Introduction to the Electronics, Trigger, DAQ and Online Design for the SuperB Detector

LNF, January 2012

S. Luitz for the SuperB ETD/Online Team
Projected Trigger Rates and Event Sizes

• Estimates extrapolated from BaBar for a detector with BaBar-like acceptance and a BaBar-like open trigger
  – Note: at $1 \times 10^{36}$ 1 nb accepted cross section = 1 kHz in rate

• Level-1 trigger rates (conservative scaling from BaBar)
  – At $10^{36}$: 50kHz Bhabhas, 25kHz beam backgrounds, 25kHz “irreducible” (physics + backgrounds)
  – $\rightarrow$ 100kHz Level-1-accept rate (without Bhabha veto)
    • 75kHz with a Bhabha veto at Level-1 rejecting 50%
    • Safe Bhabha veto at Level-1 difficult due to temporal overlap in slow detectors. Baseline: better done in High-Level Trigger
  – 50% headroom desirable (from BaBar experience) for efficient operation
  – $\rightarrow$ baseline: 150kHz Level-1-accept rate capability – headroom sufficient?

• Event size: 75–100kByte (estimated from BaBar)
  • Pre-ROM sizes reasonably well understood (400–500kByte)
  • Still some uncertainties for post-ROM event size

• High-Level Trigger and Logging
  – Expect to be able to achieve 25nb logging cross section with a safe real-time high-level trigger $\rightarrow$ 25kHz x 75kByte = 1.8 Gbyte/s logging rate
    • Will very likely change TDR baseline event size to 100kByte
  – Logging cross section could be reduced by maybe 5–10nb by using a more aggressive filter in the HLT (based on BaBar experience there is a cost vs. risk tradeoff!)
System Design Principles

• Apply lessons learned from BaBar and the LHC experiments
• Keep it simple!
  – Synchronous, fixed-latency design
  – No “untriggered” readouts
    • Except for trigger data streams from FEE to trigger processors
  – Use off-the-shelf components where applicable
  – Links, networks, computers, other components
  – Software (what can we reuse from other experiments? What can we share?)
• Modularize the design across the system
  – Common building blocks and modules for common functions
  – Implement subdetector-specific functions on specific modules
  – Carriers, daughter boards, mezzanines
  – Some ideas to provide a “clean” upgradeable interface to the data links
• Design with radiation-hardness in mind (where necessary)
• Design for high-efficiency and high-reliability “factory mode”
  – Where affordable. BaBar experience will help with the tradeoffs
  – Design for minimal intrinsic dead time (current goal: 1% + trickle injection blanking)
  – Automate. Minimize manual intervention. Minimize physical hardware access requirements.
Synchronous, Pipelined, Fixed-Latency

- Front-end electronics, Fast Control and Timing System and Trigger synchronized to global clock
- Analog signals sampled with global clock (or multiples or integer fractions of clock). Samples shifted into latency buffer (fixed depth pipeline)
- Synchronous reduced-data streams derived from some sub-detectors (DCH, EMC, ...) and sent to the Level-1 trigger processors
- Pipelined Level-1 trigger generates a trigger decision after a fixed latency referenced to global clock
- In case of an L1-accept the readout command is sent to the FCTS and broadcast to the Front-end electronics (over synchronous, fixed-latency links)
- Front-end electronics transfer data from the corresponding readout window to derandomizer buffers
- Data from derandomizer buffers are sent over optical links (no fixed latency requirement here) to the Readout Modules (ROMs)
- Every ROM applies zero suppression / feature extraction (where applicable) and combines event fragments from all its links.
- Partially event-built fragments are then sent via the network event builder into the HLT farm
SuperB ETD System Diagram

Drift Chamber
~ 35

L1 processor...

Ethernet

EMC
~ 80

L1 processor...

L1 processor...

