trigger setup
  - ROI mapping

Configuration script

- An xml file or any register based configuration would be too cryptic to specify or understand a configuration
- Instead, use a script that defines the configuration
- The script can be parsed by standard tools (lex/yacc) that verify its grammar and report inconsistencies and warnings.
- A script can be stored in a configuration database without having any problems with schema evolution. The code can deal with any backward compatibility issues.
- The tool should be able to dump a configuration in the same format which makes it easy to make changes to the scripts.
- Comments can be used in the script to illustrate its intent.

Example configuration script

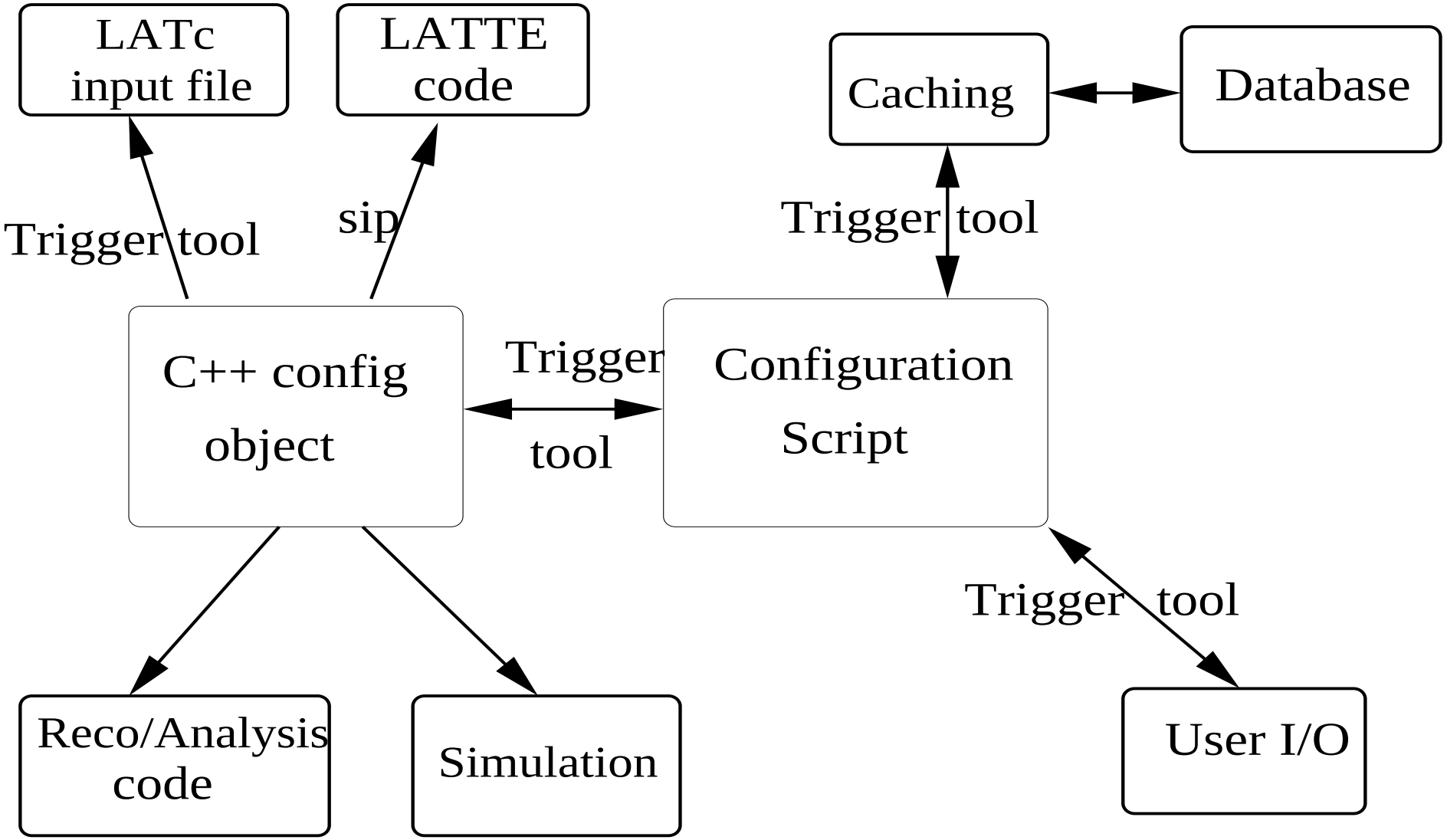
```
[Window]
Window_open_mask: TKR CALlow CALhigh EXT
Input_enable_mask: TKR, CALlow, CALhigh, CNO, ROI, EXT
Window_width=12

[Trigger Engines]
# Format is engine: actions options
Engine 0: Inhibit
Engine 1: TACK, 0-suppression
Engine 2: TACK, 4-range_readout
Engine 4: Calstrobe, TACK, 4-range_readout
Engine 6: TACK, precale=5

[Trigger definitions]
Periodic: engine 1 # highest priority, always get periodic triggers
TKR: engine 6
ROI: engine=0 # inhibit ROI
CNO: engine=1
CALhigh &! CALlow: engine=2
only CALlow: engine=2 #only one entry in the LUT is affected
Solicited: engine=4

[Periodic_Trigger]
1_PPS, Free_running, Limit = 10 Prescale = 10
#one could add more complicated configurations, e.g. autoscaling
```

Schematic view of configuration



Full trigger configuration

Scripts could be extended to provide a coherent definition of the trigger (GEM + filters).

```
[Conditions Definitions]:
Line 1: ROI, Engine 1 #GEM part

[Filter definitions]
Input=Line 1, action=Passthrough, prescale=10
```

Summary

- Same configuration object in online and offline
- Scripts are the source of the GEM configuration.
- The scripts are the part that is persisted in the database.
- A database system should be able to do caching so multiple machines can access the same data easily, e. g. in simulation.
- A ROI configuration could be added.
- Flight software filters could be added to capture the trigger configuration in one place.