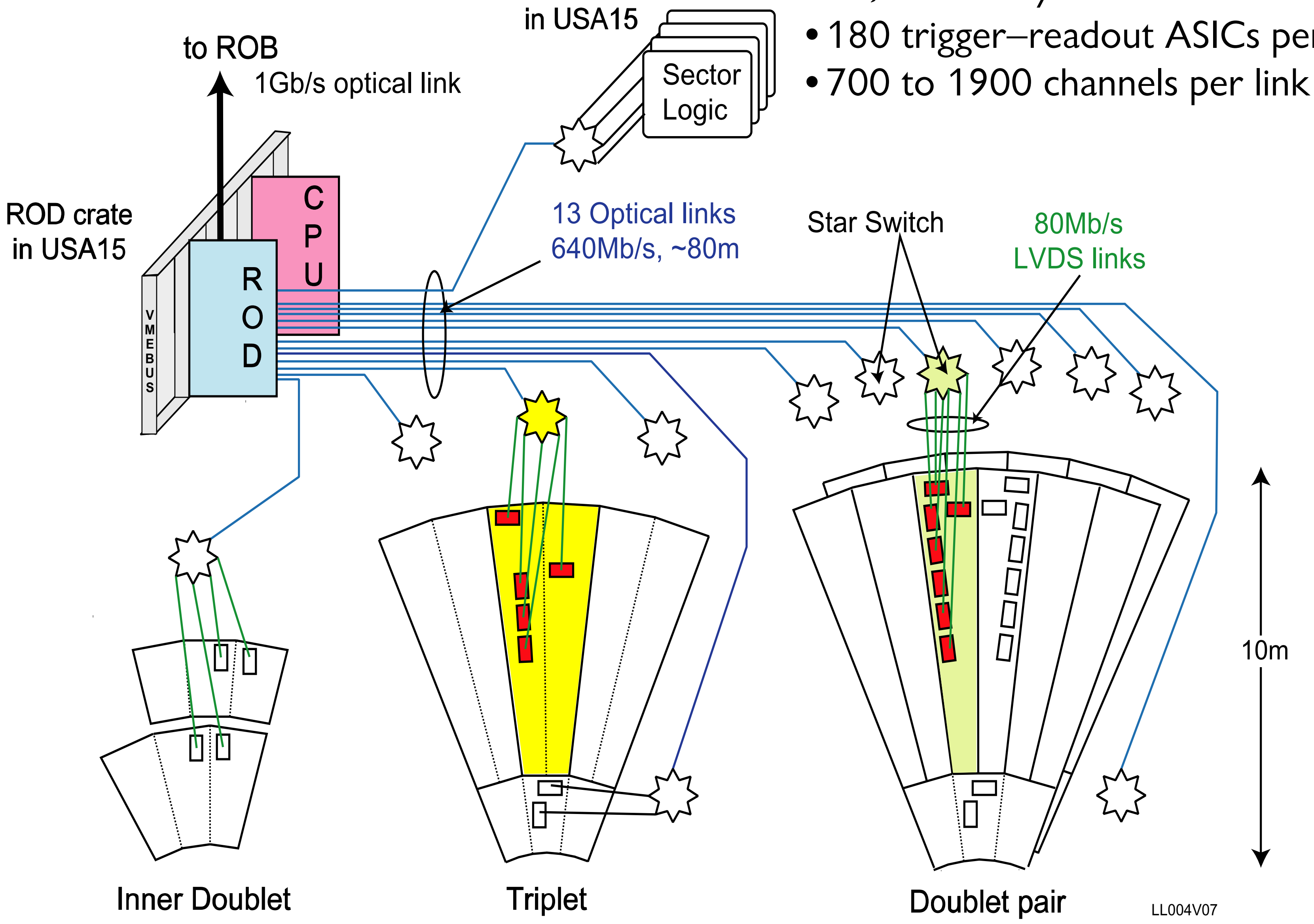
The Readout Driver for the ATLAS Muon Endcap Trigger and its architecture and design language techniques

Daniel Lellouch, Lorne Levinson, Alex Roich
Weizmann Institute of Science
Rehovot, Israel

Muon Endcap trigger chambers (TGC) readout for 1 of 16 octants

- 20,000 binary channels on 12 FE links
- 180 trigger–readout ASICs per octant
- 700 to 1900 channels per link



