

Configuration Database Issues: DT Track Finder, Global Muon Trigger, Global Trigger

Hannes Sakulin, HEPHY Vienna

On Behalf of the Vienna CMS Trigger Group

CMS Database Workshop – Configurations DB

CERN, 25th February, 2004

URL of this presentation:

<http://www.hephy.oeaw.ac.at/p3w/cms/trigger/globalMuonTrigger/trans/TriggerViennaDB24Feb2004.pdf>

- Cover only Drift Tube Track Finder, Global Muon Trigger, Global Trigger
 - ⇒ only part of the CMS trigger system

- Cover only configuration aspect
 - ⇒ configuration data and how we get it into a number of electronics boards containing FPGAs, registers, memories
 - ⇒ does not include logging, monitoring, testing, ...

- Only a first idea
 - ⇒ **material is very preliminary**
 - ⇒ **expect further iterations !!!**

➤ Intro

- ⇒ Drift Tube Track Finder (DT)
- ⇒ Global Muon Trigger (GMT)
- ⇒ Global Trigger (GT)

➤ Requirements

➤ Classification of configuration data to be stored

➤ Configuration data to be stored for DT, GMT, GT

➤ Environment: Run Ctrl, XDAQ Application, Configuration DB

➤ Configuration procedure

➤ (Open) Issues

Pipelined 40 MHz, Latency <math>< 3.2 \mu\text{s}</math>

Calorimeter Trigger

HF HCAL ECAL

Regional Calorimeter Trigger

Global Calorimeter Trigger

Muon Trigger

RPC CSC DT

Pattern comparator trigger

CSC local trigger

CSC Track Finder

DT local trigger

DT Track Finder

Global Muon Trigger

L1 Global Trigger

MIP+ ISO bits

4+4 μ

4 μ

4 μ

4 μ (with MIP/ISO bits)