SVT

~ 250 Optical links
~ 50 m

~35

~ 80

~ 400

~ 400

~ 15

~35

~ 80

~ 400

~ 15

Radiation wall

Trigger primitives

Event data

Detector

Global Level1 Trigger (GLT)

Raw L1

CLK, L1, Sync Cmds

Detector Safety System

Detector

ROMs

Event fragments

PCs Farm

Full Events

Crate Control

L1 Buffer Ctrl

Rx

L1 processor...

FEE models

throttling

Field Bus

Throttle

Clk, L1, Sync Cmds

Throttle

Clk, L1, Sync Cmds

Ethernet

ECS interface

Subdetector Specific Electronics

Pre-selection

FCTS interface

ECS interface

FE Boards

Subdetector Specific Electronics

L1 Buffer Ctrl

Tx

L3 to L5

~ 400

~ 400

~15

~ 80

~ 35

S. Luitz - Intro to SuperB ETD/Online
1% Dead Time Goal

- **Target:** \( \approx 1\% \) event loss due to DAQ system dead time
  - Not including potential trigger blanking for trickle injection
- **Assume “continuous beams”**
  - 2.1ns between bunch crossings
  - No point in hard synchronization of L1 with RF
- **1% event loss at 150kHz requires**
  66ns maximum per-event dead time
  - Assuming exponential distribution of event inter-arrival time
- **Need sufficient de-randomizer depth**
- **Challenging demands on**
  - Intrinsic detector dead time and time constants
  - L1 trigger event separation
  - Command distribution, command length (1 Gbit/s)
- **Ambitious – may need to relax this**
Level-1 Trigger

- Fully pipelined
- Input running at 7(or 14?) MHz
  - Continuous reduced-data streams from sub-detectors over fixed latency links
    - DCH hit patterns (1 bit / wire / sample)
    - EMC crystal sums, appropriately encoded
- Total latency goal: 6 ns
  - Includes detectors, trigger readout, FCTS, propagation
  - Leaves 3-4us for the trigger logic
- Trigger jitter goal: <=50-100ns to accommodate short sub-detector readout windows

Baseline: “BaBar-like L1 Trigger” with some improvements
- Calorimeter trigger
  - Cluster counts and energy thresholds
- Drift chamber trigger
  - Track counts, p_T, z-origin of tracks
  - Highly efficient, orthogonal
  - To be validated for high-lumi environment
  - Challenges: time resolution, trigger jitter and pile-up
- To be studied
  - SVT in trigger (# tracks, # tracks from/not from IP, # of back-to-back tracks in Phi)
    - Tight interaction with SVT and SVT FEE design
  - Bhabha veto
    - Baseline: Best done in HLT
Fast Control and Timing System (FCTS)

- Clock distribution
  - Synchronization with accelerator RF & revolution fiducial
- System synchronization
- Command distribution
  - L1-Accept
- Receive trigger decisions from L1
- Participate in pile-up and overlapping event handling
- Dead time management
  - Fast throttle emulates front-ends in Fast Control and Timing Master (FCTM)
  - Slow throttle via feedback from ROMs and maybe HLT (use GigE?)
  - System partitioning
    - 1 partition / subdetector
- Event management
  - Determine event destination in event builder / high level trigger farm

- Links carrying trigger data, clocks and commands need to be synchronous & fixed latency: \( \approx 1 \text{ GBit/s} \)
- Readout data links can be asynchronous, variable latency and even packetized: \( \approx 2 \text{ (?) GBit/s} \) but will hopefully improve
Common Front-End Electronics

- Digitize
- Maintain latency buffer
- Maintain derandomizer buffers, output mux and data link transmitter
- Generate reduced-data streams for L1 trigger
- Interface to FCTS
  - receive clock
  - receive commands
- Interface to ECS
  - Configure
  - Calibrate
  - Spy
  - Test
  - etc.