Input data characteristics

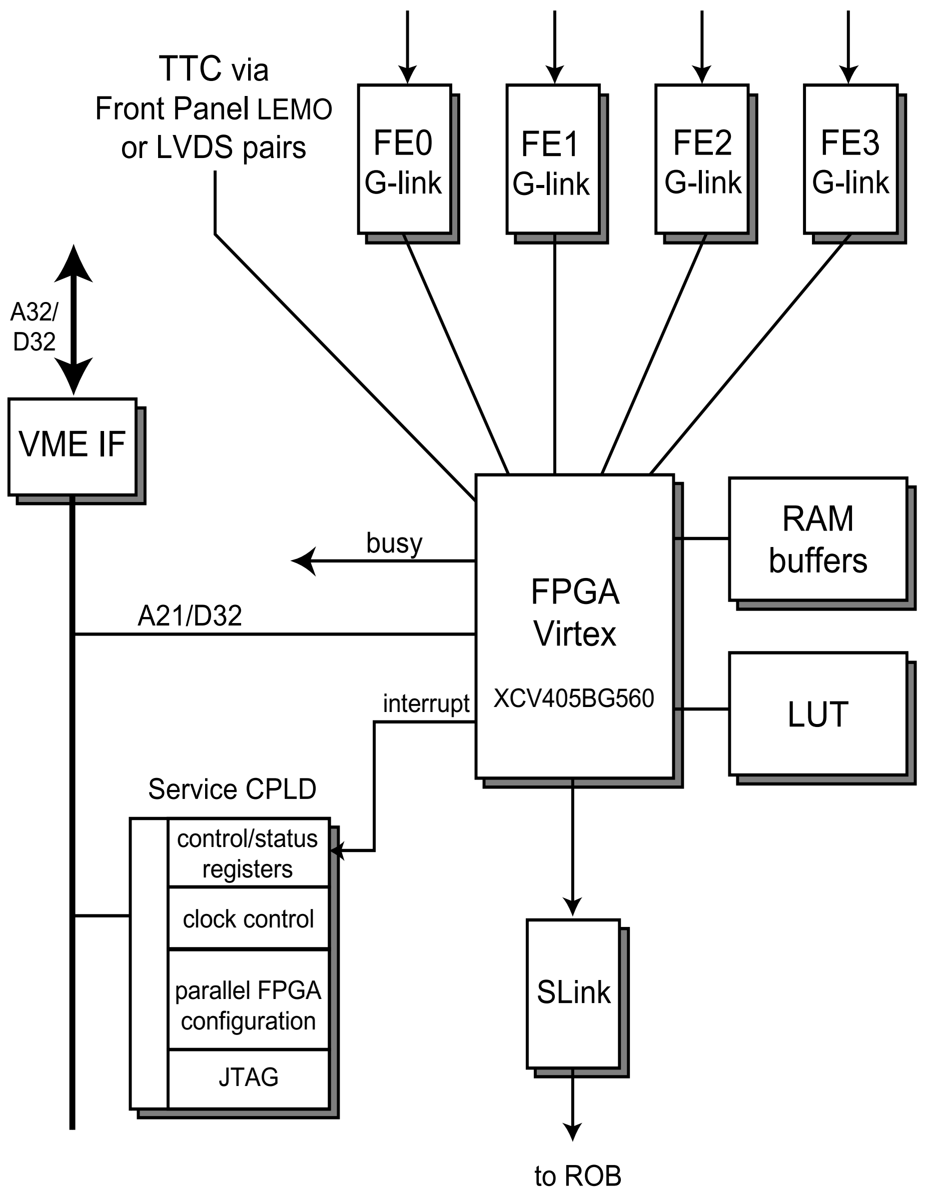
- 100 KHz L1A rate on 13 links → many small messages
- Data dominated by cavern background, but still low occupancy:
  - ~10 uncorrelated hits per octant per L1A
  - ~1.5 charged particles per octant per L1A giving correlated tracklets
  - huge uncertainty in simulation of cavern background
    → safety factor of 10 needed
  - input data is partially zero-suppressed so dominated by headers

  → headers needed to confirm synchronization of all trigger-readout ASICs
- Zero-suppression → variable length records
- Data includes output from trigger logic stages:
  on-chamber coincidence ASICs, HipT ASICs and Sector Logic
- Estimate input bandwidth of 22MB/sec per octant, without safety factor
- Tracklets (2-out-3, 3-out-of-4 coincidences from correlated background and real muons!) are needed to monitor the coincidence logic

