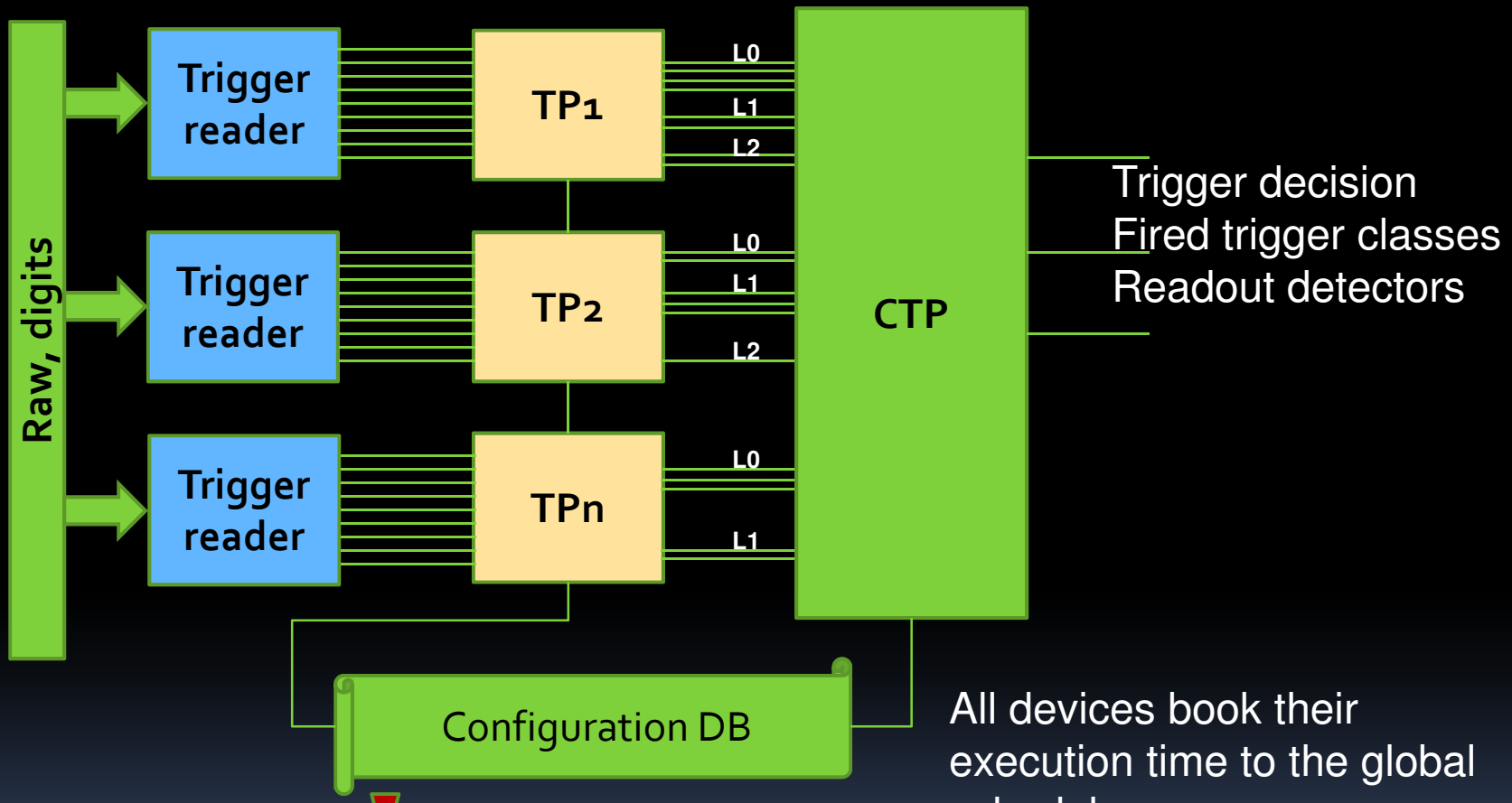
A.Gheata, ALICE offline week 26 October 2009

