svtsim as a tool to debug SVT

Bill Ashmanskas, trigger ops meeting, 2001-07-11

- svtsim library
  - is a run 2 offline package
  - ANSI C “classes” implemented:
    - emulator for each board through which SVT data flow (HF, XTFA+B, Merger, AMS+AMB, HB, TF)
      - for HF,TF, these are the same emulators used to check out new boards in the test stand
    - “cable” represents data flow
      - intermediate data are accessible
    - “maps” represent configuration data (patterns, etc.)
  - AC++ module (in C++); other C++ code for reading(SIXD+XTRD)/writing(SVTD) CDF event
    - (currently read XFLD; need to switch to XTRD)
  - runs within TRGSim++ program
  - currently only useful to experts
  - currently reads the same flat files used to configure the real boards (soon both will use database)

- current use
  - same svtsim “maps” structures/functions run on crate CPU to calculate memory images
    - thus, configuration files are identical online/offline
  - run on-crate “random test” to check emulators against boards
  - not yet: sample/verify board performance in-situ
  - run cdfSim single muons to check that a proposed set of configuration constants gives sensible results
  - replay raw data through emulator (svtsim)
    - see that distributions are reasonable
      - see attached
    - compare output bit-for-bit
      - just started this a couple of weeks ago
      - led us to find/fix a few configuration bugs
      - has drawn more attention to an event synchronization problem that we’re still tracking down
      - we’re very close to the goal of emulating SVT exactly
  - eventually incorporate sim/data checks into TrigMon

- database issues
  - ~150MB loaded into boards at start of each run
    - it would be stupid to store this verbatim
  - instead, we store patterns (~1MB/wedge), fit constants (will be ~100kB/wedge), etc., compute memory maps dynamically, and check CRC
  - these constants are orthogonal to trigger table (as are SVX pedestals, COT drift model, detector geometry information, XFT segment masks, firmware), do not take the form “cut=value,” and will typically not change from run to run
    - hardware database is the best match for our needs
  - we envision using trigger DB for some sophisticated SVT tuning, which is still many months away:
    - we want to select between 2-3 active sets of constants, depending upon trigger table (e.g. 4/4 vs 4/5 logic)
      - this was tricky to fit into trigger DB schema
    - we want SVT tracking to depend upon L1 trigger bits
  - Bill Badgett has allocated space in the hardware database; starting to use it is currently my #1 task
  - we plan to build SVX channel “kill list” for each run by reading calibration database
  - how to handle beam motion??