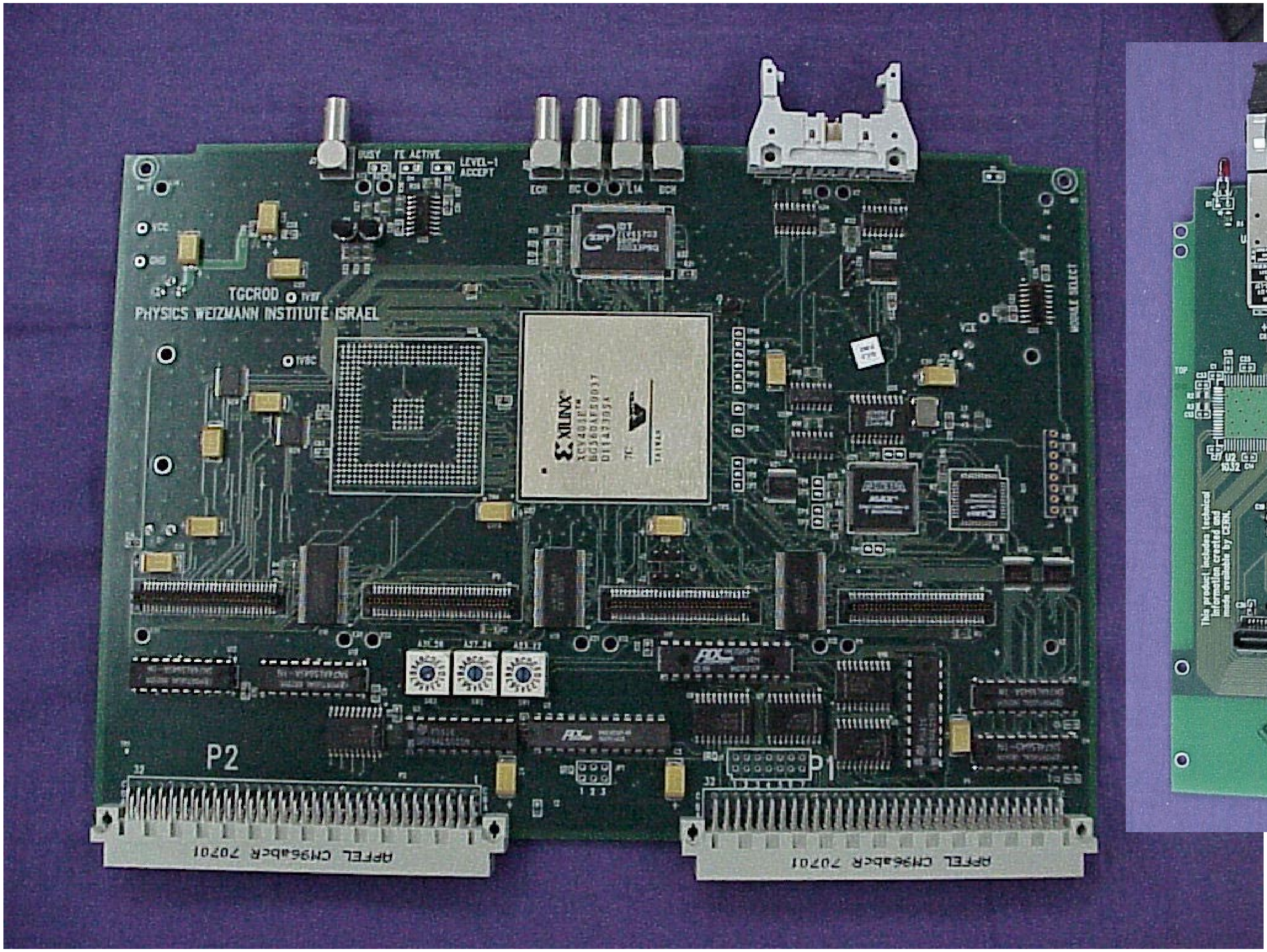
ROD – required functions

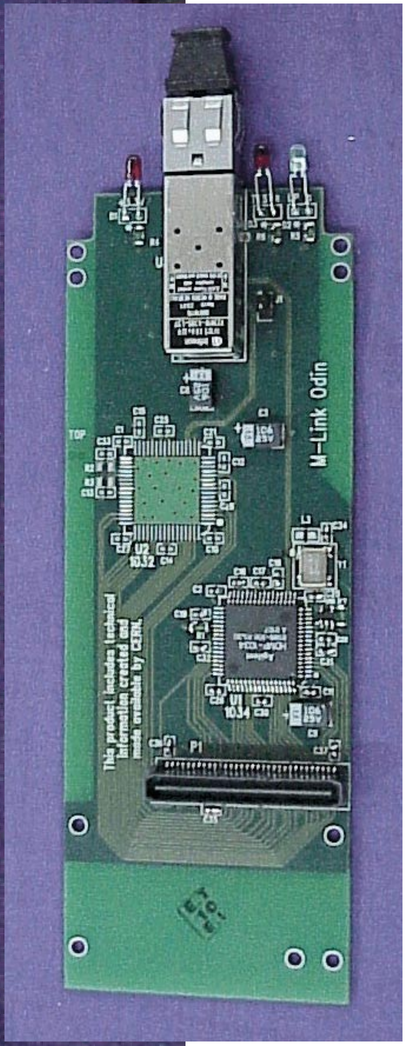
- Receive TTC information and queue expected event L1A IDs
- Collect event fragments from several Front End links corresponding to each of the L1A Event IDs
- Detect and recover from input link errors and data errors
- Assert RODBUSY to the ATLAS CTP module when necessary, but as infrequently as possible
- Extract hit and trigger data from zero-suppressed bit map
- Provide sampled full events, hits and tracklets to the ROD Crate Processor for monitoring the system
- Format events into ATLAS standard ROB format
- Send the data to the ROB and/or Rod Crate Processor
  - Respond to flow control signals from the ROB
- Requirements for calibration and monitoring

TGC ROD prototype block diagram



- Xilinx extra memory Virtex FPGA, 405EM
- VME 6U
- 3 programmable clocks
  – any frequency 1–100MHz using PLL
- 4 daughter boards for FE links
  - G-link + opto-transceiver
  - S-link compatible connector and mechanics
  - Optional TX chip on the board
- 1 S-Link connector for ROB connection
- LVDS or NIM inputs for TTC signals
- General enough architecture to use also as source of simulated data for testing
- 1MB ZBT SRAM
  - not yet needed or used (yet) for ROD
  - used by xmitter for fake event store
- 3.3V, 1.8V power generated on the board from 5V
- Service PLD