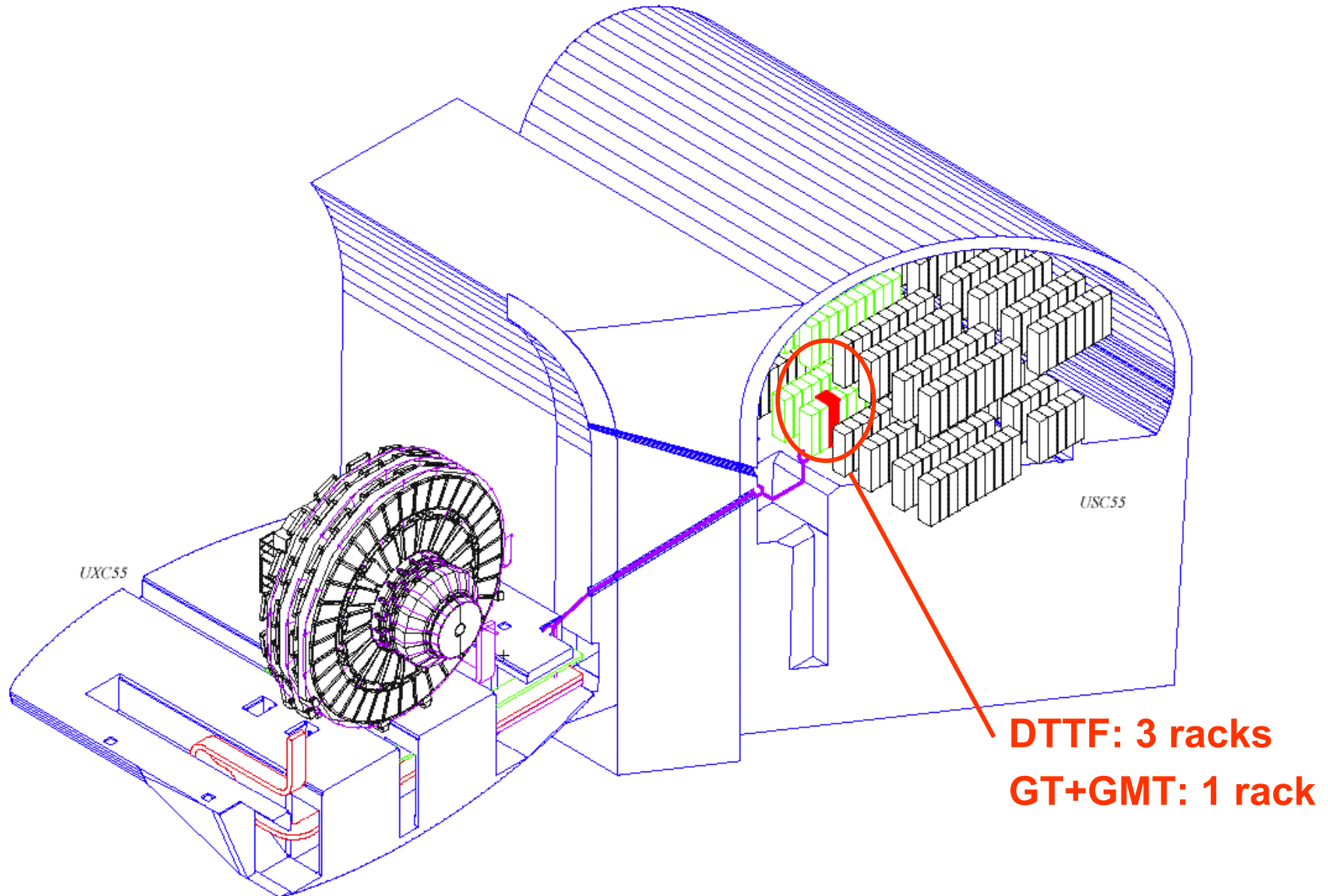
e, j, E_T , H_T , E_T^{miss}

L1 Accept

Responsibility of Vienna Trigger Group
 ⇒ Scope of this talk

max. 100 kHz

System overview (Racks)



- Quick setting up of trigger hardware for given configuration
 - ⇒ need to provide all necessary information

- Propagate configuration changes to HLT
 - ⇒ HLT checks L1 decision for some fraction of events (comparison with ORCA simulation)

- Need some simple GUI to change configuration parameters
 - ⇒ e.g. visualize settings based on eta-phi
 - ⇒ e.g. visualize bunch structure

- Need tool to select configuration (tagging)

- Traceability of Trigger Configuration
 - ⇒ Configuration becomes condition data

➤ Firmware version

- ⇒ Version of firmware to be loaded into each FPGA
- ⇒ Store in DB only the bookkeeping information and pointer to firmware
- ⇒ Store the actual firmware programming file on a server (files? http? CVS?)

➤ Register contents

- ⇒ register contents: 8, 16 (, 32) bit registers
- ⇒ may use generic solution
(e.g. GenericConfigurator by Nuno Almeida)

➤ Parameters

- ⇒ Run parameters in human-readable format
- ⇒ Register (and Look-up-Table) contents **are calculated** from these parameters
- ⇒ Need to store revisions of conversion software used to convert parameters to register contents / LUTs

➤ Store in DB

⇒ Revision nr

⇒ Firmware revision ID (32 bit)

- this is readable from the chip via VME or JTAG
- allows identification of currently loaded chip configuration

⇒ “url” of programming file

⇒ hash code / CRC code over programming file