NEW TRIGGER SIMULATION FRAMEWORK

Main features

- Replay decisions of trigger chain
- Trigger efficiency studies
- Provide base classes for describing trigger processors and their interactions
- Handle L0, L1, L2 and support time as variable
- Provide access to trigger info and serialize trigger decision in ESD

Top view

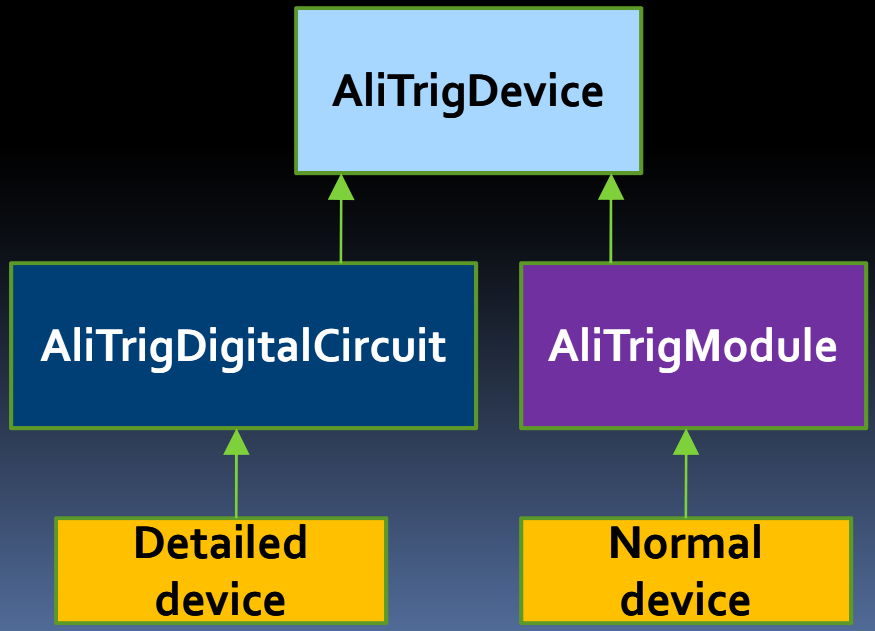

Raw, digits
Trigger reader
Trigger reader
Trigger reader
TP1
TP2
TPn
L0
L1
L2
L0
L1
L2
L0
L1
CTP
Trigger decision
Fired trigger classes
Readout detectors
Configuration DB
All devices book their execution time to the global scheduler
Global time

Current status

- New module $ALICE_ROOT/TRIG
- Base classes describing
  - Generic trigger "device"
    - Modules handling general signals or digital circuits
    - Describe input/output, connectivity and provide signal handling
    - User defined response functions
  - Signals and connectors
    - Digital data or generic messages
  - Scheduler
    - Registers and replays timed response functions from devices
- Implementation of some existing trigger modules
  - Ongoing work
- To be committed in 1-2 weeks max

(AliTrig)Device

$Oi = f(ik, \ldots il, <config>)$



Trigger devices

- Any trigger processor is a 'device' (Including CTP)
- Implement mandatory methods:
  - CreateDevice(Option_t *option): Create and configure inputs and outputs, register response functions. The implementation may be configuration dependent
  - Trigger(Int_t ioutput): Custom response function that fires (or not) a given output
- Registration of response functions to the global scheduler:
  - RegisterResponseFunction(AliTrigScheduler *s, Int_t output, Int_t delay, Int_t group): Output to be fired, delay in arbitrary global units, scheduled group number.
- Connect to other devices and register to a scheduler
  - Connect(Int_t output, AliTrigDevice *other, Int_t at_input)
  - Register(AliTrigScheduler *calendar)

AliTrigDigitalCircuit

- Arrays of TBits as inputs and outputs
- One output and one response function
- Implementations provided for basic logical circuits
- Can be used to implement a detailed circuit logic if needed