TGCROD
PHYSICS WEIZMANN INSTITUTE ISRAEL
BUSY
FE ACTIVE
LEVEL-1 ACCEPT
XILINX
P2
P1

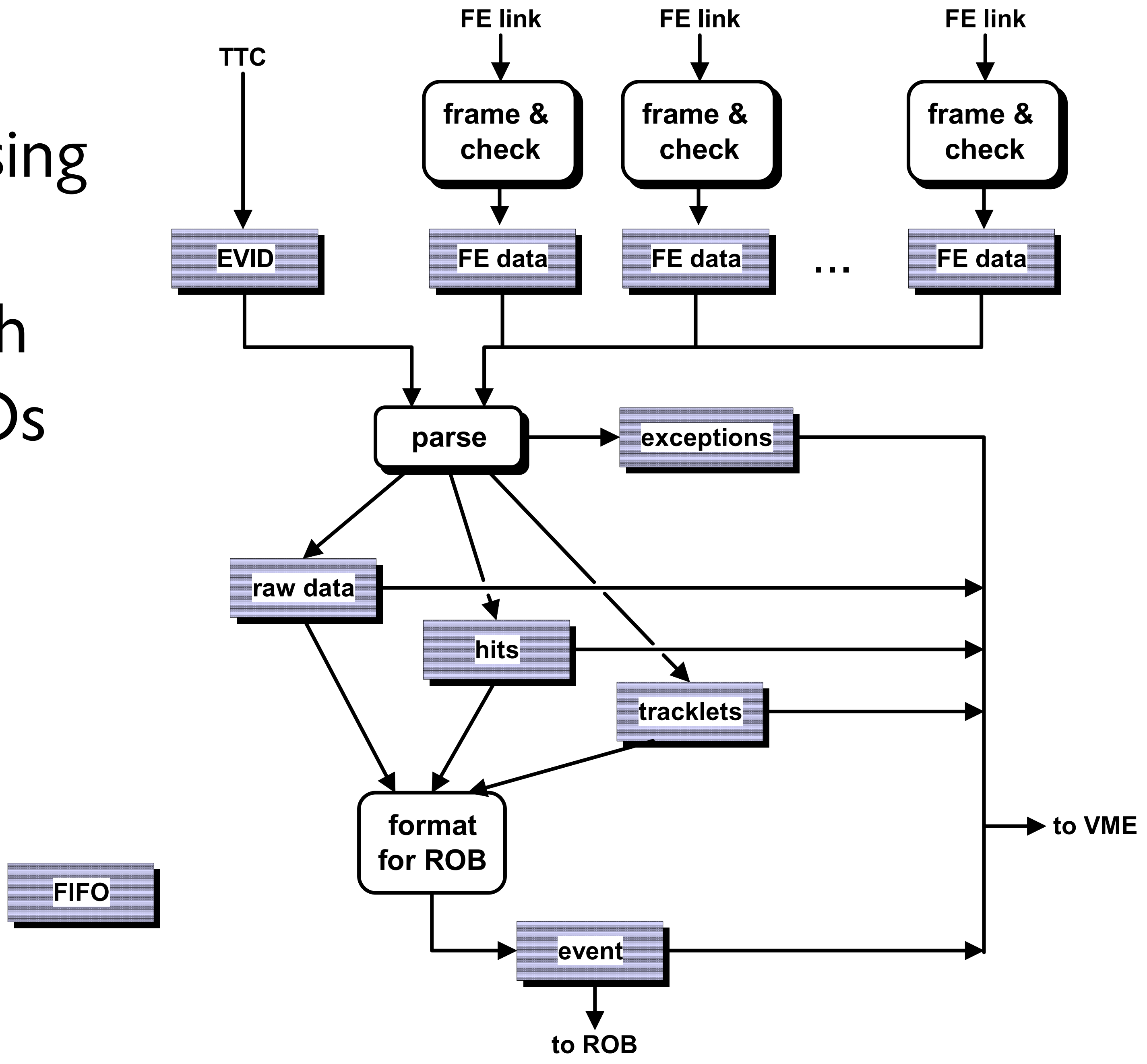

M-Link Odin
This product includes technical information created and made available by CERN.

Main relevant features of Virtex FPGA

- XCV405EM: ‘extra memory’, today: 'moderate size'
- 10800 logic cells, i.e. flip-flops
- Large amount of embedded, “block”, dual port RAM
  - 70KB (140 × ½KB width-configurable RAM blocks)
- 4 global clocks and 8 DLLs
- Low skew lines for clocks from each G-link deserializer
- Logic Cells can also be used as dual port memory (16x1 each)
- Abundant routing resources allow wide (16 and 32 bit) data paths

TGC ROD data processing (simplified) in FPGA with internal FIFOs

Pipeline of processes connected by FIFOs



Architectural elements

Design implemented using:

- graphic tools that produce VHDL
- VHDL
- Xilinx cores for FIFOs
- Handel-C with VHDL output

Synthesized together by Leonardo

Handel-C – overview

- C syntax
  - Variables and arrays are registers, or memory (internal or external)
- Additions from Occam language for Transputers:
  - par/seq indicate blocks of statements to be executed in parallel or in seq
  - channels for thread-to-thread synchronization via passing messages
    - thread of first chan I/O is suspended until second thread executes chan O/I, then xfer takes place (in one clock) and both threads continue
- Easy to make large number of variable latency parallel threads & synchronize them
- timing model: Every assignment takes one and only one clock
- advantages
  - hi level. can express procedures easily, forces synchronous design
  - rich macros help create multiple instances of HW
    - recursive macros enable building repetitive structures
- still not for linear sequential software programmers
  - need to understand the specific FPGA architecture, HW concurrency, how structures and algorithms synthesize, timing constraints
- produces VHDL or EDIF, can interface to VHDL and Xilinx cores