- consistency check of cached copy

⇒ CVS tag of firmware sources

- should include all VHDL sources, schematics, constraint files used to produce the firmware

⇒ comment / description of the version

⇒ corresponding default parameters ?

⇒ corresponding address table ?

⇒ corresponding parameter structure ?

➤ Where do we store the actual programming files?

➤ Store

- ⇒ Revision nr
- ⇒ (Set of) Register contents (8, 16, 32 bit unsigned integers)
- ⇒ Description of meaning of the contents

➤ Split into separate tables

- ⇒ registers that change rarely / registers that change frequently

➤ Store also address map ?

- ⇒ would allow evolution of the address map (e.g. with evolution of firmware)
- ⇒ linked to firmware configuration
- ⇒ foreseen in HAL?

➤ Store

- ⇒ Revision nr
- ⇒ Set of parameters
- ⇒ Description / comments
- ⇒ Revision of conversion software to convert the parameters to LUTs / registers (CVS tag)

➤ Split into separate table

- ⇒ parameters that change frequently / parameters that change rarely

➤ How to foresee an evolution of the parameter structure ?

- ⇒ e.g.: add more configurables

- Currently somewhat different ideas of how to store information
 - ⇒ e.g.: store registers / store parameters from which registers are calculated

- Different handling of configuration data
 - ⇒ DT: Change of Look-Up-Tables requires firmware change
 - ⇒ GMT: Firmware static.
Registers and Look Up Tables can be changed by VME
 - ⇒ GT: Trigger algorithm change requires firmware change.
Only thresholds changed by VME.