- Provide standardized building blocks to all sub-detectors, such as:
  - Schematics and FPGA “IP”
  - Daughter boards
  - Interface & protocol descriptions
  - Recommendations
  - Performance specifications
  - Software

**Diagram:**
- Trigger primitives to L1 processors
- Data from sub-detector
- Optical links ~ 50 m
- Event fragments to ROM
- FE Boards
- Subdetector Specific Electronics
- L1 Buffer Ctrl
- Tx

**Flowchart:**
- FCTS interface
- ECS interface
- Subdetector Specific Electronics
- Pre-selection

**Artifacts:**
- FE Electronics
- S. Luitz - Intro to SuperB ETD/Online
Readout Modules (ROMs)

- Receive data from the sub-detectors over optical links
- Reconstitute linked/pointer events
- Process data (FEX, data reduction) – optional
  - Data reduction might not be possible since raw data might be needed in the high level trigger (HLT)
- Send event fragments into HLT farm (network)

We would like to use off-the-shelf commodity hardware as much as possible.

R&D to use off-the-shelf computers with PCI-Express cards for the optical link interfaces in progress

Alternative: FPGA-based Optical-Link → 10Gig Ethernet “dumb protocol converter”
Event Builder and Network

• Combines event fragments from ROMs into complete events in the HLT farm
  – In principle a solved problem 😊 Here my preferred solution:
  – Prefer the fragment routing to be determined by FCTS
  – FCTS decides to which HLT node all fragments of a given events are sent (enforces global synchronization), distribute as node number via FCTS
    • Event-to-event decisions taken by FCTS firmware (using table of node numbers)
    • Node availability / capacity communicated to FCTS via a slow feedback protocol (over network in software)
  – Choice of network technology
    • 10Gbit/s Ethernet prime candidate
    • UDP vs. TCP … a long contentious issue
      – Pros and cons to both
      – What about RDMA?
      – What about IPv6? Some interesting features (e.g. NDP, saner multicast)
    • Can we use DCB/Converged Ethernet for layer-2 end-to-end flow control in the EB network?
  – Can SuperB re-use some other experiment’s event builder?
    – Interaction with protocol choices
High-Level Trigger Farm and Logging

• Standard off-the-shelf rack-mount servers
• Receivers in the network event builder
  – Receive event fragments from ROMs, build complete events
• HLT trigger (Level-3)
  – Fast tracking (using L1 info as seeds), fast clustering
  – 10ms/event (baseline assumption, 5-10x what the BaBar L3 needed on 2005-vintage CPUs), plenty of headroom
  – → 1500 cores needed on contemporary hardware (~150 servers 16 cores, 10 usable for HLT purposes)
  – If we have to do extensive pulse shape fitting and decomposition of overlaid events, more CPU will be needed …
  – … still … probably a “non-problem” in 2016 … 30 servers? 😊
• Data logging & buffering
  – Few TByte/node
  – Local disk (e.g. RAID1 as in BaBar)? – or –
  – Storage servers accessed via a back-end network?
  – Probably 2 days’ worth of local storage (2TByte/node)? Depends on SLD/SLA for data archive facility.
  – No file aggregation into “runs” → bookkeeping
  – Back-end network to archive facility (“Tier 0”)
    • Might be a distributed / virtual facility or located at LNF or Cabibbolab
Experiment Control System (ECS)

- Configure the system
  - Upload configuration into FEE
  - Should be fast!
- Monitor the system
  - Spy on event data
  - Monitor power supply, temperatures, etc.
- Test the system
  - Using software specifically written for the FEE
  - We do not foresee ECS-less self-test capabilities for the front-end electronics
- Proposed implementation
  - SPECS (Serial Protocol for Experiment Control System)
  - Bidirectional 10MBit/s bus designed for LHCb
Data Quality Monitoring,