Embedded FIFOs

- Allow pipeline with stages with variable latency
  - Each pipeline stage has fluctuations in its execution time and amount of output
  - Unlike a registered pipeline where each stage is one clock
- FIFOs bridge different clock domains:
  - One domain per Front End link: deserializer output clock. Defined: 40MHz
  - Main design clock: rate is a parameter: depends on design's longest path
  - S-Link output: defined maximum of 32MHz
  - TTC: 40MHz
  - VME: undefined and asynchronous
  - All inside one FPGA means FIFOs must be embedded in FPGA
- Different FE links can have different input FIFO depths, depending on the occupancy of the chambers read out
- All internal FIFO occupancies readable via VMEbus for monitoring
  - resolution of one item

Variable length data records in FIFOs

- Zero-suppression means variable length records
- Better design if data readers know in advance how much data they will read per fragment
- Use a 2nd control FIFO
  - Write N data words to the data FIFO
  - Write “N” and any flags to the control FIFO
- Reader waits for available item in the control FIFO
- (almost) all FIFOs in the ROD design are data/control FIFO pairs

Thread-to-thread pipes

- Thread-to-thread communication
  - Extension of channels to depth of more than one item
- All FIFOs in the ROD design implemented by a "pipe" object:
  - Can read and write on every clock
  - Read-on-empty and full-on-write conditions block (stall) the invoking thread until not-empty or not-full.
  - Writes intended to execute on the same clock as the pipe xfer are guaranteed to do so:
    - Stall **after** write if FIFO now full
    - E.g. **`par {pipe_write(hitpipe, hitid); hitid++;}`**
      - (example of instantiating a counter)
  - Can test **`empty`**, **`full`** and **`count`** anytime
  - Can generate Service Call to host CPU on **`~empty`**, **`almost-full`**
  - Could histogram count for occupancy monitor

Multiple "execution units" – fragment farm

- Data volume varies from link to link and from event to event
- 13 links but need only ~4 fragment processors (FP) for required throughput
- Can instantiate *n*<13 fragment processors for 13 links
- Each fragment processor is a separate thread
  - implemented in Handel-C by '*array of functions*'
- Dispatcher thread allocates each fragment to a fragment processor on the-fly
  - Use one channel per FP to receive link ID of fragment to process
  - Use another channel to acknowledge completion/ready-for-more
  - One output FIFO per fragment processor
- Another thread receives the acknowledges and updates list of free FPs
- Reduces silicon resources: number of FIFOs and logic cells
- Requires multiplexing sources to FPs, but

  reduces multiplexing from FPs to output FIFO

Service calls – SVCs

- Send a signal from FPGA HW thread to a SW process on host CPU
  - Different signal IDs for different HW thread – SW process pairs
- Exceptional conditions requiring host CPU intervention
  - e.g. FIFO almost full or unrecoverable loss of event synchronization
- Allows any number of outstanding SVCs, but only one of each type
- Scenario
  - Polling loop over all conditions posts SVCID to VME register
    - Use **`chan`** to serialize posting
  - Interrupt host CPU
  - host interrupt handler reads SVCID and acks by clearing SVCID reg
  - host interrupt handler sends signal to process subscribed to this SVCID
    - host interrupt handler is now ready for another SVC
  - Process performs service, e.g. empties FIFO
  - Process writes “done” acknowledge for this SVCID to SVCACK reg
    - This SVCID can now be issued again

