GLAST Large Area Telescope

Event Timestamps

Warren Focke
SLAC
I&T Science Verification Analysis and Calibration
Engineering Physicist
focke@slac.stanford.edu
650-926-4713

The Issue

- We had no proper timestamps
- GEM provides several data that can be combined to get low-order bits of time, but if there are long gaps in the data, you loose track of the high-order bits
- There are 2 timestamps that don't come from the GEM, and do keep the high-order bits, but they are only good to a few ms
- We can combine these to get precise relative times

GEM Timing Variables (in SVAC tuple)

- LAT timebase is a running counter of ticks (50ns)
  - 25 bits, rolls on overflow (1.67 s)
- GemTriggerTime samples timebase at window close time
- GemOnePpsTime samples timebase when 1PPS signal received
- GemOnePpsSeconds is incremented on 1PPS signal
  - 7 bits, rolls on overflow (128 s)
- Timebase can overflow between 1PPS and event
  - But only once, so we can detect it:
    - GemTriggerTime < GemOnePpsTime
- GemOnePpsSeconds overflows every 128 s
  - Not likely to roll more than once between events
    - But if it does we can't detect it from GEM variables
  - Can use other timestamps to detect multiple overflows

Coarser Timestamps (in SVAC tuple)

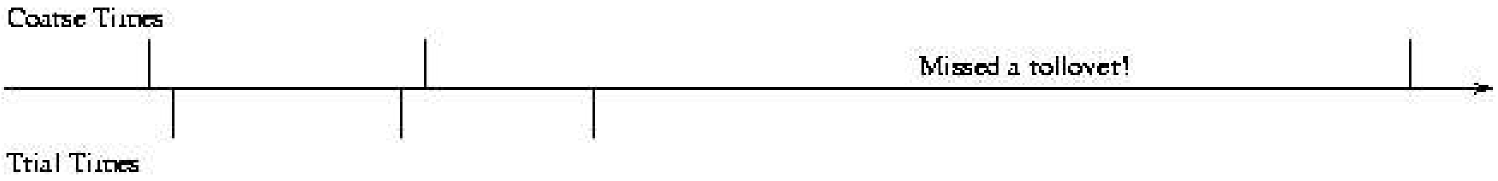
- EvtSecond, EvtNanoSecond come from vxWorks realtime clock (RTC)
  - Updated at 50 Hz
- EvtUpperTime, EvtLowerTime come from SBC CPU cycle counter
  - Updated at ~16 Mhz
  - But 1/60e-9 is closer
  - But we don't really know for sure, and even if we did, it varies by 1 part in ~1e6 (http://www-glast.slac.stanford.edu/IntegrationTest/Weekly%20Minutes/2004-02-12/EMTiming.ppt)
  - Sampled at event build time, not trigger time
    - Queuing can have odd effects

First 2 Tries

- Try to calculate when GemOnePpsSeconds will roll over based on event time using seconds/nanoseconds or upper/lower
  - This is folly
  - Don't know the offsets between the time streams, or even their relative rates, well enough to predict rollovers down to the event
- Try to use long gaps (> 128 s) in seconds/nanoseconds or upper/lower
  - Better, but still doesn't always work
  - Can give spurious rollovers for 64 < gaps < 128 s
  - Coarseness of other timestamps means you can't make an exact cutoff, and there's always a chance of a long separation sneaking into the uncertain region

Third Try

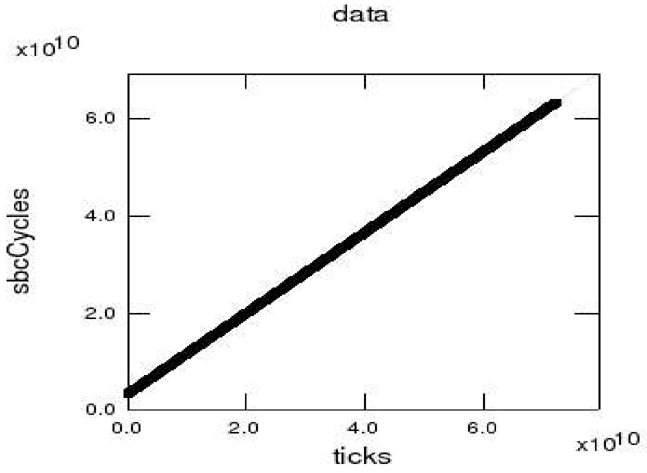
- Use GemOnePpsSeconds, GemOnePpsTime and GemTriggerTime to make trial timestamps, based on assumption that obvious rollovers are the only ones.
  - see next slide
- Compare delta times between events for trial times with deltas from coarser timestamps
- Differences should be within 10-20 ms, unless we missed a PPS rollover
  - Then they will cluster around multiples of 128 s
- Correct trial times if we missed any rollovers
  - Add an appropriate multiple of 128 s (round the difference between deltas to nearest multiple of 128) to all events after the missed roll