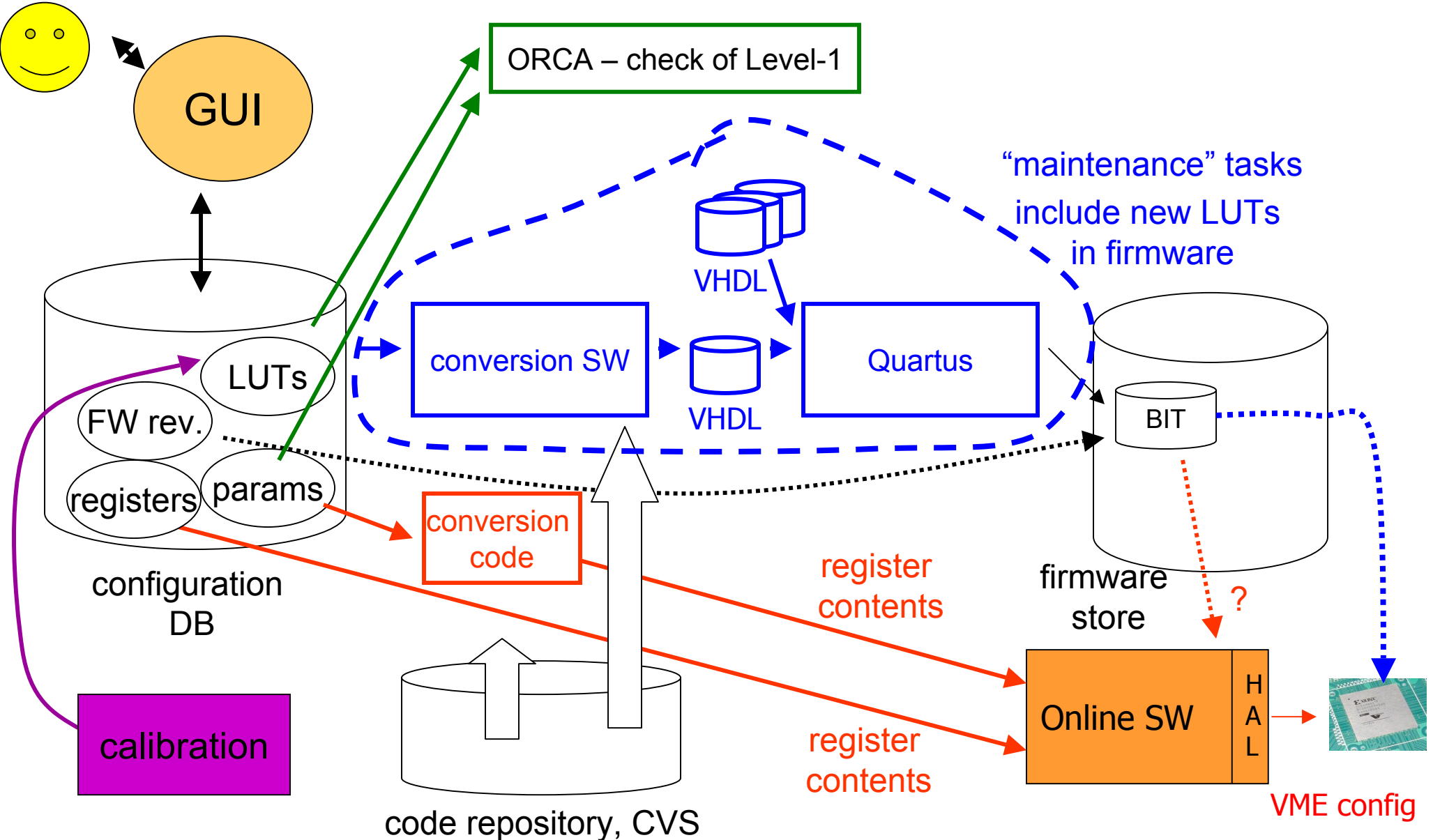
- In the following for each system ...
 - ⇒ Overview over configuration data
 - ⇒ Overview of data handling