AliTrigModule

- Generic AliTrigSignal as inputs and outputs
- Can be used to mask the details and complexity of trigger processors
  - Simplest case: reader device TP. The only input provides an object embedding an array of input values. The only output provides a custom response object
- Caveat: introduce direct dependencies between trigger devices (at least at CTP level)

AliTrigHybridModule

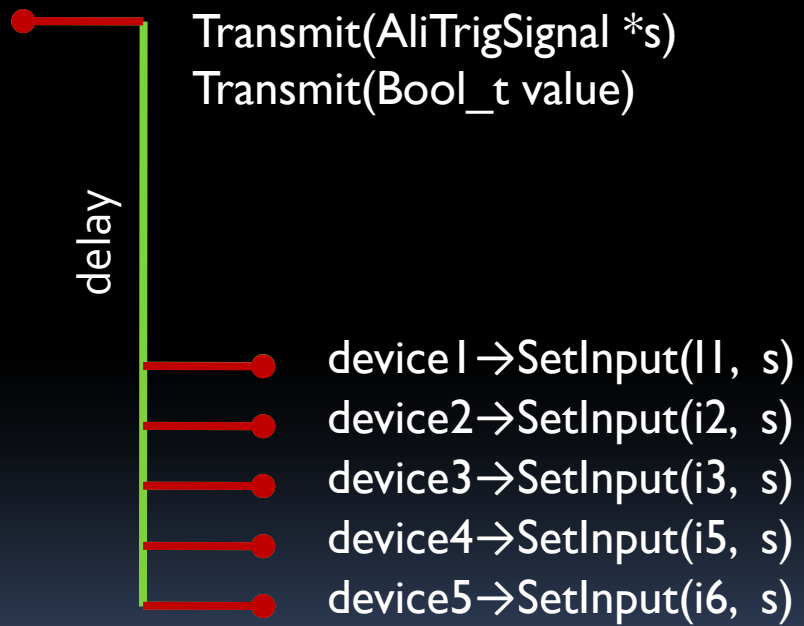
- A combination of the two: a set of binary inputs + a set of signal ones
- Useful to describe decision-type inputs or outputs (like BUSY)
- CTP module will derive from this class

AliTrigConnector

AliTrigConnector(AliTrigDevice *feeder, Int_t output)

Feeder device and slot

Transmit(AliTrigSignal *s)
Transmit(Bool_t value)



Devices and inputs

AliTrigSignal

- Containers for the state of inputs and outputs (name choice not the best – maybe AliTrigSlot ?)
  - Schema changed – notification is not done via signals , it is a scheduler that processes a trigger
- Can be active or not and embeds a value (of arbitrary type)
  - ImportData(AliTrigSignal *other): Able to import the value from another signal (of the same type)
  - Connectors can group only compatible signals
- Predefined signal types: AliTrigSignalWithBool, AliTrigSignalWithBits, AliTrigSignalWithObject