FPGA-to-host CPU pipes

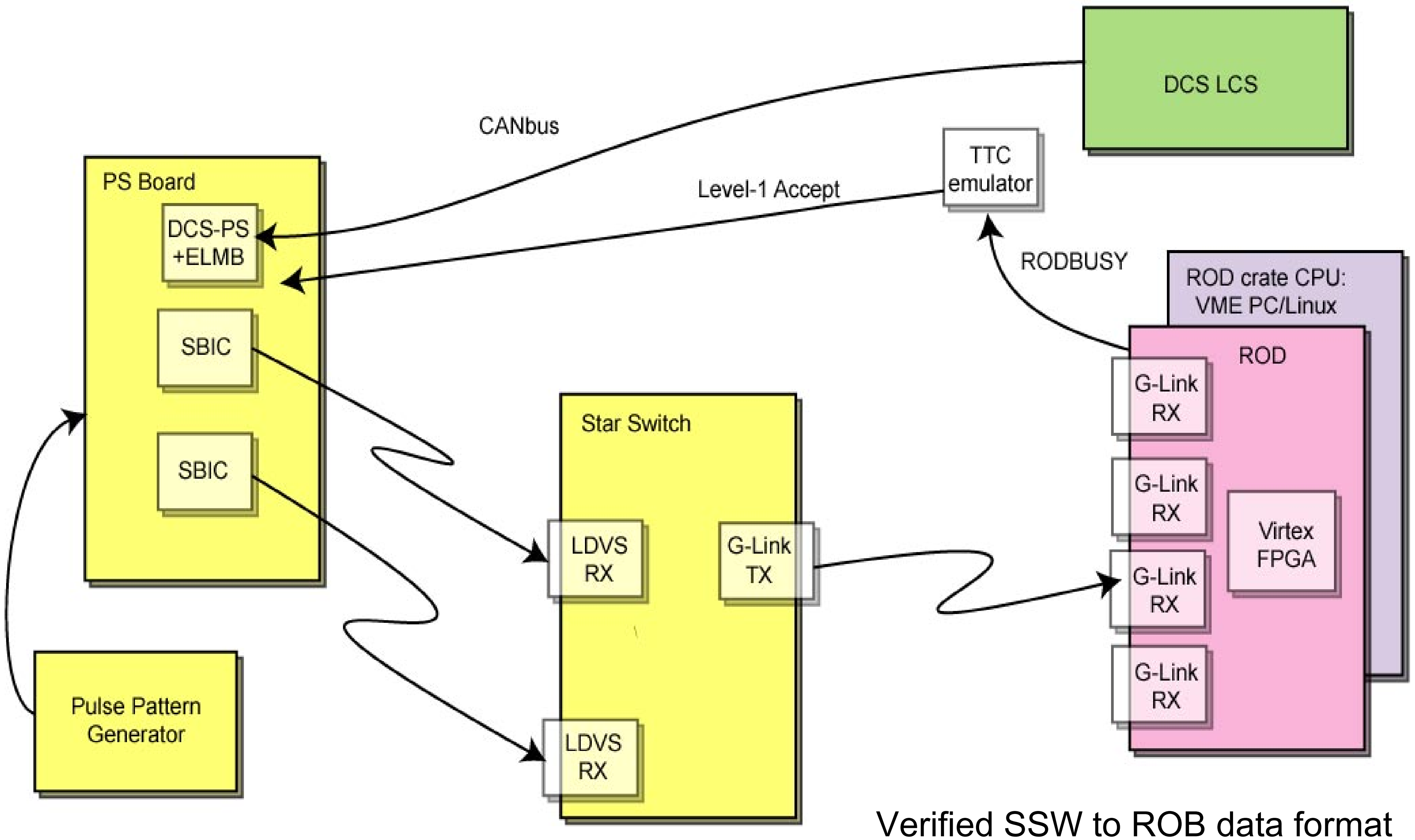
- Some pipes have their read port on the VMEbus
  - Sampled full event, hit, tracklet and message pipes
  - Host interrupted by Service Call on ~empty or almost-full
    - SVCID indicates which pipe needs service
- Exceptions, messages and debug info
  - Message pipe is written with severity/exception code and 3 bytes of data
  - Each exception is counted, logged & perhaps serviced by the host SW
  - Processes can dump values for debugging
  - Writes to this pipe can be put anywhere in a thread
    - soon: add arbitration to serialize writes from different threads

Handel-C – advantages

- More advantageous when:
  - Processing path depends on data content
  - Design needs to be frequently adapted to demand for functionality or changing requirements
  - Ultimate performance not paramount
  - Complex arithmetic or logical algorithms
  - Many special cases and possible exceptional conditions or errors
- Less advantageous when:
  - Pipelines with fixed length of data → deterministic latency
  - Pipeline stages are of fixed length, e.g. single clock
  - Few, or only simple, sequential data dependencies
    - i.e. mostly, operations are done in parallel