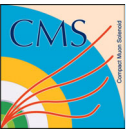
board	# brd	# FPGAs, firmware	VME registers	parameters	conv. SW (tag)
Phi Track Finder	72x	2 O(1 MB), variable** includes LUTs for extrapolation, p_T , ϕ assignment	[3 registers sync., board control few bytes] determined by parameters	parts of registers: few bytes LUTs for extrapolation, p_T , ϕ assignment* O(1 MB)	calculate registers from parts
Eta Track Finder	12x	3 O(1 MB), variable** includes LUTs for η assignment	[2 registers sync, board ctrl few bytes] determined by params	parts of registers, few bytes LUTs for η assignment* O(100 kB)	calculate registers from parts
Wedge Sorter	12x	2 O(1 MB), constant*	2 regs: sync, board ctrl 12 regs sort config few bytes		
Barrel Sorter	1x	2 O(2 MB), constant*	2 regs: sync, board ctrl 24 regs sort config few bytes		
TIM	7x	1 constant*	always defaults?		
Data Concentrator	1x	1 O(1 MB) constant*			

* not needed in HW configuration (contained in firmware), but needed by ORCA

* constant once the HW is debugged. Changed only in case of upgrade.

** changes not frequent





Configuration Data: Global Muon Trigger

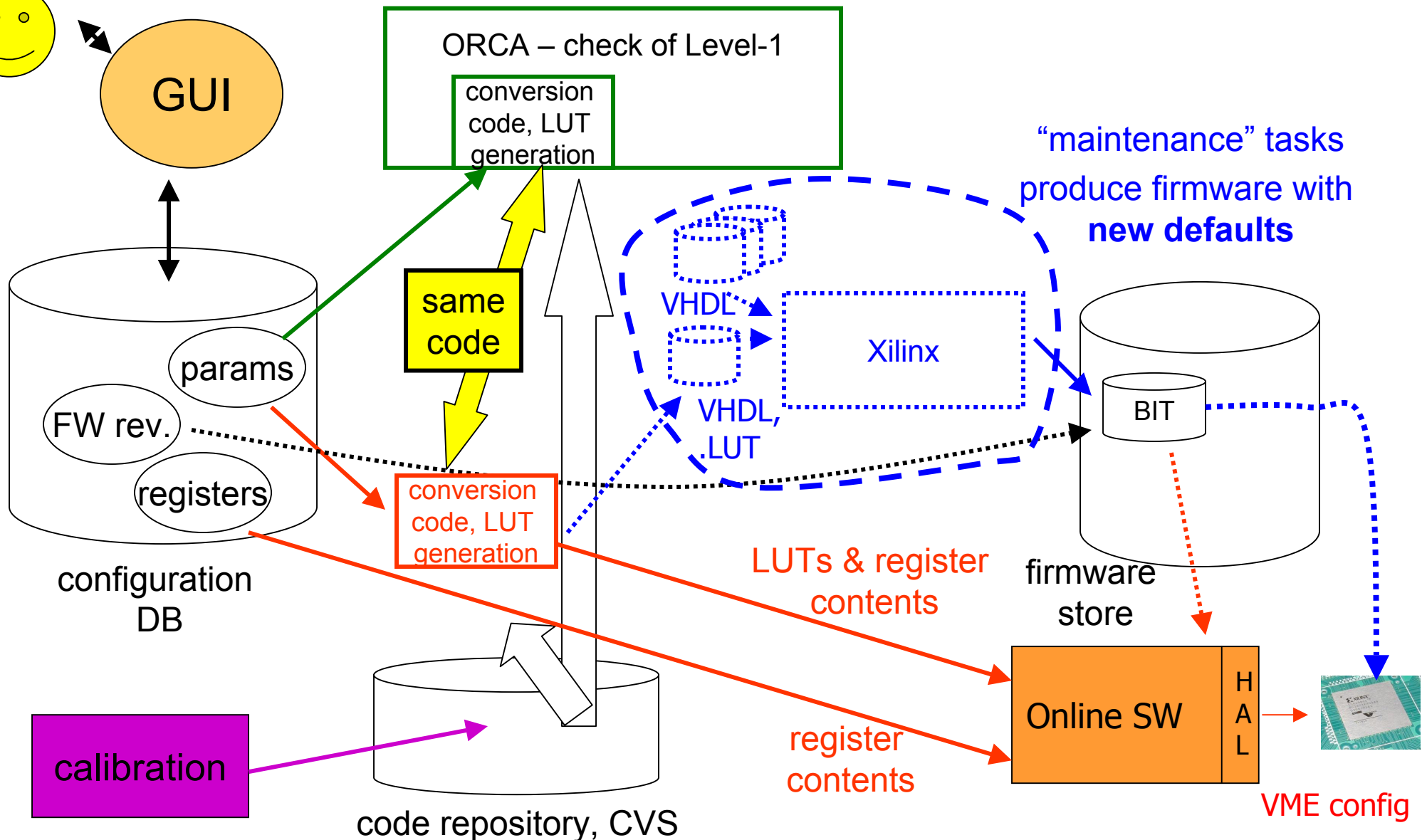


board	# brd	#FPGAs, firmware	VME registers	parameters	Conv. SW – store tag
Pipeline Synchronizing Buffer	3x	1 O(1 MB) same for each instance constant*	5 registers (sync) O(10 bytes)	-	-