• Data Quality Monitoring
  – Same concepts as in BaBar:
  – Collect histograms from HLT
  – Collect data from ETD monitoring
  – Run fast and/or full reconstruction on sub-sample of events and collect histograms
    • May include specialized reconstruction for e.g. beam spot position monitoring
  – Could run on same machines as HLT processes (in virtual machines?) or a separate small farm (“event server clients”)
  – Present to operator via GUI
  – Automated histogram comparison with reference histograms and alerting
Control Systems

• Run Control
  – Coherent management of the ETD and Online systems
    • User interface, managing system-wide configuration, reporting, error handling, start and stop data taking
• Detector Control / Slow Control
  – Monitor and control detector and detector environment
  – Interface to accelerator controls
• Maximize automation across these systems
  – Goal: 2-person shifts like in BaBar
  – “auto-pilot” mode where detector operations is controlled by the machine
  – Automatic error detection and recovery where possible
• Assume we can benefit from systems developed for the LHC, the SuperB accelerator control system and commercial systems
• Integration with accelerator control system extremely important
  – Automation of data taking operations (“the machine controls the detector and DAQ”)
  – BaBar/PEP-II experience: mutual access to “diagnostic” data extremely valuable for improving B-Factory performance, e.g.:
    • PEP-II operators routinely watching BaBar radiation & background indicators, beam spot info from BaBar, etc.
    • BaBar correlating background levels to vacuum pressures, beam currents, collimator settings, etc.
  – Needs to be designed into the system from the beginning!
  – Opportunity for joint R&D and a joint design
Machine-Detector Interface
(from a control system view)

- Experience from BaBar + talking to a few PEP-II people
- Common time base (FCTS and controls)
  - Down to single-bunch resolution (detector may be able to deliver per-bunch measurements - integrated over longer windows)
  - Trigger veto for injected bunches for a # of revolutions after injection
- APIs for data sources
  - Need to allow after-the fact injection of time-stamped data (processing of data might take minutes)
  - Multi-platform (Linux + embedded)
- Machine-Detector handshake
  - Factory mode operation (machine drives detector), trickle injection
  - Hard and soft protection systems
- Unified query interface
  - Correlate machine and detector data
- Operational independence
  - Accelerator and detector control systems have different downtimes
- Fault isolation / Security boundaries
- Federated systems?
Ancillary Components and Systems

• Electronic Logbook
  – Web based – integrated with bookkeeping
  – Joint design with accelerator electronic logbook?
    • Somewhat different requirements, though
• Databases
  – Configuration, Conditions, Ambient
• Configuration Management
  – Authoritative source of configuration
  – Log trail of configuration
  – “Provenance light”
• Software Release Management
• Too early for Detector TDR
  – “ETBD” (eventually to be designed)
Open Questions and Areas for R&D

- **Upgrade paths to $4 \times 10^{36}$**
  - What do design upfront, what do upgrade later, what is the cost?
  - How much headroom do we design into the system upfront?
- **Data link details**
  - Jitter, clock recovery, coding patterns, radiation qualification, performance of embedded SERDES
- **ROM**
  - 10GBit/s networking technology, I/O sub-system, using a COTS motherboard as carrier with links on PCIe cards, FEX & processing in software
- **Trigger**
  - Latency, time resolution and jitter, physics performance, details of event handling, time resolution and intrinsic dead time, L1 Bhabha veto, use of SVT in trigger, HLT trigger, safety vs. logging rate
- **ETD performance and dead time**
  - Trigger distribution through FCTS (length of commands), intrinsic dead time, pile-up handling/overlapping events, depth of de-randomizer buffers
- **Event builder**
  - Anything re-usable out there? Network and network protocols, UDP vs. TCP, applicability of emerging standards and protocols (e.g. DCB, Cisco DCE), HLT framework vs. Offline framework (any common grounds?)
- **Software Infrastructure**
  - Sharing with Offline, reliability engineering and tradeoffs, configuration management ("provenance light"), efficient use of multi-core CPUs
- **Control Systems**
  - Joint design of machine and detector control systems