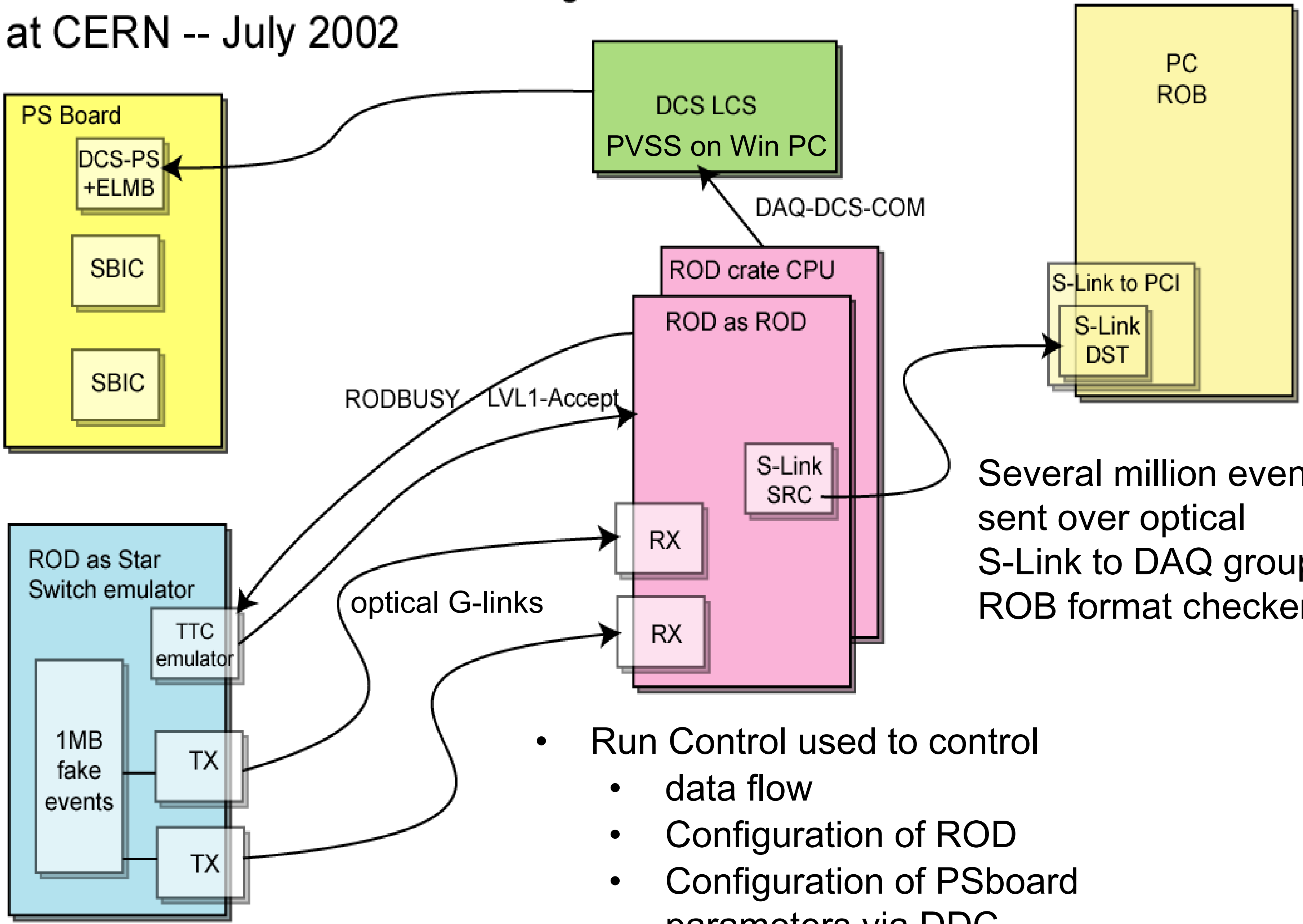
Integration and performance

ROD in the TGC Slice Test in Japan -- Nov 2001



Verified SSW to ROB data format and basic ROD functionality

TGC ROD -- DCS -- ROB integration test at CERN -- July 2002



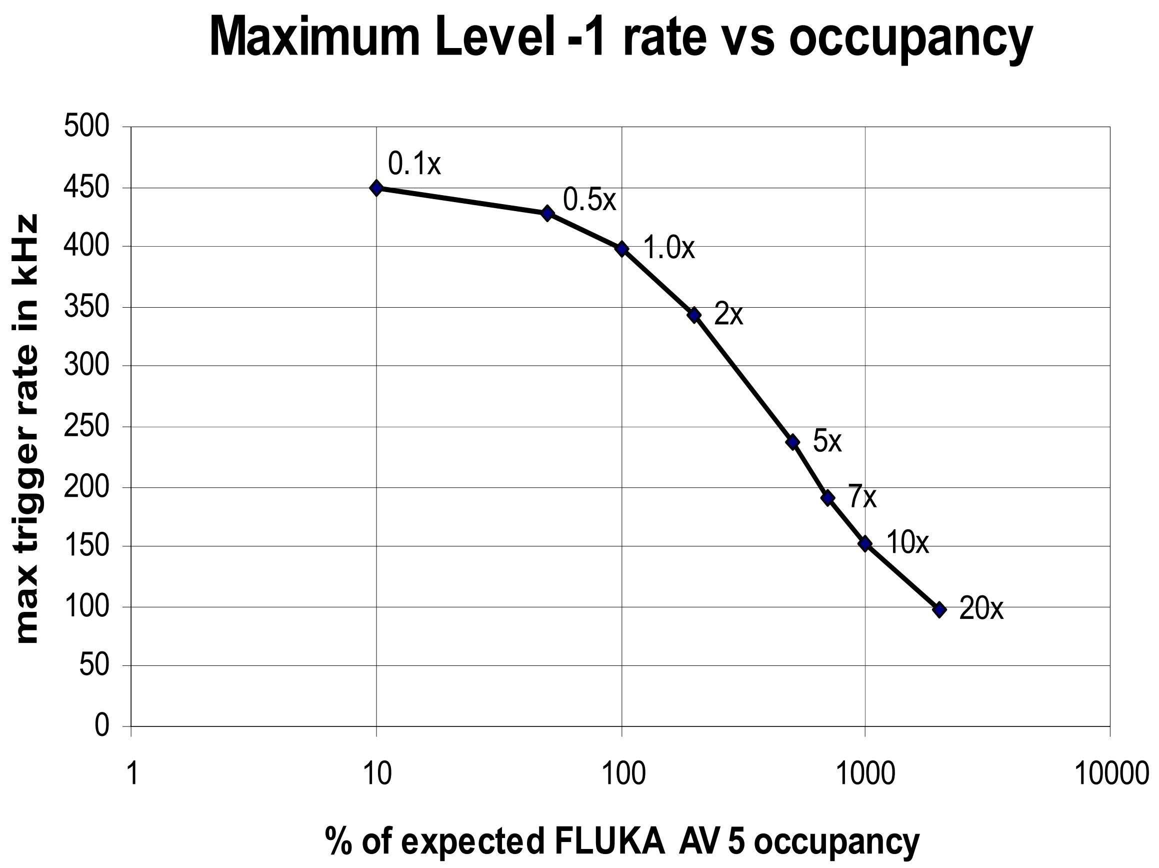
Several million events sent over optical S-Link to DAQ group ROB format checker