Global Muon Trigger Logic Board	1x	9 O(10 MB) include default values of 21 types of LUTs constant* (except for including new default LUTs and registers)	20 registers (sync) O(10 bytes)	parameters of 20 config registers, few bytes parameters of LUTs*: few bytes	a) calculate registers from parameters b) LUT generation on the fly

* LUTs (1 MB) are not stored in configuration DB but calculated on the fly when needed

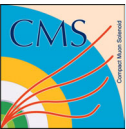
* constant once the HW is debugged. Changed only in case of upgrade.

Handling of Configuration Data: GMT

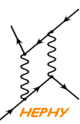


board	# brd	#FPGAs, firmware	VME registers	parameters	Conv. SW – store tag
Pipeline Synchronizing Buffer (PSB)	4x	1 O(1 MB) same for three instances, different for one constant*	16 registers (sync) 32 bytes	-	-
Global Trigger Logic Board (GTL)	2x	5 O(1 MB) <u>for 2 FPGAs:</u> frequent changes depending on trigger algorithms (firmware is a result of def.xml)	[~400 threshold regs. depending on config. O(1 kB)] determined by parameters 2 bc tables (16 kB)	def.xml configuration file. Includes algorithm definition and threshold settings. few kilobytes	GTS calculate firmware and registers from def.xml
Final Decision Logic (FDL)	1x	1 constant*	8 mask regs, 16 other regs 256 downscaling factors		

* constant once the HW is debugged. Changed only in case of upgrade.



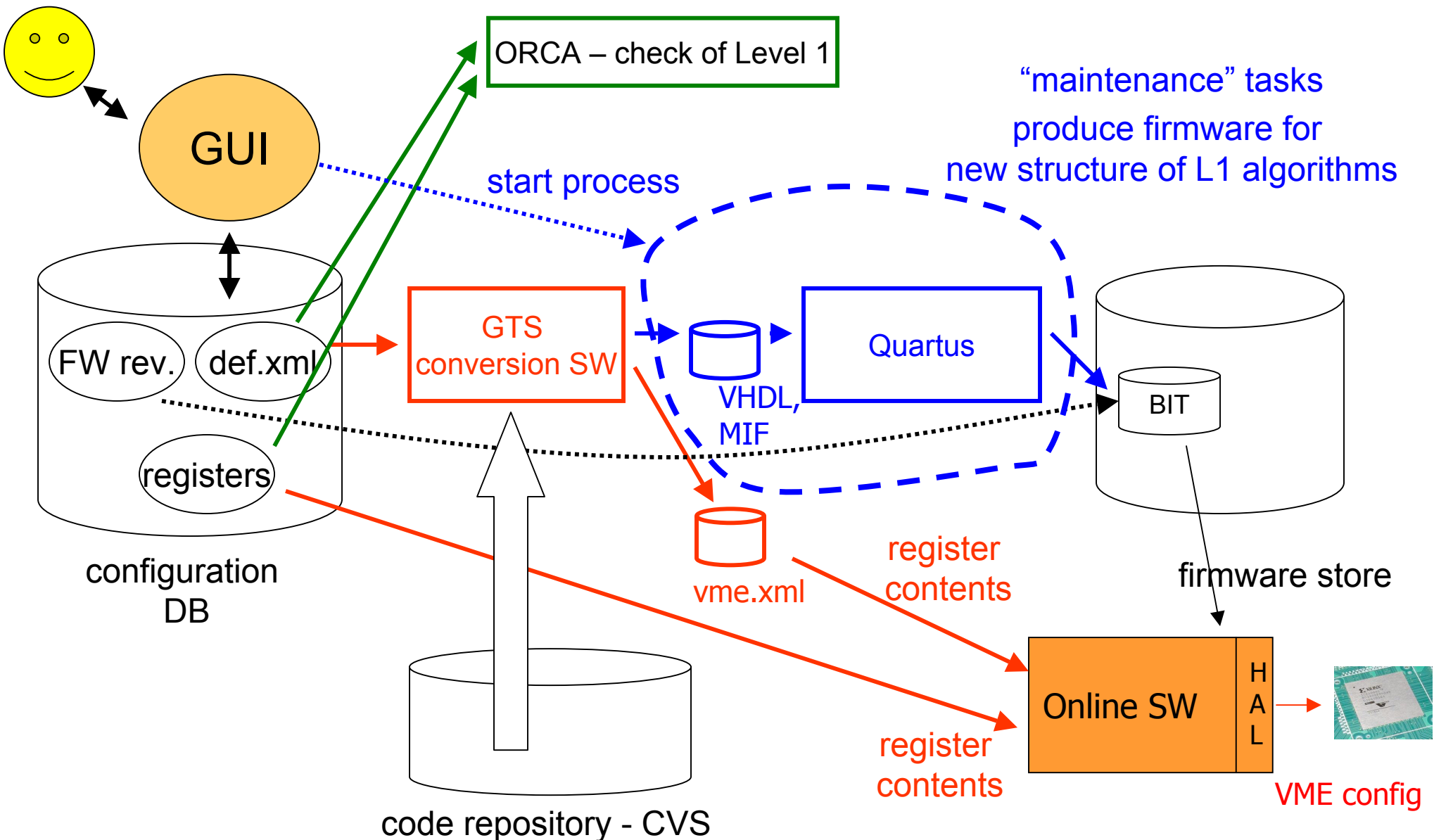
Configuration Data: Global Trigger (ff)