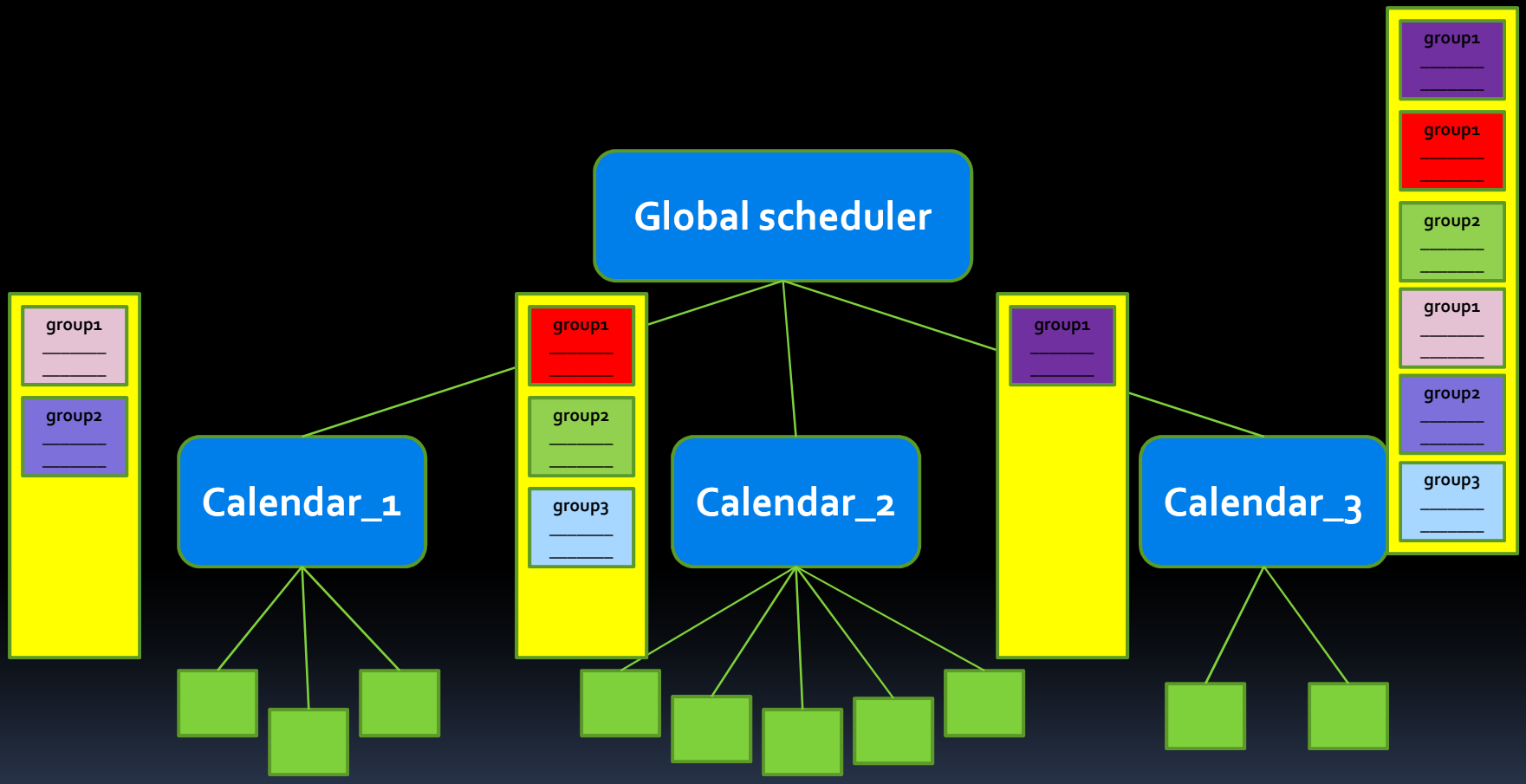
AliTrigResult

- "An object is needed for storing the trigger result and make it persistent in the ESD" (Matthias)
- Derives from a signal, so that it can be directly a device output
- Streamable object defined by each trigger processor
- Additional functionality ? We will see.

AliTrigScheduler

- Base manager class for trigger chain simulation
- Any trigger module registers to a scheduler
- A scheduler manages one or more modules
  - ConnectModules(): pure virtual method describing the connection between the managed devices
- Devices register their (timed) output response functions to their scheduler
  - Response functions are ordered in time
  - Response functions can be grouped and groups have different priorities (may override time priority) . To avoid deadlocks, non executed signal outputs are marked as inactive.
- All schedulers are registered to a global scheduler
  - At the limit: one scheduler (global one) and one priority group.

Trigger response scheduling



Conclusions

- Base classes for the new trigger framework implemented
  - To be committed in SVN in 1-2 weeks
  - libTRIGbase
- The new schema is more flexible and allowing as well detailed or coarse simulation of trigger processors
  - Allows a smooth transition from existing trigger framework
  - Much easier to extend
- Time component natively introduced in the framework
  - Can be used or not
  - Scheduling based on timed response functions + extra custom priorities for more flexibility
- Time to start convert the existing implementation of TPs to the new framework
  - Device configuration will be translated from the existing framework