- Run Control used to control
  - data flow
  - Configuration of ROD
  - Configuration of PSboard parameters via DDC

Online Software

- Pentium SBC running Linux in VME crate (Universe chip)
  - ATLAS VME driver
- ATLAS online system packages implemented
  - Run Control and Process Manager
  - Error message utility
  - Information Service: dataflow statistics error counts
  - IGUI
  - DAQ to DCS communication
    - configuration of FE electronics parameters and thresholds at Start-of-Run
  - simple use of configDB
- full readout path using KLOE buffer manager to event format checker and smart dump

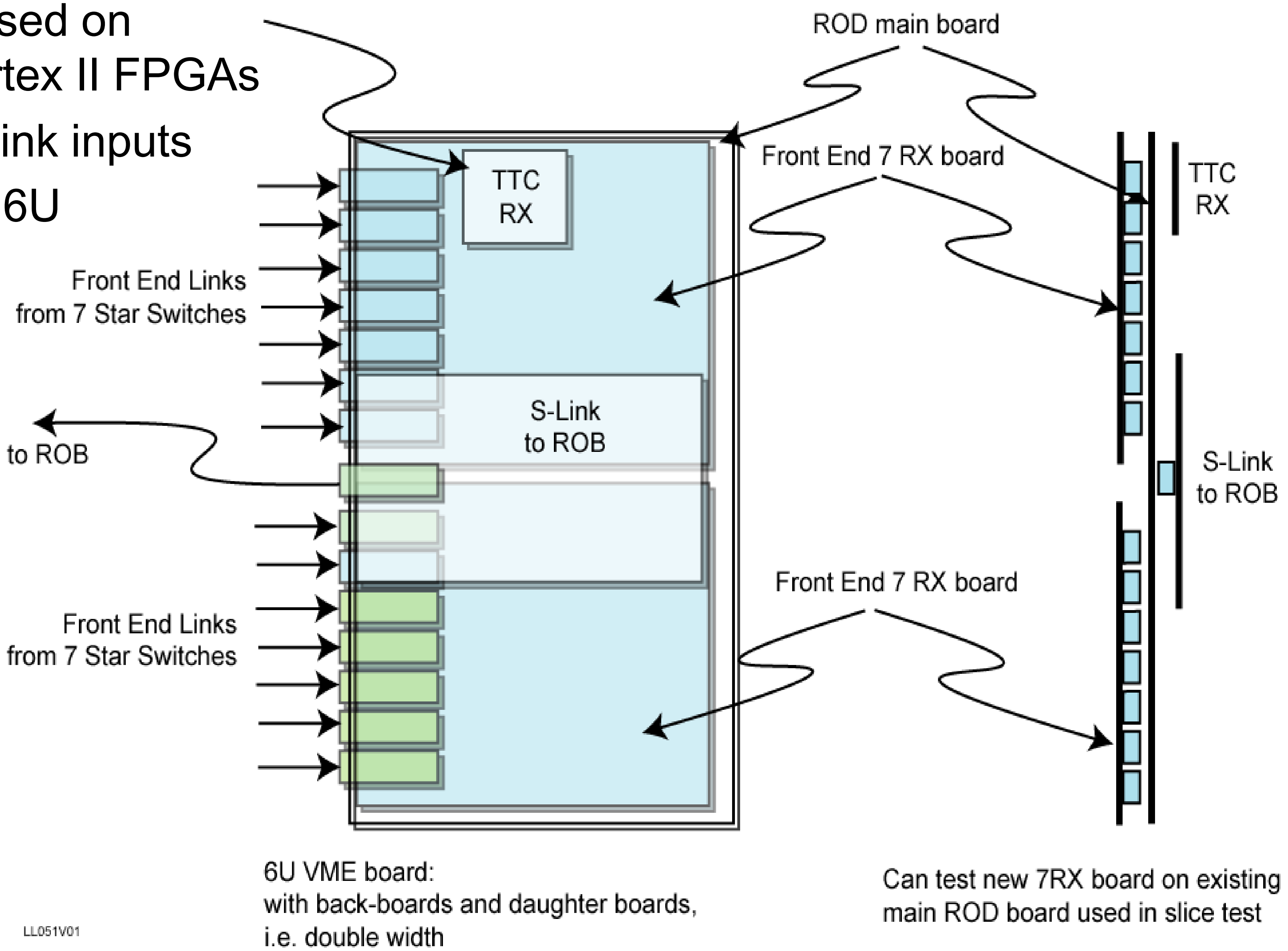
Trigger rate performance



- Used fake events from a realistic simulation using average occupancies from FLUKA simulation 'AV5' (by I. Dawson)
- uncorrelated (neutral) & correlated (charged) background
- 2 Front End links processed in series on 1 fragment processor
  - both strips & wires

Final prototype ROD

- To be based on Xilinx Virtex II FPGAs
- 2×7 FE Link inputs
- VME 64 6U



6U VME board:
with back-boards and daughter boards,
i.e. double width

Can test new 7RX board on existing main ROD board used in slice test

The end

Bon appetit!