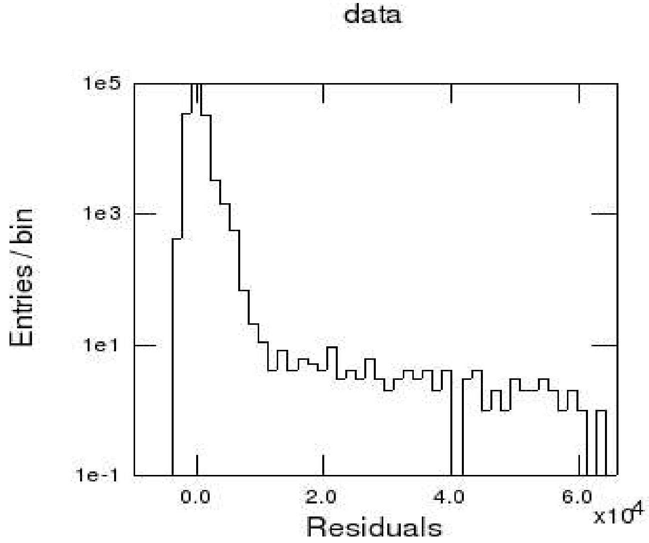


Details

- $\mathrm{trialTime}_i = (\mathrm{nPpsRoll} * 128 + \mathrm{OnePpsSeconds}_{last}) * 20e6 + (\mathrm{TriggerTime}_i - \mathrm{OnePpsTime}_{last})$
  - correct for obvious rollovers
    - $\mathrm{OnePpsSeconds}_{last} < \mathrm{OnePpsSeconds}_{last-1}$
      - nPpsRoll += 1
    - $\mathrm{TriggerTime}_i < \mathrm{OnePpsTime}_{last}$
      - TriggerTime += 2**25
- This assumes that $\mathrm{OnePpsTime}_i - \mathrm{OnePpsTime}_{i-1} == 20e6$
  - currently true, OnePps signal is faked from LAT clock
  - won't be true (?) when we get a GPS

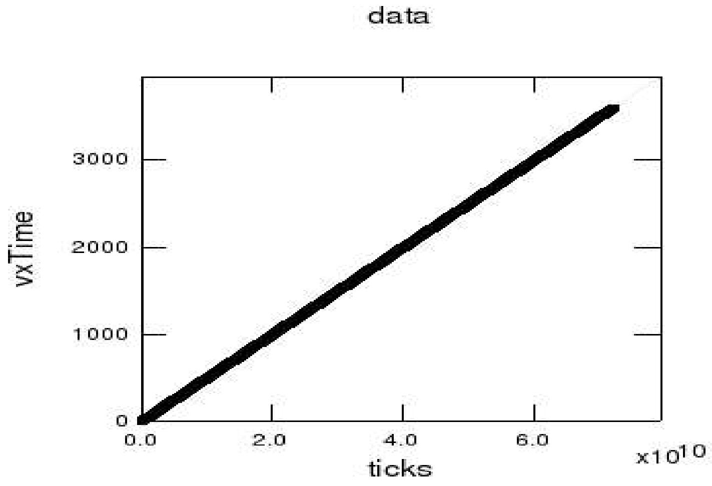
SBC Cycle Counter Performance

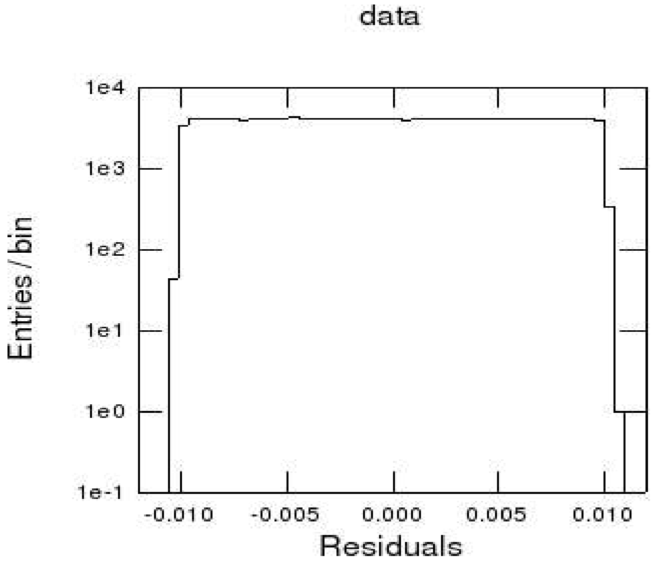




- Linear fit of SBC cycle counter vs LAT ticks
- Fit slope is 0.833329 (+/- 1e-13) cycles/tick (= 60.0003 ns/cycle @ 50 ns/tick)
- Max residual is ~ 3.6 ms

RTC Performance





- Linear fit of VxWorks RTC vs LAT ticks
- Fit slope is 49.9991 (+/- 0.0001) ns/tick
- Residuals are +/- 10 ms, as expected (RTC updates @ 50 Hz)

Wrapup

- Third method seems to work
  - for now
- This is in the SVAC tuple
- Stored as a double
  - all values are integers
  - 53 bit mantissa => 14 years to roll over