board	# brd	#FPGAs, firmware	VME registers	parameters	Conv. SW – store tag
Timing (TIM)	1x	1 constant*	47 registers: delays, readout BC table (8 kB)		
Trigger Control System (TCS)	1x	2 constant*	registers: 3 VME regs 32 common regs 64 partition regs (200 bytes) 16 BC tables (128 kB)		
Global Trigger Front End (GTFE)	1x	2 constant*	40 registers : delay, ID, event_length		

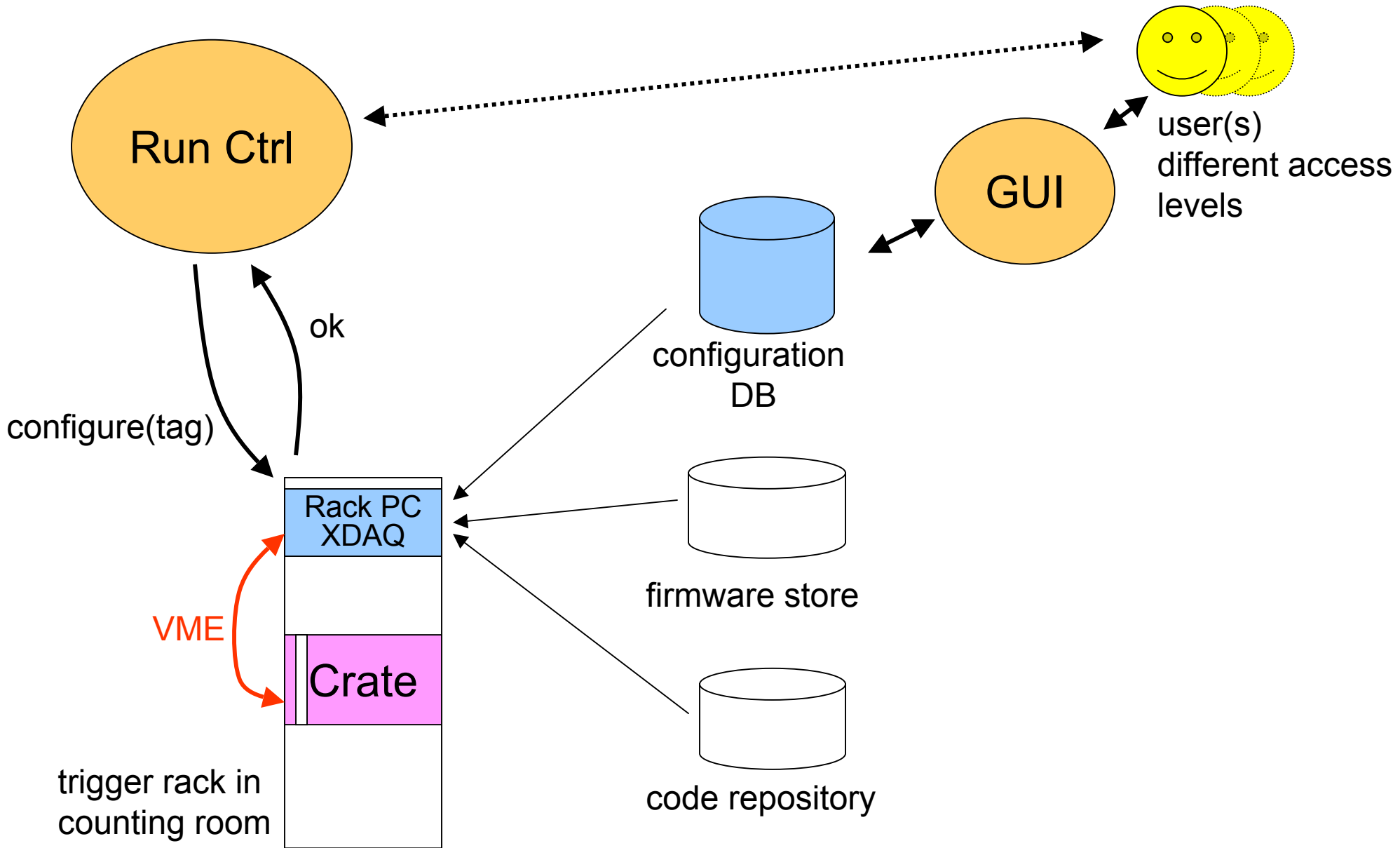
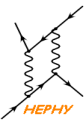
* constant once the HW is debugged. Changed only in case of upgrade.

Handling of Configuration Data: GT

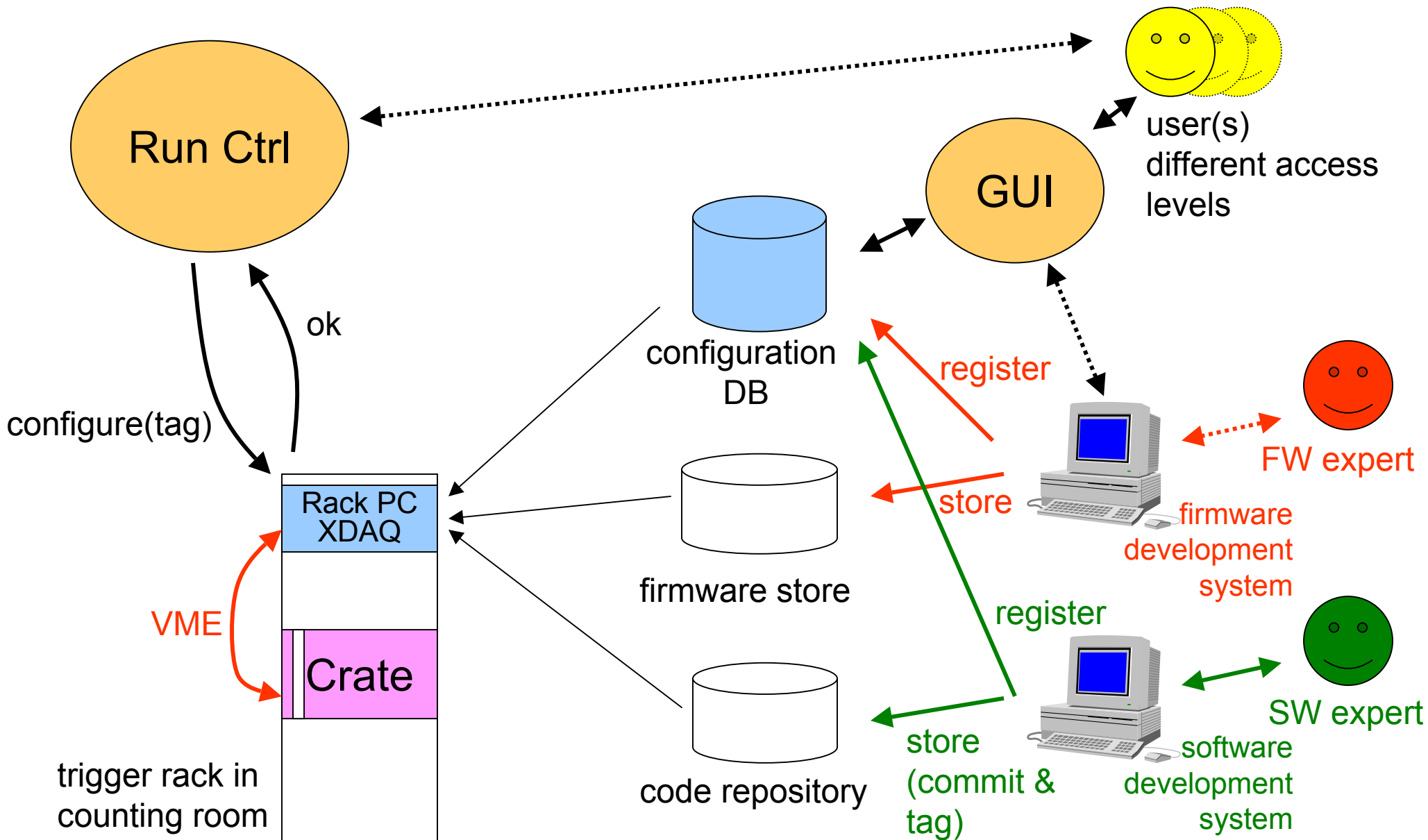


“maintenance” tasks produce firmware for new structure of L1 algorithms

Environment: Run Ctrl, XDAQ, DB



Environment: Run Ctrl, XDAQ, DB



- Receive configure command from Run Control
 - ⇒ includes indication (tag) of version to be used
- 1) Firmware configuration
 - 1) Get firmware revisions from Configuration DB
 - 2) Check if firmware revisions are already loaded (firmware ID register)
 - 3) If correct firmware version is not loaded ...
 - 1) get the firmware from the file server and download (may use local cached copy) OR
 - 2) Issue error (if online change of firmware is not foreseen)
- 2) Register configuration
 - 1) Get register values from Configuration DB
 - 2) Download to hardware
- 3) Register / LUT configuration based on parameters (optional)
 - 1) Get required conversion code revision from Configuration DB
 - 2) Check if installed and loaded
 - 3) If not, get conversion code from Code Repository, compile, link
 - 4) Get Parameters from configuration DB, convert, download to hardware

- Versioning inside Configuration
 - ⇒ Does every small change in one system require a new global tag?
- Transient Configuration / Changes to configuration
 - ⇒ e.g. masking of a hot channel
 - ⇒ e.g. during calibration runs: change of trigger pre-scaling factors
 - ⇒ may occur frequently
 - ⇒ do not want to create new global configuration
 - ⇒ but need to log the change in configuration
 - ⇒ how are the changes treated ?
 - ⇒ how are they communicated to hardware ?
- Firmware revisions
 - ⇒ there must be a mechanism to uniquely identify the firmware that is currently loaded into an FPGA (avoids unnecessary re-configurations)
- Default parameters in firmware
 - ⇒ Should the online SW know the default parameters stored in the firmware ?
 - ⇒ Could then skip writing of values equal to the default

- On what basis do we include data in the DB / reference it ?
 - ⇒ size ?
 - ⇒ binary / readable
 - ⇒ used only in hardware or also by HLT?

- Storing of crate configuration
 - ⇒ what board (unique ID) is in what slot ?
 - ⇒ linked with equipment management DB

- Interaction: Run Control – GUI – Configuration DB
 - ⇒ how do GUI for configuration DB and Run Control communicate?

- When / How can we start working with the Configuration DB
 - ⇒ is there a system available for tests

- Conversion software to convert configuration parameters to LUT contents or to registers has been developed
 - ⇒ GT: Standalone GTS converter
 - ⇒ GMT: LUT & Register C++ Classes (for use in ORCA / standalone/ in online SW)

- Configuration parameters are currently stored in files
 - ⇒ Need to move to database
 - ⇒ Need to access same data in ORCA

- Online SW is being developed (XDAQ)

- Need to develop GUI for Configuration DB
 - ⇒ modify configuration data
 - ⇒ select configuration

- Estimated available manpower for Online SW / Configuration DB (FTE)
 - ⇒ DT: 2.5, GMT: 0.5, GT: 2.5